

ИНФОРМАТИКА

7 - 9

класс

«Сорос-Кыргызстан» Фонду

Бишкек 2019

УДК 373.167.1
ББК 73. Я 721
И 74

И 74

Информатика: 7-9-класс. Окутуу кыргыз тилинде жүргүзүлгөн жалпы билим берүүчү мектептер үчүн окуу китеби / И. Н. Цыбуля, Л. А. Самыкбаева, А. А. Беляев, Н. Н. Осипова, У. Э. Мамбетакунов – Б.: «Сорос-Кыргызстан» фонду, 2019. – 205 б.

ISBN 978-9967-11-700-6

Бул окуу китеби жалпы билим берүүчү мектептердин 7-9-класстарынын окуучулары жана информатиканын негиздерин жана программалоону үйрөнүүнү баштоого даяр бардык курактагы балдарга арналат. Окуу китеби маалыматтык процесстерди терең түшүнүүгө, тармактык технологияларды, булуттук сервистерди коопсуз колдонууга, сайттарды жана роботторду программа аркылуу кандайча башкарууну жана программалоону үйрөнүүгө жардам берет. Окуу китебинин темалары информатиканы үйрөнүүнүн төрт негизги бөлүмүндө ачылып берилет, алар: Информатика жана маалымат, Компьютерлер жана программалык камсыздоо, Компьютердик тармактар жана интернет, Программалоо. Бардык бөлүмдөр материалды үйрөнүүнүн көлөмүн көбөйтүү жана акырындап тереңдетип окутуу менен, ар бир класста кайталанат. Окуу китеби «Информатика» сабагын мектеп программасынын алкагында иштөөдө да, Python программалоо тилин өз алдынча үйрөнүүдө да колдонулушу мүмкүн.

Окурмандардын кеңири чөйрөсү үчүн

Информатика. 7-9-класс

Китептин э-версиясы www.lib.kg сайтына жайгаштырылган.

Котормочу: Касымбек Жунусалиев

Текст редактору: Ысмайыл Кадыров

Техникалык эксперттер: Али Палитаев, Исабек Ташиев, Санжар Маматов, Айдай Сабырова

Арт-директор: Мария Казакова

Компьютердик калыпка салган: Малик Токталиев

Бул окуу китеби ачык билим берүү ресурсу болуп саналат жана Creative Commons Attribution 4.0. CC-BY (авторлукту көрсөтүү менен) ачык лицензиясы алдында **«Сорос-Кыргызстан» Фондунун** колдоосу менен жарыяланды.



Бул лицензия үчүнчү жактарга бүтүндөй китепти же анын каалаган бөлүгүн китептин авторлоруна сөзсүз түрдө шилтеме жасоо менен, эркин түрдө жайылтууга, туунду чыгармаларды (ремикстерди, котормолорду) түзүүгө, ыңгайлаштырууга, анын ичинде коммерциялык максаттарда да пайдаланууга мүмкүндүк берет.

Бул лицензиянын шарттары тууралуу кеңири маалымат <https://creativecommons.org/> сайтын-да берилген.

И 4306022200-18

ISBN 978-9967-11-700-6

УДК 373.167.1

ББК 73. Я 721



«Сорос-Кыргызстан» Фонду, 2019

КИРИШҮҮ

Урматтуу достор!

Силердин алдыңарда азыркы жашоонун маанилүү маңызы болгон — маалыматтык технология тармагындагы керектүү билимдерди бере турган окуу китеби турат. Биз бардыгыбыз азыркы учурда нефть жана газ сыяктуу эле баалуу ресурс болуп калган маалыматтын негизинде түзүлгөн маалыматтык коомдун бир бөлүгү болуп эсептелебиз. Силердин милдетиңер ушул баалуу ресурсту туура колдонгонго үйрөнүп, андан максимум пайда алууга жетишүү. Бул окуу китеби силерди видеофильмдерди жана веб барактарды түзүү технологиялары, маалыматты коддоо системалары жана компьютердик графика менен тааныштырат. Силер электрондук таблицаларды, презентацияларды, маалыматтар базасын кантип түзүүнү билесиңер, Wi-Fi локалдык тармагын тескегенди жана Python тилинде программалоого үйрөнөсүңөр. Силер өзүңөргө жана айлана-чөйрөгө башка көз караш менен карайсыңар, силер үчүн жаңы жана кызыктуу дүйнө ачылат! Силер үйрөнө турган заманбап маалыматтык технологиялар айлана-чөйрөгө болгон көз карашыңарды бир топ өзгөртөт. Бул окуу китеби өзүңөрдүн өсүү деңгээлиңерди баалап тургандай болуп түзүлгөн: ар бир теманын аягында теориялык билимиңерди текшерүү үчүн суроолор жана тапшырмалар берилип турат. Андан тышкары анын жардамында өзүңөрдүн практикалык көндүмүңөрдү өнүктүрүү үчүн компьютердик практикумдар берилген.

Окуу китебинде төмөнкүдөй шарттуу белгилер кездешет:

Шарттуу белгилер



ЭСИҢЕ ТУТ

– маанилүү маалымат, аны жакшы эстеп калуу керек.



БУЛ КЫЗЫКТУУ!

– тема боюнча кошумча маалымат.



АНЫКТАМА

– жатка билүү керек болгон теориялык маалыматтар.



СУРООЛОР ЖАНА ТАПШЫРМАЛАР

– окуу китебинин түшүндүрүүчү текстинде өзүн өзү көзөмөлдөө үчүн.



КОМПЬЮТЕРДИК ПРАКТИКУМ

– компьютерде өз алдынча аткаруу үчүн тапшырмалар.

Дүйнө кайраттуу жана билимге умтулгандарга ачылат – өзүңөргө суроо бергиле, аларга жооп издегиле, өздүк тартипти сактагыла, эксперимент жүргүзүүдөн коркпогула. Силердин колуңардан баары келет!

МАЗМУНУ

7-КЛАСС

Мазмуну

1 Информатика жана маалымат

- 10 Компьютер адамдын жашоосунда
- 12 Маалыматтык процесстер жана малыматты сактоо
- 16 Тексттик маалыматты коддоо

2 Компьютер жана ПК

- 20 Программалык камсыздоолордун түрлөрү жана курамы
- 22 Электрондук таблицалар
- 27 Презентациялар

3 Программалоо

- 32 Python программалоо тили
- 38 Маалыматтардын тиби жана алар менен болгон амалдар
- 42 Шарттуу операторлор
- 45 while жана for циклдери

4 Компьютердик тармактар жана интернет

- 50 Татаал издөө суроо-талаптары
- 51 Сайт конструкторлору
- 54 Электрондук почта жана булуттук сервистер

8-КЛАСС

1 Информатика жана маалымат

- 60 Логикалык туюнтмалар жана амалдар
- 63 Логиканын мыйзамдары
- 66 Логикалык туюнтмаларды чыгаруу

2 Компьютер жана ПК

- 72 ПК жана лицензиянын түрлөрү
- 75 Маалыматтар базасы

3 Программалоо

- 82 Татаал шарттар: and, or, not

- 84 Тизмелер, кортеждер жана сөздүктөр
- 87 Циклдик алгоритмдер
- 92 Камтылган шарттуу амалдар жана циклдер
- 96 Функциялар
- 102 Массивдер
- 107 Саптар жана алар менен болгон амалдар
- 113 Саптарды форматтоо
- 115 Python тилинде графика менен иштөө

4 Компьютердик тармактар жана интернет

- 122 Компьютердик тармактар
- 127 Интернет протоколдордун түрлөрү
- 131 Стилдердин каскаддык таблицалары (CSS)

9-КЛАСС

1 Информатика жана маалымат

- 140 Маалыматтык сабаттуулук
- 143 Шифрлөө жана электрондук-санариптик кол тамга
- 146 Графикалык маалыматты коддоо

2 Компьютер жана ПК

- 154 Компьютердик графика
- 158 Робототехникага киришүү

3 Программалоо

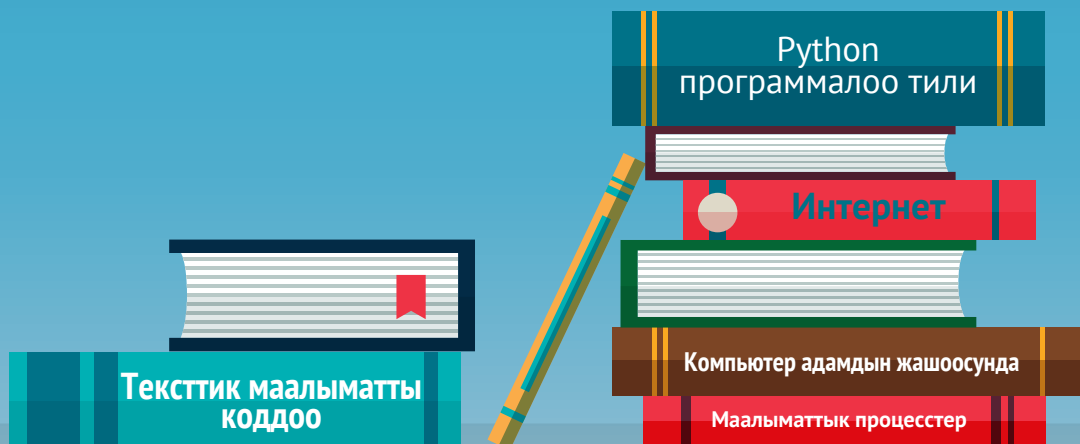
- 166 Рекурсия
- 170 Массивдерди иштетүү алгоритмдери
- 177 Тизмелерди сорттоо
- 182 Матрицалар

4 Компьютердик тармактар жана интернет

- 188 Келечектин технологиялары
- 192 Санариптик дүйнөдөгү коопсуздук



7 – класс



1

- бөлүм



Информатика жана маалымат

1.1-тема:**Компьютер адамдын жашоосунда**

Айлана-чөйрөнү карагылачы, бизди курчап тургандардын бардыгы: класстагы проектор, үйдөгү муздаткыч, көчөдөгү жол чырак, заманбап мультфильмдер жана онлайн оюндардан тартып атомдук станциялар жана Айдагы роботторго чейин компьютерлердин жардамында түзүлгөн жана башкарылат.

Маалыматтык технологиялар (МТ) биздин күндөлүк жашообузга сиңип бара жатат. Алар турмушубузду ыңгайлуу жана коопсуз кылып, эң негизгиси кандай гана маселе болбосун аларды тез арада чечкенге жардам берет. Азыр биз көз ирмемде керектүү маалыматты онлайн энциклопедиялардан таап же болбосо башка өлкөдөгү досубузга билдирүү жөнөтө алабыз.

Учурда жасалма интеллект, виртуалдуу жана толукталган реалдуулук, жасалма органдарды өндүрүү үчүн биотехнологиялар сыяктуу фантастиканын чегиндеги технологиялар кеңири колдонула баштады. Мындан бир канча жыл мурда принтерде үч өлчөмдүү басып чыгаруу деген фантастика эле, азыр болсо бүтүндөй үйдү басып чыгаруучу принтерлер жасалып чыкты.

Ошол эле учурда глобалдуу компьютерлештирүү заманында компьютердин адамдын дене-бой жана психикалык ден соолугуна тийгизген таасири жөнүндө да суроо жаралат.

Компьютерде көп олтуруу адамдын нервдик, эндокриндик, иммундук системаларынын өзгөрүүсүнө алып келип, анын сөөк-булчуң аппаратына жана көзүнө терс таасирин тийгизет.

Компьютер жана башка электрондук түзүлүштөр менен иштөөдө адамга тийгизген зыяндуу факторлор:

АНЫКТАМА:

Гиподинамия – бул организмдин кыймыл активдүүлүгүнүн аздыгы, жалпы дене-бой активдүүлүгүнүн азайышы.



Технологиялардын өнүгүүсүнүн жакшы жактарына мисалдар:

- 1 ачык электрондук энциклопедияларга эркин кирүү;
- 2 маалыматтарды дароо жиберүү;
- 3 интернетти такси чакырууга, билеттерге заказ берүүгө пайдалануу;
- 4 онлайн дүкөндөрдөн буюмдарды сатып алуу;
- 5 болжолдоо үчүн жасалма интеллектти пайдалануу;
- 6 реалдуу объекттерди 3Dда басып чыгаруу.

Компьютерде иштөөдө өзүңөрдүн ден соолугуңарга зыян келтирбеш үчүн төмөнкү эрежелерди сактоо керек:

- компьютерде үзгүлтүксүз иштөө режими 20 мүнөттөн ашпашы керек;
- тыныгууларда жеңил көнүгүүлөрдү жасап туруу керек;
- көзүңөр ачышса, көрүүңөр кескин начарласа, колуңардын манжаларында ооруксунууну сезсеңер же жүрөгүңөрдүн тез-тез согушун байкасаңар тез арада иш ордуңарды таштап, мугалимге билдиргиле;
- көздөн экранга чейинки аралык — 50-70 см (сунулган колдун аралыгындай болуш керек);
- моюн кичине ийилген абалда;
- экран көздүн деңгээлинен кичине төмөнүрөөк жайгашышы керек;
- дене түз, ийиндер түшкөн жана эркин абалда болушу керек;
- тизелер жана чыканактар тик бурч боюнча жайгашышы керек;
- стулдун чети тизенин артына тийбей турушу керек;
- буттар түз турушу жана бири-бирине учкаштырылбашы керек.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) Компьютердин адамдын жашоосуна тийгизген оң жана терс таасирлери боюнча таблицаны толуктагыла:

ОҢ ТААСИРИ		ТЕРС ТААСИРИ	
1	Баарлашуунун жеткиликтүүлүгү	Компьютердик көз карандылык	
2	Каалагандай маалыматтын жеткиликтүүлүгү	“Керексиз маалыматтар менен мээни толтуруу”	
3	Жумуштарды автоматташтыруу	Экрандан окуу татаалдыгы	
4	Жекече окуу мүмкүнчүлүгү	Спам, фрагменттүү окуу	
5	Көрсөтмөлүүлүк презентацияларды түзүү мүмкүнчүлүгү	Реалдуулукту виртуалдуу дүйнө менен алмаштыруу	

1.2-тема:

Маалыматтык процесстер жана маалыматты сактоо

Адам маалымат дүйнөсүндө жашайт. Ар бир адам маалыматты өзүнүн эс тутумунда, андан тышкары ар түрдүү сырткы алып жүрүүчүлөрдө: кагазда, магниттик, оптикалык дисктерде жазуу түрүндө сактайт. Маалыматты алуу, сактоо, иштетүү жана берүү менен байланышкан процесстер **маалыматтык процесстер** деп аталат.

Качан гана айлана-чөйрөнү сезүү органдарыбыздын жардамы менен сезүүдө – биз *маалыматты кабыл алган* болобуз. Башка адамдар менен баарлашуу процессинде бир убакта биз *маалыматты алабыз* жана *беребиз*. Дүйнөдө туура багыт алуу үчүн алынган *маалыматтарды эстеп* калабыз, б.а. маалыматты сактайбыз. Кайсы бир максаттарга жетүү үчүн жана туура чечим кабыл алуу үчүн биз *маалыматты иштетебиз*.



Маалыматтын көлөмүн өлчөө

Сактоо формасы боюнча маалыматты аналогдук жана санариптик деп экиге бөлүшөт. Эки форманын айырмасы – бул аналогдук маалымат үзгүлтүксүз, ал эми санариптик болсо – дискреттүү. Маалыматты сактоо үчүн адатта анын көлөмүн өлчөө керек.

Информатикада маалыматты өлчөө үчүн эки негизги ыкма колдонулат:

1

Мазмундук ыкма

2

Алфавиттик ыкма

Мазмундук ыкма

Мындай ыкмада маалыматтын көлөмү алынган билдирүүнүн маалымат берүүчүлүгүнө жараша болот. Эгерде билдирүү маалымат бере албаса (билдирүүнү кабыл алуучу үчүн белгилүү маанини бербесе), анда маалыматтын көлөмү нөлгө барабар.

Мындай учурда билдирүүнүн маалымат берүүчүлүгү андагы **пайдалуу** маалыматтын көлөмүнө көз каранды болот. Андай маалымат кайсы бир кырдаалдагы белгисиздикти толугу менен жоёт же аны азайтат.

Мисалы, адамдын кийинки аялдамадан чыгышы жөнүндөгү маалыматтагы белгисиздик «Ооба» же «Жок» деген жооптун эки вариантынын бирин тандоо менен аныкталат.

Ыктымалдыгы барабар болгон окуялардагы маалыматтын көлөмүн (i) аныктоо үчүн $N=2^i$ формуласы колдонулат. Мында N – бул мүмкүн болгон окуялардын саны.

1-маселе. Жүргүнчүдөн сурашты: «Сиз кийинки аялдамадан чыгасызбы?» – «Ооба», – деп жооп берди ал. Жооп канча маалыматты камтыйт?

Чыгаруу:

$$N = 2^i$$

$N=2$, («Ооба» же «Жок» деген жооптун варианттарынын мүмкүн болгон саны).

Анда:

$$2=2^i$$

$$i=1$$

Жооп: 1 бит.

1-мисал. Китеп жогорку же төмөнкү текченин биринде турат. Китеп төмөнкү текчеде турат деген билдирүү белгисиздикти туптуура эки эсеге азайтат жана 1 бит маалыматты камтыйт.

2-мисал. Информатика боюнча олимпиадага 4 окуучу катышат. 2-катышуучу эң көп балл алды деген билдирүү, баштапкы белгисиздикти туура 4 эсе (эки жолу экиден) азайтат жана 2 бит информацияны камтыйт.

Алфавиттик ыкма

Мындай ыкмада маалыматтын көлөмү каалагандай тилдеги (табигый же формалдуу) текстте колдонулган символдордун маалыматтык салмагына көз каранды болот.

Алфавиттик ыкмада алфавиттеги цифраларды жана тыныш белгилерди кошкондогу символдордун толук санын билдирген **алфавиттин кубаттуулугу** түшүнүгүн колдонушат.

Мисалы: орус тамгаларынын жана колдонулган символдорунун алфавитинин кубаттуулугу 54: 33 тамга + 10 цифра + 11 тыныш белги, кашаа, бош орун.

Эң аз кубаттуулукка компьютерде колдонулган алфавит (машианын тили) ээ. Аны экилик алфавит деп аташат, анткени ал «0» жана «1» деген 2 маанини гана алат. Экилик алфавиттин символунун маалыматтык көлөмү маалыматты өлчөөнүн бирдиги катары кабыл алынган жана 1 бит деп аталат.

Бит боюнча алфавиттеги бир символдун маалыматтык көлөмүн төмөнкү таблица боюнча эсептөөгө болот:

Варианттардын саны
(алфавиттин кубаттуулугу)

2 4 8 16 32 64 128 256 512 1024

Маалыматтагы
биттин саны

1 2 3 4 5 6 7 8 9 10

Ошентип, алфавиттин кубаттуулугунун формуласы мындай болот:

$$M=2^K,$$

мында **M** – алфавиттин кубаттуулугу,

K – биттин саны (символдун маалыматтык салмагы).

Билдирүүдөгү маалыматтын санын (**S**) аныктоо үчүн, ошол тексттеги символдордун санын (**N**), берилген алфавиттеги бир символду коддоо үчүн кеткен биттердин санына (**K**) көбөйтүү керек:

$$S=N \cdot K.$$

1-маселе. Алфавит 32 тамганы камтыйт. Анын бир тамгасы кандай канча көлөмдөгү маалыматты камтыйт?

Чыгаруу:

Алфавиттин кубаттуулугу $M=32$.

1) $32=2^5$, демек бир символдун салмагы $K = 5$ бит болот.

Жообу: 5 бит.



АНЫКТАМА

Табигый тил – адамдардын ортосунда баарлашуу үчүн колдонулган тил (орус, кыргыз, англис ж.б. тилдер)

Формалдуу тил – билдирүү түзүлүшүнүн так эрежелери менен мүнөздөлгөн жасалма тил: ноталар, Морзе алиппеси, химиялык элементтердин формуласы, программалоо тилдери.

2-маселе. 64 символдук алфавиттин тамгалары менен жазылган билдирүү 20 символду камтыйт. Байт менен бул канча көлөмдөгү маалыматты камтыйт?

Чыгаруу:

Алфавиттин кубаттуулугу $M=64$.

1) $64=2^6$, демек бир символдун салмагы $K=6$ бит.

2) $20 \text{ символ} * 6 \text{ бит} = 120 \text{ бит}$.

3) $120 \text{ бит} : 8 = 15 \text{ байт}$.

Жообу: 15 байт.

3-маселе. Бир уруу 32 символдук, ал эми экинчи уруу 64 символдук алфавитке ээ. Уруулардын башчылары бири-бири менен кат алышты. Биринчи уруунун катында — 80 символ, ал эми экинчисиникинде — 70 символ камтылган. Каттардагы маалыматтын көлөмдөрүн салыштыргыла.

Чыгаруу:

Биринчи уруу: $2^K = 32$, $K = 5$ бит – ар бир символду алып жүргөн маалыматтын көлөмү, $80 * 5 = 400$ бит.

Экинчи уруу: $2^K = 64$, $K = 6$ бит – ар бир символду алып жүргөн маалыматтын көлөмү, $70 * 6 = 420$ бит.

Жообу: экинчи уруунун катындагы маалыматтын көлөмү көбүрөөк.

4-маселе. 12288 битти камтыган билдирүү канча килобайтты түзөт?

Чыгаруу:

1 килобайт = 1024 байт, 1 байт = 8 бит.

$12288 : 8 : 1024 = 1,5 \text{ Кб}$.

Жообу: 1,5 Кб

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Эгерде үй 8 кабаттуу болсо, «Бакыт 5-кабатта жашайт» деген билдирүүдөн канча бит маалымат алынды?
- 2) 16 символдук алфавиттеги 384 символдон канча килобайт маалыматты түзүүгө болот?
- 3) Программалоо тилинин алфавити А дан Z ке чейинки чоң жана кичине тамгаларды жана арифметикалык амалдардагы белгилерди камтыйт. Бул программалоо тилинин алфавитинин кубаттуулугу кандай?

1.3-тема:

Тексттик маалыматты коддоо

Каалагандай текст символдордон, ошондой эле тамгалардан (баш же кичине), цифралардан, тыныш белгилерден, атайын символдордон, мисалы «=», «(», «&» ж.б.у.с. жана сөздөрдөн арасындагы боштуктардан турат. Компьютердин эс тутумунда символдордун ордунда алардын номерлери – сандык коддору сакталат.

Текстти стандарттык коддоо

Коддук таблица – сандык код жана символдордун ортосундагы дал келүүчүлүктү түзүүчү таблица. Алгач бир символду коддоо үчүн 1 байт (8 бит) колдонулган. Мындай коддук таблица 256га чейин (28) символду гана камтыйт. Көп тилдүү тексттерди көрсөтүү үчүн 1991-жылы **Юникод** (англ. **Unicode**) деп аталган жаңы эл аралык стандарт пайда болгон. Мында 1 символго 2 байт бөлүнөт, ал болсо 65536 символду коддоого мүмкүндүк берет. Юникод стандартынын толук спецификациясы дүйнөдөгү бардык алфавитти өз ичине камтыйт.

2017-жылы Юникод стандартынын 10.0 версиясы кабыл алынган. Ал 137 439дан көп ар түрдүү символдорду камтыйт. Азыркы учурда көп байттуу коддоолор колдонулат (бир символду көрсөтүүгө бир нече байт керектелет).

ASCII таблицасы

ASCII (англ. **American Standard Code for Information Interchange**) – маалымат алмашуу үчүн америкалык стандарттык код.

ASCII бул ондук цифраларды, латын жана улуттук алфавиттерди, тыныш белгилерди жана башкаруучу символдорду көрсөтүү үчүн кодго айландырууну түшүндүрөт.

Одөн 32ге чейинки коддор – бул атайын символдор, өзүнө кийинки сапка өтүү, бош орун, киргизүү ж.б. кодун камтыйт. 33төн 127ге чейинки коддор интернационалдык деп аталат жана латын алфавитине, цифраларга, тыныш белгилерине, арифметикалык амалдарга туура келет.



АНЫКТАМА

Код – бул маалыматты көрсөтүү үчүн шарттуу белгилердин жана эрежелердин системасы.

Коддоо – бул берилген коддун жардамында маалыматты көрсөтүү.

Улуттук алфавиттин тамгаларын чагылдыруу үчүн 128ден 255ке чейинки коддогу символдорду колдонушат. Улуттук алфавитте бир эле кодго ар кандай символдор дал келет. Ошондуктан текстти туура көрсөтүү үчүн ага тиешелүү коддук баракты (программаларды тескөөдө) орнотуу керек.

1-маселе. Тексттик файл 32 барактан турат. Ар бир баракта 40 сап, ар бир сапта 48 символ бар. Ар бир символ 8 бит менен коддолгон КОИ-8 коддоосундагы макаланын көлөмүн аныктагыла.

Чыгаруу:

Макаладагы символдордун санын аныктайбыз: $32 \cdot 40 \cdot 48 = 61\,440$.
Бир символ бир байт менен коддолот. 10^{24} байт 1 килобайтты түзөт, ошондуктан макаланын маалыматтык көлөмү: 60 Кб.

Жообу: 60 Кб

2-маселе. Unicode дун бир коддоосунда ар бир символ 16 бит менен коддолот. Бул коддоодо төмөнкү сүйлөмдүн өлчөмүн аныктагыла:

Жети өлчөп, бир кес!

Чыгаруу:

Сүйлөмдө 20 символ бар. Демек, Unicode коддоосундагы бул сүйлөмдүн өлчөмү: $20 \cdot 16 = 320$ бит.

Жообу: 320 бит.

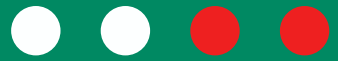
Кириллицаны чагылдыруу үчүн ASCII коддоосунун ар түрдүү бир канча стандарттары бар, анын ичинде: CP1251 (Windows), KOI-8 (UNIX), MacCyrillic (MacOS) ж.б. (1-тиркемени кара).

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Эгерде браузерде сиздин барагыңыз окулбаган текст менен көрсөтүлүп калса эмне кылыш керек?*
- 2) Эмне үчүн стандарттык ASCII коддоосу бардык алфавиттерди көрсөтүүгө жетишсиз?*
- 3) Стандарттык ASCII коддоосунун жардамы менен кытай тилиндеги текстти көрсөтүүгө болобу? Эмне үчүн, түшүндүргүлө?*
- 4) KOI-8 коддоосу колдонулуп жатса, төмөнкү фраза эстин кандай көлөмүн ээлей турганын аныктагыла: **Молекулалар атомдордон турушат.***



- бөлүм



Компьютер жана ПК

2.1-тема:

Программалык камсыздоолордун түрлөрү жана курамы

Салттуу түрдө программалык камсыздоону (ПК) эки түргө бөлүшөт: системалык жана колдонмо ПКга.

Системалык ПК – бул эсептөө системасынын компоненттери болгон процессор, байланыш жана перифериялык түзүлүштөрүн башкаруучу программалардын жыйындысы.



Системалык программалык камсыздоонун эң негизги бөлүгү болуп, төмөндөгү функцияларды аткарган операциялык системасы эсептелет:

- компьютердин ресурстарын колдонууну бөлүштүрөт жана дайындайт (процессор, ЫСТ, дисктер);
- компьютердин ресурстарын колдонууну жана программаларды колдонуунун убактысын пландайт;
- компьютердин ишин контролдойт.

Операциялык системага адам менен компьютердин баарлашуусун камсыз кылган графикалык колдонуучунун интерфейси да кириши мүмкүн. Операциялык системаны башка программалар иштөө үчүн чөйрө деп айтсак да болот.

Көп маселелүүлүк – компьютерде бир эле убакта бир нече ишти аткарууга мүмкүндүк берүүчү механизм: мисалы, музыка угуу жана ошол эле учурда тексттик редактордо иштөө.

Көп маселелүүлүктү камтыбаган система



Көп маселелүүлүктү камтыган система



Көп маселелүүлүктү камтыган системаларда компьютер көбүрөөк натыйжалуу колдонулат.

Виртуалдуу эс тутумдун механизми андан ары система ал бөлүктү ЫСТтын уландысы катары карагандай кылып, сырткы эс тутумдун (катуу дисктен же башка алып жүрүүчүдөн) бир бөлүгүн бөлүп алуу мүмкүндүгүн берет. Жыйынтыгында компьютер чоң көлөмдөгү ЫСТ менен иштөөгө мүмкүндүк алат.

Колдонмо программалык камсыздоо – бул колдонуучунун компьютердеги конкреттүү маселелерди аткаруусун камсыз кылуучу программалар.



Азыркы кезде кеңири тараган колдонмо ПКнын бири болуп, офистик документтер менен иштөөгө арналган OpenOffice.org эркин программасы эсептелет. Ал төмөнкүлөр менен иштейт:

- тексттик документтер менен – OpenOffice.org Writer;
- электрондук таблицалар менен – OpenOffice.org Calc;
- презентациялар менен – OpenOffice.org Impress;
- вектордук сүрөттөр менен – OpenOffice.org Draw;
- маалыматтар базасы менен – OpenOffice.org Base;
- математикалык формулалар менен – OpenOffice.org Math.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Силер колдонгон колдонмо программаларды алардын аткарган функцияларын кошуп санап бергиле.
- 2) Компьютердин иштеши үчүн кайсы системалык программалар сөзсүз керек?
- 3) Жаңы принтерди компьютерге кошуунун алгоритми кандай?

2.2-тема:

Электрондук таблицалар

Чоң көлөмдөгү сандык маалыматарды иштетүү, анализдөө жана сорттоо үчүн электрондук таблицалар (ЭТ) колдонулат.

Алардын жардамы менен силер төмөнкүлөрдү жасай аласыңар:

- ар кайсы уячада эле эмес, ар кайсы бетте жайгашкан берилиштерди бириктирип эсептөөлөрдү жүргүзө аласыңар;
- график жана диаграммалардын жардамы менен маалыматтарды визуалдаштыра аласыңар.



Электрондук таблица төмөнкүлөрдү камтыган **Барактардан** турат: **Мамычалар** (латын алфавитинин тамгалары менен белгиленет) жана **Саптар** (цифралар менен белгиленет).

Сап менен мамычанын кесилиши **Уячаны** түзөт. Активдүү уячанын дареги аттар талаасында көрсөтүлөт, мисалы, A1. Уячаларда сандар, формулалар же текст жайгашышы мүмкүн.

Уячалардын тобу **Диапазонду** түзөт. Ал кош чекит менен бөлүнүп жазылат, мисалы, A1:B5. ЭТнын элементтерин бөлүп алуу ыкмаларын силер 2-тиркемеден көрүп алсаңар болот.

Электрондук таблицаларда берилиштерди киргизүү өзгөчөлүктөрү

Берилиштер активдүү уячага гана киргизилет. Ушул уячага маалыматтарды кайталап киргизүүдө мурунку киргизилген маалымат өчүп калат, ошондуктан редакциялоо үчүн F2 баскычын басуу керек.

Эгерде киргизилип жаткан тексттин узундугу уячанын энинен чоң болсо, анда ал саптын бөлүгү гана көрүнүп турат.



БУЛ КЫЗЫКТУУ!

Эгерде бир нече баракты бөлүп көрсөтсөк, силер бардык бөлүнгөн барактарга бир мезгилде маалыматты кошо аласыңар жана форматтасаңар болот.

	A	B	C	D	E	F
1	Эгерде кошуна уячалар бош болсо, анда сап толугу менен көрүнөт					
2	Эгер тол»	Саптын бөлүгү эле көрүнөт				
3	Ташымалды колдонсо болот.	Уячаны чоюп коюуга да болот				
4						

Маалыматты толугу менен көрсөтүү үчүн, уячанын энин өзгөртүү керек же сапты ташыганга уруксат бериш керек.

Электрондук таблицада берилиштерди сорттоо жана чыпкалоо



Сорттоо – эгерде мамычада бир типтеги берилиштер жайгашса (текст, сандар же даталар) аспаптар панелиндеги баскычтарды же **Берилиштер** менюсундагы **Сорттоо** командасын колдонуп аларды өсүү же кемүү тартибинде сорттоого болот.



Фильтр

Фильтр (чыпка) – экранга берилген шартка дал келген гана берилиштерди чыгарууга мүмкүндүк берет. Тандалган диапазонго чыпканы орнотууну **Берилиштер** менюсунун же Аспаптар панелиндеги баскычтын жардамында жүргүзүүгө болот.

Шарттуу форматтоо берилген шартка дал келген берилиштерди шрифттин гарнитурасын, өлчөмүн же түсүн өзгөртүү аркылуу бөлүп көрсөтүү үчүн колдонулат. Ал үчүн:

1. **Формат** менюсунан **Шарттуу форматтоо** командасын тандап алуу керек.
2. Шартты уячанын маанисине же формулага колдонула.
3. Жасалганын стилин орнотула.

Формулары киргизүү

Формулары киргизүү ар дайым барабардык белгиси «=» менен башталат, андан соң формуланын өзү жазылат. Мисалы $=A4+16$. Мисалы, эгер $A4$ уячасына биз 20 санын жазсак, анда $B4$ кө $=A4+16$ формуласын жазып, *Enter* баскычын бассак, $B4$ уячасында 36 саны пайда болот.

Формула сабында учурда колдонуп жаткан формула жазылып турат жана аны редакциялоого болот. Татаал туюнтмаларды жазууда «+», «-», «*», «/», «^» белгилерин колдонсо болот. Мисалы: $A1+C5*B4$

Салыштырмалуу жана абсолюттук шилтемелер

Формулаларда уячалардын дарегине шилтемелер колдонулат. Шилтемелерди салыштырмалуу, абсолюттук жана аралаш деп бөлүшөт.

Салыштырмалуу шилтемелерде формулаларды көчүрүүдө баштапкы берилиштердеги шилтемелер өзгөрөт (мисалы $A1, B3$).

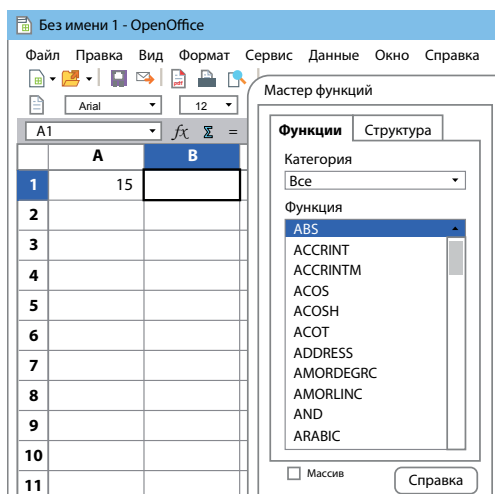
Абсолюттук шилтемелерде формулары которууда же көчүрүүдө баштапкы берилиштери менен уячанын белгиленген дареги колдонулат. Ал шилтеме доллар белгиси менен белгиленет (мисалы $\$A\1).

Аралаш шилтемелерде болсо абсолюттук катары мамычанын дареги (мисалы, $\$A1$) же сапчанын дареги (мисалы, $A\$1$) эсептелет.

Функциялар

Электрондук таблицаларда төмөнкү категориялардагы функциялардын кеңири жыйындысы жеткиликтүү:

- маалыматтар базасы менен иштөөчү;
- дата жана убакытты иштетүүчү;
- финансылык;
- маалыматтык;
- логикалык;
- математикалык;
- массивдер менен иштөөчү;
- статистикалык;
- тексттик;
- кошумча.



Мисалы, «математикалык» категориясынан **СУММ** деген функцияны тандап, бөлүнүп алынган диапазондогу маанилердин суммасын эсептесеңер болот. «Статистикалык» категориясынан **СРЗНАЧ** функциясын тандап, белгиленген диапазондогу маанилердин орточо маанисин чыгарсаңар болот.

Диаграммалар

Электрондук таблицаларда жайгашкан сандык маалыматтарды көрсөтмөлүү кылуу үчүн диаграммаларды жана графиктерди колдонушат.

Колдонуучунун белгилүү маселелерин чечүү үчүн ар түрдүү типтеги диаграммалар сунушталат:

- айланма
- гистограмма
- сызыктуу
- чекиттик ж.б.



Диаграмманын негизги элементтери

- Диаграмма аймагы
- Түзүү аймагы
- X горизонталдык огу – категориялар огу
- У вертикалдык огу – маанилер огу
- Берилиштер катары
- Легенда
- Диаграмманын аты
- Октордун аты
- Берилиштердин жазылышы
- Берилиштер чекити

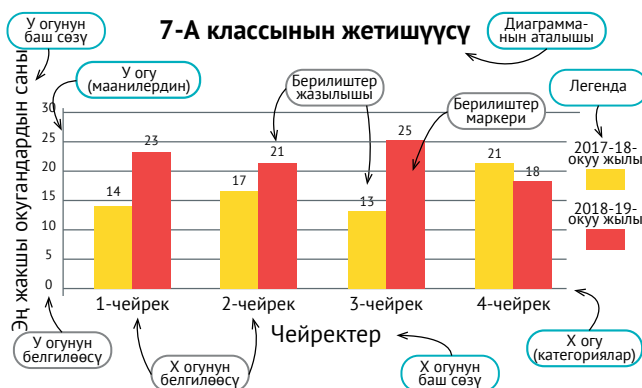


Диаграмма түзүү мисалы:

Дем алыш күндөрүндөгү үй-бүлө мүчөлөрүнүн чыгымдары

Аты	Ишемби	Жекшемби
Ата	200	500
Апа	400	600
Байке	150	300
Эже	300	200

Үлгү боюнча үй-бүлө мүчөлөрүнүн дем алыш күндөрдөгү чыгымдарынын таблицасын түзөлү.

1-кадам. Диаграмма түзүү үчүн берилиштер турган диапазонду бөлүп алышыбыз керек.

2-кадам. *Вставить* менюсунан **Диаграммы** командасын тандайбыз.

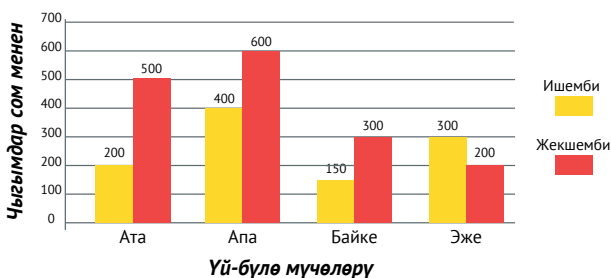
3-кадам. Ачылган *Мастер диаграмм* терезесинде диаграмманы жайгаштыруу ордун, анын тибин (гистограмма) жана атын көрсөтөбүз.

4-кадам. Берилиштер катарынын атын көрсөтүү, тор сызыктарын көрсөтүүнү тандоо, легендалардын жайгашкан ордун көрсөтүү.

Диаграмманы алабыз → **Үй-бүлө мүчөлөрүнүн дем алыш күндөрдөгү чыгымдары**

Диаграмманын параметрлерин өзгөртүү үчүн тиешелүү объектке ЧОБ басып, чыккан контексттик менюдан керектүү команданы тандап алгыла.

Эгерде анын негизинде диаграмма түзүлгөн берилиштерди өзгөртсөңөр, диаграмма автоматтык түрдө кайра түзүлөт.



КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) Учурдагы окуу жылына жылнаама түзгүлө.
- 2) Электрондук таблицада Пифагордун таблицасын түзгүлө.
- 3) Мектеп баскетболчулар командасы 12 окуучудан турат. Эгерде окуучулардын аттары жана алардын бойлорунун узундуктары белгилүү болсо окуучулардын боюнун өсүүсү боюнча диаграмма түзгүлө.
- 4) Музейдин бир жумалык кирешесин эсептегиле, эгер төмөнкүлөр белгилүү болсо:
 - ар бир күнү сатылган билеттердин саны;
 - чоңдордун билетинин баасы – 40 сом;
 - балдардын билетинин баасы чоңдордукуна караганда 25%га арзан.
 Музейдин күндөлүк кирешеси боюнча диаграмма (график) түзгүлө.
- 5) «ЕСЛИ» функциясын колдонуу менен, ар бир абонент электр энергиясы үчүн канча төлөөрүн аныктагыла. Төмөнкү шартты эске алгыла: биринчи 700 кВт/саат үчүн абонент 0,77 сом, ал эми андан ашса 1 кВт/саат электр энергиясына төлөм 2,16 сомго чейин көбөйөт.

	A	B	C	D	
1	№	Кардар	Электр энергиянын саны	Төлөм	
2	2	Асанова	135		
3	3	Давыдов	79		

2.3-тема:

Презентациялар

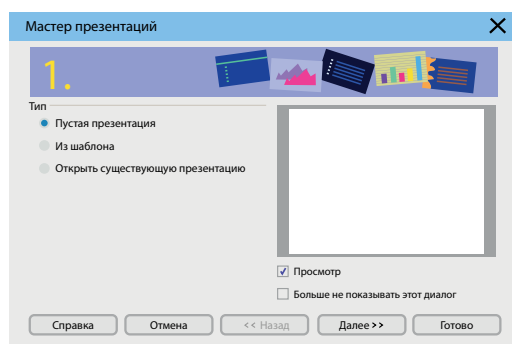


OpenOffice.org Impress программасы текстти, графикалык объекттерди, диаграммаларды, анимацияларды, мультимедиа жана башка элементтерди колдонуп презентация түзүүгө мүмкүндүк берет.

OpenOffice.org Impress программасынын мүмкүнчүлүктөрү:

- эффекттери менен шаблондун негизинде слайддарды түзүү (анимация, өтүү эффекттери);
- ар түрдүү макеттердин негизинде слайддарды түзүү;
- диаграммаларды колдонуу менен презентация түзүү;
- презентацияларды автоматтык же кол режиминде демонстрациялоо.

«Мастер презентаций»дин жардамында презентация түзүү



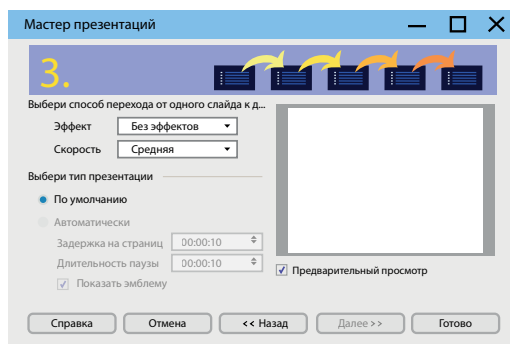
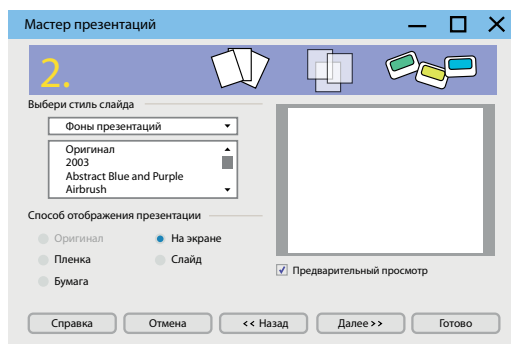
OpenOffice.org Impress программасын ишке киргизүүдө экранда «Мастер презентаций» терезеси пайда болот. «Мастер презентаций» шаблондорду жана редакциялоо үчүн ар кандай мүмкүнчүлүктөрдү колдонуу менен презентация түзүүгө жардам берет.

1-кадам. «Мастер презентаций» терезесинде презентациянын тибин тандагыла:

- «Пустая презентация» – жаңы презентацияны түзөт;
- «Из шаблона» – шаблондор тизмесинен жаңы презентацияны түзөт;
- «Открыть существующую презентацию» – мурда сакталган презентацияны ачат.

2-кадам. Кийинки терезеде «Выберите стиль слайда» группасында жогорку тизмеден дизайндын эки стилинин бирин тандагыла: презентацияны же презентация фонун.

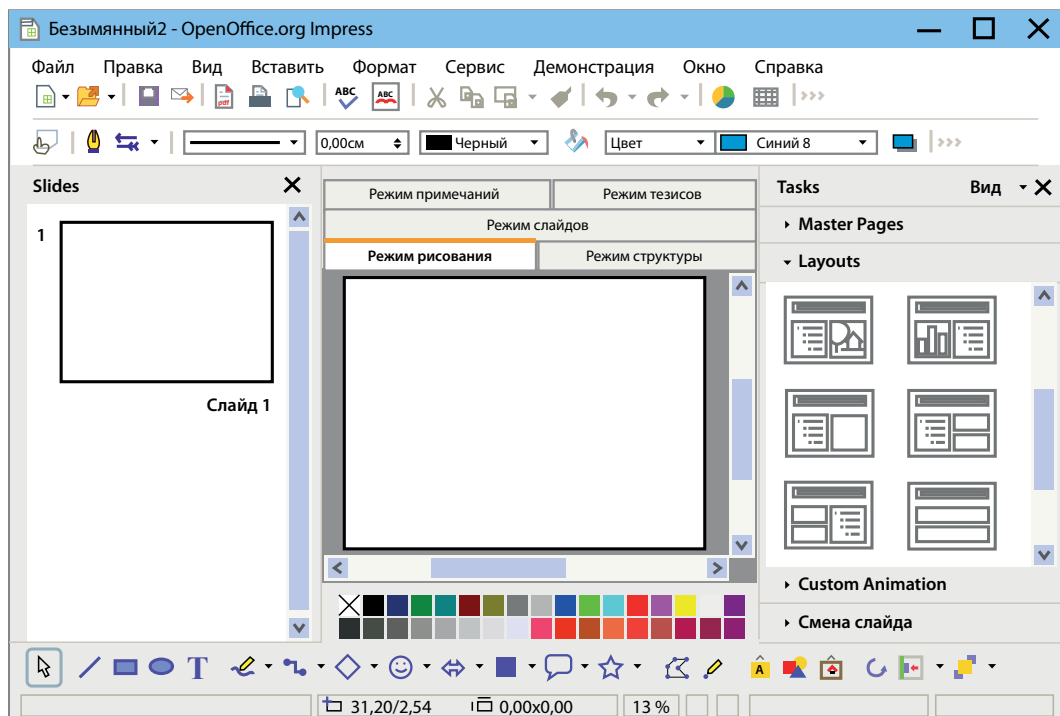
3-кадам. «Способ отображения презентации» группасынан чагылдыруунун бир тибин тандагыла: *Оригинал, Пленка, Бумага, На экране же Слайд* жана «Далее» баскычын басуу керек.



4-кадам. «Выберите способ перехода от одного слайда к другому» группасынын «эффект» тизмесинен презентация үчүн эффектти тандагыла. «Скорость» тизмесине эффекттин иштөө ылдамдыгын орноткула, ал эми «Выберите тип презентаций» группасынан презентациянын убактысын аныктагыла.

Ошентип мастердин жардамы менен презентация түзүү аяктайт. Аягына чыгуу үчүн «Готово» баскычын басуу керек.

Силердин алдыңарда презентацияны форматтоо үчүн терезе ачылат.



Слайддарды форматтоо

Слайддарда тексттерди (шрифтти, өлчөмүн өзгөртүү, түздөө ж.б.) жана графикалык объекттерди (өлчөмдөрүн, жарыктыгын, агуучулугун өзгөртүү) форматтоо каалагандай тексттик редакторлордо жүргүзүлгөндөй эле ишке ашырылат.


Титулдук слайдды жасалгалоо үчүн «Заголовок, слайд» макетин орнотуп, презентациянын баш сөзүн киргизгиле.

Кийинки слайдды кошуу үчүн «Вставка» менюсунан «Слайд» командасын тандоо керек же ошол слайддан ЧОБду басып чыккан контексттик менюдан «Новый слайд» командасын тандап алуу керек.

Слайдга тексттик талаа коюу үчүн чыккан менен **Т** пиктограммасын басуу керек. Андан соң талааны коё турган орунга ЧСБны басып коюу керек.

Графикалык объектти коюу үчүн «Вставка-Изображение-Из файла» менюсун колдонушат.

Презентациянын тексттерден айырмасы, анда анимация кошуу мүмкүнчүлүгү бар. Графикалык объекттерге жана текстке маселелер панелиндеги «эффекты» группасынан ар кандай эффекттерди коюуга болот.

Презентацияны демонстрациялоо үчүн Аспаптар панелиндеги Демонстрация  баскычын, же F5 баскычын басуу керек.

Демонстрацияны каалаган жеринен бүтүрүү үчүн, же аягына чыгуу үчүн ESC баскычын же чыккандын баскычын басуу керек.

Презентацияны сактоодо файлдын тибине көңүл бургула. Эгерде силердин презентацияңар башка программаларда да мисалы, Microsoft PowerPointте ачылсын десеңер, файлдын тиби деген талаада Microsoft PowerPoint – версия – (.ppt) деп көрсөтүү керек.

Презентацияны видео форматта сактоо үчүн өзүңөрдүн ишиңерди башка форматка экспорттош керек, ал үчүн «Файл» менюсунан «Экспорт» командасын тандоо керек.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Анимацияны колдонуп «Космонавтиканын тарыхы» деген темада презентация түзгүлө жана аны фильм түрүндө сактагыла.

3

- бөлүм

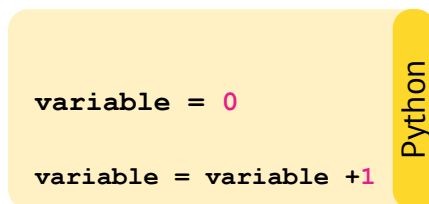
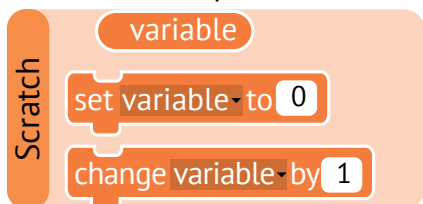


Программалоо

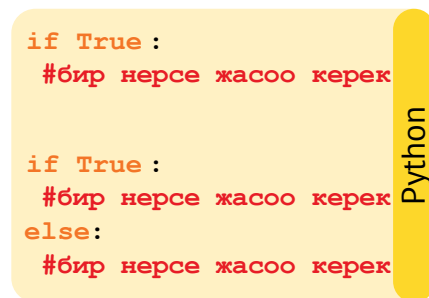
3.1-тема:

Python программалоо тили

Python программалоо тили – бул жаңы үйрөнүп жаткандар үчүн да жеткиликтүү болгон, ар түркүн багыттагы программаларды түзүү үчүн колдонулган аспап. Эгерде силер Scratchте блоктор менен программа түзүп жүргөн болсоңор, анда Python тилинде программа түзүү силерге оңой эле болот. Келгиле, командалар Python жана Scratch тилдеринде кандай жазыла тургандыгын салыштыралы.



Циклдер ушундай көрүнөт:



Python коду оңой окулат, ал эми интерактивдүү кабыкчасы болсо программаны киргизип, ошол замат жыйынтыгын алганга мүмкүндүк берет.

Бүгүнкү күндө бул программалоо тилинде банктар, телекоммуникациялык компаниялар үчүн программалар жазылат, көптөгөн аналитиктер ушул тилдин жардамындагы маалыматтар менен иштешет. Өтө жөнөкөй синтаксистин натыйжасында бул тилде программалоону баштоо жеңил.

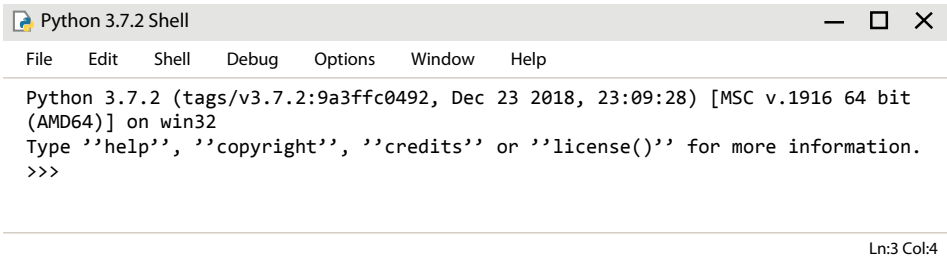
Python тилинде программалоо үчүн өзүңөрдүн компютериңерге акысыз программалоо чөйрөсүн орнотушуңар керек. Аны <https://www.python.org/downloads/> сайтынан жүктөп аласыңар.

Python менен бирге компютерде программаларды киргизүү үчүн IDLE – иштетүү чөйрөсү да орнотулат.

Python-командаларды жазуу, сактоо жана аткаруу үчүн төмөнкү кадамдарды жасоо керек болот:

1 -кадам. IDLEни ишке киргизүү

Түзгөн программаңардын жыйынтыгын көрүп тургандай консоль терезеси ачылат.



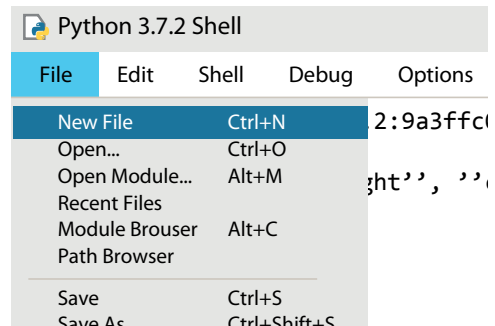
2 -кадам. Жаңы файлды түзүү

Жаңы программаны жазуу үчүн өзүнчө файл түзүү керек. Ал үчүн **File** менюсунан **New File**ды тандап алабыз.

3 -кадам. Программаны киргизүү

Ачылган терезеде өзүңүздүн программаңызды киргизиңиз, мисалы:

```
print('Salam!')
```



4 -кадам. Программаны сактоо

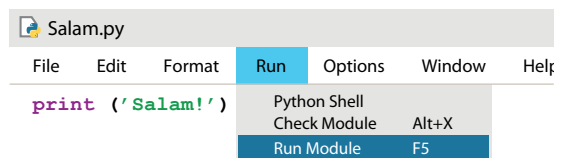
File менюсунан **Save As** дегенди тандап, файлыңар үчүн жаңы атты киргизгиле жана **Save** командасын баскыла. Биз алдын ала My scripts папкасын түзүүнү жана ошол жерге өз программаңардын бардыгын сактоону сунуштайбыз.

5 -кадам. Программаны ишке киргизүү

Run менюсунан **Run Module** командасын тандагыла (же тез иштетүү үчүн F5 баскычын баскыла)

Даяр! Консолдо силер төмөнкү маалыматты көрүшүңөр керек:

```
>>>
Salam!
```



Программа биринчиден файлга жазылып (адатта **.py** кеңейтилиши менен) жана ишке киргизүүдө толугу менен аткарылган ыкма *программалык режим* деп аталат. Мындай программалык файл Pythonдо скрипт деп аталат (англ. **Script** – сценарий).

Pythonдо андан тышкары бир да операторду (команданы) камтыбаган бош программалар да болот. Көбүнчө булар комментарийлер (адатта **#** белгисинен кийин жазылат) – интерпретатор менен иштетилбеген түшүндүрмөлөр.

бул бош программа



АНЫКТАМА

Интерпретатор – бул силердин программаңарды окуп, андагы камтылган нускамаларды аткаруучу программа.

Python тилиндеги функциялар

Pythonдо көптөгөн функциялар бар, аларды чакырууда андагы камтылган командаларды программалар аткарышат.

print()

print функциясы маалыматты экранга чыгарып берет. Ал өзгөрмөлөрдүн маанилерин, ошондой эле туюнтманын маанисин да чыгара алат.

input()

input функциясы тескерисинче, колдонуучу тарабынан программага берилиштерди киргизүүгө мүмкүндүк берет.

randint()

randint функциясы белгилүү диапазондон кокустук санды кайтарат.

*Кашааларга бул функция иштей турган маанилери көрсөтүлөт.

print функциясы апострофко же тырмакчаларга алынган символдорду экранга чыгарат. Бул команда аткарылгандан кийин автоматтык түрдө кийинки жаңы сапка өтөт.

Өзгөрмөлөр

Эки санды кошууну аткаруучу программанын алгоритмин түзөбүз:

- 1) колдонуучудан бүтүн эки санды сурайт;
- 2) аларды кошот;
- 3) кошуунун жыйынтыгын чыгарат.

Маселенин шартында эске сакташ

керек болгон маалыматтар пайда болду дейли. Ушулар үчүн өзгөрмөлөрдү колдонушат. Өзгөрмөлөр (каалагандай эс уячасындай) бир гана маанини сактай алат. Ага жаңы маанини жазууда мурункусу өчүрүлөт жана аны эч кандай калыбына келтирүүгө да болбойт.



АНЫКТАМА

Өзгөрмө – бул атка, тибине жана маанисине ээ болгон чоңдук. Өзгөрмөнүн мааниси программанын аткарылуу учурунда өзгөрө алат.

Python тилинде өзгөрмөлөр биринчи колдонулганда эле, тагыраак айтканда аларга алгачкы маанилерди ыйгарууда эсте түзүлөт.

Мисалы ыйгаруу операторун аткарууда:

```
a = 4
```

эсте **a** аталыштагы жаңы өзгөрмө түзүлөт («бүтүн сан» тибиндеги объект). Эми ушул ат менен өзгөрмөгө кайрыла берсек болот: анын маанисин окуйбуз жана өзгөртөбүз.

Өзгөрмөлөрдүн атында латын тамгаларын (кичине жана баш тамгалар айырмалуу болот), цифраларды жана астына сызылган сызыкты («_») колдонсо болот.

Колдон келсе өзгөрмөлөргө ал кандай роль аткара тургандыгын ошол замат түшүнө тургандай тиешелүү аттарды бериш керек.



ЭСИҢЕ TUT

Өзгөрмөнүн аты цифра менен башталбаш керек, анткени трансляторго кайсы жерден цифра, ал эми кайсы жерден өзгөрмөнүн аты баштала тургандыгын айырмалоо кыйын болуп калат.

Pythonдо өзгөрмөнүн тиби автоматтык түрдө аныкталат. Өзгөрмөнүн тиби жөнүндө кеңири маалыматты кийинки темадан карайбыз.

Алгоритмди жазуу үчүн биз үч маселени чечишибиз керек:

- 1 клавиатурадан эки санды киргизип, аларды өзгөрмөлөргө жазуу (аларды **a** жана **b** деп атайлы);
- 2 бул эки санды кошуп, жыйынтыгын **c** деген үчүнчү өзгөрмөгө жазуу;
- 3 **c** өзгөрмөсүнүн маанисин экранга чыгаруу.

Өзгөрмөнүн маанисин киргизүү үчүн **input** оператору колдонулат:

```
a = input()
```

Бул сап аткарылгандан кийин система өзгөрмөнүн маанисин клавиатурадан киргизүүнү күтөт. Enter баскычын басканда, бул маани **a** өзгөрмөсүнө жазылып калат. **input** операторун чакырууда кашаанын ичине түшүндүрмө билдирүүнү жазып койсо болот:

```
a = input('Бүтүн санды киргиз: ')
```

Эки маанини кошуп, жыйынтыгын **c** өзгөрмөсүнө жазуу өтө оңой:

```
c = a + b
```

«=» символу – бул ыйгаруу оператору, анын жардамы менен өзгөрмөнүн маанилери өзгөртүлөт. Ал төмөнкүдөй аткарылат: «=» символунун оң жагындагы туюнтма аткарылат, андан соң анын жыйынтыгы сол жагында жазылган өзгөрмөгө ыйгарылат. Ошондуктан, мисалы

`i = i + 1`, бул команда `i` өзгөрмөсүнүн маанисин бирге көбөйтөт.

`c` өзгөрмөсүнүн маанисин экранга `print` оператору менен чыгарабыз:

```
print (c)
```

Бардык программа:

Python тилиндеги программа

```
a = input ('Бүтүн сан киргизиңиз:')
b = input ('Бүтүн сан киргизиңиз:')
c = a + b
print (c)
```

Экранга чыккан жыйынтыгы

```
Бүтүн сан киргизиңиз: 2
Бүтүн сан киргизиңиз: 3
23
```

Биз көрүп тургандай эки сан кошулган жок: программа аларды биринин артына экинчисин жазып, бириктирип эле койду. Анткени мындай жазууда `input` оператору киргизилген маанилерди сан катары эмес, символ катары кабыл алат.

Бул катаны оңдош үчүн маанилерди киргизүүдө алынган символдук сапты бүтүн санга өзгөртүп түзүшүбүз керек. Бул болсо `int` (англ. *integer* – бүтүн) функциясынын жардамында ишке ашат.

```
a = int (input())
b = int (input())
```

Дагы бир варианты бар: эки сан эки сапта эмес бир сапта бош орун менен ажыратылып киргизилет. Бул учурда маанини киргизүү кыйыныраак:

```
a, b = map (int, input().split())
```

`map()` – берилген функцияны тизменин бардык элементтерине тиешелүү кылат; биздин учурда бул саптагы элементтерди бүтүн санга айландырган `int()` функциясы.

`split()` – сапты маселелерге жараша бөлүктөргө бөлөт; жыйынтыгында тизмени кайтарат.

Жыйынтыгында `map ()` функциясынын ишинен кийин биз эми сандардан турган жаңы тизмени алабыз. Биринчи киргизилген сан (тизменин биринчи элементи) `a` өзгөрмөсүнө, ал эми экинчиси – `b` өзгөрмөсүнө жазылат.

Берилген функцияларды эске алуу менен программаны кайра жазалы:

Python тилиндеги программа

```
a, b = map(int, input().split())
c = a + b
print(c)
```

Экранга чыккан жыйынтыгы

```
2 3
5
```

Эми программа туура иштеп жатат – клавиатурадан киргизилген эки санды кошуп жатат. Бирок мунун эки кемчилиги бар:

- 1) берилиштерди киргизүүдө колдонуучудан эмне талап кылынып жаткандыгы билинбейт (канча, кандай сандарды киргизүү керек);
- 2) жыйынтыгы эч нерсени билдирбеген сан түрүндө чыгат.

Ошондуктан программа менен колдонуучунун ортосунда диалогду түзүү үчүн программаны мындайча өзгөртсө болот:

Python тилиндеги программа

```
print('Эки бүтүн сан киргизиңиз:')
a, b = map(int, input().split())
c = a + b
print(a, '+', b, '=', c, sep='')
```

Экранга чыккан жыйынтыгы

```
Эки бүтүн сан киргизиңиз:
2 3
2 + 3 = 5
```

Колдонуу үчүн маалыматты өзүңүз каалагандай жазсаңыз болот.

Жыйынтыгын чыгарууда үч өзгөрмөнүн маанисин жана `print` операторунда бөлүнгөн эки символду «+» жана «=» чыгаруу керек.

```
print(a, '+', b, '=', c)
```

`print` операторундагы ашыкча боштуктарды өчүрүү үчүн бөлгүч (же сепаратор, англ. *separator*) – `sep` колдонулат.

```
print(a, '+', b, '=', c, sep='')
```

Бул жерде биз бош бөлгүчтү (бош сап) орноттук. Бөлгүч катары каалагандай белгини көрсөтсө болот. Мисалы, эгерде командада `sep = '*'` деп көрсөтсөк

```
print(1, 2, 3, 4, sep='*')
```

анда экранда төмөнкү жыйынтык чыгат: `1*2*3*4*`

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Убакыт интервалы саат, минута жана секунда менен берилген. Ошол эле интервалды секундада гана көрсөтө турган программаны түзгүлө.

3.2-тема:

Маалыматтардын тиби жана алар менен болгон амалдар

Өзгөрмөлөрдү колдонуудан мурда алардын типтери менен таанышалы. Мүмкүн болгон аракеттердин көптүгү силердин тандап алган тибиңерге көз каранды болот.

Маалыматтар тиби

Python тилиндеги негизги маалыматтар типтерин тизмелейли:

int – бүтүн маанилер;

float – чыныгы сан маанилери (бөлчөктүү бөлүгү менен сандар);

bool – логикалык маанилер, **True** (чындык «ооба») же **False** (жалган «жок»);

str – символ же символдук сап, б.а. символдордун чынжыры (катары).

Pythonдогу бүтүн өзгөрмөлөр өтө чоң (же, тескерисинче сөз терс сандар жөнүндө болуп жатса кичинекей) болушу мүмкүн: интерпретатор автоматтык түрдө эсептөөнүн жыйынтыгын сактоого керек болгон өлчөмдө эстин аймагын бөлүп берет. Ошондуктан Pythonдо көп орундуу сандар менен эсептөөлөрдү так жүргүзүү жеңил.

Чыныгы сандарды жазууда бүтүн бөлүгү бөлчөк бөлүгүнөн үтүр менен эмес, чекит менен ажыратылат. Мисалы:

x = 123.456

Логикалык өзгөрмөлөр **bool** тибине кирет жана **True** (чындык) же **False** (жалган) маанилерин алат.

Python тилиндеги программа

```
a = 10
b = 3
c = a/b
print ('c =', float (c))
```

Экранга чыккан жыйынтыгы

c = 3.3333333333333335

```
a = 10
b = 3
c = a/b
print ('c =', int (c))
```

c = 3

```
a = 10
b = 3
c = a/b
print ('c =', bool (c))
```

c = True

Арифметикалык туюнтмалар жана аракеттер

Pythonдо каалагандай арифметикалык амалдарды аткараса болот. Арифметикалык туюнтмалар сапка жазылат. Алар сандарды, өзгөрмөлөрдүн аттарын, арифметикалык амалдардын белгилерин, кашааларды (аракеттердин иретин өзгөртүү үчүн) жана функцияларды чакырууларды камтышы мүмкүн. Мисалы,

$$a = (c + 5 - 1) / 2 * d$$

Амалдардын иретин аныктоодо амалдардын артыкчылыгы (улуулугу) колдонулат. Алар төмөнкү иретте аткарылат:

- 1 кашааанын ичиндеги амалдар;
- 2 солдон оңго даражага көтөрүү (**);
- 3 солдон оңго көбөйтүү (*) жана бөлүү (/);
- 4 солдон оңго кошуу жана кемитүү.

Тең күчтүү туюнтмалардын жазуулары

`a = b = 0`

`b = 0`
`a = b`

`a += b`
`a -= b`
`a *= b`
`a /= b`

`a = a + b`
`a = a - b`
`a = a * b`
`a = a / b`

Бөлүүнүн жыйынтыгы ар дайым чыныгы сан болоорун эстен чыгарбаш керек. Ал эмес бөлүнүүчү жана бөлүүчү да бүтүн болуп, бири-бирине бөлгөндө бүтүн сан чыкса да ал чыныгы сан. Бөтөн сандарды бөлгөндө бүтүн сан алуу үчүн «//» операторун колдонушат. Ал эми бөлүүнүн калдык бөлүгүн алуу үчүн «%» операторун колдонушат (алар көбөйтүү жана бөлүүдөй эле артыкчылыкка ээ).

Python тилиндеги программа

```
d = 85
a = d // 10
print (a)
b = d % 10
print (b)
```

Экранга чыккан жыйынтыгы

8
5

Терс сандар үчүн

```
print (-7 // 2)
print (-7 % 2)
```

-4
1

Сандар теориясынын көз карашы боюнча, калдык – бул оң сан, ошондуктан $-7 = (-4) * 2 + 1$, башкача айтканда -7ни 2ге бөлгөндөгү тийинди -4кө барабар, ал эми калдык 1ге барабар болот.

Pythonдо даражага көтөрүү амалы эки жылдызча менен белгиленет: «**». Мисалы $y = 2x^2 + z^3$ туюнтмасы мындай жазылат:

$$y = 2*x**2 + z**3$$

1-маселе. Үч бурчтуктун аянтын табуучу программа түзгүлө, эгерде анын негизинин узундугу жана бийиктиги белгилүү болсо.

Геометриядан белгилүү болгондой үч бурчтуктун аянты үч бурчтуктун негизинин (a) жарымын анын бийиктигине (h) көбөйткөнгө барабар:

$$S = \frac{1}{2} ah$$

Бул формуланы мындай жазсак да болот: $s=(a*h)/2$

Эми программаны түзөлү жана үч бурчтуктун негизинин узундугун **6 см** жана бийиктигин **4 см** деп киргизели.

Python тилиндеги программа

```
a = float(input('Негизинин маанисин киргизиңиз: '))
h = float(input('Бийиктигинин маанисин киргизиңиз: '))
s = (a*h)/2
print ('Жообу: s=', s)
```

Экранга чыккан жыйынтыгы

```
Негизинин маанисин киргизиңиз: 6
Бийиктигинин маанисин киргизиңиз: 4
Жообу: s= 12.0
```

Кокустук сандар

Кокустук бүтүн санды алуу үчүн алгач Pythonго **randint** функциясын жүктөйлү. Ал үчүн консоль терезесинде **import** командасын колдонобуз:

```
>>> from random import randint
>>> randint (1, 10)
7
```

randint() функциясы биринчи сандан экинчи санга чейинки диапазондо кокустук санды тандап алды. Биздин мисалда ал 7 цифрасын тандады.

Стандарттык функциялар

Python тилиндеги көптөгөн стандарттык функциялар кызматы боюнча топ-торго бөлүнгөн. Ар бир топ **модуль** деп аталган өзүнчө файлга жазылган. Математикалык функциялар **math** модулунда топтолгон.

Бул модульду кошуу үчүн импорт командасы (модульду жүктөө) колдонулат – `import math`.

Төмөндө сандар менен иштөөчү кээ бир функциялар көрсөтүлгөн:

Команда	Аткарган аракети
<code>int(x)</code>	x чыныгы санын калдык бөлүгүн алып салуу менен бүтүн санга айландыруу
<code>round(x)</code>	x чыныгы санын жакынкы бүтүн санга чейин тегеректөө
<code>ceil(x)</code>	жакынкы чоң санга чейин тегеректөө
<code>floor(x)</code>	төмөнкү кичинекей санга чейин тегеректөө
<code>abs(x)</code>	сандын модульн эсептөө (абсолюттук чоңдугун)
<code>sqrt(x)</code>	x санынын квадраттык тамырын чыгаруу
<code>sin(x)</code>	синус x (радиан менен көрсөтүлөт)
<code>cos(x)</code>	косинус x (радиан менен көрсөтүлөт)
<code>tan(x)</code>	тангенс x (радиан менен көрсөтүлөт)

Функцияларга кайрылуу үчүн чекиттик жазуу колдонулат: алгач модульдун атын, чекиттен кийин функциянын атын көрсөтүү керек:

```
print (math.sqrt(x))
```

Python тилиндеги программа

```
x = 25
print (math.sqrt(x))
```

Экранга чыккан жыйынтыгы

5.0

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) Температура Цельсий градуста берилген. Берилген температураны Фаренгейт градуста туюндуруучу программаны түзгүлө. Туюндуруу үчүн формула – $t(F) = 9/5 * t(C) + 32$.
- 2) Туюнтманын маанисин эсептөөчү программаны жазгыла:

$$\frac{(a + b) * c + d * \frac{e}{f}}{k + p * \frac{b}{a} + g} * \frac{4}{5}$$

3.3-тема:

Шарттуу операторлор

Буга чейинки караган мисалдарда операторлор биринин артынан бири удаалаш аткарылган **сызыктуу** программаларды жазууга мүмкүндүк берген. Алардын аткарылышы киргизилген маалыматка көз каранды эмес.

Көпчүлүк реалдуу маселелерде кандай маалыматтар келип түшкөнүнө жараша аракеттердин ирети бир аз өзгөрүшү мүмкүн.

6-класста берилгендей, эгерде аракеттин эки вариантынан тандаш керек болсо, анда алгоритмди жазуу үчүн тармактуу конструкция колдонулат. Python тилинде тармактануу шарттуу операторлор аркылуу ишке ашат. Маанисине карата шарттуу операторлор программаны кайсы бир жол боюнча багыттайт. Мисалы, өрт сигналдык системасынын программасы билдиргичтерден алынган маалыматтар температуранын жогорулашынан же түтүн каптоодон кабар берсе, тынчсыздануу сигналын таратышы керек.

if шарттуу оператору биринчиден шартты текшерет жана андан кийин гана андан аркы нускама боюнча аткаруу же аткарбоо чечимин кабыл алат. **if** оператору кандай иштешин түшүнүү үчүн шартты текшерүү жана тандоого типтүү маселелерден карап көрөлү.

1-маселе. Жашы 21ден ашкандар үчүн гана уруксат бере турган программаны түзөлү.

```
a = int(input('Өзүңүздүн жашыңызды киргизиңиз: '))
if a >= 21:
    print('Уруксат')
else:
    print('Уруксат эмес')
```

if операторундагы шарт кашаага алынбастан жазылат жана кош чекит («:») менен жыйынтыкталат. Шарттын кийинки «бутактары» жаңы саптан оңго жылдыруу менен жазылат.

Эгерде **if** операторунан кийин жазылган шарт туура (чындык) болсо, анда кийинки **else** шарттуу операторуна чейинки бардык командалар аткарылат. Эгерде **if** операторунан кийин жазылган шарт туура эмес (жалган) болсо, анда түз эле **else** операторунан кийинки командалар аткарылат. Биздин мисалда, эгер жашты 13 деп киргизсек, анда **else** шарттуу операторунан кийинки нускама аткарылат: б.а. уруксат берилбейт.

Python тилинде операторлордун сол жактагы чеги боюнча жылышы да чоң мааниге ээ. Көңүл бурсаңар **if** жана **else** сөздөрү бир деңгээлде башталат, ал эми ички блоктогу бардык командалар бул деңгээлге караганда оң жакка бирдей аралыкка жылышкан. Жылдыруу үчүн табуляция символу колдонулат: Tab баскычына бир жолу же төрт жолу бош орун баскычын басуу.

Эгерде бир нече окшош шарттарды киргизүү керек болсо, анда андан кийин нускамалар блогу кеткен кошумча **elif** (**else** – **if**тин кыскартылганы) блогун колдонсо болот.

```
a=int(input('Өзүңүздүн жашыңызды кир-
гизиңиз: '))
if a >= 21:
    print('Уруксат')
elif a >= 18:
    print('Жарым-жартылай уруксат')
else:
    print('Уруксат эмес')
```



ЭСИҢЕ TUT

Python тилинде шарттуу операторлуу нускамалардын баш сөзүнүн аягына сөзсүз кош чекит коюлат.

Салыштыруу операторлору

Салыштыруу операторлору эки маанини бири-бири менен салыштырып жыйынтыгында True же False деген маанисин берет.

Математикалык символ	Python оператору	Мааниси	Мисал	Жыйынтык
<	<	Кичине	1 < 2	True
>	>	Чоң	1 > 2	False
≤	<=	Кичине же барабар	1 <= 2	True
≥	>=	Чоң же барабар	1 >= 2	False
=	==	Барабар	1 == 2	False
≠	!=	Барабар эмес	1 != 2	True

2-маселе. Компания элдин оюн билүү боюнча сурамжылоо жүргүзүп жатат жана аларды 20дан 70 жашка чейинки адамдар кызыктырат. Суралуучунун жашын сураган жана ал ошол белги боюнча «туура келээрин» же «туура келбесин» сураган программа түзгүлө.

▼ өзгөрмөсүнө адамдын жашы жазылсын дейли. Анда программанын керектүү фрагменти төмөнкүдөй болот:

```
if v >= 20 and v <= 70:
    print ('подходит')
else:
    print ('не подходит')
```

Python тилинде кош барабарсыздыктарга уруксат берилет, мисалы:

```
if A < B < C:
```

деген төмөндөгү менен бир мааниде:

```
if A < B and B < C:
```

3-маселе. Шарттуу операторлорду жана салыштыруу операторлорун колдонуп, файлды сактоо программасын жазгыла.

```
ans = input('Сиз файлды сактагыңыз келеби? (ооба/жок)')
if ans == 'ооба':
    print('Сактоо үчүн папканы тандаңыз')
if ans == 'жок':
    print('Маалыматтар өчүрүлөт, андан ары колдоно албайсыз')
else:
    print('Ката. Жооптун мындай варианты жок')
```

Чыгуунун жыйынтыгы кандай жооп тандалгандыгына көз каранды болот:

1-вариант:

Сиз файлды сактагыңыз келеби?
(ооба/жок) ооба
Сактоо үчүн папканы тандаңыз

2-вариант:

Сиз файлды сактагыңыз келеби?
(ооба/жок) жок
Маалыматтар өчүрүлөт, андан ары колдоно албайсыз

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) Үч сан берилди. Алардын ичинен канча сан бирдей экенин аныктоочу программа түзгүлө.
- 2) Үч кесиндинин узундуктары берилди. Бул кесиндилер үч бурчтуктун жактары боло ала тургандыгын текшерүүчү программаны түзгүлө.
- 3) Натуралдык сан берилди. Бул сан жуп экенин, 4кө эселүү экенин аныктоочу программаны түзгүлө.
- 4) Сом, доллар жана евро валюталарын алмаштыруу боюнча программа жазгыла.



ЭСИҢЕ ТУТ

= өзгөрмө үчүн маанини ыйгарат (эгер $a = b$ болсо, анда a b болуп калат);

== эки маанини салыштырат (эгер $a == b$ болсо, анда бул салыштырууга суроо-талап, жана программа True же False деген жыйынтыкты чыгарат).

3.4-тема:

while жана for циклдери

Циклдер шарттуу операторлор сыяктуу эле программалоонун маанилүү бөлүгү болуп саналат. Алардын жардамы менен коддун кээ бир бөлүктөрүн кайталатууну уюштурса болот. Python тилинде циклдерди жазуу үчүн эки түрдөгү командалар колдонулат: **while** жана **for**.

while цикли

«While» англис тилинен «ошондой болгон учурда» деп которулат, башкача айтканда цикл (командалардын блогу) берилген шарт аткарылмайынча кайталана берет. Ал үчүн ар бир циклдин кадамынын башында шартты текшерүү аткарылат. Ошондуктан ал баштапкы шарты бар цикл деп аталат.

1-маселе. 1ден 5ке чейинки бардык бүтүн сандарды экранга чыгаралы.

Python тилиндеги программа

```
d = 0
while d < 5:
    d += 1
    print (d)
```

Экранга чыккан жыйынтыгы

```
1
2
3
4
5
```

Баштапкы шарт мындайча текшерилет: эгерде *d* өзгөрмөсүнүн мааниси баштапкы учурда 5тен чоң же ага барабар болсо, анда цикл бир жолу да кайталанбайт.

2-маселе. Мындай мисалды карайлы: бүтүн оң сандуу ондук ситемада цифралардын санын аныктоо керек. Баштапкы сан бүтүн типтеги *n* өзгөрмөсүнө жазылган деп эсептейли.

Маселени чыгаруу үчүн, мааниси циклдин ар бир өтүшүндө өзгөрүп туруучу **эсептегич** өзгөрмөнү колдонобуз. Цифралардын санын эсептөө үчүн ар бир өтүштө эсептегичти чоңойтуу менен бул цифраларды башынан же аягынан бирден бөлүп алып туруш керек. Эсептегичтин баштапкы мааниси нөлгө барабар, анткени алгоритмди аткарууга чейин бир да цифра табыла элек. Акыркы цифраны бөлүп алууда санды бөлчөксүз 10 санына бөлүп коюу жетиштүү. Сандарды бөлүп алуу жана эсептегичти көбөйтүү амалдарын санда канча цифра болсо ошончо жолу аткаруу зарыл.

Качан гана кийинки 10го бөлүүнүн жыйынтыгында бүтүн бөлүгү нөлгө барабар болгондо, бул цикл аяктады дегенди түшүндүрөт.

Python тилинде программа мындай жазылат:

```
count = 0
while n > 0:
    n = n // 10
    count += 1
```

Циклдин айлануусунун саны киргизилген сандын цифрасына барабар болот, башкача айтканда баштапкы берилишке көз каранды. Эгерде циклдин башындагы шарт бузулбаса, анда цикл чексиз иштей берет. Бул учурда «программа циклден чыкпай калды» деп айтышат. Циклден чыкпай калган программаны токтотуу үчүн Ctrl+C баскычын консоль терезесинде басуу керек.

for цикли

for цикли командаларды керектүү жолу кайталап, программаны кыскартууга мүмкүндүк берет. Жогорку мисалда **for** циклин колдонолу:

Python тилиндеги программа

```
for i in range (5):
    print (i)
```

Экранга чыккан жыйынтыгы

```
0
1
2
3
4
```

Бул жерде *i* өзгөрмөсү (муну циклдин өзгөрмөсү деп аташат) Одөн 5ке чейинки, 5 өзү кирбейт (б.а. Одөн 4кө чейин) диапазондо (**in range**) өзгөрөт. Ошентип цикл туптуура 5 жолу кайталанат.

while менен жазылган программага окшош жооп алуу үчүн **for** циклин колдонуп программаны өзгөртөлү:

Python тилиндеги программа

```
d = 0
while d < 5 :
    d+=1
    print ( d )
```

```
for i in range (1,6):
    print ( i )
```

Экранга чыккан жыйынтыгы

```
1
2
3
4
5
```

2-маселе. Эки санынын 2^1 нен 2^{10} уна чейин даражаларын чыгарабыз (k = экинин даражалары).

Тең күчтүү туюнтмалардын жазуулары

```
k = 1
while k <= 10 :
    print ( 2**k )
    k += 1
```

```
for k in range (1,11):
    print ( 2**k )
```

Биринчи вариантта **k** өзгөрмөсү үч жолу колдонулат: баштапкы маанисин ыйгарууда, циклдин шартында жана циклдин тулкусунда (1ге чоңойтуу).

Экинчи вариантта **k** өзгөрмөсү баштапкы жана акыркы маанилериндеги эки сандын диапазонунда берилет, мында акыркы маани диапазонго **кирбейт**.

Циклдик өзгөрмөнүн өзгөрүү **кадамы** берилбесе 1ге барабар болот. Эгерде аны өзгөртүү керек болсо **range** сөзүнөн кийин кашаанын ичинде үчүнчү (кошумча) санды киргизишет – бул керектүү **кадам**. Мисалы, мындай цикл 2 санынын так даражаларын гана чыгарат:

Python тилиндеги программа

```
for k in range(1,11,2):
    print ( 2**k )
```

Экранга чыккан жыйынтыгы

2
8
32
128
512

Циклдин ар бир кадамы менен циклдин өзгөрмөсү өсүүдө эле болбостон, кемүүдө да болушу мүмкүн. Ал үчүн баштапкы мааниси акыркы маанисинен чоң, ал эми кадам – терс болуш керек. Төмөнкү программа 5тен 1ге чейинки натуралдык сандардын квадраттарын кемүү тартибинде чыгарат:

Python тилиндеги программа

```
for k in range(5,0,-1):
    print ( k**2 )
```

Экранга чыккан жыйынтыгы

25
16
9
4
1

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) А жана В бүтүн сандарын алган жана Адан Вга чейинки диапазондогу бардык натуралдык сандардын квадраттарын чыгара турган программаны түзгүлө.
- 2) Натуралдык сан берилген. Ал сандын цифраларынын суммаларын чыгарып берген программаны жазгыла.
- 3) Узундугу 20 метр болгон тактай берилген. Бул тактайдан узундуктары 1,5 м жана 2 м болгон бүтүн сандагы канча минималдык кесиндиин даярдоого боло турганын эсептөөчү программаны түзгүлө.

4

- бөлүм



Компьютердик тармактар жана интернет

4.1-тема:

Татаал издөө суроо-талаптары

Издөө системаларынан так маалыматты табуу үчүн атайын операторлор колдонулат. Мисалы, Беляевдин авторлугундагы жаныбарлар жана канаттуулар жөнүндөгү китептерди табалы.

- 1** **жаныбарлар & канаттуулар & Беляев же +жаныбарлар +канаттуулар +Беляев.**

Изделүүчү документте бардык берилген сөздөр камтылат. Ал сөздөр документте катар же ар кайсы бөлүгүндө жайгашуусу мүмкүн.

- 2** **«Кыргызстандагы жаныбарлар жана канаттуулар»**

Силер издөөнү кандай берсеңер так ошондой сөз тартибин камтыган веб барактар көрсөтүлөт. Мындай талап так цитатаны, ырдын же фильмдин аталышын табуу керек болгондо колдонулат.

- 3** **Жаныбарлар жана канаттуулар –Африка**

Суроо-талаптагы сөздүн алдына «минус»(-) белгиси коюлса, анда издөө жыйынтыкта ошол сөз кездешпеген веб барактар берилет.

- 4** **Жаныбарлар|канаттуулар же жаныбарлар OR канаттуулар**

Эки суроо-талаптын ортосундагы OR оператору ушул сөздөрдүн жок дегенде бирөөсү кезиккен барактарды табууга мүмкүндүк берет.

- 5** **Жаныбарлар жана канаттуулар site:www.kyrgyzstantravel.net**

Берилген домендин же сайттын чегинде гана издөөнү ишке ашырат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Качан издөөдө «+» белгисин, качан «-» белгисин киргизүү керек?
- 2) Үй мышыктарын багуу боюнча маалыматты издөөгө татаал суроо-талапты түзгүлө. Издөөдөн чоң мышыктарды (мисалы арстан, жолборс) жана сатуу, сатып алуу жөнүндөгү сунуштарды алып салгыла.
- 3) «ЖЕ» логикалык амалды белгилөө үчүн суроо-талапта «|» белгиси, ал эми логикалык «ЖАНА» амалы үчүн «&» белгилери колдонулат. Алынган суроо-талаптарды издөө сервери таап чыккан барактардын саны боюнча өсүү тартибинде жайгаштыргыла.

Код**Суроо-талап****А**

Конан Дойль & Г. Бичер-Стоу & Джером К. Джером

Б

Конан Дойль | (Г. Бичер-Стоу & Джером К. Джером)

4.2-тема:

Сайт конструкторлору

Дүйнөдөгү биринчи сайт пайда болгондон бери аларды түзүү технологиялары өтө өзгөрдү. Эгерде мурда ар бир сайт «нөлдөн» баштап жазылса, азыркы учурда интернеттин өнүгүшү жана веб-сайттардын өтө тездикте көбөйүшү менен аларды жаратууга атайын конструкторлор иштелип чыккан. Алардын негизинде HTML тилин билбестен эле, ал эмес баштапкы деңгээлдеги колдонуучу да өзүнүн сайты жарата алат.

Бул конструкторлор сайттын мазмунун башкаруу системалары же «кыймылдаткытар» (content management system – CMS) деп аталышат. Алардын жардамы менен сайттын барактарын түзүүгө жана калыпка келтирүүгө, сайттын мазмунун редакциялоого: документтер, сүрөт, видеофайл ж.б. менен толтурууга болот.

Ар түрдүү CMSтерде каалагандай ойлор үчүн көпчүлүк учурларда даяр ар кандай дүкөн-сайттардын, визитка-сайттардын, лэндингдердин (кайсы бир аракетти жасоого чакырган сайттар) шаблондору бар. Бирок эгерде сайт өтө эле өзгөчө болсо анда албетте аны нөлдөн баштап жазуу сунушталат.

Статистика боюнча азыркы сайттардын 99%ы ар түрдүү CMSтерди колдонуу менен түзүлгөн. Алардын кээ бирлери дүйнөдө өтө популярдуу: Wordpress, Drupal, Joomla ж.б.



БУЛ КЫЗЫКТУУ!

Дүйнөдөгү биринчи веб-сайт **info.cern.ch** 1991-жылдын 6-августунда британиялык окумуштуу-ойлоп табуучу Тим Бернерс-Ли тарабынан түзүлгөн. Бул өтө жөнөкөй, эч кандай графиканы камтыбаган жана World Wide Web технология эмне үчүн керектигин түшүндүргөн тексттик веб-сайт болгон.



АНЫКТАМА

CMS – мазмунун (контентин) башкаруу системалары (англ. content management system, CMS) – бул сайттын мазмунун чогуу түзүү, редакциялоо жана башкаруу процессин камсыз кылууга жана уюштурууга колдонулган программалык камсыздоо.

Лэндинг (англ. landing page) – анын негизги максаты сайтты колдонуучуну максаттуу аракетти жасатууга арналган веб-барак. Мисалы, бир нерсе үчүн добуш берүү, кайсы бир товар менен таанышуу жана аны сатып алуу, китепке заказ берүү ж.б.

Wix – сайт түзүү үчүн платформа

Анча чоң эмес визитка-сайтын, лэндинг же блогду тез арада түзүү үчүн эң популярдуу платформалардын бири болуп Wix сервиси эсептелет. Wixте сайттын даяр көптөгөн кооз шаблондору бар. Ал жерден конструктордо-гудай эле силер барактарды кошуп же ашыкчасын алып салсаңар, стилин, тексттерди, фондук сүрөттөрдү, баскычтарды ж.б. өзгөртсөңөр болот.

Веб-барактарды түзүүдөн мурда силер сайттын контенти жана структу-расын аныктап алышыңар керек жана ошого жараша сайттын шаблонун тандайсыңар.

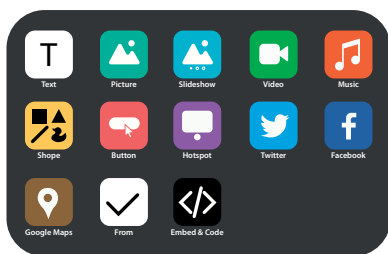
The image shows the Wix website builder interface with several callouts explaining its features:

- Бул баскыч сайттын менюсун тууралайт, анын жардамы менен сайттын барактарына гипершилте-мелерди кошсо болот.** (This button configures the site's menu, with its help you can add hyperlinks to the site's pages.)
- Силер жүктөгөн файлдар-дын баарын көрсөтөт.** (It shows all the files you have uploaded.)
- Сиздин сайтыңызга байланыш-кан блогду алып барууга мүмкүн-чүлүк берет.** (It gives you the opportunity to start a blog related to your site.)
- Муну менен адамдарды жазып жана аларга эскертүү-лөрдү жөнөтсө болот.** (With this, you can write people and send them notifications.)
- «Сайт» баскычынын жардамы менен киргизилген өзгөртүүлөрдү сактоо-го, болжолдуу версия-сын көрүүгө же сайтты жарыялоого болот.** (With the help of the 'Site' button, you can save changes, view a preview, or publish the site.)
- Барактын фонун жасалгалоого мүмкүндүк берет. Фон бир түстүү болушу мүмкүн, андан тышкары фон катары Wix галереясынан же силердин компютериңерден алынган фото же видео болушу да мүмкүн.** (It allows you to design the page background. The background can be a single color, or you can choose from the Wix gallery or your own photos/videos.)
- Анын жардамы менен баракка буларды койсо болот: гипершил-теме-баскыч («Андан ары окуу», «Кирүү», «Ок»), видео жана фотолорду, Wixтин стандарттык жыйнагынан графикалык сүрөт-төрдү ж.б.** (With its help, you can add to the page: hyperlinks ('Read more', 'Enter', 'OK'), videos and photos, standard Wix graphics, etc.)
- Wixтин кошумча тиркемеле-рин колдонууга мүмкүндүк берет. Ыңгайлуу иштөө үчүн тиркемелерди «Акысыз» категориясы менен сорттоп алгыла. Мында html-кодду (мисалы башка сайттан викторинанын коду), сайтка кирүүчүлөр эсептегичин же социалдык тармактарга шилтемени койсо болот.** (Wix provides additional plugins. For convenient work, sort them by 'Free' category. Here you can find HTML code (for example, a quiz from another site), a site visitor counter, or social media links.)
- Сайттын доменин өзгөртүүгө же начар көргөндөр үчүн версиясын редакциялоого мүмкүндүк берет.** (It allows you to change the domain or edit a previous version for a better look.)
- «Инструменты» баскычында сайттын өлчөмдөрү туурала-нуучу сызгычтар тескелет.** (In the 'Tools' button, you can find guides for adjusting the site's dimensions.)
- Андан тышкары «Сайт на мобиль-ном» баскычы менен сайттын мобилдик версиясын жасалга-ласа болот.** (Additionally, with the 'Site on mobile' button, you can customize the mobile version of the site.)

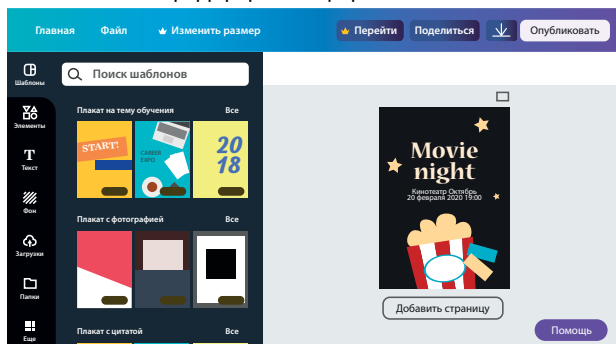
Интернетте Wix платформасын тереңдетип окутуучу көп материалдар бар. Эсиңерде болсун, бул платформанын жогорку деңгээлдеги кээ бир тейлөөлөрү акылуу болушу мүмкүн.

Сайттын мазмунун ар түрдүү визуалдык контенттерди, мисалы, лонгриддерди жана кооз презентацияларды кошуу менен да кызыктуураак кылсаңар болот. Аларды түзүү үчүн интернетте көптөгөн түрдүү платформалар бар.

Readymag деген конструктор лонгриддерди жасоо үчүн эң ыңгайлуу онлайн-платформалардын бири болуп саналат. Анын жардамы менен макалага текстти, сүрөттү, слайдшоуну, видео-файлдарды жана музыканы оңой эле кошсо болот.



readymag.com – лонгриддерди түзүү үчүн платформа



canva.com – графика жана дизайн үчүн сервис.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Графика жана дизайн үчүн платформаны колдонуп мектебиңердеги Бүтүрүү балынын афишасын түзгүлө.

АНЫКТАМА

Лонгрид (англ. longread) – ар кандай мультимедиялык элементтердин (фотография, сүрөттөр, видео, диаграмма ж.б.) жардамында блокторго бөлүнгөн узун текст (макала).

Контент (англ. мазмун) – бул колдонуучуга арналган маалымат жана тажрыйба. Сайттын контенти – бул анын ичинде камтылгандын бардыгы: текст, сүрөт, видео ж.б.

Башка онлайн-сервис www.canva.com силерге тез арада постерлерди, сайт үчүн баннерлерди, презентацияларды, соцтармактар үчүн сүрөттөрдү түзүүгө жардам берет. Текст, сүрөт жана фото менен болгон бардык аракеттер түздөн түз сайтта жүргүзүлөт.

4.3-тема:

Электрондук почта жана булуттук сервистер

Электрондук почта – интернетте эң көп колдонулган сервистердин бири. Электрондук почта менен иштөө үчүн почта тиркемелерин, мисалы, Outlook Express же жөн эле браузерди колдонсо болот.

Бүгүнкү күндө эң популярдуу почта сервиси болуп Gmail эсептелет. Gmailге катталуу (Google аккаунтту түзүү) Google аспаптарын бекер жана мыйзамдуу түрдө колдонууга мүмкүндүк берет. Алар: электрондук почта, Google диск, социалдык тармак Google+, Play Market, YouTube сайттары ж.б. Аккаунт түзүү үчүн www.gmail.com сайтына киргиле жана катталгыла.

Катталууда анкетада өзүңөрдүн чыныгы аты-жөнүңөрдү жана туулган датаңарды көрсөтүү керек. Анткени бурмаланган маалыматтар Googleдун кээ бир сервистерине кирүүгө мүмкүнчүлүктү чектеп коюшу мүмкүн. Электрондук почтанын дареги белгилүү формада жазылат жана «@» символу менен бөлүнгөн эки бөлүктөн турат:

Дарек ээсинин_аты@сервердин_доменидик_аты

Дарек ээсинин_аты – бул колдонуучунун аты (логин).

Тамгалардын ар кандай регистрин, символдорду жана цифраларды колдонуп, ишенимдүү паролду пайдалануу керек.

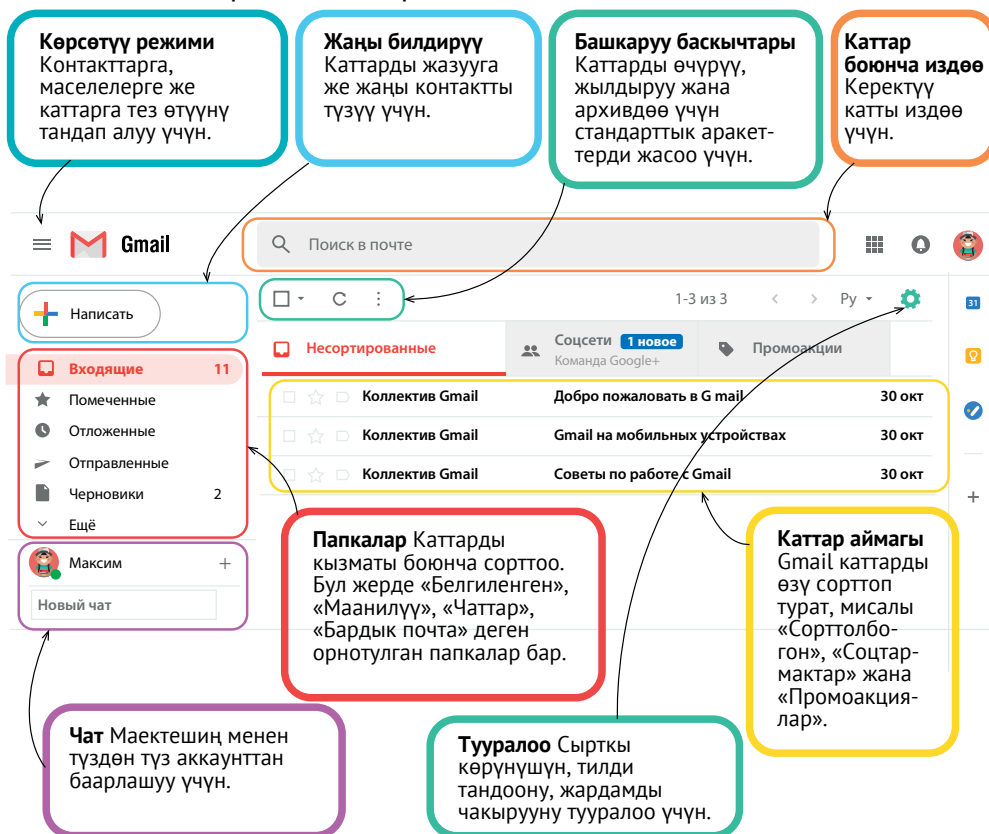
Google тиркемелери

Google Disk – маалыматты интернетте виртуалдык дискте сактоо үчүн сервис. **Google дисктин** курамына онлайн тексттик документтерди, электрондук таблицаларды, презентацияларды ж.б. түзүүгө жана редактирлөөгө мүмкүндүк берүүчү тиркемелер да кирет. Документтин ээси башка колдонуучулар менен бирдиктүү иштөө максатында аны башкаларга жеткиликтүү кылып уруксат берсе да болот ж.б.

Google Calendar – жолугушууларды пландоо үчүн ыңгайлуу сервис. Бул жерде жолугушуунун убактысын белгилеп, электрондук почта аркылуу башка катышуучуларга чакыруу жөнөтсө болот.

Google Translate – бир тилден башка тилге сөздөрдү, сүйлөмдөрдү жана веб-барактарды которууга мүмкүндүк берет. Бардыгы болуп тилдердин багыттары 1000ге жакын, мисалы, кыргыз тилинен орус, япон, серб, хинди тилдерине же тескерисинче.

Gmail почталык сервисинин терезеси:



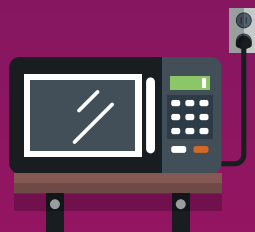
Google Photo – санариптик фотосүрөттөрдөн коллаж жана слайд-шоуларды түзүүгө жана иштетүү үчүн атайын сервис.

Google Maps – дүйнө жүзүнүн спутниктен тартылган сүрөттөрү жана карталары (Ай менен Марсты кошуп), маршруттарды издөөсү менен автомобиль жолдорунун картасы менен иштөөгө мүмкүндүк берет.

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Google календарда мектеп каникулдары жана майрамдары боюнча эскертүүнү даярдагыла.
- 2) Google Maps аркылуу Кыргызстандын эң чоң көлдөрүн таап, алардын рельефтик сүрөттөлүштөрүн алгыла.
- 3) Google Maps аркылуу Эйфель мунарасынан Париждеги Майыптар үйүнө чейинки маршрутту тапкыла жана белгиленген жерге жөө жүрсө, автомобиль менен же велосипед менен барса, канча убакыт кетээрин аныктагыла.





 —класс



1

- бөлүм



Информатика жана маалымат

1.1-тема:

Логикалык туюнтмалар жана амалдар

Күнүмдүк жашообузда «логикалуу», «логика» деген сөздөрдү биз көп колдонобуз. Байыркы грек тилиен которгондо «логика» – «ой жүгүртүү жөнүндөгү илим» деп которулат. Бул туура ойлонуу, ырастоолорду далилдөө жана чечим чыгаруу жөнүндөгү илим.

Логика чындыкка жетүү ыкмаларын сезимдерге эмес, мурда алынган билимдерге таянуунун натыйжасында изилдейт. Логика математиканын бир бөлүмү жана көптөгөн илимдерде колдонулат, ал эми бул алгебрасы информатиканын негиздеринин бири болуп эсептелет.



БУЛ КЫЗЫКТУУ

1847-жылы англиялык математик Джордж Буль логикалык айтымдарды изилдөө үчүн татаал сөз түрүндөгү айтымдарды кыскача символдор менен белгилеген математикалык ыкманы колдонууну сунуштаган. Кийин математиканын бул бөлүмү логика алгебрасы же «буль алгебрасы» деген атты алган. Анын жардамында логикалык айтымдарды жазышат, эсептешет, жөнөкөйлөтүшөт жана өзгөртүп түзүшөт. Буль алгебрасы өтө жөнөкөй, анткени ар бир өзгөрмө «чындык» же «жалган» деген эки маанини эле алышы мүмкүн.

Логика алгебрасы айтымдардын мазмунун териштирбестен, алардын чындык же жалган экенине гана карайт. Ал алгебралык эсептөөлөр жолу менен курама (татаал) айтымдардын чындык же жалган экенин аныктоого мүмкүндүк берет. Логикалык айтымдарды латын алфавитинин баш тамгалары менен белгилешет:

$A = \{ \text{Жер тоголок} \},$
 $B = \{ \text{Күн — бул планета} \}.$



АНЫКТАМА

Жөнөкөй логикалык айтым – бул маанисин жоготпостон туруп кичирейтүүгө же бөлүүгө мүмкүн болбогон айтым.

Татаал айтым логикалык амалдарга (операцияларга) дал келген логикалык байламталар менен бириккен жөнөкөй айтымдардан турат.

Логикалык туюнтма – бул логикалык амалдар (операциялар) менен бириккен логикалык айтымдар.

Негизги логикалык амалдар (операциялар)

✓ Дизъюнкция (кошуу) – бул эгерде жөнөкөй логикалык айтымдардын бирөөсү гана чын болсо, анда чындык, жана экөөсү тең жалган болгондо гана жалган маанисин алган татаал логикалык туюнтма. Логикалык кошуу эки (же андан көп) айтымдын «же» байламтасы менен биригишинен түзүлөт.

Мисалы: *Апамдын маанайын көтөрүү үчүн идиш-аякты жууп коюш керек же ага чай сунуштоо керек.*

∧ Конъюнкция (көбөйтүү) – бул жөнөкөй логикалык айтымдардын экөөсү тең чындык болгондо гана чындык маанисин алган, ал эми калган бардык учурларда жалган маанисин алган татаал логикалык туюнтма. Логикалык көбөйтүү эки (же андан көп) айтымдын «жана» байламтасы менен бирөөгө биригишинен түзүлөт.

Мисалы: *Максаттарга жетүү үчүн билим жана тырышчаактык керек.*

→ Импликация (ээрчүү) – бул биринчи айтымы шарт, ал эми экинчиси натыйжасы болгон татаал логикалык туюнтма. Мындай логикалык туюнтма, чындыктан жалган келип чыгат дегенден башка бардык учурда чындык маанисин алат. Логикалык ээрчүү «эгерде...анда...» деген сөз айкашынын жардамында эки айтымдын бирөөгө биригишинен түзүлөт.

Мисалы: *Эгерде сан 6га бөлүнсө, анда ал 3кө да бөлүнөт.*

¬ Инверсия (тануу) – бул эгерде баштапкы логикалык айтым чындык болсо, тануунун жыйынтыгы жалган жана тескерисинче баштапкы логикалык айтым жалган болсо, тануунун жыйынтыгы чындык болгон татаал логикалык туюнтма.

Логикалык тануу баштапкы логикалык айтымга «эмес», же «мындай туура эмес» сөз байламталарын кошуу жолу менен түзүлөт.

Мисалы: *«Кыргызстандын борбору – Бишкек».*

Инверсиядан кийин: «Кыргызстандын борбору – Бишкек ЭМЕС».

≡ Эквиваленттүүлүк (барабардык) – бул качан жөнөкөй логикалык айтымдардын экөө тең бирдей чындык маанисин алганда гана чындык болгон татаал логикалык туюнтма. Логикалык барабардык «качан» жана «качан гана» сөз айкаштарынын жардамында эки айтымдын бирөөгө биригүүсүнөн түзүлөт.

Мисалы: *Жеңиш чокусу – Кыргызстандагы эң бийик тоо чокусу.*



Логикалык амалдар үчүн чындык таблицасы

Аракеттердин аты	Кошуу	Көбөйтүү	Тануу	Ээрчүү	Барабардык
	Дизъюнкция	Конъюнкция	Инверсия	Импликация	Эквиваленттүүлүк
Белгилениши	\vee	\wedge	\neg	\rightarrow	\equiv
Чындык таблицасы	$0 \vee 0 = 0$ $0 \vee 1 = 1$ $1 \vee 0 = 1$ $1 \vee 1 = 1$	$0 \wedge 0 = 0$ $0 \wedge 1 = 0$ $1 \wedge 0 = 0$ $1 \wedge 1 = 1$	$\neg 0 = 1$ $\neg 1 = 0$	$0 \rightarrow 0 = 1$ $0 \rightarrow 1 = 1$ $1 \rightarrow 0 = 0$ $1 \rightarrow 1 = 1$	$0 \equiv 0 = 0$ $0 \equiv 1 = 0$ $1 \equiv 0 = 0$ $1 \equiv 1 = 1$
	же	жана	эмес	эгерде ... анда	качан гана

(Белгилейбиз: 0 – бул жалган айтым, 1 – чындык)

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) Формула менен берилген татаал туюнтмаларды кадимки тилде жазып бергиле. A = «Тимурга тарых жагат», B = «Зояга информатика жагат», C = «Рита тарых боюнча видеоролик жасай алат».

а) $A \wedge B$; б) $B > C$; в) $A \wedge C$; г) $(\neg A) > (\neg C)$; д) $A \vee B \vee C$; е) $(A \wedge B) > C$

2) Берилген айтым чындык болгон X бүтүн санын атагыла: $X > 6$ ЭМЕС ЖАНА $X > 4$

3) Таблицада издөө серверине болгон суроо-талаптар берилген. Суроо-талапта «ЖЕ» логикалык амалын белгилөө үчүн «|» символу, ал эми «ЖАНА» логикалык амалын «&» менен белгилөө колдонулат. Ар бир суроо-талап үчүн A дан G га чейинки тамгаларга туура келген анын коддору көрсөтүлгөн. Суроо-талаптардын коддорун солдон оңго карай табылган беттердин сандарынын өсүү тартиби боюнча жайгаштыргыла. Мында табылган беттердин сандары ар башка.

Код	Суроо-талап
А	Эльфтер Гномдор Хоббиттер Орктор
Б	(Эльфтер & Гномдор) Орктор
В	Эльфтер & Гномдор
Г	Эльфтер Хоббиттер

1.2-тема:

Логиканын мыйзамдары

Татаал логикалык туюнтманын маанисин табууда логиканын мыйзамдарын билүү керек, алар арифметиканын мыйзамдарына окшош.

МЫЙЗАМ

БАЯНДООСУ (ФОРМУЛИРОВКАСЫ)

1. Теңдештик (окшоштук) мыйзамы $A = A$

Ар кандай айтым өзүнө барабар.

2. Орун алмаштыруу (коммутативдик) мыйзамы

$$A \wedge B = B \wedge A$$

$$A \vee B = B \vee A$$

Айтымдардын үстүнөн жүргүзүлгөн аракеттердин жыйынтыгы, бул айтымдардын кандай ирээтте алынгандыгына көз каранды эмес. (Кошулуучулардын ордун алмаштыруудан сумма өзгөрбөйт. Көбөйтүүчүлөрдүн ордун алмаштыруудан көбөйтүндү өзгөрбөйт).

3. Топтоштуруу (ассоциативдик) мыйзамы

$$(A \vee B) \vee Z = A \vee (B \vee Z)$$

$$(A \wedge B) \wedge Z = A \wedge (B \wedge Z)$$

Бирдей белгилерде кашааларды каалагандай жайгаштырса, же таптакыр алып салса болот.

4. Бөлүштүрүү (дистрибутивдик) мыйзамы

$$(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$$

$$(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$$

Жалпы айтымды кашаанын сыртына чыгаруу эрежесин аныктайт.

Кошумча мыйзамдар:

Тең күчтүүлүк мыйзамы

$$A \vee A = A$$

$$A \wedge A = A$$

Константаларды чыгарып салуу мыйзамы

$$A \vee 1 = 1, A \vee 0 = A$$

$$A \wedge 1 = A, A \wedge 0 = 0$$

Жутуу (сиңирүү) мыйзамы

$$A \vee (A \wedge B) = A$$

$$A \wedge (A \vee B) = A$$

Жабыштыруу мыйзамы

$$(A \wedge B) \vee (\neg A \wedge B) = B$$

$$(A \vee B) \wedge (\neg A \vee B) = B$$

Жалпы инверсия мыйзамы
Де Моргандын мыйзамы

$$\neg(X \vee Y) = \neg X \wedge \neg Y$$

$$\neg(X \wedge Y) = \neg X \vee \neg Y$$

Эки жолку тануу мыйзамы

$$(\neg X) = X$$

Карама-каршылыксыздык мыйзамы

$$X \wedge \neg X = 0$$

Үчүнчүсүн алып салуу мыйзамы

$$X \vee \neg X = 1$$



Логикалык маселелерди чыгаруу логикалык айтымдарды формалдаштыруу жолу менен жүргүзүлөт.



АНЫКТАМА

Формалдаштыруу – бул берилген айтымды символдордун жардамы менен формалдуу жазуу.

Маселе:

Ажара, **Диана**, **Ира** жана **Тамара** ар түрдүү чет тилдерин окушат: англис, француз, кытай жана япон тилдери.

Андрей кыздардан ар бири кайсы тилдерди окушаарын сураганда,

Тамара мындай жооп берди: «Мен япон тилин».

Ал эми калган кыздар: «Бул сыр, эми өзүң таап көр, эгерде:

Ажара кытай тилин окуса,

Диана кытай тилин окубайт,

Ира болсо француз тилин окубайт.

Ошондой эле бул жерде айтылган үч ырастоонун бирөөсү чындык, ал эми экөө – жалган».

Андрейге туура жообун табуу үчүн ой талкуу ыкмасын колдонуп жардам берели.

3 ырастоо берилип жатат:

1-ырастоо. **Ажара** кытай тилин окуйт,

2-ырастоо. **Диана** кытай тилин окубайт,

3-ырастоо. **Ира** француз тилин окубайт.

Кыскартып жазалы:

A=K

D≠K

I≠Φ

Шарт боюнча – булардын ичинен бирөө гана туура.

1. Биринчи ырастоону туура деп эсептеп көрөлү:

*«**Ажара** кытай тилин окуйт»*

Анда, маселенин шарты боюнча башка эки ырастоо жалган болуп чыгыш керек. Мында эгер **Диана кытай тилин окубайт** болсо, демек ал **кытай тилин окуйт экен да**.

A = K=1

D ≠ K=0

I ≠ Φ=0

Демек **Ажара** да жана **Диана** да кытай тилин окуп калышат да. Ал эми баштапкы шартта кыздар ар түрдүү тилдерди окушат деп жазылган, натыйжада бул божомол туура эмес жана биринчи ырастоо чындык болушу мүмкүн эмес.

A = K = 1

D ≠ K = 0

Демек D=K

I ≠ Φ = 0

2. Эми экинчи ырастоо чындык деп эсептейли:

«**Диана** кытай тилин окубайт».

Анда маселенин шарты боюнча биринчи жана үчүнчү ырастоо да туура эмес. Мындан эч кимиси кытай тилин окубайт деген жыйынтык келип чыгат. Демек мында да биздин божомол туура болбой чыкты.

$$\begin{aligned} A &= K=0 \\ D &\neq K=1 \\ I &\neq \Phi=0 \end{aligned}$$

3. Маселенин шарты боюнча үч ырастоонун бирөө гана

туура болушу керек. Биринчи караган эки ырастоо жалган болгондуктан демек үчүнчү: «**Ира** француз тилин окубайт» деген ырастоо чындык, ал эми биринчи экөө туура эмес экен да.

$$\begin{aligned} A &= K=0 \\ D &\neq K=0 \\ I &\neq \Phi=1 \end{aligned}$$

Демек:

$A \neq K=1$ **Ажара** кытай тилин окубайт (француз же кытай тили калат экен),
 $D = K=1$ **Диана** кытай тилин окуйт.

Демек биз,

Диана кытай тилин окуйт тургандыгын аныктадык.

Анда **Ира** менен **Ажара** кайсы тилдерди окушат?

«**Ира** француз тилин окубайт» болсо демек, ал англис тилин окуйт да.

Анда француз тилин **Ажара** окуйт экен да.

Жооп:

Тамара япон тилин окуйт, **Диана** кытай, **Ира** англис, ал эми **Ажара** француз тилин окушат.



СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

Индира, Эмил жана Лена байыркы идишти таап алышты. Бул идиш жөнүндө алардын ар бири экиден божомолдорун айтышты:

Индира: Бул скифтерден калган идиш жана ал V кылымдарда жасалган.

Эмил: Бул идиш согдийлердики жана III кылымдарда пайда болгон.

Лена: Бул идиш скифтердики эмес жана IV кылымдарга таандык.

Тарых мугалими балдарга, ар бириңер эки божомолдун бирөөндө гана туура айтып жатасыңар деди.

Бул идиш качан жана кайда жасалган?



1.3-тема:

Логикалык туюнтмаларды чыгаруу

Маселелердин негизинде логикалык туюнтмаларды түзүүнү карайлы:

Элестеткиле, силерге суу астындагы кеменин сигнал берүү системасын программалоо маселеси берилди. Эгерде үч кыймылдаткычтын экөө иштен чыгып калса, авариялык сигналды бергендей кылуу керек.

A – «Биринчи кыймылдаткыч иштен чыкты».

B – «Экинчи кыймылдаткыч иштен чыкты».

C – «Үчүнчү кыймылдаткыч иштен чыкты».

X – «Авариялык кырдаал».

Сигнал берүү системасы «X» кырдаалында иштейт.

«X» абалын формула түрүндө көрсөтсөк болот:

$$X = (A \text{ жана } B) \text{ же } (A \text{ жана } C) \text{ же } (B \text{ жана } C)$$

Биз формалдаштырууну аткардык.

Логикалык аракеттердин символдорунун жардамы менен бул туюнтма мындай жазылат:

$X = (A \vee B) \wedge (A \vee C) \wedge (B \vee C)$ Мындай сигнал компьютерге түшүнүктүү болот.

Логикалык туюнтмаларды айтымдар алгебрасынын жардамында чыгаруу

Логикалык аракеттерди аткаруу ирээти:

1 Кашааларда жайгашкан туюнтма.

2 Тануу.

3 Логикалык көбөйтүү.

4 Логикалык кошуу.

5 Логикалык ээрчүү.

1-маселе. Татаал логикалык туюнтманы чыгаруу:

Жөнөкөй туюнтмалар берилген: $A=\{3=6\}$, $B=\{2<3\}$, $C=\{4<1\}$.

Курама туюнтманын чындык экенин аныктагыла: $(\neg A \vee A \wedge B) \wedge \neg C$

Чыгаруу алгоритми:

1) Жөнөкөй туюнтмалардын чындык экенин аныктайбыз:

$A=\{3=6\}$ жалган (0)

$B=\{2<3\}$ чындык (1)

$C=\{4<1\}$ жалган (0)

2) Курама туюнтмага маанилерин коёбуз:

$(\neg A \vee A \wedge B) \wedge \neg C = (\neg 0 \vee 0 \wedge 1) \wedge \neg 0$

3) Тануу аракетин эске алуу менен кашаанын ичинде жайгашкан туюнтманын маанилерин аныктайбыз:

$(\neg 0 \vee 0 \wedge 1) = 1 \vee 0 \wedge 1$

4) Логикалык көбөйтүүнү аткарабыз: $0 \wedge 1 = 0$

5) Логикалык кошууну аткарабыз. $1 \vee 0 = 1$

6) Ортодогу жыйынтыкты алабыз:

$(\neg 0 \vee 0 \wedge 1) = 1$

7) Баштапкы туюнтманын маанисин аныктайбыз:

$1 \wedge \neg 0 = 1 \wedge 1 = 1$ (чындык)

Жооп: $(\neg A \vee A \wedge B) \wedge \neg C$ курама туюнтмасы чындык.

2-маселе. Берилген сандардын кайсынысы үчүн төмөндөгү айтым чындык:
(сан<100) ЭМЕС ЖАНА (жуп сан) ЭМЕС? 115, 108, 35, 8.

Чыгаруу алгоритми:

Туюнтманы (сан >=100) ЖАНА (жуп сан) түрүндө жазалы.

Текшеребиз:

1) Чындык, анткени эки айтым тең чындык: 115 саны 100дөн кичине эмес жана жуп сан эмес. Бул туура жооп.

2) Жалган, анткени экинчи айтым жалган: 108 – жуп сан эмес.

3) Жалган, анткени биринчи айтым жалган: 35 саны 100дөн кичине эмес.

4) Жалган, анткени экинчи айтым жалган: 8 – жуп сан эмес.





Чындык (аныктык) таблицасын түзүү

Логикалык туюнтманын чын экенин аныктоо үчүн чындык таблицасын колдонууга болот.

Чындык таблицасы татаал айтымга кирген жөнөкөй айтымдардын бардык маанилеринде кандай маанини ала тургандыгын көрсөтөт.

Чындык таблицасынын жардамында логикалык туюнтмаларды чыгаруу алгоритми:

- 1 логикалык туюнтмадагы n өзгөрмөлөрүнүн санын эсептеп чыгуу;
- 2 $m=2n$ формуласы боюнча таблицадагы саптардын санын эсептөө, мында n өзгөрмөлөрдүн саны;
- 3 формуладагы логикалык аракеттердин санын эсептөө;
- 4 приоритетин жана кашааларды эсепке алуу менен логикалык аракеттердин удаалаштыгын аныктоо;
- 5 мамычалардын санын аныктоо: өзгөрмөлөрдүн саны + аракеттердин саны;
- 6 киргизилүүчү өзгөрмөлөрдүн топтомун жазып алуу;
- 7 4-пунктта көрсөтүлгөн удаалаштыкта логикалык амалдарды аткаруу менен чындык таблицасын мамычалар боюнча толтуруу.

Мисал: чындык таблицасын жардамы менен $\neg A \vee B$ туюнтмасынын чындык экенин аныктагыла.

- 1 Логикалык туюнтмадагы өзгөрмөлөрдүн санын аныктайбыз: 2.
- 2 $m=2^n$: формуласы менен таблицадагы саптардын санын аныктайбыз: $2^2=4$.
- 3 Формуладагы логикалык аракеттердин санын аныктайбыз: 2.
- 4 Логикалык аракеттердин аткарылуусунун удаалаштыгын орнотобуз: биринчи – логикалык тануу, экинчиси логикалык көбөйтүү.
- 5 Таблицадагы мамычалардын санын аныктайбыз: өзгөрмөлөрдүн саны + амалдардын саны $= 2 + 2 = 4$.

- 6** Киргизилген өзгөрмөлөрдүн топтомун жазып алабыз.
- 7** А жана В өзгөрмөлөрүнүн маанилеринин мүмкүн болгон бардык варианттарын толтуруу менен чындык таблицасын мамыча боюнча толтурууну жүргүзөбүз.
- 8** 4-пунктта көрсөтүлгөн удаалаштыкка жараша логикалык амалдарды аткарабыз.

$$\neg A \vee B$$

A	B	$\neg A$	$\neg A \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

СУРООЛОР ЖАНА ТАПШЫРМАЛАР

- Логикалык туюнтманы логика алгебрасы тилинде жазгыла:
 - Каникул учурунда биз театрда же циркке барабыз;
 - 15 саны 3кө же 5ке бөлүнөт;
 - Эгерде сандын цифраларынын суммасы 3кө калдыксыз бөлүнсө, анда сан да 3кө бөлүнөт.
- А айтылышы «х – так сан», В айтылышы «х 5ке бөлүнөт». $A \vee B$ логикалык кошуусу кандай жыйынтык берет?
- $((1 \vee 0) \vee 1) \vee 0$ логикалык туюнтмасынын маанисин тапкыла.
- Жөнөкөй айтымдар берилген:

$$A = \{3=6\}, B = \{2 < 3\}, C = \{4 < 1\}$$
 Курама айтымдын чындык экендигин аныктагыла:

$$(A \vee \neg B) \wedge \neg (\neg A \vee C).$$
- a, b, c, – төмөнкүдөй маанилерге ээ болгон логикалык чоңдуктар болушсун:

$$a = \text{чындык}, b = \text{жалган}, c = \text{жалган}.$$
 Эми төмөнкү логикалык туюнтмалар үчүн чындык таблицасын түзгүлө:
 - a же b жана c эмес;
 - a жана b жана/же c жана b.
- Силер окуп үйрөнгөн логиканын мыйзамдарын чындык таблицасынын жардамы менен далилдегиле.



- бөлүм



Компьютер жана ПК

2.1-тема:

ПК жана лицензиянын түрлөрү

Силер көргөндөй программалык камсыздоосуз (ПК) эч бир компьютерде иштөөгө мүмкүн эмес. Ал компьютерге автоматтык түрдө орнотулушу мүмкүн (мисалы, операциялык системаны орноткондо, анын стандарттык ПКнын курамына жөнөкөй тексттик редактор, калькулятор, графикалык редактор ж.б. кирет) же болбосо колдонуучунун каалоосу менен кошумча орнотулат.

Программалык камсыздоо



Проприетардык (эркин эмес) ПК



Эркин ПК

Белгилүү болгондой, каалагандай программа бул адамдын интеллектуалдык ишмердүүлүгүнүн жыйынтыгы, демек – анын менчиги.

Мисалы, сиз дүкөндөн уюлдук телефон сатып алдыңыз. Эми бул сиздин менчигиңиз, а бирок телефондун дизайны мурдагыдай эле телефонду иштеп чыгуучуларга жана дизайнерлерге тийиштүү. Эгерде сиз бул телефондордун нускаларын (копияларын) чыгарып баштасаңыз, анда аны иштеп чыгуучулардын автордук укуктарын бузган болосуз.

Ушундай эле кырдаал программаларга да тиешелүү. Программанын өзү бул иштеп чыгуучунун интеллектуалдык менчиги, а сиз болсо бул программанын иштөөсүнүн жыйынтыгын гана сатып аласыз. Мыйзамдын көз карашы боюнча сиз программаны колдонуу укугуна гана ээ боло аласыз. Берилген программаны колдонуу шарттарынын бардыгы адатта программаны орнотуп жатканда чыгуучу лицензиялык макулдашууда көрсөтүлөт. Лицензиялык макулдашууга макулдугуңузду берүү менен сиз программаны иштеп чыгуучулардын бардык шарттарын кабыл



ЭСИҢЕ ТУТ

Автордук укук бир эле программаларга эмес, фильмдерге, музыкага, ал эмес графикалык объектерге (сүрөт, картинка) да таралат.

аласыз. Эгерде сиз лицензиялык макулдашуунун бир эле шартын бузсаңыз (андан ары таратууга тыюу салынгандыгын, өзгөртүп түзүүдө ж.б.), анда сиздин программа легалдуу болбой калат.

Эркин ПКнын өнүгүү тарыхы

Акы төлөнүүчү лицензиялык (проприетардык) ПКны өнүгүшүнө каршы салмакта өткөн кылымдын 80-жылдарынан баштап дүйнө жүзүндөгү программалоочулардын арасында **эркин (Open-source software) ПКны** өнүктүрүү кыймылы да пайда болгон.

Эркин ПК — бул бир гана акысыз ПК эле эмес, андай программа ачык баштапкы коду менен берилип, каалоочулар үчүн аны андан ары өзүнүн муктаждыктары үчүн өзгөртүп түзүп алуу мүмкүнчүлүгү да каралган. Ачык баштапкы код ачык лицензиянын негизинде таралат.



Ричард Столлман

Ушундай ачык лицензиялардын бири катары **General Public License (GPL)** эсептелет. Анын түзүүчүсү ачык ПК кыймылынын негиздөөчүсү Ричард Столлман болуп саналат.

Мындай лицензияны колдонуунун өзгөчөлүгү болуп, каалаган колдонуучу мындай ПКны өзгөртүүгө, толуктоого жана таратууга толук укугу бар экендиги эсептелет. Бардык жаңы өзгөртүүлөрдүн коду да башкалар үчүн ачык болушу керек.

Ушундай лицензия менен Linux, MySQL, Asterisk ж.б. өзөктөрү таралат. ПКны лицензиялоо үчүн экинчи популярдуу ачык лицензия катары **Apache License 2.0** эсептелет. Бул лицензия колдонуучу тарабынан өз алдынча жазылган коддун бөлүгүн жаап коюуга мүмкүндүк берет.



Эң популярдуу **Android** операциялык системасынын көпчүлүк баштапкы коддору ушул Apache 2.0 эркин лицензиясы менен таратылат. Азыркы учурда Android ОС де ар кандай түзүлүштөр, «интернет буюмдар» жана «акылдуу үйлөрдөн» тартып сыналгы, ноутбук жана автомобилдерге чейин иштейт. Бирок көпчүлүк учурларда Androidди смартфондордо жана планшеттерде кеңири колдонушат.

Интернеттеги маалыматты колдонуунун легалдуулугу (мыйзамдуулугу)

Интернеттеги материалдарды колдонууда жана файлдарды жөнөтүүдө тармактагы көпчүлүк маалыматтар автордук укук жөнүндөгү мыйзам (бул

жөнүндө веб-сайттарда түздөн түз көрсөтүлбөсө дагы) менен корголгон-дугун эске алуу керек. Бул берилген маалыматты сиз болгону көрүүгө гана мүмкүндүгүңүз бар дегенди түшүндүрөт. Аны сиз көчүрүп алууга, таратууга жана башкаларга берүүгө укугуңуз жок. Көпчүлүк учурларда мындай маалыматтар © [автордун аты-жөнү] копирайт белгиси менен белгиленет. Ал маалыматты колдонуу же иштеп чыгуу үчүн маалыматтын же сайттын ээсинен сөзсүз түрдө уруксат алуу керек дегенди түшүндүрөт.

Андыктан интернеттен маалыматты көчүрүп алуудан мурда берилген материал анын автору тараптан эркин таратылаарына толук ынанууңуз зарыл. Андан тышкары «акысыз» жана «эркин» колдонуу бир маанини бербейт. Акысыз колдонууда материалды жөн гана акысыз көрүп жана окууга болот, бирок аны андан ары таратууга болбойт.

Ал эми эркин колдонуу деген түшүнүк кененирээк маанини берет. Ал материалды бир гана көрүүдөн жана окуудан тышкары, аны кайра иштетүүгө (мисалы, тексттин бөлүгүн доклад үчүн алууга же фильмдин кайсы бир бөлүктөрүн колдонууга) жана андан ары таратууга да мүмкүндүк берет.

Интернетте авторлору башкаларга толук колдонуу укуктарын берген жана эркин таратылган билим берүү материалдары **ачык билим берүү ресурстары** деп аталат. Эң ири ачык билим берүү ресурсуна Википедия кирет. Андагы бардык макалалар Creative Commons ачык лицензиясы менен берилет.

Бул лицензия менен түзүүчү-авторлор бардыгына анын эмгегин эркин колдонуп, таратууга жана кошумчалоого уруксат берет. Ал эмес силер кармап турган бул окуу китеби да ачык билим берүү ресурсу. Китептин биринчи барагынан силер ачык лицензиянын белгисин таба аласыңар:



СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Эмне үчүн компьютердик каракчылык коомго зыян алып келет?
- 2) Open-source программаларга мисал келтиргиле жана алардын кызматын сүрөттөп бергиле.
- 3) Интернеттен алынган санариптик материалдарды толуктап иштетүүнүн кандай ыкмаларын билесиңер? Алардын кайсыларына авторлорунан уруксат алыш керек?

2.2-тема:

Маалыматтар базасы

Азыркы коомдун эң маанилүү маалыматтык ресурстарынын бири болуп маалыматтар базасы (МБ) эсептелет. Азыркы мезилде көптөгөн ири интернет-ресурстар МБсын маалыматтарды сактоо үчүн колдонушат, анткени алар бирдей касиеттер топтомуна ээ болгон объекттердин группасы жөнүндөгү берилиштерди уюштурулган тартипте сактоого мүмкүндүк берет.

МБнын негизги параметрлери бул анын физикалык жана логикалык көлөмдөрү. Физикалык көлөм байттар, Кб, Мб ж.б. менен туюнтулат, ал эми логикалык болсо жазуулардын санына көз каранды. Эң ири маалыматтар базасын ондогон терабайт жана миллиарддаган жазуулар түзөт.

Берилиштерди иреттөөнү берилиштер арасындагы байланыштарды жана алардын үстүнөн мүмкүн болгон аракеттерди түшүндүргөн эрежелердин топтомун **маалыматтар базасын уюштуруу модели** деп аташат.

Маалыматтар базасынын негизги үч моделин айырмалашат:

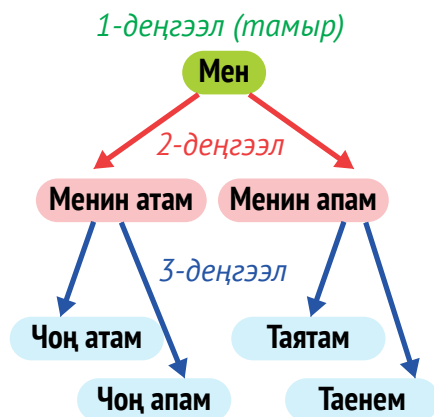
- 1 Иерархиялык
- 2 Тармактык
- 3 Реляциялык

Иерархиялык модель чокулары жазуулардан турган дарак сымал түзүлүштү элестетет. Мисалы, компьютерде сакталып турган файлдар жөнүндө маалыматтарды чагылдырган каталогдордун структурасы. Бул модель тамыры деп аталган биринчи деңгээлдеги бир чокуга гана ээ. Тамырдан тышкары ар бир деңгээлдин түйүндөрү жогорку деңгээлдин бир гана түйүнү менен жана төмөнкү деңгээлдин бир нече түйүнү менен байланышкан. Башкача айтканда мындай байланыш «бирөөдөн – көптүккө» деп аталат.



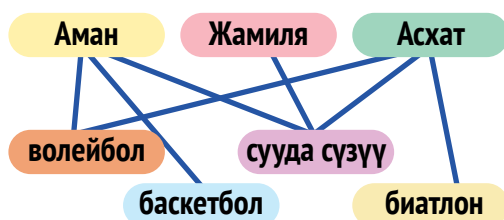
АНЫКТАМА

Маалыматтар базасы (мындан ары МБ) – бул эсептөө системасында алардын эффективдүү издөө жана иштетүү максатында логикалык системалаштырылган маалыматтар топтому.





Тармактык модель вертикалдык байланыштан тышкары горизонталдык байланыштарга да ээ, б.а. бир деңгээлдин ар бир элементи башка деңгээлдин каалаган сандагы элементтери менен байланышкан болушу мүмкүн. Мисал катары бир класста окугандардын кызыгуусу жөнүндө маалыматтар сакталган МБ айтсак болот: бир окуучунун көптөгөн кызыгуулары болушу мүмкүн, ал эми бир кызыгууга көп окуучулар ээлик кылышы мүмкүн. Мындай байланыш «көптөн – көпкө» деп аталат.



Реляциялык модель (англ. Relation – катыш) – бул бири-бири менен байланышта болгон таблицалардын жыйындысы. Реляциялык маалыматтар базасында байланыштардын бардык түрү колдонулушу мүмкүн: «бирөөдөн – бирөөгө», «бирөөдөн – көпкө», «көптөн – көпкө».

Реляциялык МБда сапчалар жазуулар деп, ал эми мамычалар – талаалар деп аталат. Ар бир таблица андагы ар бир жазууну идентификациялоого мүмкүндүк берген ачкычтык талааны камтуусу зарыл. Ошондуктан көптөгөн азыркы маалыматтар базасын башкаруу системалары (мындан ары МББС) реляциялык МБна ориентир алышат.

Эң популярдуу МББС: MySQL, Oracle, PostgreSQL, SQLite, Firebird, DB2, Microsoft Access, Microsoft SQL Server.



Open Office.org Base маалыматтар базасын башкаруу системасы

Ар кандай категориялардагы маалыматтарды камтыган татаал бири-бири менен байланышкан берилиштерди иштетүү үчүн маалыматтар базасын башкаруу системалары (МББС) колдонулат.

Open Office.org Base мисалында МББС иштөө принцибин карап көрөлү. Мисалы, бизге сүрөттөрдүн онлайн галереясын түзүү керек болду дейли. Ал үчүн ар бир сүрөт боюнча бардык актуалдуу маалыматтар сакталган маалымат базасын түзүү керек:

- сүрөттүн автору,
- качан тартылган жылы,
- жанры (портрет, натюрморт же пейзаж),
- сүрөттүн аты.

Эми, онлайн галереянын көрүүчүсү сайтка кирип, көрүү үчүн конкреттүү автордун гана сүрөтүн, же бир гана портреттик жанрды тандап алышы мүмкүн.

Программанын негизги терезесинин сол жагында негизги төрт баскыч жайланышкан: *таблицалар, суроо-талаптар (запросы), формалар жана отчеттор*. Алар төмөнкүлөрдү түзүүгө мүмкүндүк берет:

- **Таблицалар** – маалыматтарды сактоо үчүн;
- **Суроо-талаптар** – берилген критерий боюнча маалыматты тандоо үчүн;
- **Формалар** – берилиштерди таблицага тез жана ыңгайлуу киргизүү үчүн;
- **Отчеттор** – колдонуучу үчүн маалыматты ыңгайлуу түрдө чыгарып берүү жана кагазга басып чыгаруу үчүн.

Маалыматтар базасын түзүү

Программаны ишитетип баштаганда автоматтык түрдө даяр МБсын ачуу жана түзүү Мастер ишке кирет. Даяр МБсын түзүү үчүн:

- 1 «Жаңы МБ түзүү» пунктун тандап, «Кийинки» баскычын баскыла
- 2 Экинчи кадам ар бирине экиден жооп болгон эки суроого ээ. 1-суроого жооп адатта «Ооба, МБсын каттоо» болот, жана 2-суроого жооп, «Базаны редактирлөө үчүн ачуу» болот. «Даяр (Готово)» баскычын баскыла.
- 3 Андан ары маалыматтар базасын кайсы бир ат менен, мисалы Gallery деген ат менен сактоо.

МБнын таблицасын түзүү

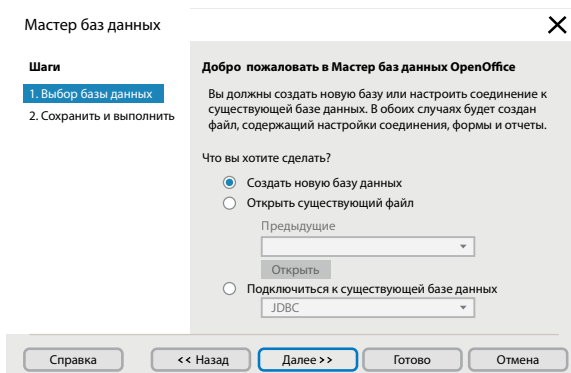
МБнын таблицасын түзүү үчүн «МБ (База данных)» терезесинен «Таблицалар» белгисин тандап жана таблица түзүүнүн кайсы бир ыкмасын тандап алуу керек:

- Таблицаны дизайн режиминде түзүү, б.а. өз алдынча талаанын атын коюу, маалыматтар тибин тандоо ж.б.



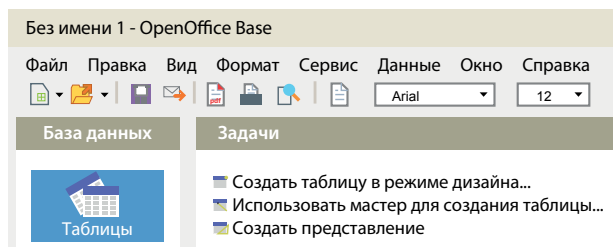
ЭСИҢЕ TUT

МБда сапчалар жазуу, ал эми мамычалар талаа деп аталышат. Ар бир таблица андагы ар бир жазууну идентификациялоого мүмкүндүк берүүчү ачкычтык талааны (ID) камтыш керек.



- Таблица түзүү үчүн мастерди колдонуу, б.а. даяр талаалары менен таблицалардан тандап алуу.

МБ түзүү үчүн «Дизайн режиминде таблица түзүү» дегенди тандайбыз, ачылган терезеде «Талаанын атын» жана «Талаанын тибин» киргизебиз:



Название поля	Тип поля
автор	Текст [VARCHAR]
название	Текст [VARCHAR]
жанр	Текст [VARCHAR]
год	Дата [DATA]

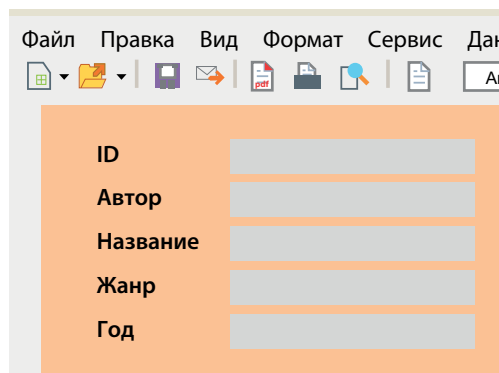
Таблицанын талаалары ар кандай типти алышы мүмкүн, Мисалы:

- Тексттик
- Логикалык
- Сандык
- Дата/убакыт

Файлды биринчи жабууда жана сактоодо, программа «Баштапкы ачкычты түзөйүнбү?» деп сурайт. Мында «Ооба» дегенди тандоо керек, натыйжада сиздин таблицага «ID» деген ат менен дагы бир талаа кошулат.

Форманы түзүү

Эми колдонуучунун интерфейсин түзөлү, б.а. МБнын таблицасына бериштерди киргизүүгө мүмкүн болгон форманы түзөбүз. Ал үчүн Форма баскычына басып «Форма мастера (Мастер форм)» пунктун тандап алгыла. Силер түзгөн таблицанын негизинде «Мастер форма» силерди жөнөкөй кадамдар аркылуу алып өтөт. Ушул жерден силер форманын сырткы келбетин да тандап аласыңар.



Суроо-талаптарды түзүү

Маалыматтар базасынын таблицаларынын негизинде суроо-талаптарды да түзсө болот, б.а. силерге керектүү берилиштерди гана чыпкалап (филтирлеп) алса болот. Суроо-талапты ачып, силер сураган параметрлер менен берилиштерди таблица түрүндө көрө аласыңар.

Таблица жана формалар сыяктуу эле суроо-талапты да «Суроо-талап мастери» режиминде жана «Суроо-талап дизайны» режиминде түзө аласыңар.

Суроо-талапты аныктоо үчүн талааларды көрсөтүү шартын жана анда киргизиле турган маалыматтар базасынын талааларынын атын көрсөтүү керек. Мисалы биздин галереянын маалыматтар базасында конкреттүү авторлордун сүрөтүн жана портреттик жанрдагы гана түрүн тандап алууга болот.

Отчетторду түзүү

Отчет – бул силердин МБдагы берилиштерди чагылдырган тексттик документ. Силер отчеттун өлчөмүн, сырткы келбетин жана структурасын өзгөртсөңөр болот.

Отчеттор бир же бир нече таблицадагы берилиштерди көрсөтүшү мүмкүн. Алар анын негизинде түзүлүп жаткан ар бир таблицалардын же суроо-талаптардын талааларынын каалагандай комбинациясынан турушу мүмкүн.

Отчетту тез арада түзүү үчүн «Отчеттерду түзүү мастерин» колдонсо болот. Ал жерде этап менен:

- а) берилиштердин булагын тандоо (берилиштер алына турган таблица же суроо-талап);
- б) көрсөтүлө турган талааларды тандап алуу керек;
- в) отчеттун сырткы келбетин аныктоо (талаалардын жайланышы, элементтерди чагылдыруу тиби жана фон).

Эми даяр отчетту кагазга басып чыгарсаңар болот.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1)** Иерархиялык жана тармактык маалыматтар базасына мисал келтиргиле. Open Office каражаттарынын жардамы менен алардын схемаларын түзгүлө.
- 2)** Талаалардын төмөнкү типтерин колдонуп «Менин досторум» маалыматтар базасын түзгүлө: тексттик, сандык, дата/убакыт, логикалык.
- 3)** Төмөнкү маалыматтарды камтыган китепкананын маалыматтар базасын иштеп чыккыла: китептин аты, авторлордун Ф.А.А, басмакананын аты, чыккан жылы, беттердин саны, иллюстрациялардын саны, конкреттүү китептин нускаларынын саны, ушул китеп берилген окуучулардын саны. Маалыматтар базасы таблицаны, суроо-талапты жана форманы камтысын.

3

- бөлүм



Программалоо



3.1-тема:

Татаал шарттар: and, or, not

Программалоодо шартты туура коё билүү өтө маанилүү. Көпчүлүк учурда шарттар татаал болушат, б.а. «ЖАНА», «ЖЕ» жана «ЭМЕС» логикалык операторлору (байламта) менен бириккен бир нече курама шарттардан турушат. Python тилинде алар «and», «or», «not» деген англис сөздөрү менен жазылат.

and логикалык оператору (логикалык көбөйтүү)

Туюнтмада **and** байламтасы менен бириккен курама шарттардын баары тең **True** маанисине барабар болсо, анда татаал шарт **True** маанисин кайтарат. Эгер эки туюнтманын бирөө эле жалган болсо, анда шарт жалган:

```
x = 5
if x < 10 and x % 3 == 0:
    print('True')
else:
    print('False')
```

Бул жерде жооп False болот, анткени шарттын экинчи бөлүгүнө ылайык берилген 5 саны 3кө калдыксыз бөлүнбөйт. Эгерде биз туюнтманы $x \% 3 == 0$ **and** $x < 10$ деп жаза турган болсок, ал кайрадан эле False маанисин кайтармак. Бирок мындагы экинчи салыштыруу $x < 10$ интерпретатор тарабынан аткарылмак эмес, аны аткаруунун кажети жок. Анткени биринчи туюнтма ($x \% 3 == 0$) жалган болгондуктан **and** операторунун болушу бардык туюнтманы жалганга чыгарды.

or логикалык оператору (логикалык кошуу)

Эгерде жок дегенде бир туюнтма True маанисине ээ болсо True маанисин кайтарат:

```
x = 5
if x < 10 or x % 3 == 0:
    print('True')
else:
    print('False')
```

Бул жерде жооп True болот, анткени шарттын биринчи бөлүгүнө ылайык берилген 5 саны 3кө калдыксыз бөлүнбөсө дагы 10 санынан кичине. Мына ушул үчүн эгерде эки туюнтманын бирөөсү эле True маанисин кайтарса, анда экинчи туюнтма бааланбайт, анткени **or** оператору баары бир True маанисин кайтарат.

not логикалык оператору (логикалык тануу)

Not унардык оператору чындыкты жалганга кайтарат, ал эми жалганды чындыкка кайтарат. Унардык дегенибиз, анткени ал **and** жана **or** операторлорундай болуп анын оң жагында же сол жагында турган туюнтмаларга эмес андан кийин турган бир эле туюнтмага колдонулат.

1-вариант:

```
x = 8
print (not x < 15)

False
```

2-вариант:

```
x = 8
print (not x > 15)

True
```

Эгерде бир туюнтмада бир эле убакта бир нече же бардык логикалык операторлор колдонулса, анда аткаруу тартиби төмөнкүдөй болот:

- 1) катыш (<, >, <=, >=, ==, !=)
- 2) not («ЭМЕС»)
- 3) and («ЖАНА»)
- 4) or («ЖЕ»)

Аракеттердин иретин өзгөртүү үчүн тегерек кашаалар колдонулат. Кашаалар пайда болгон учурдагы эсептөөлөрдүн иретинин өзгөрүшүн мисалда карап көрөлү:

1-вариант:

```
a=4
b=6
c=8
result = c==8 or b<a and not a < 7
print (result)
```

Жыйынтык:

True

Эмне үчүн экендигин түшүндүрөлү:

```
c == 8 or b < a and not a < 7
  True   False   True
           False
           False
           False
           True
```

2-вариант:

```
a=4
b=6
c=8
result= (c==8 or b<a) and not a < 7
print (result)
```

Жыйынтык:

False

Эмне үчүн экендигин түшүндүрөлү:

```
(c == 8 or b < a) and not a < 7
  True   False   True
           False
           True
           False
           False
```

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) *and* операторунун жардамы менен бирөө чындыкты, экинчиси – жалганды көрсөткөн эки татаал логикалык туюнтманы түзгүлө.
- 2) Жогорудагы маселени *or* операторун колдонуп аткаргыла.



3.2-тема:

Тизмелер, кортеждер жана сөздүктөр

Көпчүлүк учурда бизге көптөгөн окшош маалыматтарды бир файлда чогултууга туура келет, мисалы, окуучулардын тизмеси же маалымдамадагы телефон номерлер. Python тилинде мындай маалыматтардын топтомун **тизмелерде, кортеждерде жана сөздүктөрдө** уюштурса болот.

Тизме (list) – бул белгилүү иретте жайгашкан элементтерден турган структура. Ар бир элементке ага кайрылууга мүмкүн болгон номер (же индекс) туура келет. Тизмени түзүү үчүн квадраттык кашаанын ичине үтүр менен ажыратылып, анын бардык элементтери тизмектелет.

Мисалга өзүбүздүн үй-бүлө мүчөлөрүбүздүн тизмесин түзөлү:

```
>>> myfamily = ['father', 'mother', 'sister', 'brother']
```

Бул учурда биздин тизме myFamily деген өзгөрмөдө сакталат. Качан тизме түзүлгөндөн кийин, бул тизме менен иштей турган программаны жазсак болот. Мисалга, цикли колдонуп, үй-бүлөнүн ар бир мүчөсү менен саламдашуу программасын жазалы:

Python тилиндеги программа

```
myfamily = ['father', 'mother',  
'sister', 'brother']  
for item in myfamily:  
    print('Hello', item)
```

Экранга чыккан жыйынтыгы

```
Hello father  
Hello mother  
Hello sister  
Hello brother
```

Тизме ар кандай типтеги объекттерди камтышы мүмкүн. Бир эле тизмеге бир убакта сапты, сандарды, башка типтеги объекттерди киргизүүгө болот:

```
objects = [1, 2.6, 'Hello', True]
```

Тизмелерди бири-бирине кошсо болот, анда жаңы тизме эки тизмедеги тең элементтерди камтып калат:

```
x = [1, 2, 3, 4]  
y = [5, 6, 7, 8]  
z = x + y  
print (z)
```

Жыйынтык:

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Тизмелер менен ар түрдүү көп амалдарды жасаса болот:

<code>x in A</code>	<code>x</code> элементи <code>A</code> тизмесинде бар же жок экенин текшерет. <code>True</code> же <code>False</code> маанисин кайтарат.	<pre>A = [1, 2, 3, 4, 5, 6, 7, 8] print (2 in A) Жыйынтык: True</pre>
<code>min(A)</code>	<code>A</code> тизмесинен эң кичине элементти табуу.	<pre>A = [1, 2, 3, 4, 5, 6, 7, 8] print (min(A)) Жыйынтык: 1</pre>
<code>max(A)</code>	<code>A</code> тизмесинен эң чоң элементти табуу.	<pre>A = [1, 2, 3, 4, 5, 6, 7, 8] print (max(A)) Жыйынтык: 8</pre>

Кортеж (tuple) тизме сыяктуу эле элементтердин удаалаштыгынан турат. Бирок андагы элементтерди өзгөртүүгө, кошууга же өчүрүүгө болбойт. Кортеждерди түзүү үчүн үтүр менен бөлүнгөн анын маанилери жайгашкан тегерек кашаалар колдонулат:

```
user = ('Timur', 23, 1.10.1998)
print(user)
```

Кортеждерде объекттердин касиеттерин сактоо ыңгайлуу, мисалы, атын, жашын, туулган датасын. Эгерде кортеж болгону бир элементтен турса, анда кортеждин жалгыз элементинен кийин үтүр белисин коюу керек:

```
user = ('Tom',)
```

Сөздүктөр (dictionary) – бул ар бир элементи индекстин ордуна уникалдуу ачкычка ээ болгон маалыматтардын структурасы. Сөздүктүн элементтерин өзгөртө берсе болот. Сөздүктөрдү түзүү үчүн фигуралык кашаалар колдонулат ({}):

```
dictionary = {ачкыч1:мааниси1, ачкыч2:мааниси2, ...}
```

`myschool` атындагы сөздүктү түзөлү:

```
myschool = {'5 klass':'Anara, Kanat, Pavel', '6 klass':
'Chyngyz, Tina, Emil'}
```

Бул сөздүктө ачкыч катары класстын аттары, ал эми мааниси катары – ошол класстарда окугандардын аттары берилди.



Сөздүккө аны жаңы ачкыч менен белгилеп маанилерди кошсо болот:

```
myschool['7 klass'] = 'Elena, Ainura, Dastan'  
print (myschool)
```

Жыйынтык:

```
{ '5 klass': 'Anara, Kanat, Pavel', '6 klass': 'Chyngyz,  
Tina, Emil', '7 klass': 'Elena, Ainura, Dastan' }
```

Элементтин маанисин өзгөртүү үчүн, анын ачкычына жаңы маани берип коюу керек:

```
myschool = { '5 klass': 'Anara, Kanat, Pavel', '6  
klass': 'Chyngyz, Tina, Emil' }  
myschool['6 klass'] = 'Matvei, Tina, Salima'  
print (myschool)
```

Жыйынтык:

```
{ '5 klass': 'Anara, Kanat, Pavel', '6 klass': 'Matvei, Tina, Salima' }
```

for циклин колдонуп, экранга сөздүктүн ачкычтарын эле чыгарса болот:

```
for i in myschool:  
    print(i)
```

Жыйынтык:

```
5 klass  
6 klass
```

Же болбосо сөздүктүн маанисин гана чыгарса болот:

```
for i in myschool:  
    print(myschool[i])
```

Жыйынтык:

```
Anara, Kanat, Pavel  
Chyngyz, Tina, Emil
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

«Балыктар» сөздүгүн түзгүлө, ал эми анын элементтерин 3 түргө бөлгүлө: «дарыя», «көл» жана «деңиз» балыктары деп. Экранга биринчи сөздүктүн ачкычтарын, андан кийин элементтерин чыгаргыла.

3.3-тема:

Циклдик алгоритмдер

Каалагандай алгоритмдер 3 конструкциянын жардамы менен жазылаары далилденген: цикл, шарттуу операторлор, сызыктуу алгоритмдер менен.

Силер билгендей, **цикл** – бул окшош аракеттерди көп жолу аткаруу.

ОШОНДОЙ ЭЛЕ КАРА

3.4-тема 7-класс

while жана **for** циклдери

7-класста биз **while** жана **for** циклдерин үйрөнүп баштаганбыз. **For** цикли **while** циклинен кайсы бир командаларды алдын ала белгилүү санда кайталоо үчүн колдонулгандыгы менен айырмаланат. Ал эми **while** цикли тескерисинче кайсы бир аракеттерди канча жолу кайталай тургандыгы бизге белгилүү болбогон учурларда колдонулат. Ошону менен бирге бизге ошол аткарылганга чейинки циклди кайталаш керек болгон шарт белгилүү.

For циклинин колдонулушун кеңири карап көрөлү. For циклинин Python тилинде жазылышы төмөнкү схема боюнча жүрөт:



Цикл башталганда диапазондун биринчи элементинин мааниси белгиленген аракет аткарыла турган өзгөрмөгө ыйгарылат. Циклдин экинчи өтүшүндө өзгөрмөгө диапазондун экинчи элементинин мааниси ыйгарылат. Жана ушул сыяктуу диапазондогу бардык элементтер менен берилген аракеттер жүргүзүлүп бүтмөйүнчө кайталана берет. Келгиле мисал карайлы: **letter** өзгөрмөсүнө ар бир жолу **Python** сабынын жаңы элементи ыйгарылып турсун. Print командасы экранга бул саптын ар бир тамгасын бирден чыгарат:

```

for letter in 'Python':
    print (letter, 'тамгасы', )
>>>
P тамгасы
y тамгасы
t тамгасы
h тамгасы
o тамгасы
n тамгасы
  
```



Төмөнкү мисалда ар бир кийинки өтүүдө өзгөрмөнүн мааниси берилген диапазондогу санга көбөйүп турат:

```
f = 12
for i in range(1, 6):
    f = f + i
print (f)
>>>
27
```

«for i in range(1,6)» цикли беш жолу аткарылат (6 – кирбейт). Циклдин ар бир кадамында f өзгөрмөсү i санына өсүп турат. Баштапкы мааниси f = 12. Циклде маанилери өзгөрүп турат:

1-өтүү: $f = 12 + 1 = 13$
2-өтүү: $f = 13 + 2 = 15$
3-өтүү: $f = 15 + 3 = 18$
4-өтүү: $f = 18 + 4 = 22$
5-өтүү: $f = 22 + 5 = 27$

Кыскача мындай кылып жазсак болот: $f = 12 + 1 + 2 + 3 + 4 + 5 = 27$

Range функциясынын аргументтери төмөнкүдөй берилет:

- **range (x)** – 0 дөн x ке чейинки маанилерди алат, бирок x – диапазонго кирбейт;
- **range (y, x)** – у тен x ке чейинки бардык маанилерди алат, мында да x диапазонго кирбейт;
- **range (y, x, s)** – у тен x ке чейинки бардык маанилерди s кадамы менен алат.

Мисалы:

```
for i in range(0, 15, 3):
    print(i)
```

Берилген мисалда for цикли 0 дөн 15 ке чейинки маанилерди 3 кадам менен алат, жыйынтыгында ал ар бир үчүнчү санды чыгарып берет:

```
>>>
0
3
6
9
12
```

Андан тышкары кадам үчүн терс сандарды да колдонсо болот, анда цикл маанилерди тескери багытта тандап ала баштайт:


```
for i in range(100,0,-20):
    print(i)
>>>
100
80
60
40
20
```

Удаалаштыкты белгилүү санда кайталаган **for** циклинен айырмаланып **while** цикли саны менен эмес, логикалык шарты менен жетектелет. Ошондуктан кодду канча жолу аткараарынын так санын билүүнүн кажети жок.

While циклинин коду логикалык шарт чындык маанисинде (True) болуп турганга чейин кайталана берет.

1-маселе. Ушул циклдин негизинде оюндун программасын түзүп көрөлү. Мында колдонуучу компьютер тарабынан катылган санды табышы керек:

```
import random #кокустук сандар китепканасын жүктөйбүз
number = random.randint(1, 25) #компьютер кокустук санды тандайт
choices = 0 #choices өзгөрмөсүнө аракеттердин санын жазабыз
while choices < 5: #циклди 5 аракетке чейин аткарат
    print('1ден 25ке чейинки санды тап:') #колдонуучуга
    санды киргизүүнү сунуштайт
    guess = input()
    guess = int(guess) #киргизилген сан бүтүн болуш керек
    choices = choices + 1 #ар бир аракетте эсептөө 1ге өсүп турат
    if guess == number: #эгерде киргизилген сан катылган санга
        барабар болсо
            break #программаны токтот
```

choices өзгөрмөсүнө 0 мааниси ыйгарылган. Ал санды табуу боюнча жасалган ар бир аракет сайын көбөйө берет. Биз программа чексиз циклге түшүп калбашы үчүн аракеттердин санын 5өө менен чектейбиз.

Программа иштеп жатат, бирок ал колдонуучуга эч кандай жыйынтыкты билдирбейт: колдонуучу катылган санды таптыбы же тапкан жокпу, билбейт. Жыйынтыгы мындай көрүнөт:

```
1ден 25ке чейинки санды тап:
5
1ден 25ке чейинки санды тап:
16
```



```
1ден 25ке чейинки санды тап:
7
1ден 25ке чейинки санды тап:
18
1ден 25ке чейинки санды тап:
10
>>>
```

Ал үчүн колдонуучуга анын саны катылган сандан чоң же кичине экенин билдирип тургандай шарттуу операторлорду киргизели. Бул болсо санды тезирээк тапканга жардам берет:

```
import random
number = random.randint(1, 25)
choices = 0
while choices < 5:
    print('1ден 25ке чейинки санды тап:')
    guess = input()
    guess = int(guess)
    choices = choices + 1
    if guess < number: #сан катылган сандан кичине болсо
        print('Менин саным сеникинен чоң')
    if guess > number: #сан катылган сандан чоң болсо
        print('Менин саным сеникинен кичине')
    if guess == number: #сан катылган санга барабар болсо
        break
if guess == number:
    print('Азамат! Сен санды' +str(choices)+ 'аракеттен кийин тап-
тың!')
else:
    print('Тилекке каршы сен санды тапкан жоксуң. Мен' + str(number) +
'санын каттам')
```

Эгерде программаны ишке киргизсек, анда колдонуучу менен баарлашуу варианты төмөнкүдөй болот:

```
1ден 25ке чейинки санды тап:
6
Менин саным сеникинен чоң
```

```

1ден 25ке чейинки санды тап:
17
Менин саным сеникинен кичине
1ден 25ке чейинки санды тап:
14
Менин саным сеникинен чоң
1ден 25ке чейинки санды тап:
15
Азамат! Сен санды 4 аракеттен кийин таптың!
>>>>

```

Эми программа колдонуучуга санды табуу үчүн жардамдашат. Мисалы, компьютер 15 санын катса, ал эми колдонуучу 17 санын киргизсе, программа киргизилген сан катылган сандан чоң экендигин көрсөтөт.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) Төмөнкү программанын иштөөсүнүн жыйынтыгында алынган у өзгөрмөсүнүн маанисин жазгыла:

```

y = 5
for i in range(2,6):
    y = y + 4 * i
print (y)

```

2) Узундугу 20 м тактай берилген. Бул тактайдан узундугу 5 м жана 2 м болгон канча минималдык бүтүн кесиндилерди даярдоого боло тургандыгын чыгаруучу программаны түзгүлө.

3) Берилген код боюнча алгоритмдин ар бир кадамындагы өзгөрмөлөрдүн маанилерин таблицкага жазгыла:

```

k=4   p=1040   m=2
while p != m*m:
    k=k+1
    p=p-4
    m=m*2
print (k)

```

k	p	m	m*m

4) Майнкрафттын каарманы Алекстин минутасына 4 минерал чыгара турган машинасы бар. Ар бир 100 минералга дал ушундай эле минутасына 4 минерал чыгарган машина курууга болот. Бир сааттан кийин Алекстин канча машинасы боло тургандыгын аныктоочу программаны жазгыла.



3.4-тема:

Камтылган шарттуу амалдар жана циклдер

7-класста биз силер менен **if** жана **else** операторлору кандай иштей тургандыгын көрдүк эле. Программада алар аткаруунун эки вариантын көрсөтөт. Бирок программанын алгоритми экиден көп жолду тандоону сунушташы мүмкүн, мисалы үчөөнөн, төртөөнөн же андан көптөн.

Шарттуу операторлордун ичинде ар кандай операторлор, анын ичинде башка шарттуу операторлор да камтылышы мүмкүн. Мисалы бизде мештин ичиндеги билдиргичтен алынган температуранын көрсөтмөсү бар дейли. Эми ал жогорубу, төмөңбү же нормадабы, аныкташ керек. Нормалдуу деп 200дөн 250 градус Цельсийге чейинки температура эсептелет. Өзгөрмөдө температура сакталат. Бир эле шарттуу оператор жетишсиз, анткени мында үч мүмкүн болгон жыйынтык бар. Маселенин чыгарылышын төмөндөгүдөй жазса болот:

```
t = int(input('Температураны киргизгиле'))
if t > 250:
    print('Мештеги температура өтө жогору')
else:
    if 200 <= t <= 250:
        print('Мештеги температура нормада')
    else:
        print('Мештеги температура нормадан төмөн')
```

Барабардыкты текшерген **if** шарттуу оператору **else (антпесе)** блогу-нун ичинде жайгашкан, ошондуктан ал камтылган шарттуу оператор деп аталат. Бул мисалдан көрүнүп тургандай, аны колдонуу бир нече варианттан бирөөнү тандап алууга мүмкүндүк берет. Эгерде **else** операторунан кийин эле дагы бир **if** кетсе, **elif (else-if** тин кыскартылганы) сөзү менен «каскаддык» тармакты колдонсо болот. Мисалы: берилген **x** жана **y** координаталары боюнча координаттык тегиздиктин чейректерин аныктоо керек. **x** жана **y** өзгөрмөлөрүндө клавиатурадан киргизилген бүтүн сандык маанилер сакталат.

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
```

```

    print('Биринчи чейрек')
elif x > 0 and y < 0:
    print('Төртүнчү чейрек')
elif y > 0:
    print('Экинчи чейрек')
else:
    print('Үчүнчү чейрек')

```

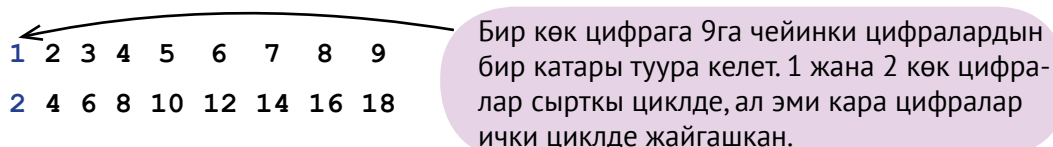
Келтирилген программада `if`, ..., `elif` шарттары кезеги менен текшерилет жана биринчи чыныгы шартка дал келген блок аткарылат. Эгерде бардык текшерилип жаткан шарттар жалган болсо, анда `else` блогу аткарылат, эгерде ал бар болсо.

Камтылган циклдер

Татаал маселелерде көпчүлүк учурда циклдин ар бир кадамында өзү да циклдик алгоритмди түзгөн түрдүү берилиштерди иштетүүнү аткарууга туура келет. Бул учурда «циклдин ичиндеги цикл» же «камтылган цикл» конструкциясы алынат.

Цикл *камтылган* деп аталат, эгер ал башка циклдин ичинде жайгашса. Биринчи жүрүштө сырткы цикл ички циклди чакырат. Ал өзүнүн акырына чейин аткарылгандан кийин башкаруу кайрадан сырткы циклдин тулкусуна берилет. Экинчи жүрүштө сырткы цикл кайрадан ички циклди чакырат, жана ушинтип бул процесс сырткы цикл аяктамайынча кайталана берет.

1-маселе. Экранга көбөйтүүнүн таблицасын чыгарабыз. Ал үчүн сырткы циклде 1ден 9га чейинки сандарды терип чыгуу керек. Ал сандардын ар бирине ички циклде 1ден 9га чейинки сандарды терип чыгуу керек.



Бир көк цифрага 9га чейинки цифралардын бир катары туура келет. 1 жана 2 көк цифрлар сырткы циклде, ал эми кара цифралар ички циклде жайгашкан.

Мындагы ички циклде биринчи катардагы сырткы жана ички циклдердин өзгөрмө-эсептегичтерин көбөйтүш керек.

Ушундай жол менен сырткы циклдин бир аткаруусуна ички циклдин 9 аткаруусу туура келет жана көбөйтүүнүн таблицасынын бир сабы түзүлөт. Ар бир саптан кийин жаңысына өтүү керек: бул ички цикл аткарылып бүткөндөн кийин, сырткы циклде жөргүзүлөт.



Андан тышкары таблицаны тургузуу үчүн форматталган киргизүүнү колдонуш керек, б.а. мамычалардын туурасын ($\backslash t$) берүү керек, антпесе сандар жылышып калат, анткени ар бир саптагы цифралардын саны ар башка.

Биздин код мындай болуп көрүнөт:

```
for i in range(1,10): #1ден 10го чейинки биринчи көбөйтүүчү
    for j in range(1,10): #1ден 10го чейинки экинчи көбөйтүүчү
        print(i*j, end='\t')
    print()
```

Жыйынтык:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

2-маселе. 2ден 100гө чейинки интервалдагы бардык жөнөкөй сандарды табуучу программаны түзөлү.

Жөнөкөй сан – бул 1 санына жана өзүнө гана калдыксыз бөлүнгөн сандар. Мисалы 5 – жөнөкөй сан, анткени ал 1ге жана 5ке гана калдыксыз бөлүнөт, ал эми 6 – курама сан, анткени 6га жана 1ден тышкары ал 2ге жана 3кө да калдыксыз бөлүнөт.

- Бир жөнөкөй сан болуп эсептелбейт, ошондуктан биздин мисалда диапазон 2 цифрасынан башталат.
- Шарт боюнча, эгерде n саны 2ден n ге чейинки диапазондо бөлүүчүгө ээ эмес болсо, анда ал жөнөкөй. Ал эми бул интервалда жок дегенде бир бөлүүчү чыгып калса, анда сан курама.
- n санынын кандайдыр бир k санына бөлүнүүчүлүгүн текшерели, эгер калдык нөлгө барабар болсо, анда n саны k санына бөлүнөт.
- Эгер жок дегенде бир бөлүүчүсү табылса, анда сан курама жана бул маселеде андан аркы бөлүүчүлөрдү издөөнүн кажети жок. Ал үчүн $n \% k == 0$ болсо, анда **break операторунун жардамында циклден тез арада чыгуу** аткарылат.
- `flag` өзгөрмөсү сандын жок дегенде бир өзгөрмөсү бар же жок экенин көрсөтөт.

Ошентип программабызды мындай кылып жазсак болот (мында n, k – бүтүн сандык өзгөрмөлөр):

```
for n in range(2, 101):
    flag = False
    for k in range(2, n):
        if n % k == 0:
            flag = True
            break
    if not flag:
        print (n)
```



ЭСИҢЕ TUT

Цикл n жолу кайталанышы үчүн диапазондун акыркы саны $n+1$ болуш керек.

3-маселе. Экранда эки түрдөгү символдордун жардамы менен «тик бурчтук» тарталы. Тик бурчтуктун четтери «0» символу менен, ал эми анын ичи «1» символу менен тартылсын.

Мейли тик бурчтуктун узундугу 10 символго жана туурасы 7 символго барабар болсун. Сырткы цикл саптарды терип жатып, биринчи жана акыркы цифраларга 0дү коюш керек. Эгерде сап биринчи же акыркысы болсо (эсептөө 0дөн башталгандыктан 0- жана 6-саптар), 0 башынан аягына чейини тизилип чыгат. Калган бардык башка учурларда 1 цифрасын коёбуз. Программаны жазалы:

```
for i in range(7):          #сапты 7 жолу чыгарабыз
    if i==0 or i==6:        #эгерде сап 1-чи же акыркы болсо
        for j in range(20): #бардык 20 жолу
            print('0', end='') #0дү чыгарабыз
        else:               #антпесе
            print('0', end='') #0дү чыгарабыз
            for j in range(1, 19): #1 жана 19-дан башкасына
                print('1', end='') #1 цифрасы менен чыгарабыз
            print('0', end='')
    print()
```

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Айдын номерин алып, ошого тиешелүү жыл мезгилин чыгаруучу же ката жөнүндө маалымат берүүчү программаны жазгыла.
- 2) 5 кг, 10 кг жана 15 кг алма бата турган ящиктер берилди. 100 кг алманы бөлүштүрүү үчүн канча ар кандай өлчөмдөгү ящик керек боло тургандыгын аныктоочу программаны түзүү керек.



3.5-тема:

Функциялар

Мурда силер интерпретатордун өзүндө **орнотулган** функцияларды колдонуп келгенсиңер, мисалы:

`print()` – экранга тегерек кашаанын ичиндегилердин бардыгын чыгарат;
`str()` – берилиштерди саптык типке өзгөртүп түзөт;
`int()` – берилиштерди бүтүн санга өзгөртүп түзөт;
`float()` – бүтүн сандарды бөлчөк типке өзгөртүп түзөт;
`round()` – санды модулу боюнча чоң жагына тегеректейт.

Булардан башка биз тигил же бул маселелерди аткартуу үчүн өзүбүздүн функцияларды түзүп алсак болот. Бул үчүн Python тилинде эгер кайсы бир алгоритм (же фрагменти) кайталанып жатса, аны функция катары формага келтирсе боло турган мүмкүнчүлүгү каралган.

Ал үчүн жаңы функцияга ат берип жана анын алгоритмин баяндоо керек. Мындан кийин программада функциянын атын жазганда эле өзүнүн кирген жана чыккан берилиштери менен тиешелүү алгоритм ишке кирет. Функцияны аткаргандан кийин программанын иши функцияны чакырган командадан кийин кайра улана берет.

Мисалга, программанын бир нече жеринде экранга «Программада ката» деген билдирүүнү чыгарыш керек болуп жатат. Аны мындайча жасасак болот:

```
print ('Программада ката')
```

Бул чыгаруу операторун керек болгон жердин бардыгында коё берсек, анда бул эсти толтуруп жибегиши мүмкүн. Эгерде билдирүүнүн текстин өзгөртүш керек болуп калса, анда бул чыгаруу операторлорун бүткүл программанын ичинен издеш керек болот. Мына так ушундай учурлар үчүн кошумча алгоритм – функциялар колдонулат. Аларга программанын каалаган жеринен кайрыла берсе болот. `error` функциясын жазалы:

```
def error():  
    print ('Программада ката')  
n = int (input())  
if n < 0:  
    error()
```

Биз `error` деген жаңы функцияны киргиздик.

Функциянын аты **def** (англ. *define* – аныктоо) ачкыч сөзү менен башталат, андан кийин гана функциянын уникалдуу аталышы берилет (мисалы: **def sum**). Аталышынан кийин функциянын параметрлери киргизилген кашаалар жана кош чекит коюлат. Функциянын тулкусу жылдыруу менен жазылат. Функцияны программанын башка жеринде иштетиш үчүн, анын аталышы менен (кашааларын кошуп) чакырыш керек. Мисалы: **error()**.

Эгерде кандайдыр бир аракеттер программанын ар кайсы жерлеринде бир нече жолу кайталанса, анда функцияны колдонуу кодду бир топ кыскартууга мүмкүндүк берет. Кээде өтө чоң программаларды жөнөкөйлөтүп жана ыңгайлуу кылуу үчүн бир нече функцияларга бөлүп алышат. Анын татаал алгоритмдеринин өзүнчө этаптарын функция түрүндө көрсөтөт. Мындай ыкма бардык программаны түшүнүктүү кылат.

Функция жана алардын аргументтери

Функцияларга аргументтерди – аткарылуучу аракеттерди өзгөртүү үчүн кошумча берилиштерди берсе болот.

Мисалы таблица же бөлүүчү сызыкты тартуу үчүн экранга бир символду көп жолу чыгарыш керек дейли. *n* өзгөрмөсү үчүн бул маселени чечүүчү программаны мындай жазсак болот:

```
n = 125 #ушунча жолу
s = '-' #символ
while n > 0:
    print (s, end = '')
    n -= 1;
```

Аяктоочу символ (адатта “жаңы сап” символу) – **end** аталыштагы аргументи менен **print** функциясынын чакырылышына көңүл буралы.

Кайталануучу символдун чыгаруу циклинин кошумча алгоритмин функция түрүндө жазсак болот. Бул функцияга *аргументин* бериш керек – символ жана аны канча жолу кайталаш керектигин көрсөткөн сан. Анда:

```
def print_char(s, n): #аргументи менен функциянын аты
    k = n
    while k > 0:
        print (s, end = '')
        k -= 1
print_char('-', 10) #аргументтер
```



ЭСИҢЕ ТУТ

Функциянын аты кичинекей латын тамгаларынан туруш керек, ал эми сөздөр бири-биринен төмөнкү сызык символу менен ажыратылышы керек. Бул кодду окуу үчүн ыңгайлуу кылат (snake case).



Негизги программа `print_char` функциясын чакыруунун болгону бир командасын гана камтыйт. Кашаанын ичинде тире («-») символун 10 жолу чыгарыш керектигин көрсөтүүчү функциянын аргументи көрсөтүлгөн.

Глобалдык жана локалдык өзгөрмөлөр

Көпчүлүк учурларда функцияларды берилиштерди иштетүү үчүн колдонушат. Бул берилиштер глобалдык же локалдык болушу мүмкүн. **Локалдык өзгөрмөлөр** функцияга анын атынан кийин тегерек кашаанын ичинде көрсөтүлгөн аргументтер аркылуу берилет. Локалдык өзгөрмөлөр ошол функциянын гана «көрүнүү зонасында» жайгашат жана программанын калган бөлүгүнө жеткиликтүү эмес. Ал эми **глобалдык өзгөрмөлөр** программанын бардыгында жеткиликтүү. Аларга аты боюнча кайрылса болот жана аны менен байланышкан маанилерди алса болот.

1-маселе. Программанын негизинде өзгөрмөлөрдүн типтерин карайлы:

```
def rectangle():
    a = float(input('Туурасы: '))
    b = float(input('Бийиктиги: '))
    s = a*b
    print('Аянты: ', s)
def triangle():
    a = float(input('Негизи: '))
    h = float(input('Бийиктиги: '))
    s = 0.5*a*h
    print('Аянты: ', s)
    figure = input('1-тик бурчтук, 2-тик бурчтук:')
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()
```

Бул маселеде 5 өзгөрмө бар, анын ичинде `figure` гана глобалдуу `rectangle()` функциясындагы `a` жана `b` жана `triangle()` функциясындагы `a` жана `h` өзгөрмөлөрү – локалдык. Ошону менен бирге ар кайсы функциядагы локалдык өзгөрмөлөр – ар башка өзгөрмөлөр.

Функциядан маанилерди кайтаруу

Ар бир функция белгилүү бир жыйынтыкты берет. А түгүл силер жыйынтык катары маанисин кайтарууну көрсөтпөсөңөр да, ал баары бир `None` (эч нерсе) деген жыйынтыкты берет.

Функциянын мааниси эмнеге барабар экендигин көрсөтүү үчүн артынан маани-жыйынтыгы жазылган **return** (англ. *кайтаруу*) операторун колдонушат.

Жыйынтыгы сан, символ, символдук сап же каалагандай башка объект болушу мүмкүн.

2-маселе. Сандын цифраларын кошууну эсептөөчү функцияны түзөлү (мисалы, 147 саны үчүн сандарды кошуу керек: $1+4+7=12$). Цифраларды кошууну акыркысынан баштайлы, биздин мисалда бул 7 цифрасы.

- 1 Сандын акыркы цифрасын алуу үчүн, санды 10го бөлгөндөгү калдыгын алуу керек ($147 \% 10 = 7$).
- 2 Алынган калдыкты баштапкы мааниси нөлгө барабар болгон «суммага» ($sum = 0$) кошобуз. Эми сумма 7ге барабар.
- 3 Андан соң биз бүтүн сандык бөлүү операторун колдонуп, сандын акыркы цифрасын «бөлүп салабыз» ($147 // 10 = 14$).
- 4 $14 > 0$ болгондуктан биз циклдин башына кайрылабыз. Цикл n мааниси нөлгө барабар болгонго чейин уланат.
- 5 Мындан кийин кайрадан 4 цифрасын бөлүп салабыз ($14 \% 10 = 4$) жана аны суммага кошобуз ($4 + 7 = 11$).
- 6 Аны бардык сандан бөлүп алабыз ($14 // 10 = 1$).
- 7 Акыркы бир орундуу санды суммага кошобуз ($11 + 1 = 12$).

Жыйынтыгы: 12

Ушундай жол менен биздин программа төмөнкүдөй жазылат:

```
n = int(input('Санды киргизиңиз: '))
def digits_sum (n):
    total = 0
    while n > 0:
        total += n%10
        n = n // 10
    return total
#негизги программа
print (digits_sum(n))
```

3-маселе. Берилген сандын цифраларынын суммасы 3кө бөлүнөөрүн же бөлүнбөсүн аныктоочу программаны түзөлү. Ошондой эле эгер алынган сумманын саны бир цифрадан көп болсо, анда сумма бир цифрадан турганча аракетти кайталатуу керек.



Мисалы, 123456789 санынын цифраларынын суммасы 45ке барабар. Сан эки орундуу, демек дагы цифралардын суммасын табабыз: $4 + 5 = 9$. 3кө бөлүнгөн 9 санын алдык ($9\%3==0$). Демек баштапкы 123456789 саны 3кө бөлүнөт. Бул алгоритмден төмөнкү программага келебиз:

```
def sum(n):
    sum = 0
    while n>0:
        sum += n % 10
        n = n // 10
    return sum

#негизги программа
k = int(input('Санды киргизиңиз: '))
while k > 9: #цифралардын суммасы бир цифра болгонго чейин
    k = sum(k) #функцияны чакырабыз
if k%3==0:
    print ('Сан 3кө бөлүнөт')
else:
    print ('Сан 3кө бөлүнбөйт')
```

Функцияларды бир гана негизги программдан эле эмес башка функциялардан да чакырса болот. Мисалы, үч ар түрдүү сандардын ортосундагысын (б.а. эки сандын ортосунда жайгашкан санды) табуучу функция мындай аныкталышы мүмкүн:

```
def middle ( a, b, c ):
    mi = min ( a, b, c )
    ma = max ( a, b, c )
    return a + b + c - mi - ma
```

Программа `min` жана `max` даяр функцияларын колдонду. Муну чыгаруунун идеясы, эгер үч сандан максималдык жана минималдык санды кемитип салса, так ошол үчүнчү сан калат дегенде.

Функция бир нече маанини кайтарышы да мүмкүн. Мисалы эки санды бөлүүдөн алынган тийиндини да, калдыкты да чыгаруучу программа төмөнкүдөй жазылат:

```
def divmod ( x, y ):
    d = x // y
    m = x % y
    return d, m
```

Функциянын жыйынтыгын эки ар түрдүү өзгөрмөлөргө жазса болот:

```
a, b = divmod ( 7, 3 )
print ( a, b ) #2 1
```

Эгерде бир эле өзгөрмөнү көрсөтсөк, биз кортежди алабыз – тегерек кашаага алынган элементтердин катары:

```
q = divmod ( 7, 3 )
print ( q ) #(2, 1)
```

Кээде программада бир эле жолу колдонулуп жана бир нече аргументи менен анча татаал эмес аракеттерди аткарган функцияны түзүү үчүн **lambda-функцияларды** колдонушат. lambda-функция бул анонимдик функция, б.а. **def** сыяктуу өзүнүн атына ээ эмес. Функциянын жазылышы **lambda** сөзүнөн башталат жана бош орундан кийин функциянын аргументтери көрсөтүлөт. Андан соң кош чекиттен кийин жыйынтыгы функцияда кайтарылган амалдар көрсөтүлөт. Мындай функцияны эки санды көбөйтүү мисалында карайлы:

```
multiple = lambda x, y: x * y #2 аргументи менен lambda-функция
print (multiple (2, 5)) #жыйынтык 10
```

Жогорку мисалда биз lambda-функцияны **multiple** өзгөрмөсүнө ыйгардык. Бирок мындай функцияны бир сап менен эле өзгөрмөнү колдонбостон да жазсак болот:

```
print ((lambda x, y: x * y)(2, 6))
```

Программалоодогу негизги көндүмдүрдүн бири – бул бир эле кодду бир нече жолу жазбоо болуп саналат. Качан код кайталана баштаса, бул кайталанган коддун бөлүгүн функцияга айлантуу керектигин түшүнүш керек.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) 3 берилген санды өсүү тартибинде экранга чыгаруучу функцияны жазгыла.
- 2) Эки натуралдык сандын эң чоң жалпы бөлүүчүсүн табуучу функцияны жазгыла.
- 3) Сандардын төмөнкүдөй белгилерин чыгаруучу функцияны жазгыла: кайтарылган жыйынтыгы менен аргументи бүтүн сан болуш керек, 0 – эгерде аргумент 0 болсо, -1 эгер сан терс болсо, 1 – эгер сан оң болсо.



3.6-тема:

Массивдер

Азыркы компьютерлердин эң негизги кызматы – бул көп көлөмдөгү маалыматты иштетүү. Ошону менен бирге маалымат сакталган миндеген (же миллиондогон) уячанын ар бирине кайрылып туруш керек. Мындай учурларда ат бир уячага эмес, ар бир уяча өзүнүн номерине ээ болгон уячаларын тобуна берилет. Эстин мындай аймагы массив деп аталат.

Массив – бул жалпы атка ээ болгон эсте жакын жайгашышкан (кошуна уячаларда) өзгөрмөлөрдүн тобу. Массивдеги ар бир уяча уникалдуу номерге (индекске) ээ.

Python тилинде бир өлчөмдүү массивдер элементтердин тизмеси түрүндө болот. Ошондуктан массивдер менен иштөө үчүн тизмелерди колдонушат (берилиштер тиби list). Python тилинде тизме – бул ар бири өзүнүн номерине (индекс) ээ болгон элементтердин жыйындысы. Номерлөө ар дайым нөлдөн башталат, катары боюнча экинчи элемент 1 номерине ээ ж.б.

Тизмени түрдүү жолдор менен түзсө болот. Анын эң жөнөкөй ыкмасы – элементтердин тизмесин үтүр менен ажыратып, чарчы кашаанын ичинде жазуу:

```
a = [1, 3, 4, 23, 5]
```

Тизме – бул динамикалык структура, анын өлчөмүн программанын иштөө убагында өзгөртсө (элементтерди өчүрсө же кошсо) болот.

Тизмелерди «+» белгисинин жардамында «кошсо» болот. Ошентип жогорку мисалды мындай жазса да болот:

```
a = [1, 3] + [4, 23] + [5]
```

Бирдей тизмелерди кошуу көбөйтүү менен алмаштырылат «*». Нөлдөр менен толтурулган 10 элементтен турган тизме мындай түзүлөт:

```
a = [0]*10
```

Мындан татаал учурларда тизмелердин генераторлору колдонулат. Анда жаңы түзүлгөн тизменин элементтери циклди колдонуу менен толтурулат:

```
a = [i for i in range(10)]
```

ОШОНДОЙ ЭЛЕ КАРА

3.2-тема 8-класс

Тизмелер, кортеждер,
сөздүктөр

Силер билгендей `for i in range(10)` цикли Одөн 9га чейинки `i` нин бардык маанилерин коюп чыгат. `for` сөзүнүн алдындагы туюнтма (биздин учурда – `i`) – бул ар бир `i` үчүн тизменин улам кийинки элементине жазыла турган өзгөрмө. Берилген мисалда тизме `i` өзгөрмөсү удаалаш алып жаткан маанилер менен толтурулат:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Ушуну эле, `range` функциясынын жардамы менен алынган берилиштер менен тизме түзүү үчүн `list` функциясын колдонуп алсак болот:

```
a = list (range(10))
```

Тизмени бул сандардын квадраттары менен толтуруу үчүн мындай генераторду колдонсо болот:

```
a = [i*i for i in range(10)] #[0,1,4,9,16,25,36,49,64,81]
```

Генератордун жазуусунун аягында тандап алуу шартын кошуп койсо болот. Бул учурда тизмеге ушул шартты канааттандырган циклде тандалып алынган элементтер гана камтылат.

Мисалы, төмөнкү генератор Одөн 9га чейинки диапазондогу бардык жуп сандардын тизмесин түзөт:

```
a = [i for i in range(10) if i%2 ==0] #[0,2,4,6,8]
```

Көпчүлүк учурда тесттик жана окуу программаларында массивди кокустук сандар менен толтурушат. Муну генератордун жардамында кылса болот:

```
from random import randint  
a = [randint(20,100) for x in range(10)]
```

Мында 10 элементтен турган массив түзүлөт жана `[20, 100]` диапазонундагы кокустук сандар менен толтурулат. Ал үчүн `random` модулуна импорттолгон `randint` функциясы колдонулат.

Тизменин узундугу (андагы элементтердин саны) `len` функциясы менен аныкталат. Мисалга:

```
a = [1, 3, 4, 23, 5]  
n = len(a) #5
```



Мындан ары бардык мисалдарда биз n элементтен (бүтүн сандардан) турган a тизмеси массив турүндө түзүлдү деп эсептейбиз. i өзгөрмөсү тизменин элементинин индексин түшүндүрөт.

Циклди пайдалануу менен 3 элементтен турган тизмени түзүү үчүн төмөнкү программаны жазабыз:

Python тилиндеги программа

```
n = 3
a = [0]*3
for i in range(n):
    print('a[' + i + ']= ', sep='', end='')
    a[i] = int(input())
```

Экранга чыккан жыйынтыгы

```
a[0] = 1
a[1] = 2
a[2] = 3
```

n элементинен турган массив түзүп, анын маанилерин киргизүүнү тизме генераторунун жардамында жүргүзсө да болот.

```
a = [int(input()) for i in range(n)]
```

Бул жерде колдонуучу `int` функциясынын жардамында бүтүн санга өзгөртүлүп түзүлгөн n сандагы элементтерди киргизиш керек жана бул сан массивге кошулат.

Киргизүүнүн дагы бир варианты бар, мында бардык массив бир эле сапта киргизилген. Бул учурда `input` функциясынан алынган сапты `split` ыкма-сынын жардамы менен бөлүктөргө ажыратуу керек:

```
data = input()
s = data.split()
print (s)
```

Тизмени чыгаруу үчүн `print` функциясын колдонобуз. Тизменин ар бир элементин чыгаруу үчүн анын индексин чарчы кашаада жана бирден элементти сапта көрсөтүү менен төмөнкү программаны пайдаланабыз:

Python тилиндеги программа

```
a = [1, 2, 3, 4, 5];
for i in range(0, len(a)):
    print('a[' + i + ']= ', a[i], sep='')
```

Экранга чыккан жыйынтыгы

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```


Элементтерди иргөө

Тизменин элементтерин иргөө, биз тизменин бардык элементтерин циклде карап жана керек болсо анын ар бири менен кандайдыр бир аракеттерди жасайбыз дегенди түшүндүрөт. Циклдин өзгөрмөсү 0 ден $n-1$ ге чейин өзгөрөт, мында n – тизменин элементтеринин саны:

```
for i in range(0,n):  
    a[i] += 1
```

Бул мисалда **a** тизмесинин бардык элементтери 1 ге көбөйөт.

Эгерде тизмени өзгөртүүнүн кереги жок болсо, анда анын элементтерин иргөө үчүн мындай циклди колдонуу ыңгайлуу:

```
a = [1,2,3,4,5]  
for x in a:  
    print(x)
```

Мында **print(x)** тин ордуна «**x**» өзгөрмөсүнө жазылган элементи менен иштеген каалагандай башка операторлорду колдонсо болот. Көңүл бурсаңар, циклдин тулкусундагы «**x**» өзгөрмөсүнүн өзгөрүшү катаны берет.

Көптөгөн маселелерде берилген шартты канааттандырган тизмедеги бардык элементтерди табуу жана аларды иштетүү талап кылынат. Мындай маселелердин эң жөнөкөйү – керектүү элементтерди эсептөө болуп саналат. Бул маселени чечүү үчүн баштапкы мааниси нөл болгон өзгөрмө-эсептегичти киргизүү керек. Андан соң циклде тизменин бардык элементтерин карайбыз. Эгерде каралып жаткан элементте берилген шарт аткарылса, анда эсептегичти бирге көбөйтөбүз.

a массивинде класстагы балдардын бою жөнүндө маалымат жазылган дейли. Бойлору 120 см ден чоң бирок 150 см ден кичине болгондордун санын аныктайлы. Төмөнкү программада **count** өзгөрмө-эсептегичи колдонулду:

```
count = 0  
for x in a:  
    if (120 < x < 150):  
        count += 1
```

Эми маселе кичине татаалдансын: балдардын орточо боюн табуу керек. Ал үчүн кошумча өзүнчө бир өзгөрмөдө бардык керектүү маанилерди кошуу, ал эми циклдин аягында бул сумманы бардык маанилердин санына бөлүү керек. Сумма жыйналган **sum** өзгөрмөсүнүн баштапкы мааниси да нөлгө барабар болуш керек:



```
count = 0
sum = 0
for x in a:
    if 120 < x < 150:
        count += 1
        sum += x
print (sum / count)
```

Тизменин элементтерин суммалоо – бул көп таралган амал, ошондуктан Python тилинде элементтерди суммалоо үчүн атайын `sum` деген орнотулган функция бар:

```
a = [1,2,3,4,5]
print (sum(a))
```

Анын жардамында мурунку маселени бир аз тыканыраак чечсек болот: биринчиден кошумча массивге бардык керектүү элементтерди бөлүп алуу, андан кийин алардын суммасын жалпы санына (тизменин узундугуна) бөлүп коюу керек.

Жаңы тизмени түзүү үчүн шарттуу операторду колдонобуз:

```
b = [x for x in a if 120 < x < 150]
print (sum(a)/len(a))
```

Тандоо шарты **a** тизмесинен **b** тизмесине шартты канааттандырган гана элементтерди берди. Ал эми класстагы балдардын бойлорунун орточосун чыгаруу үчүн жаңы тизменин элементтеринин суммасын алардын санына бөлүп коюу жетиштүү.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) Тизмеде окуучулардын баалары сакталган. Клавиатурадан киргизген бааларга барабар болгон тизменин элементтеринин номерлерин чыгаруучу программаны түзгүлө.
- 2) Дене тарбия сабагында окуучулардын бойлорун тизмеге жазышты. Бул тизмеден эң бийик жана эң жапыз бойлуу окуучуну табуучу программа түзгүлө.

3.7-тема:

Саптар жана алар менен болгон амалдар

Адегенде компьютерлер эсептөөчү машиналар катары колдонулган, азыр болсо алардын негизги кызматы тексттик маалыматтарды иштетүү болуп бара жатат. Python тилинде текст менен иштөөчү негизги тип – бул саптар (**str** тиби) англ. **string**.

Сап – бул бир же кош тырмакчанын ичине алынган символдордун удаалаштыгы: **'Бул сап' = "Бул сап"**

Тизмелерден (массивдерден) айырмаланып, саптар берилиштердин структурасына кирбейт. Ошону менен бирге саптарды иреттелген элементтердин удаалаштыгы катары карап, алар менен тизменин элементтериндей эле иштесе болот.

```
>>> s = 'Бул сап'
>>> s[0] #көрсөтүлгөн индекси менен элементтерди кайтарат
'Б'
>>> s[5:] #5-индекстен акыркысына чейинки элементтер
'ап'
```

Бирок Python тилинде саптар өзгөртүлбөйт. Башкача айтканда берилген саптын кайсы бир бөлүнгөн элементин башкасына алмаштырууга болбойт. Мындай учурда программа ката деп чыгарат. А бирок берилген саптын символдорунан керектүү өзгөртүүлөрдү киргизип **жаңы сапты** түзсө болот.

Сапты клавиатурадан киргизип, андагы бардык «а» тамгаларын «б» тамгаларына алмаштырып, экранга чыгаруунун толук программасын көрөлү:

```
s = input('Сапты киргизиңиз: ')
s1 = ''
for c in s: #s сабындагы бардык символдорду иргеп чыгат
    if c == 'a': #өзгөрмөнүн мааниси «а» га дал келсе
        c = 'б' #анда аны «б» тамгасына алмаштырабыз
    s1 = s1 + c
print (s1)
```

Бирок бул ыкма өтө жай иштейт. Символдорду алмаштырыш керек болгон практикалык маселеде эң жакшысы даяр **replace** методун колдонуу керек.



Мындан тышкары көпчүлүк учурда тексттин сабын иштетүү керек болот: бул саптын бөлүгүн (сапча) алуу, эки сапты бир сапка бириктирүү, же саптын бөлүгүн өчүрүү. Мисалы, саптарды **бириктирүү** (чиркештирүү) үчүн «+» оператору колдонулат. Бул амал конкатенация деп аталат:

```
s1 = 'Кутман'
s2 = 'күн'
s = s1 + ' ' + s2 + '!'
print (s)
>>>
Кутман күн!
```

**ЭСИҢЕ TUT**

Саптын узундугу саптын касиети len дин (length) жардамы менен аныкталат. Ал үчүн n өзгөрмөсүнө бош орундары менен кошуп саптагы белгилердин санын (бүтүн сан) берүүчү s сабы жазылат:

n = len(s)

Саптарды иштетүү үчүн үзүмдөрдү колдонуу

Python тилинде көп учурда саптын аныкталган бөлүгүн иштетүү үчүн белгилеп алган үзүмдөр (англ. slicing) колдонулат. Үзүмдөр мындайча жазылат: [X:Y]. X – үзүмдүн башталышы, Y – үзүмдүн аягы. Y индексиндеги символ үзүмгө кирбейт. Алгач көрсөтүлбөсө, биринчи индекс 0гө, ал эми экинчиси – саптын узундугуна барабар.

Мисалы, s[3:8], s сабындагы 3-символдон 7-символго чейин (б.а. 8-ге чейин бирок 8 өзү кирбейт) үзүмдү түшүндүрөт.

Мааниси

s = 'китепканачы' сапчасы үчүн мисалдар

Саптын бөлүгүн (сапча) белгилеп алуу үчүн

```
s1 = s[2:8]
#s1 сабына 3-элементтен баштап 8-элементи
кошулган маани жазылат
>>> тепкан
```

Саптын бөлүгүн өчүрүү үчүн

```
s1 = s[:3] + s[9:]
#башынан баштап 3-элементке чейин, жана
9-элементтен аягына чейин кесип алабыз
жана аларды s1 сабында сактайбыз
>>> китагы
```

Саптын ичине жаңы фрагмент коюу

```
s1 = s[:3] + 'ABC' + s[3:]
#3-элементтен кийин ABCны кошуп кетебиз
>>> китABCепканачы
```

Сапты реверстөө (аны тескерисинче буруу)

```
s1 = s[::-1] #сөздү тескери жазып чыгарат
>>> ычанакпетик
```

Берилген кадам аркылуу элементтерди тандоо

```
s1 = s[::2]
#ар бир экинчи символду кайтарат
>>> ктпааы
```

Саптар методу

Python тилинде саптар менен иштөө үчүн көптөгөн камтылган методдор бар. Алардын ичинен кызыктууларын карап көрөлү.

1 upper жана **lower** методдору сапты тиешелүү түрдө жогорку жана төмөнкү регистрлерге өткөрөт. **title** методу болсо биринчи эле тамгаларды жогорку регистрге, калганын төмөнкү регистрге өткөрөт:

```
s = 'aAbB cC'
s1 = s.upper() # 'AABB CC'
s2 = s.lower() # 'aabb cc'
s2 = s.title() # 'Aabb Cc'
```

2 split методу сапты бош орундар боюнча бөлүүгө мүмкүндүк берет. Жыйынтыгында сөздөрдөн тизме алынат. Эгерде колдонуучу программада ар бири өзүнчө тизмедегидей иштетилсин деп, бир сапта катар сөздөрдү же сандарды киргизүү керек болсо, анда split методун колдонсо болот:

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.split() # ['Дүйшөмбү', 'Шейшемби', 'Шаршемби', 'Бейшемби', 'Жума']
```

3 join методу тескерисинче тизмеден сапты курайт. Ал үчүн алдына сап-бөлгүч коюлат, ал эми кашаанын ичинде тизме берилет:

```
s = ['Дүйшөмбү', 'Шейшемби', 'Шаршемби', 'Бейшемби', 'Жума']
s1 = '-'.join(s) # Дүйшөмбү-Шейшемби-Шаршемби-Бейшемби-Жума
```

4 find методу саптын бөлүгү (сапча) менен иштейт. Ал саптагы сапчаны издейт жана табылган сапчанын биринчи элементинин индексин кайтарат. Эгерде сапча табылбаса анда 1ди кайтарат.

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.find('Шаршемби') # жообу: 18, сапчанын 1-элементи - «Ш» тамгасынын индекси
```

Мындан тышкары бул метод менен берилген үзүмдөгү саптын элементинин индексин табууга болот. Үзүмдү көрсөтүү үчүн, анын башталышын жана аягын үтүр менен ажыратылган цифралар менен берүү керек. Эгерде экинчи цифра көрсөтүлбөсө, анда издөө саптын аягына чейин жүргүзүлөт:

```
s2 = s.find('ү', 4) # 8, 4-индексден аягына чейинки бөлүгүндөгү биринчи «ү» нүн индекси.
```



`s1 = s.find ('ү', 0, 9) #2, башынан 9-индекске чейинки бөлүгүндөгү биринчи «ү» нүн индекси`

5 `replace` методу бир сапчаны башкасына алмаштырат. Бул учурда баштапкы сап өзгөргөйт, болгону жаңы сапка модификацияланат (өзгөртүп түзүлөт). Ал болсо жаңы `s1` өзгөрмөсүнө ыйгарылат:

```
s = 'Дүйшөмбү Шейшемби Шаршемби Бейшемби Жума'
s1 = s.replace ('б', 'в') #ДүйшөмВү-ШейшемВи-ШаршемВи-Бей-
шемВи-Жума
```

Кээде бизге сапты бөлүш керек болот. Бөлүнгөн бөлүгү жаңы саптан башталгыдай кылып, бул учурда биз `\n` белгисин колдонобуз.

```
print ('Дүйшөмбү \n Шейшемби \n Шаршемби \n Бейшемби \n
Жума') #бардык сөздөр мамыча түрүндө чыгат.
```

Эгерде жаңы сапты жылдыруу менен чыгаруу керек болсо, анда `\t` белгисин колдонобуз.

```
print ('Дүйшөмбү \n\t Шейшемби \n\t Шаршемби \n\t Бейшемби
\n\t Жума') #бардык сөздөр мамыча түрүндө чыгат, ар бир кийин-
ки саптын алдына орун ташталат.
```

Жогорудагы үйрөнгөн командаларды колдонуп, сапты иштетүүнүн мисалын карап көрөлү.

1-маселе. Клавиатурадан атын, фамилиясын жана атасынын атын камтыган сап киргизилет, мисалы:

Айтматов Чыңгыз Төрөкулович

Ар бир эки сөз бири-биринен бош орундар менен ажыратылган, саптын башында бош орун жок. Бул сапты иштетүүнүн натыйжасында фамилия жана инициалдарын эле камтыган жаңы сап пайда болуш керек:

Айтматов Ч. Т.

Чыгаруу:

1 Сапты клавиатурадан киргизебиз:

```
s = input('фамилия, атыңыз жана атаңыздын атын киргизиңиз: ')
```

2 Киргизилген сапты бош орун менен ажыратылган өзүнчө сөздөргө бөлүп чыгабыз. Ал үчүн `split` методун колдонобуз. Бул массивде үч элемент болот: `fio[0]` – фамилиясы, `fio[1]` – аты, `fio[2]` – атасынын аты:

```
fio = s.split()
```

3 Инициалдары менен фамилияны чогултабыз:

```
fioshort = fio[0]+' '+fio[1][0]+'.'+ fio[2][0]+ '.'
```

Толук программа мындай көрүнөт:

```
s = input('Фамилия, атыңыз жана атаңыздын атын киргизиңиз: ')
fio = s.split ()
fioshort = fio[0] + ' ' + fio[1][0] + '.' + fio[2][0] + '.'
print (fioshort)
```

Саптарды салыштыруу жана сорттоо

Биз алфавит боюнча сорттоону сөздү тезирээк табуу үчүн колдонобуз. Эгерде сөздүктөрдө сөздөр алфавит менен жайгашпаганда, анда биз бир сөздү издөө үчүн эле бир топ убакыт коротмокпуз. Python тилинде саптарды сорттоо кандай принцип менен түзүлгөн?

Көрсө, сандар сыяктуу эле тамгалар да өзүнүн салмагына ээ экен. Алфавитте биринчи турган тамга жеңилирээк, б.а. «а» «б» га караганда жеңил, ошондуктан сорттоодо ал биринчи чыгат. Мындан саптарды да сандар сыяктуу эле салыштырууга болот деген жыйынтык келип чыгат.

Сөздөрдү салыштырууда, алгач биринчи тамгалары салыштырылат, эгерде алар айырмаланса, анда салыштыруунун жыйынтыгы аныкталат. Андан кийинки тамгалар салыштырылбай калат. Ал эми эгерде биринчи тамгалары барабар болсо, анда кийинки 2 элементи салыштырылат жана ушул сыяктуу аягына чейин кетет. Мисалы «паровоз» сөзү «пароход» сөзүнө караганда кичине: алар 5-тамгада айырмаланышат жана «в» < «х».

Эгерде силерде текшерүү үчүн символдор бүтүп калса, анда кыска сап узун сапка караганда кичине, ошондуктан «пар» < «парк». Баш тамгалар кичине тамгадан жеңил, анткени сөздүн башында турушат.

Бирок компьютер «алфавиттик тартипти» кайдан «билет»? Көрсө, саптарды салыштырууда символдордун ASCII жана Unicode коддору колдонулат экен. Демек:

«ПАР» < «Пар» < «пар»

«ПАР» жана «Пар» деген сөздөрдү салыштыралы. Биринчи символ эки сөздө тең бирдей, ал эми экинчиси айырмаланат – биринчи сөздө баш тамга, ал эми экинчисинде ошол эле тамга, бирок кичине. Символдор таблицасында баш тамгалар кичине тамгадан биринчи турушат, ошондуктан кичине коддорго ээ. Ошондуктан «А» < «а», «П» < «п» жана «ПАР» < «Пар».



Ал эми башка символдор (цифралар, латын тамгалары) менен кандай болот? Коддук таблицада цифралар ирети менен жайгашкан жана латын тамгаларынан мурун турушат; латын тамгалары – орус тамгаларынан мурун, орус жана латын тамгаларынын баш тамгалары тиешелүү тилдердин кичине тамгаларынан мурун турат. Ошондуктан

«5STEAM» < «STEAM» < «Steam» < «steam» < «ПАР» < «Пар» < «пар»

Мисалы, сөздүн ичиндеги тамгаларды сорттоо үчүн, программаны мындай жазуу керек:

```
s = 'Дүйшөмбү'
s1 = ''.join(sorted(s))
print(s1)
>>> Д б й м ш ү ү ө
```

1-маселе. Клавиатурадан бир нече сөздөрдү (мисалы фамилияларды) киргизүү жана аларды экранга алфавиттик тартипте чыгаруу керек.

Бул маселени чыгаруу үчүн, үтүр менен ажыратылып киргизилген фамилияларды тизмеге кайра жазып алуу ыңгайлуу, андан соң sorted методу менен сорттоп коюу керек:

```
s = input('Фамилияңызды киргизиңиз: ') #Абакиров Муканова
Бебинов Семенова Запруда
s1 = s.split() #сапчалар менен тизме түзөт ['Абакиров', 'Му-
канова', 'Бебинов', 'Семенова', 'Запруда']
s2 = ' '.join(sorted(s1))
print(s2)
>>> Абакиров Бебинов Запруда Муканова Семенова
```

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Клавиатурадан бир нече сөз киргизүүнү жана ошол сөздөрдүн ичинен эң кыска сөздүн узундугун аныктоочу программа түзгүлө.
- 2) isdigit() саптык методу сап жалаң гана цифралардан тураарын текшерип берет. Киргизүүдө программа эки бүтүн санды сураган жана алардын суммасын чыгарып берүүчү программаны түзгүлө. Туура эмес киргизүү учурунда программа ката деп токтоп калбастан, кайрадан эле сандарды сурап тургандай кылыш керек.

3.8-тема:

Саптарды форматтоо

Python тилинде саптарды форматтоо шаблондун кайсы бир ордуна башка текстти коюу дегенди түшүндүрөт. Ордуна коюу ошол замат жүргүзүлөт. Мисалы, форматтоону колдонуу менен даяр шаблон аркылуу мындай чакыруу билеттерин жасоого болот:

Урматтуу Алина!
Сизди чакырабыз: Ачык эшиктер күнүнө.
Окуянын датасы: 1-май
Урматтоо менен Тимур.

Бир жолу ордуна коюу мындай жазылат:

```
print ('Урматтуу {}'.format ('Алина'))
```

Башкача айтканда, шаблондун текстинен кийин бош фигуралуу кашаалар коюлат, андан кийин **.format ()** команданын кашаасынын ичине коюу керек болгон маанилери көрсөтүлөт. Программа ишке киргенде, ордуна коюлуучу текст фигуралуу кашаалардын ордуна коюлат.

Бир нече коюулар болсо, фигуралуу кашаага сөздөрдүн индекстери, ал эми сөздөрдүн өзү **.format**тан кийин кортежде коюлат. Ал шаблон мындай жазылат:

```
print ('Урматтуу {0}! \n Сизди чакырабыз: {1}.\n Окуянын датасы: {2} \n Урматтоо менен {3}.'.format ('Алина', 'Ачык эшиктер күнүнө', '1-май', 'Тимур'))
```

Текстти **.format**ты колдонбостон башка ыкма менен да форматтаса болот. Бул ыкма бир аз туура эмес жана эскирип калган деп эсептелет. Бирок, эгер силер коддон % белгисин кезиктирсеңер, анда бул жерде форматтоо колдонулду деп эсептесеңер болот. Биздин чакыруу шаблону бузду % операторунун жардамында жазып көрөлү:

```
print ('Урматтуу %s! \n Сизди чакырабыз:%s.\n Окуянын датасы: %d %s \n Урматтоо менен %s.' % ('Алина', 'Ачык эшиктер күнүнө', '1-май', 'Тимур'))
```

```
>>>
```

```
Урматтуу Алина!  

Сизди чакырабыз: Ачык эшиктер күнүнө.  

Окуянын датасы: 1-май  

Урматтоо менен Тимур.
```



Силер байкадыңарбы, кээ бир жерде биз %d деп, кээ бир жерде %s деп жаздык? Алар ордуна коюуга эмнени колдонуп жатканыбызды аныктайт:

%s – сапты коёт;

%d – бүтүн санды коёт;

%f – бөлчөк санды коёт.

Санды сапка жана сапты санга өзгөртүп түзүү

Практикалык маселелерде көбүнчө символдордун катары түрүндө жазылган сандарды сандык мааниге жана тескерисинче айландырууга туура келет. Бул үчүн Python тилинде атайын стандарттык функциялар бар:

int – сапты бүтүн санга айландырат

```
s = '123'  
N = int ( s ) #N = 123
```

float – сапты чыныгы санга (бөлчөк) айландырат

```
s = '123.456'  
X = float ( s ) #X = 123.456
```

str – бүтүн жана бөлчөк сандарды сапка айлантат

```
N = 123  
s = str ( N ) #s = '123'  
X = 123.456  
s = str ( X ) #s = '123.456'
```

Эгерде сапты санга айландыра албай калса (мисалы, сапта тамгалар камтылса), анда ката пайда болот да программа аяктайт.

Бөлчөк сандар үчүн (float тиби), бөлчөк бөлүгүнөн канча белгини чыгарууну көрсөтүп койсок болот, мисалы:

```
x = 34.8589578  
print ( '{:.2f}'.format (X) ) #34.86  
print ( '{:.3f}'.format (X) ) #34.859
```

Эгерде чоң сандарда үтүрдү биз разряддарды бөлүү үчүн колдонгубуз келсе, анда мындай жазабыз:

```
print ( '{:,.2f}'.format (10001.23554) ) #10,001.24
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

Колдонуучудан логин жана паролду сураган шаблонду жазгыла. Эгерде логин же пароль туура эмес болуп калса, «мындай логин жана пароль табылган жок» деген билдирүүнү чыгарсын. Ал эми логин жана пароль туура болсо, анда колдонуучунун атын көрсөткөн саламдашууну чыгаргыла.

3.9-тема:

Python тилинде графика менен иштөө

Turtle модулунун жардамында сүрөт тартуу

Python программалоо чөйрөсүндө жөнөкөй сүрөт тартуу үчүн Turtle (ташбака) модулу колдонулат. Ал төмөнкү команда менен кошулат:

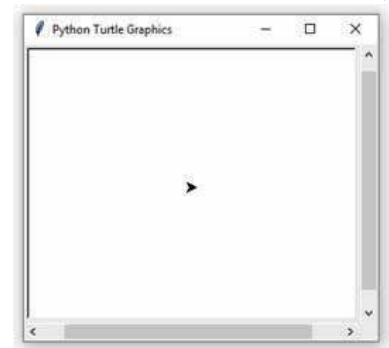
```
from turtle import* #ташбака менен иштөөчү командаларды жүктөйт
reset () #позицияны нөлгө айлантат жана калемди күйгүзөт
```

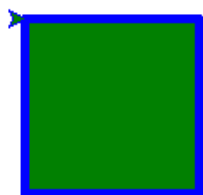
Программаны ишке киргизүү менен силер борборунда калем (ташбака) жайгашкан графика үчүн терезени көрө аласыңар. Мисалы, **forward** командасы ташбаканы алдыга жылдырат. Кашаанын ичинде пиксель менен кадамдардын саны көрсөтүлөт. **right** жана **left** бурулуу командалары үчүн кашаанын ичинде бурууга керек болгон градустардын саны көрсөтүлөт. Эми квадратты тартып көрөлү:

```
from turtle import*
reset ()
forward (100) #алдыга карай 100 пик-
селге кыймылда
right (90) #90°ка оңго бурул
forward (100)
right (90)
forward (100)
right (90)
forward (100)
```

Кодду кыскартуу үчүн **for** циклин жана фигурга дагы кошумча параметрлерди берели:

```
from turtle import*
width (5) #сызыктын жоондугу - 5 пиксель
color('blue','green') #1-түс - сызыктын, 2-түс - боёктун түсү
begin_fill() #аймакты толтуруу үчүн ташбакага байкоо жүргүзүү
for i in range (4):
    forward (100)
    right (90)
end_fill() #begin_fill()ден баштап, аймакты түс менен толтуруу
>>>
```





Up() командасын колдонуп, калемди көтөрүп, башка орундан баштап сүрөттү (же текстти) тартып баштасаңар болот. Башка сүрөттү тартуудан мурун кайра калемди **down()** командасы менен түшүрүү керек.

Айлананы тартуу үчүн **circle(r,n)** командасы колдонулат, мында r – айлананын радиусу, n – биз тарткан айлананын бөлүгү, градус менен. Эгер $n = 180$ градус болсо, анда калем жарым айлана сызат, $n = 360$ градус болсо, толук айлана сызат.

Чекитти тартуу үчүн **dot(r, color)** командасы колдонулат, мында r – чекиттин радиусу (пиксель менен), $color$ – чекит тартыла турган түс.

1-маселе. Айлана чиели, анын узундугу боюнча берилген сандагы чекиттер бирдей бөлүштүрүлүп жайгашсын:



```
from turtle import*
def circ(d, r, rBig): #параметрлери менен circ
    функциясы: чекиттердин саны, чекиттин радиусу, айлананын радиусу
    for i in range(d):
        circle(rBig, 360 / d) #чекиттердин санын айлана боюнча
        бөлүштүрөбүз
        dot(r, 'red')
up()
goto(150, 0) #калемди 150 пикселге оңго жылдырабыз
setheading(90) #калемди 90° ка бурабыз
down() #калемди сүрөт тартууну баштоо үчүн түшүрөбүз
circ(15, 10, 150) #параметрлерге маанилерин беребиз
screen.mainloop() #программанын аткарылышын токтотобуз
```

Фигураларды чыгаруудан башка графикалык терезеде текстти да тартса болот. Ал үчүн параметрлери менен **write()** командасы колдонулат:

```
write(text, move, align, font = (fontname, fontsize, fontstyle))
```

- **text** параметрине тырмакчага алынып текст өзү жазылат;
- **align** параметри «left», «right», «center» маанилерин алат жана тексттин абалын ташбакага салыштырмалуу өзгөртөт, маанилери тырмакчада жазылат;
- **font** параметри fontname, fontsize, fontstyle маанилерин алат:
 - **fontname**ге тырмакчада шрифттин аты жазылат;
 - **fontsize** шрифттин өлчөмү үчүн жооп берет;
 - **fontstyle** тексттин стилине жооп берет (**normal** – кадимки, **bold** – кара, **italic** – курсивдүү, **bold italic** – кара, курсивдүү текст).

2-маселе. Форматталган текстти чыгаруучу программа түзөлү:

```
from turtle import*
color ('blue')
write('Биз Python тилин үйрөнөбүз', 'center', font=('Roboto', 24, 'bold italic'))
>>>
```



clear() жана **reset()** командалары сүрөттөрдөн экранды тазалайт жана калемди борборго жайгаштырат.

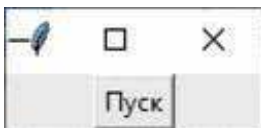
Графикалык объекттерди түзүү үчүн Tkinter менен иштөө

Tk графикалык китепканасы менен иштөө үчүн Tkinter модулу жогорку деңгээлдеги графика жана анимация менен иштөөгө мүмкүндүк берет. Turtle модулу сыяктуу эле tkinter модулун да кошуп алабыз:

```
from tkinter import*.
```

3-маселе. Алгач кадимки эле баскычты (кнопка) түзүп көрөлү. Ал үчүн Button виджетин колдонобуз, кашаанын ичинде баскычтын параметрлерин беребиз. text маанисинин жардамы менен баскычтын атын жазабыз. Төмөнкү программаны иштетели:

```
from tkinter import*
tk = Tk() #tk ат менен Tk
объектисин түзөбүз
btn = Button(tk, text='Пуск')
#«Пуск» тексти менен баскыч
түзөбүз
btn.pack() #баскычты терезе-
нин ичинде жайгаштырабыз
>>>
```



Бирок бул баскычты баскандан эч нерсе чыкпайт. Баскычты басуу кандайдыр бир аракет менен коштолсун үчүн программага жыйынтыгы **command** командасы аркылуу чыга тургандай функция киргизели:



БУЛ КЫЗЫКТУУ!

Виджеттер – бул кээ бирлери бардык программалоо тилдеринде стандарттык болгон, программанын графикалык интерфейсин түзүү үчүн атайын базалык блоктор. Мисалы бул кнопкалардын, желекчелердин же жылдырып көрүү тилкелеринин виджеттери. Алар аттары менен гана айырмаланышы мүмкүн: мисалы, классикалык желекчелер (checkbox), Tkinter де check button деп аталышат.

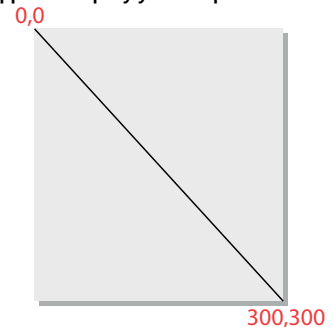
```
from tkinter import*
def btn_act(): #консолдо жыйынтык чыгара турган функция
    print('Оюн башталды!')
tk = Tk()
btn = Button (tk, text='Пуск', command=btn_act) #баскычка
басканда функциядагы билдирүү чыгарылат
btn.pack()
```

Эми баскычка баскан сайын консолдо ушул маалымат чыгып турат:

```
>>> «Оюн башталды!»
```

Башка виджет – Canvas (англ. холст) берилген аянтта сүрөт тартууга мүмкүндүк берет. Сүрөт тартуу холсттун өлчөмдөрүн: холсттун туурасын (width) жана бийиктигин (height) берүү менен башталат.

Холсттогу сүрөттүн баштапкы чекитин белгилөө үчүн X, Y координаталары колдонулат. Координаттар горизонталь боюнча (X) сол четинен, ал эми вертикаль боюнча (Y) жогорку четинен канча пикселге жылышкандыгын аныктайт.



Сызыкты түзүү үчүн **create_line()** методу колдонулат, кашаанын ичинде 4 сан көрсөтүлөт. Биринчи экөө сызыктын башталыш координаталары, кийинки экөө – акыркы координаталары болот. Төмөнкү программаны жазалы:

```
from tkinter import *
tk = Tk()
canvas = Canvas(tk, width=300, height=300)
canvas.pack()
canvas.create_line(0, 0, 300, 300)
```

Айлана сызуу үчүн **create_oval()** методу колдонулат:

```
canvas.create_oval(10, 10, 80, 80, outline= 'red', fill=
'green', width=2)
```

Берилген жазууда биринчи 4 параметри фигураны чектөө координаталарын аныктайт. Б.а., бул ичинде айлана сызыла турган квадраттын жогорку сол бурчтун жана төмөнкү оң жак бурчтарынын x жана y координаталары.

4-маселе. Холстто диаметрлери жана түстөрү кокустук сан менен алынган тегеректерди сызуучу программаны түзөлү.

```

from random import* #random модулуна random.randint жана choice
функцияларын жүктөйбүз
from tkinter import*
size = 500 #size өзгөрмөсүн киргизебиз
tk = Tk()
diapason = 0 #тегеректердин санын чектөө үчүн diapason өз-
гөрмөсүн киргизебиз
my_canvas = Canvas (tk, width=size, height=size) #size өз-
гөрмөсүнүн маанисин колдонуп холст түзөбүз
my_canvas.pack() #холстту терезенин ичине жайгаштырабыз
while diapason <1000: #цикл ушул шартка чейин кайталанат
    color = choice(['green', 'red', 'blue', 'orange',
'yellow', 'pink', 'purple', 'violet', 'magenta', 'cyan'])
#тегеректердин түстөрүн кокустан тандоо үчүн тизме түзөбүз
    x1 = randint(0,size) #x, y коорд-ды кокустан тандоо
    y1 = randint(0,size)
    d=randint(0,size/5) #тегеректердин диаметрлерин каала-
гандай тандоо, бирок size/5 тен чоң эмес
    my_canvas.create_oval(x1,y1,x1+d,y1+d,fill=color) #теге-
ректерди түзөбүз жана кокустан тандал-
ган түс менен ичин боёйбуз
    tk.update()
    diapason+=1 #циклдин кадамы, эсеп-
тегич

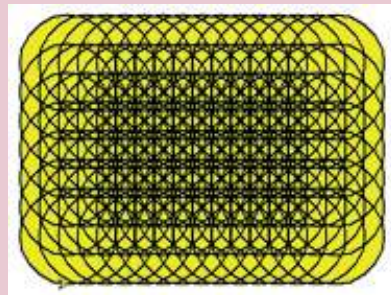
```



КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) Turtle модулуна жардамында айланалардан килем жасагыла. Мында айланалар бир түс менен, ал эми килемдин фону башка түс менен боелсун.

2) Tkinter модулуна жардамында холстко жоондугу жана түсү кокустан тандалган сызыктарды чыгаруучу программаны түзгүлө.



4

- бөлүм



Компьютердик тармактар жана интернет

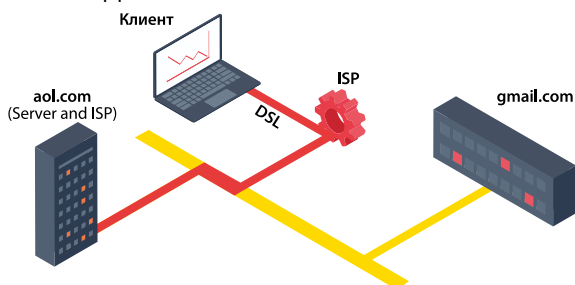
4.1-тема:

Компьютердик тармактар

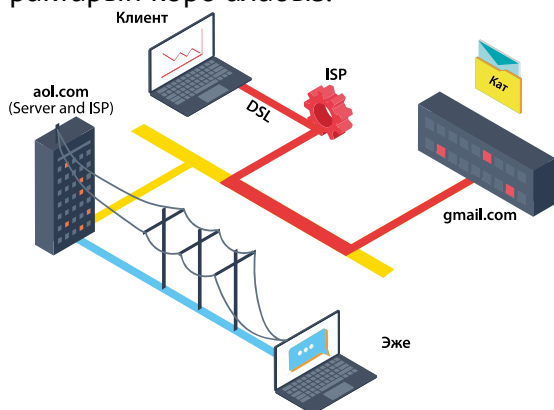
Компьютердик тармак – бул маалыматты берүү каналдары менен байланышкан компьютерлердин системасы. Компьютерлер өз ара атайын өткөргүчтөр менен (Ethernet кабелдери, була оптикалык кабелдер) же аларсыз деле (Bluetooth, Wi-Fi ж.б.) байланышы мүмкүн. Компьютердик тармактар локалдуу (бир имараттын чегинде) же глобалдуу (интернет) болушу мүмкүн.

Интернет глобалдуу тармагы кантип иштээрин карап көрөлү.

Мисалы, биз **aol.com** сайтына кирип, акыркы жаңылыктарды окугубуз келип жатат дейли.



Алгач биз провайдердин тармагына туш болобуз, андан кийин провайдер бизди интернет тармагы аркылуу тиешелүү сайтка багыттайт. Ошондон кийин гана биз **aol.com** сайтынын барактарын көрө алабыз.



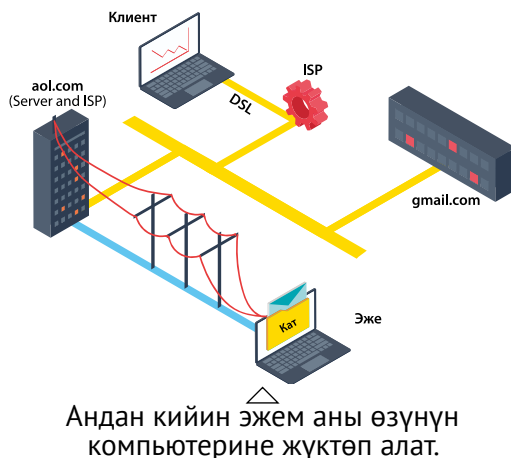
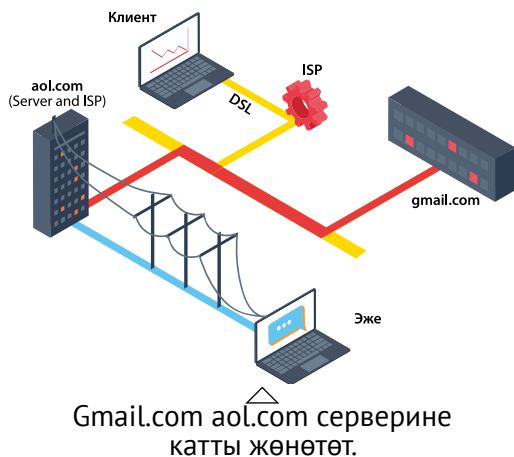
БУЛ КЫЗЫКТУУ!

IP-даректерди домендик аттарга өзгөртүп түзүү үчүн атайын **DNS** (домендик аттардын Сервери, Domain Name Server) серверлер колдонулат.

Бул серверлер домендик аттардын IP-даректер менен дал келген атайын таблицаларын сакташат. Суроо-талап боюнча алар домендик аттарды IP-даректерге жана тескерисинче - IP-даректерди домендик атка өзгөртүп түзө алышат. Домендик аттарды биз браузерге веб-барактарды табуу үчүн киргизебиз. Алар **URL** (Universal Resource Locator) деп да аталышат.

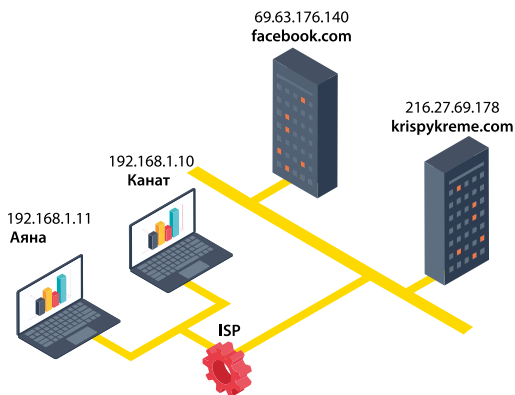


Башка мисал: биз эжеме электрондук катты жөнөткүбүз келип жатат. Эжемдин интернетти телефондук байланыш кабели боюнча **aol.com** провайдерине кошулган. А биздин почтабыз – **gmail.com**до. Эжемдин почтасы **aol.com** сайтында. Алгач **gmail.com** сайтында өзүбүздүн почтабызга киришибиз керек (мисалы, *my@gmail.com*).



Андан соң катты жазып, эжемдин дарегин киргизебиз (*sister@aol.com*) жана «жөнөт» баскычын басабыз.

Интернетте билдирүүнү жөнөтүүдө компьютер аны **пакеттер** деп аталган кичинекей бөлүктөргө бөлүп чыгат. Акыркы серверде билдирүү кайрадан керектүү тартипте чогулуп калат. Билдирүү бир эле электрондук кат түрүндө болбостон, веб-баракча, Твиттердеги билдирүү же каалагандай башка маалымат болушу мүмкүн.



Мисалга сиз жана сиздин ата-энеңиз да үйдөгү интернет менен колдонуп жатасыңар. Мындай учурда силер Wi-Fi аркылуу үй маршрутизаторго туташкан болосуздар.

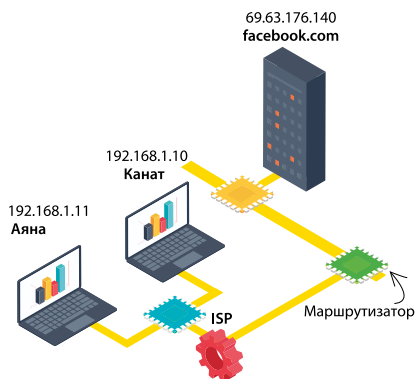
Эгерде сүрөткө карасак, анда сиздин пакеттер жана ата-энеңиздин пакеттери интернетке бир чекит аркылуу кетип жатканын көрүүгө болот. Башкача айтканда сиздин ата-энеңизге тиешелүү маалымат жана сиздин досторуңуз менен болгон маалымат алмашуу да, бардыгы бир орунда чогулат.



Интернеттен сизге тиешелүү каттар сиздин ата-энеңизге эмес өзүңүзгө гана келиш үчүн эмне кылуу керек?

Ушул учурда бизге IP-даректер жана маршрутизаторлор жардамга келет. Интернет тармагында бардык колдонуучулар автоматтык түрдө IP-дарек алышат. Компьютерлер, уюлдук телефондор, планшеттер жана интернетте бири-бирине маалымат жөнөтүп жаткан түзүлүштөр өзүнүн IP-даректерине ээ болушат. Интернеттин түрдүү бөлүктөрүн бириктирген түзүлүштөр маршрутизаторлор деп аталышат.

Маршрутизаторлор интернетте пакеттерди жөнөтүшөт. Ар бир пакет жакынкы маршрутизаторго барат. Мисалы сайтка кирүүдө силердин пакеттиңер адатта 10дон 20га чейинки маршрутизаторлордон өтөт.



Сүрөттө сигналдын маршрутизаторлор арасындагы кыймылы берилген.

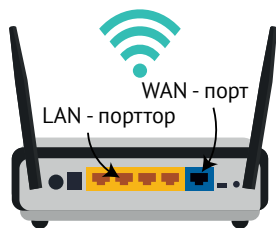
Тармакка берилген пакеттер бир нече катмарлардан турат: 1-катмар – компьютериңиздин IP-дареги; 2-катмар – маршрутизатордун IP-дареги; андан кийин пакет өткөн бардык маршрутизаторлордун IP-даректери жазылат.



ЭСИҢЕ ТУТ

- **IP-дарек** – бул интернет тармагына кошулган ар бир түзүлүшкө ыйгарылган атайын номер.
- **DNS (Domain Name Service)** – бул сайттын аттарын IP-дарекке которуучу сервис.
- **URL (Universal Resource Locator)** – бул сайттын универсалдуу көрсөткүчү, эстеп калууга жеңил дарек (мисалы, www.code.org).
- **Интернет (Internet)** – бул бири-бири менен жалпы тармак аркылуу биригишкен компьютерлердин жана серверлердин группасы.
- **Серверлер (Servers)** – бул берилиштерди жөнөтүп жана суроо-талаптарга жооп берип турган кубаттуу компьютерлер.
- **Була оптикалык кабелдер (Fiber optic cable)** – бул маалыматты берүү үчүн жарык колдонулган атайын кабелдер.
- **Wi-Fi** – бул берүү радиотолкундарды колдонуп санариптик сигналды берүү ыкмасы.
- **DSL** – бул телефондук же телевизиондук кабелди колдонуп маалыматты берүү ыкмасы.
- **Пакеттер (Packets)** – бул сигналды өтө тыкандык менен берүү үчүн чоң маалымат пакеттеринен түзүлгөн маалыматтын кичинекей үлүштөрү.

Локалдык тармакты жана Wi-Fi тармагын тескөө (орнотуу)



Локалдык тармактар бир имараттын чегинде бири-биринен өтө алыс эмес жайгашкан компьютерлердин ортосунда маалымат алмашууну ишке ашырат.

Үйдөгү (локалдык) Wi-Fi тармагын орнотуп көрөлү. Ал үчүн интернет-провайдерге туташылган роутер керек. Аны кабель же Wi-Fi аркылуу тескөөгө болот.

Кабель аркылуу кошууда: кабель бир учу менен компьютерге кошулат, 2-учу менен – роутердин LAN сайгычына кошулат. Андан кийин роутердин WAN сайгычына силердин интернет провайдердин кабелин кошуу керек.

Роутерди Wi-Fi аркылуу орнотууда роутерди күйгүзгөндөн кийин (адатта тармактын аты сиздин роутердин маркасындай болот) пайда болгон Wi-Fi тармагына компьютерди кошуп коюу жетиштүү болот. Бул тармак ачык болот. Бул тармакка планшет же телефон аркылуу кошулуп компьютерсиз эле маршрутизаторду орнотсо да болот. Бул учурда роутер өзү WAN-кабели аркылуу провайдердин тармагына туташкан болуш керек.

Андан ары силер браузерден 192.168.0.1. же 192.168.1.1. даректери аркылуу роутердин тескөөсүнө киришиңер керек. Мында колдонуучунун атын жана сыр сөзүн сураган терезе чыгат. Ага роутердин алдында наклейкада көрсөтүлгөн логин жана паролду киргизесиңер.

Роутердин тескөө терезеси ачылат.

WAN Connection Type: PPTP/Russian PPTP

User Name: username

Password:

Connect Disconnect Disconnected

Server IP Address/Name: [Field]

IP Address: 0.0.0.0

Subnet Mask: 0.0.0.0

Gateway: 0.0.0.0

DNS: 0.0.0.0 0.0.0.0

Internet IP Address: 0.0.0.0

Internet DNS: 0.0.0.0 0.0.0.0

MTU Size (in bytes): 1420 (The default is 1420, do not change unless necessary.)

Max Time: 15 minutes (0 means remain active at all times.)

WAN Connection Mode: ☒ Connect on Demand ☐ Connect Automatically ☐ Connect Manually

Save

WAN Connection Type дегенден ачылып түшкөн тизмеден силердин интернет-провайдер колдонгон байланышуу тиби тандагыла. Адатта бул PPPoE тиби болот.

Андан кийин Username Password деген талаага сиздин провайдерден берилген логин жана паролду киргизгиле.

Dynamic IP пункту тандагыла.

Тескөөдө Network – WAN тиркемесине өткүлө.

Ар бир этапта тескөөнү сактап туруу үчүн Save баскычын басып тургула.



Wi-Fi тармагыбызды сыр сөз менен коргоо үчүн Wireless - Wireless Settings тиркемесине өтүңүз.

Ал жерден коопсуздуктун тибин WPA/WPA2 – Personal тандоо керек. PSK Password талаасында силердин Wi-Fi тармакты коргогон паролду ойлоп таап жазгыла.

Save баскычын басып, орнотууларды сактап турууну унутпагыла.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Качан силер аны браузерде ачканда силердин пакетиңер google.com серверине жеткенге чейинки өткөн бардык серверлерди жазып чыккыла.
- 2) Роутер кошулганда силердин провайдериңер кандай DNS – серверлерди берээрин жазып чыккыла.
- 3) Алмаз кагазга IP-дарегин жазып алып, кийин кокустап ал баракты майдалап айрып алды. Жыйынтыгында IP-даректердин фрагменттери менен 4 айрынды алынды. Бул айрындылар А, Б, В жана Г тамгалары менен белгиленди. IP-даректи калыбына келтирип көргүлө. Жообунда IP-даректин ирээтине туура келгендей кылып айрындыларды белгилеген тамгалардын удаалаштыгын көрсөткүлө.

А.

Б.

В.

Г.

.64

3.13

3.133

20

4.2-тема:

Интернет протоколдордун түрлөрү

Интернет тармагындагы протоколдор жана тармактагы жумуш

Бир компьютерди экинчисине жөнөкөй туташтыруу – бул тармакты түзүү үчүн зарыл болгон кадам. Бирок бир эле бул кадам аздык кылат. Маалыматты бере башташ үчүн, компьютерлер бири-бирин «түшүнөөрүнө», алар өз ара «маалымат алмаша ала турганына» ынаныш керек. Компьютерлер тармакта кантип «маалымат алмашышат»? Мына ушул мүмкүнчүлүктү камсыз кылуу үчүн «протоколдор» деп аталган атайын каражаттар иштелип чыккан.

Протокол түшүнүгү бир эле компьютердик индустрияга гана тиешелүү эмес. Ал эмес интернет менен эч качан тааныш болбогондор да протоколдорду колдонуп иштеген түзүлүштөр менен кезигишет. Мисалы, кадимки телефондук канал да өзүнүн протоколуна ээ болот. Ал аппараттарга байланыш каналынын экинчи учундагы абоненттин жооп бергендиги же сигналдын үзүлгөндүгү жөнүндө фактыны тактайт. Телефондук трубкадагы үн сигналы – бул телефондордун тили.

Компьютерлердин да бирдиктүү тили бар, бул – **протокол**.

Интернет ишинде колдонулуучу негизги протоколдор:

- **TCP/IP** – пакеттерди берүү;
- **POP3/SMTP** – электрондук каттарды алуу/берүү;
- **FTP** – файлдарды берүү;
- **HTTP/HTTPS** – веб-барактарды берүү;
- **IMAP4** – электрондук каттарды алуу/берүү.

TCP/IP протоколу

IP протоколу TCP/IP протоколдорунун негизин түзөт. Ал туташтырууну орнотпогон протоколдордун тибине кирет. IP протоколу билдирүүлөрдү жеткирүүнүн ишенимдүүлүгүнө кепилдик бербейт, анткени маалыматты жеткирүү тастыкталбайт.



АНЫКТАМА

Протокол – бул тармак аркылуу маалыматты берүүнү жүргүзгөн тиешелүү эрежелер.



Берилиштердин агымын IP протоколу белгилүү бөлүктөргө – **IP-пакеттерге** же **дейтаграммаларга** бөлөт жана ар бир IP-пакетти көз карандысыз пакет катары карайт. IP протоколунун негизги кызматы тармактар арасында IP-пакеттерди берүү болуп саналат.

IP протокол билдирүүлөрдү ишенимдүү жеткирүүгө кепилдик бербегендиктен, бул маселени чечүү үчүн TCP протоколун колдонушат. TCP протоколу компьютерлер арасында логикалык байланышты орнотот. Маалыматтарды берүү алдында сеансты баштоого суроо-талап жөнөтүлөт. Кабыл алуучу тарабынан тастыктоо жөнөтүлөт, эгер белгилүү убакытта тастыктоо болбосо, анда суроо-талап кайрадан жөнөтүлөт.

TCP протоколу берилиштердин агымын **сегменттерге** бөлөт. Ар бир сегменттин баш сөзүндө контролдук суммасы берилет.

- Эгерде жөнөтүүдө берилиштер бузулган болсо TCP протоколу ошол контролдук сумма аркылуу муну аныктай алат. Бузулган сегмент жок кылынат, ал эми булакка эч нерсе жөнөтүлбөйт.
- Эгерде берилиштер бузулбаса, анда алар тиркемеде чогултулат, ал эми булакка тастыктоо жөнөтүлөт. Сегменттерди транспорттоо үчүн TCP/ IP протоколу аларды IP-пакетинин кабыгына жайгаштырат.

HTTP протоколу

HTTP протоколу (*Hypertext Transfer Protocol* – *Гипертекстти берүү протоколу*) интернеттен веб-барактарды натыйжалуу берүү үчүн иштелип чыккан. HTTP негизинде биз тармактагы барактарды алардын укмуштуудай кооздугу менен көрүүгө мүмкүнчүлүк түзүлдү. HTTP протоколу World Wide Web системасынын негизи болот.

Силер браузердин интерфейсин колдонуп, HTTP командаларын бересиңер. Электрондук чыккандын баскычы менен шилтемеге басууда, браузер веб-серверден ошол шилтеме көрсөткөн ресурстун берилиштерин – мисалы, веб-барактын маалыматтарын сурайт.



ЭСИҢЕ ТУТ

Интернет тармагынын протоколдору деңгээлдерге бөлүнөт, бир протоколдордун иштеши үчүн башка протоколдор керек болот. Мисалы, HTTPS/HTTP, FTP, SMTP, POP3, IMAP протоколдору компьютерлер арасында маалыматтарды берүү үчүн TCP/IP протоколун колдонушат.

Веб барактын мазмуну браузерде ойдогудай чагылдырылышы үчүн, ал өзгөчө тексттик белгилердин – гипертекстти белгилөө тили (HyperText Markup Language, HTML) менен берилет.

Мындан тышкары учурда **HTTP-S (HTTP Secure)** протоколу – гипертекстти кошумча коргоо менен берүүгө колдонулат. Бул учурда веб-сервер жана колдонуучунун ортосундагы бардык берилип жаткан маалыматтар шифрленет. Бул болсо колдонуучуга интернетте маанилүү маалыматты, кредиттик карталардын аттарын, сыр сөздөрүн, номерлерин ж.б. коопсуз алмашууга жардам берет.

HTTP протоколу боюнча силер кайрылган интернет ресурстарынын даректери болжолдуу түрдө төмөнкүдөй болот: <http://www.code.org>.

FTP протоколу

FTP (File Transfer Protocol – Файлдарды берүү протоколу) – файлдар менен алмашууга мүмкүндүк берет. FTP протоколу боюнча файлдарды берүү **FTP-сервер** жана **FTP-клиенттин** ортосунда жүрөт. FTP-сервер FTP-клиенттен келген суроо-талаптарды – файлдарды жүктөө үчүн программаларды иштетет.

FTP-клиентке файлды берүү процессинин башында серверге колдонуучунун атын жана сыр сөздү жөнөтүү талап кылынат. Ал колдонуучуну идентификациялоого мүмкүндүк берет. FTP-серверди колдонуучу идентификациясыз эле киргендей кылып орнотуп койсо да болот. Бул учурда колдонуучу **анонимдүү** (anonymous) деп аталат башкача айтканда серверге колдонуучунун бир эле атын (anonymous) билдирип турат. Сыр сөз мындай учурда эске алынбайт.

Эми биз электрондук почталардын иши мүмкүн болбогон, SMTP, POP3 жана IMAP протоколдорду карап көрөлү. Бул протоколдор бир гана максат үчүн колдонулат – электрондук билдирүүлөр алмашууну уюштуруу үчүн. Мисалы билдирүүнү жөнөткөн протокол аларды кайра кабыл ала албайт жана тескерисинче. Ошол себептен мындай протоколдор түгөйү менен иштешет.

SMTP протоколу

SMTP (Simple Mail Transfer Protocol – Жөнөкөй почталык берүү протоколу) электрондук билдирүүлөрдү жөнөтүүнү камсыз кылат.



Бирок SMTP сервер билдирүүлөрдү кабыл алуучу тарапта чогултууга жөндөмдүү эмес. Ошондуктан почтаны кабыл алууда дагы бир почталык протокол – POP3 же IMAP протоколу керек.

POP3 протоколу

POP3 (Post Office Protocol 3 – Почталык кызмат протоколу) адресат тарабынан электрондук билдирүүнү кабыл алууну камсыз кылат. Бул протоколдун негизинде почта сервер аркылуу кабыл алынат жана ал жерде топтолот. Почталык клиент программасы почтаны мезгил менен текшерип турат жана билдирүүлөрдү локалдык компьютерге жүктөйт.

Ошентип, почтаны жөнөтүү SMTP менен ишке ашат, ал эми кабыл алуу – POP3 жардамы менен жүрөт. Мына ошондуктан почталык клиентти орнотууда SMTP серверинин аты менен бирге POP3 серверинин атын да киргизүү талап кылынат.

IMAP протоколу

IMAP (Internet Message Access Protocol – интернеттин электрондук почтасына кирүүгө мүмкүндүк алуу протоколу) POP3кө окшош, келген каттар менен иштөө үчүн кызмат кылат, андан тышкары кошумча функцияны камсыз кылат. Тактап айтканда почтаны локалдык компьютерге сактабастан туруп, ачкычтык сөз менен издөөгө мүмкүндүк берет. Бул протоколду колдонгон почталык программа сервердеги каттарды кабыл алуучунун компьютерине жүктөбөстөн эле иштетүүгө мүмкүндүк берет. Демек каттар, алардын толук мазмундагы файлын серверден улам жөнөтүп жана кабыл алып турбастан эле, колдонуучунун компьютеринде жеткиликүү болот. Каттарды жөнөтүү үчүн SMTP протоколу колдонулат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Качан силер электрондук почтаңарда катты ачып окууда жана жөнөтүүдө браузер кандай протоколдорду колдонот?*
- 2) Силер колдонуп жүргөн сайттардын ичинен кайсылары HTTPS протоколун колдобойт?*

4.3-тема:

Стилдерин каскаддык таблицалары (CSS)

Кадимки HTML документ форматтоо тегдеринин жардамы менен тексттин түсүн жана өлчөмдөрүн орнотууга мүмкүндүк берет. Эгерде сайттагы бир типтүү элементтердин параметрлерин өзгөртүш керек болуп калса, анда тегдерди табыш жана өзгөртүү үчүн бардык барактарды карап чыгуу керек болот. Анын ордуна тексттин түсүн, өлчөмдөрүн ж.б. параметрлерин стилдерде сактай турган башка – CSS технологиясын колдонсо жеңил болот.

Стиль деп HTML документтин элементтерине, анын сырткы келбетин тез өзгөртүү үчүн колдонулган форматтоо эрежелеринин жыйындысын айтышат. Стилдер бир аракет менен эле форматтоону окшош элементтердин бүткүл группасына колдонгонго мүмкүндүк берет, мисалы, бардык баш сөздөрдүн көрүнүшүн өзгөртүү ж.б.

Стилдер жөнөкөй HTMLге караганда форматтоо үчүн көп мүмкүнчүлүктөрдү берет, андан тышкары өзүнчө сырткы файлда сактала алат. Браузер мындай документтерди колдонуучунун компьютеринде локалдуу сактайт (кэштейт), ошондуктан сайттын жүктөлүүсү тез болот.

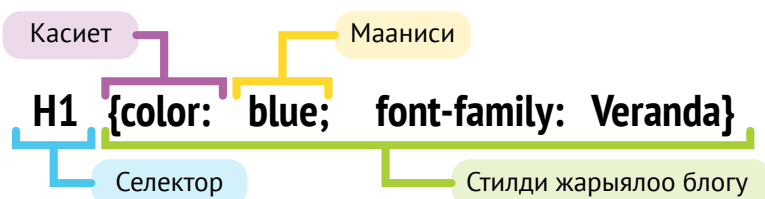
HTML документтеги элементтерди форматтоонун бардык жеткиликтүү касиеттери 9 категорияга бөлүнөт:

- **Шрифт** – шрифттердин типографиялык касиеттерин орнотот;
- **Түс жана фон** – тексттин түсүн жана фонду аныктайт, андан тышкары сүрөттү фон катары колдонот;
- **Текст** – текстти түздөө, форматтоону аныктайт;
- **Блок** – блокуу элементтерди форматтоо касиети;
- **Визуалдык форматтоо** – элементтерди чагылдыруу блоктору, аларды позициялоо жана тизмелерди чагылдыруу менен байланышкан касиеттери;
- **Фильтрлер жана өтүүлөр** – мультимедиялык эффекттерди жана графикалык сүрөттөрдү өзгөртүп түзүүнү аныктайт;
- **Басып чыгаруу (печать)** – барактын бөлүнүшүнүн спецификацияларын аныктайт;
- **Псевдокласстар** – @import, cursor, important касиеттери ишке киргизилет;
- **Калган башка касиеттери.**



CSSтин синтаксиси

CSS стили ар дайым стилди жарыялоо блогунун сол бөлүгүнөн орун алып, чарчы кашаага алынган селектордон (HTML документтин элементинен) турат.



Берилген мисалда H1 тегине жасалгалоонун жаңы стили колдонулду, эми ал көк түстө жана анын шрифти Verdana болду.

Селектор – бул берилген стилдер кайсы элементтерге таандык экендигин аныктоочу конструкция.

Селекторлордун 4 тиби эң көп колдонулат:

- Элементи боюнча;
- Классы боюнча;
- ID боюнча;
- Контексттик селектор.

Элементи боюнча селектор

Эгерде стилди биринчи деңгээлдеги бардык баш сөздөргө колдонуш керек болсо, анда селектор катары <H1> элементи колдонулат. Мисалы, биринчи деңгээлдеги бардык баш сөздөр күңүрт боз түстө, Arial шрифти менен жана тамгалардын арасындагы аралык 10 пикселден болуп чыгыш үчүн стиль мындай жазылат:

```
h1{color:#666;font-family:Arial;letter-spacing:10px}
```

Класс боюнча селектор

Белгилүү тегге конкреттүү касиеттерди колдонуш үчүн каалаган элементте пайдаланууга боло турган class деген атайын атрибут бар.

Мисалы, менюдагы шилтемелерди өзгөчө кылуу үчүн жаңы классты киргизебиз: class=«menu».

```
a.menu {font-style: italic;font-weight: bold}
```

HTML код болсо мындай болот:

```
<a href="about.html" class="menu">Мектеп жөнүндө</a> (Өзгөчө шилтеме)
```

```
<a href="links.html">Ссылки</a> (Кадимки шилтеме)
```

ID боюнча селектор (Идентификатор боюнча)

Эгерде класс боюнча селектордо элементтин аты менен классты бөлүү үчүн чекит колдонулса, ID боюнча селектор үчүн # белгиси колдонулат. Элементтин атын калтырып кетсе да болот, анда берилген стилди first идентификаторундагы баардык элементтерге колдоно берсе болот.

```
a#first {font-style: italic;font-weight: bold}
```

```
<a href="about.html" id="first">Мектеп жөнүндө</a>
```

Контексттик селектор

Контексттик селекторлор башка элементтин ичинде жайгашкан элементтердин группасы үчүн бирдиктүү стилди колдонууга мүмкүндүк берет. Мисалы, таблица – бул элемент, ал эми бардык мамычалар – бул элементтер группасы.

Берилген мисалда div теги менен бөлүнгөн бардык группадагы шилтемелер үчүн

```
<div id="first">
```

```
  <a href="index.html">Негизги барак</a>
```

```
  <a href="about.html">Мектеп жөнүндө </a>
```

```
</div>
```

first идентификатору менен стиль колдонулат.

```
a#first {font-style: italic;font-weight: bold}
```

Стиллерди документке кошуу

Бардыгы болуп стилдерди кошуунун төрт ыкмасы бар:

- Стиллерди HTML документтин ичине камтуу;
- HTML документтен сырткы Стиллер документине шилтеме берүү;
- Стиллер документин HTML документине импорттоо;
- Стиллерди түздөн түз HTML документинин саптарына кошуу.



Стилдерди HTML документтин ичине камтуу

CSS менен байланышкан бардык маалымат HTML документтин башында жайгашат жана <BODY> тегинин ички мазмуну менен байланыш болбойт.

Мисал:

```
<html>
<style type="text/css">
  <!--
  h1{color:green;font-family:Impact}
  p{background:yellow;font-family:'Courier New'}
  -->
</style>
<head>
  <title>Камтылган стилдер</title>
</head>
<body>
  <h1> Мисал 1</h1>
  <p>Мисал 2</p>
</body>
</html>
```

Камтылган стилдер ушул гана HTML документке колдонулат. Эгер силер Стилдерди жалгыз барак үчүн колдоном десеңер, анда бул ыкма эң ыңгайлуу болуп саналат.

Стилдерди импорттоо

Импорттолгон стилдер өзүнчө файлда жайгашкан болот. Аларды документтин ичинде жайгашкан өзүңөрдүн стилиңер менен бириктирсеңер да болот.

Мисал:

```
<html>
<style type="text/css">
  <!--
  @import url(mytyles.css)
  h1{color:green;font-family:Impact}
  -->
</style>
<head>
  <title>Камтылган стилдер</title>
```

```
</head>
<body>
  <h1>Мисал 1</h1>
  <p>Мисал 2</p>
</body>
</html>
```

Mystyles.css файлы мындай көрүнөт:

```
h1{color:orange;font-family:Impact}
p{background:yellow;font-family:'Courier New'}
```

Бул мисалда браузер биринчи Mystyles.css (@import сапчасы ар дайым биринчи болуш керек) файлынан стилдерди импорттойт, андан кийин буга стилдердин камтылган командаларын кошот. Веб-бетте тигил да бул да стилдер колдонулат.

Стилдерди түздөн түз HTML документинин тулкусуна кошуу

Бул учурда стилди HTML документтин баш сөзүнө кошуунун кереги жок. Элементтин стилинин орнотулган атрибуттары браузерде чагылдыруу үчүн бүткүл маалыматты өзүндө камтыйт. Бул стиль берилген элемент үчүн гана колдонулат.

Мисал:

```
<html>
<head>
  <title>Камтылган стилдер</title>
</head>
<body>
  <h1 style="color:green;font-family:Impact"> Мисал 1</h1>
  <p style="background:yellow;font-family:'courier'">Мисал
2</p>
</body>
</html>
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

«Мен жана космос» деген веб-баракты түзгүлө жана баш сөз, шилтемелер үчүн аныкталган өзүңөрдүн CSS стилиңерди импорттогула.

`</CODE>`



PYTHON

`void setup() { pinMode (13, OUTPUT);`

`// устанавливаем и используем пин как выход`

`pinMode (12, OUTPUT); pinMode (11, OUTPUT); pinMode (10`

`pinMode (10`





9 – класс



1

- бөлүм



Информатика жана маалымат

1.1-тема:

Маалыматтык сабаттуулук

Бүгүнкү күндө, маалыматтын агымы күндөн күнгө көбөйүп жатканда, керектүү маалыматты тандоо жана баалоо ар бир адам үчүн эң маанилүү көндүм болуп тургандыгы шексиз. Бул көндүм маалыматтык сабаттуулук деп аталат жана коюлган маселени чечүү үчүн компьютердик тиркемелерди (программаларды) колдоно алууну түшүндүргөн компьютердик сабаттуулуктан айырмаланат.

Маалыматтык сабаттуулук төмөнкүлөргө мүмкүндүк берет:

- конкреттүү маалымат муктаждыктарды аныктаганга;
- маалымат булактарын табууга;
- маалыматты кабыл алууга же тандоого;
- маалыматтын сапатын баалоого жана талдоого;
- маалыматты сактоого жана ирээттөөгө;
- маалыматты натыйжалуу жана этика менен колдонууга.

Маалыматты издөөдө аны сын көз менен баалоо билгичтиги маанилүү ролду ойнойт. Интернеттен маалыматты алып жатып, өзүңөргө беш суроо бергиле:

- Макаланын автору ким, ал жазып жаткан тема бонча эксперт болуп саналабы?
- Макаланын мазмуну жалпы сайттын тематикасына дал келеби?
- Макаланы жарыялоо датасы барбы?
- Эмне үчүн мен бул маалыматка ишенүү керек деп эсептейм?

Өзгөчө азыркы кездеги «жаңы медиа» деп аталган интерактивдик жалпыга маалымат каражаттарынан (социалдык тармактар, жаңылыктар каналы ж.б.) алган маалыматты туура баалай билүү керек.

Мындай медиалардын негизги максаты болушунча көп колдонуучулардын көңүлүн бурдуруу жана аудиториянын көз карашына олуттуу таасир этүү болуп саналат.



АНЫКТАМА

Маалыматтык сабаттуулук – бул адамдын маалыматка муктаж экендигин мойнуна алуу, аны издеп, тандап, баалап жана пайдалана билүү жөндөмү.

Ал үчүн ар кандай ыкмалар колдонулат: олуттуу материалдарды кызыктыруучу оюн формасында берүү, же болбосо көпчүлүк учурда чындыкка коошпогон, үрөй учурган контентти камтыган материалдарды жарыялоо. Андан тышкары маалыматты көп учурда өздөрүнүн кызыкчылыгы үчүн өзгөртүп колдонушат (манипуляциялашат).

МААЛЫМАТ МЕНЕН ЭМНЕ КЫЛСА БОЛОТ?

Жок нерсени ойлоп таап, аны чындык катары көрсөтүүгө

Толук эмес бир тараптуу берүү менен бурмалоого

Өзүңдүн оюнду жана комментарийлерди кошуп редакциялоого

Манипулятор үчүн пайдалуу болгондой интерпретациялоого

Кайсы бир маанилүү бөлүктөрүн жашырып, айтпай коюуга

Маалыматты бурмалоо окурмандарга туура эмес, жалган ой-пикирдин калыптанышына алып келет.

Социалдык тармактарда каалаган адам маалымат каналы болгондуктан көптөгөн фейктер пайда болуп жатат.



АНЫКТАМА

Фэйк (англ. *fake* – «жасалма, жалган») маалыматты берүүдө болсо – «жалган маалымат» деген маанини түшүндүрөт.

Бүгүнкү күндө эмнелерди «фейк» деп аташат:

Фотошопто жасалган сүрөттөр, адаштыруу максатында же башка окуяны иллюстрациялоо үчүн атайын монтаждалган видеороликтер. Мисалы, бир нече жыл мурун аскердик аракеттерде тартылган видеону акыры күндөрдө болгон окуя катары көрсөтүшөт.

Бардыгы эле чын - бышыгын ажырата албаган жалган жаңылыктар (мурда «гезит өрдөк» деп аталгандар азыр «төгүндүлөр» («вбросы») деп аталат.

Социалдык тармактардагы башка адамдын (көбүнчө белгилүү адамдын) аты менен түзүлгөн барактар.



Фейктерди жаратуу көпчүлүк учурда алдамчылык менен байланышкан. Ал акча каражатын чогултуу болушу да мүмкүн (мисалы, «смартфондорду ойнотуу»), ошондой эле жалган пикирлерди жарыялоо менен белгилүү товарларды сатууну көбөйтүү болушу да мүмкүн. Андан тышкары фейктерди жарыялоо менен алдамчылар көп сандагы көрүүнү (трафикти генерациялоо) ишке

ашыра алышат. Мисалы, абдан таасирдүү баш сөздөрдүн жардамында: «Сенсация: англис тилин 1 ай ичинде үйрөн!», «Ал 2 апта ичинде 20 кг га арыктады, сенин колуңан да келет!» ж.б.

Спам

«Информациялык таштандынын» тараган дагы бир түрү болуп спам эсептелет. Спам – бул көпчүлүк учурда жөнөтүүчүнү тактоого мүмкүн болбогон сиз каалабаган жарнамалык жөнөтмөлөр. Мындай билдирүүлөр электрондук почта, социалдык тармактар аркылуу, форумдар, комментарийлер жана СМС-билдирүүлөр аркылуу таралат. Статистикага таянсак, азыркы кездеги бардык электрондук каттардын 90%ы – бул спам. Спамерлер (спамды жөнөткөндөр) өздөрүнүн спамдары үчүн кабыл алуучулардын даректерин ар түрдүү сайттардан чогултушат. Качандыр бир убакта силер ал сайттарга өзүңөрдүн маалыматыңарды (логин, эл.почта ж.б.) киргизгенсиңер.

Негизи спамдар силерди кайсы бир шилтемелер менен өтүүңөрдү же шек жараткан тиркемелерди ачууга багыттайт. Албетте спамды өчүрүү эң жеңил болгону менен кээде капыстан спамдагы шилтемени басып алуу олуттуу көйгөйгө кабылдырышы мүмкүн: вирус жукутуруп алуу же сиздин өздүк маалыматтарыңардын уурдалышына жол берүү.

Интернет-алдамчыларга алданып калбаш үчүн сын көз менен ой жүгүртүү зарыл, башкача айтканда ар кандай шек жараткан маалыматтын алгачкы булагын табууга аракеттенүү керек.

Эң негизгиси – интернетте айтылгандардын бардыгына эле ишене бербөө.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) «Фейк» деген эмне? Окуу жылынын башталышын 15-сентябрга жылдырыптыр деген маалымат «фейк» болуп эсептелеби? Эмне үчүн?
- 2) Социалдык тармактагы «фейктер» менен сыналгыдагы «фейктер» бири-биринен айырмаланабы? Алар эмнеси боюнча айырмаланышы мүмкүн экендигин түшүндүргүлө.
- 3) Өзүңөрдүн мини-изилдөөңөрдү жүргүзгүлө: «Фейк» окуяңарды 10 досуңарга айтып, алардын канчасы силерге ишенгенин эсептегиле.
- 4) Таблица түзгүлө: спамдын кандай түрлөрү болот жана алардын ар бири кандай зыян алып келиши мүмкүн?

1.2-тема:

Шифрлөө жана электрондук-санариптик кол тамга

Санариптик кылымда электрондук документтер өзгөчө мааниге ээ болуп келүүдө.

Электрондук документ – бул электрондук алып жүрүүчүдө сакталган документ. Каалагандай юридикалык маанидеги келишимдер же маалымктар сыяктуу кагаз документтердей эле электрондук документтерге да кол койсо болот. Ал үчүн электрондук-санариптик кол тамга колдонулат.

Электрондук документтер кагазга караганда бир топ артыкчылыктарга ээ:

- Электрондук документтерди иштетүү жана жеткирүү убактысынын тездиги (каалагандай документке аралыктан эле кол коюп, аны электрондук почта менен жөнөтүү).
- Чыгашалардын азайышы: документти кагазга чыгаруунун, почта менен жөнөтүүгө төлөөнүн ж.б. кереги жок.

Башкача айтканда, мурда кандайдыр бир укуктарды же акчаны берүүдө, мисалы үйдү алып-сатуу же болбосо мамлекеттик мекемеден маалымкат алуу, ошол документ берген органда маалымкат алуучу адамдын сөзсүз болуусу менен гана ишке ашкан. Андан тышкары бизге калемсап менен толгон токой кагаздарга кол коюп чыгууга туура келген. Операторлор адамдын аты-жөнүн тактоо үчүн паспортунун же туулгандыгы тууралуу күбөлүгүнүн түп нускасын көрсөтүүнү өтүнүшкөн.

Санариптик кылымда көптөгөн ушул сыяктуу аракеттерди интернет аркылуу аралыктан жүргүзүүгө болот.

Негизги шарт – оператор сизден келген маалыматты аутентификациялап (текшерүүдөн өткөрүп) туруусу керек. Бул үчүн ар кандай технологияларды колдонушат, мисалы, банкоматтан акча алуу үчүн 4 орундуу санды киргизүү керек, ал болсо сиздин картаңызга кирүүгө мүмкүнчүлүк түзөт, ал эми сиздин каттарыңызды окуу үчүн электрондук почтага паролуңузду киргизүү керек болот.



АНЫКТАМА

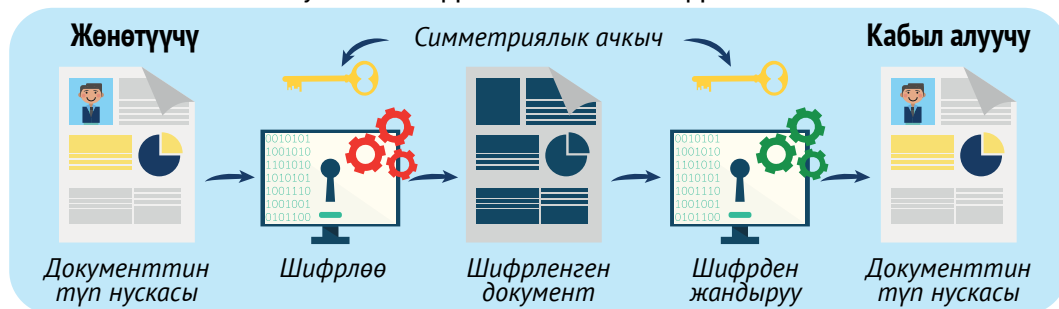
Аутентификация – бул бир нерсенин чындык экенин текшерүү процесси. **Аутентификациянын** мисалы катары маалыматтар базасынын серверинде сакталган пароль менен колдонуучу киргизген паролду салыштыруу болуп саналат.

Бирок, мамлекет катышкан электрондук документтерге, мисалы, келишимге, арызга кол коюу үчүн **электрондук-санариптик кол тамганын (ЭСКТ)** болуусу шарт. ЭСКТ – бул калем сап менен коюлуучу кол тамга сыяктуу эле уникалдуу кол тамга, болгону электрондук файлдар үчүн гана колдонулат. ЭСКТ берилген электрондук документке ким кол койгонун жана документ кол коюлгандан кийин өзгөргөнүн же өзгөргөбөгөнүн аныктоого мүмкүндүк берет.

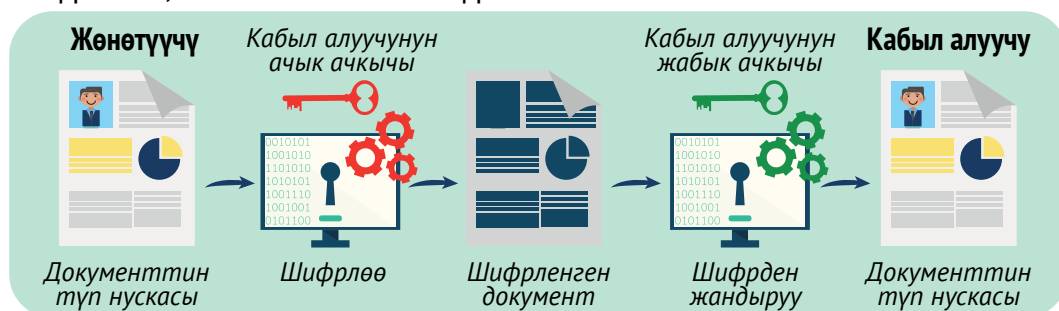
Башка сөз менен айтканда ЭСКТ – бул документти шифрлөө механизми болуп саналат.

Тарыхта шифрлөөнүн негизги эки түрү колдонулган:

- Симметриялык – бир гана ачкыч колдонулат, башкача айтканда бир эле ачкыч менен документ шифрленет жана шифри чечмеленет.



- Асимметриялык – эки ачкыч колдонулат: бир ачкыч менен документ шифрленет, экинчиси менен шифри чечмеленет.



Асимметриялык шифрлөөдө, эгерде бирөө сизге шифрленген билдирүүнү жөнөткүсү келсе, ал сиздин ачык ачкычыңызды (сиз аны бардык каалоочуларга бересиз) алып, билдирүүнү шифрлеп, анан сизге жөнөтөт.

Ал эми сиз болсо, ал билдирүүнү алгандан кийин өзүңүздүн жабык ачкычыңыз (ал сизге эле белгилүү) шифрленген билдирүүнү чечмелеп аласыз.

Ошентип, кол тамганы текшерүү аны кайрадан эсептөө жана текшерүүчү тараптагы ачкыч менен салыштыруу болуп эсептелет. Текшерүүдө кол тамга же туура, же туура эмес деген билдирүү чыгарылат. Эгерде кол тамга туура эмес деп чыкса, анда туура эмес ачкыч колдонулду же документ өзгөрүүгө дуушар болгон.

Акырында айтаарыбыз, өзүнүн корголушунун негизинде асимметриялык шифрлөө эң кеңири таралган. Аны **HTTPS** протоколун колдогон сайттар, мессенджерлер (эстегилечи, Watsapта «сиздин билдирүүлөр өтмө шифрлөө менен корголгон» деп жазылып турат), wi-fi-роутерлер, банк системалары жана башкалар колдонуп келишет. **Электрондук кол тамга** да асимметриялык криптографияга негизделген. Андан тышкары асимметриялык криптографиянын негизинде **блокчейн** алгоритми түзүлгөн, өз кезегинде бардык криптовалюталар, анын ичинде **биткоин** да анын негизинде түзүлгөн.



БУЛ КЫЗЫКТУУ!

Блокчейн (англ. *blockchain*) – маалыматты камтыган жана эреже менен түзүлгөн блоктордун үзгүлтүксүз удаалаш чынжыры. Көпчүлүк учурда чынжыр блокторунун нускалары бири-бирине көз карандысыз көптөгөн ар башка компьютерлерде сакталат.

Башка сөз менен айтканда блокчейн – бул берилиштер сакталган жай жалпы процессор менен байланышпаган бөлүштүрүлгөн маалыматтар базасы. Чынжырдагы бардык колдонуучулар базадагы маалымат менен башкалар эмне кылып жаткандыгын көрө алышат, бирок алар өздөрүнө тиешелүү гана маалымат блокторун өзгөртө алышат. Бул болсо бардык процесстерди тунук кылат, андыктан бул базада кандайдыр бир документти өзгөртүү мүмкүн эмес.

Блокчейн технологиясынын артынан эч бир дүйнөлүк банкка же кайсы бир өлкөнүн экономикасына көз каранды болбогон **биткоин** жаңы криптовалютасы пайда болду. Анын өзүнүн баасы (курсу) бар, аны интернетте сатып алса жана сатса болот. Биткоиндин эң негизги өзгөчөлүгү – бул анын анонимдүүлүгү.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Симметриялык жана асимметриялык шифрлөөлөрдүн негизги артыкчылыктарын атап бергиле.
- 2) Өзүңөрдүн жашооңордон мисал келтиргиле, симметриялык жана асимметриялык аутентификациялоо кай жерлерде колдонулат?

1.3-тема:

Графикалык маалыматты коддоо

Сактоо формасы боюнча графикалык маалыматтарды **аналогдук** (кагаздагы сүрөт) жана **санариптик** (компьютердик графика) деп бөлүшөт.

Компьютердик графика – бул информатиканын бир бөлүмү, ал эсептөө техникасынын жардамы менен графикалык маалыматты алуу, түзүү жана иштетүү ыкмаларын окутат.

Бардык маалыматтын түрлөрү сыяктуу эле компьютердеги сүрөттөлүштөр да экилик коддордун удаалаштыгы түрүндө коддолот. Мындан тышкары, санариптик сүрөттөлүштөрдөгү ар бир чекит да өзүнүн түстүк кодуна ээ. Канчалык чекит жана колдонулган түстөр көп болсо, сүрөттөлүш ошончолук сапаттуу болот.

Түзүү ыкмасы боюнча графикалык сүрөттөлүштөрдү 3 топко бөлсөк болот:

Вектордук – бул параметрлери сан түрүндө, мисалы: өлчөмдөрү, чокуларынын координаталары, жантаюу бурчу, контурдун жана боёктун түсү сакталган жөнөкөй геометриялык фигуралардын топтому түрүндө коддолгон сүрөттөлүштөр.

Вектордук графиканын артыкчылыктары: сапатын жоготпостон масштабдоо мүмкүнчүлүгү, салыштырмалуу өтө чоң эмес маалыматтык көлөм.

Фракталдык – компьютердин эсинде сакталган формула боюнча түзүлгөн сүрөттөлүштөр. Фракталдык графиканын артыкчылыгы: түзүү жөнөкөйлүгү, чексиз масштабдоо мүмкүнчүлүгү – сүрөттүн татаалдыгын көбөйтүү практика жүзүндө файлдын көлөмүнө таасирин тийгизбейт.

Растрдык – ар бири өзүнүн координатасына жана түсүнүн кодуна ээ болгон пикселдердин көптүгүнөн түзүлгөн сүрөттөлүш. Растрдык графиканын артыкчылыгы: түстү берүү тактыгы жана сүрөттөлүштүн реалдуулукка жакындыгы.



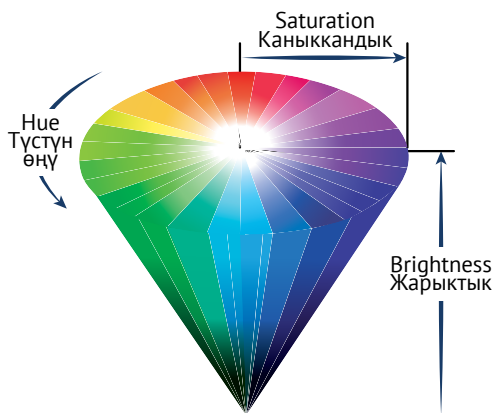
АНЫКТАМА

Дискреттөө – бул графикалык маалыматты аналогдук формадан дискреттик формага өзгөртүп түзүү, башкача айтканда үзгүлтүксүз графикалык сүрөттөлүштү өзүнчө элементтерге бөлүп чыгуу.

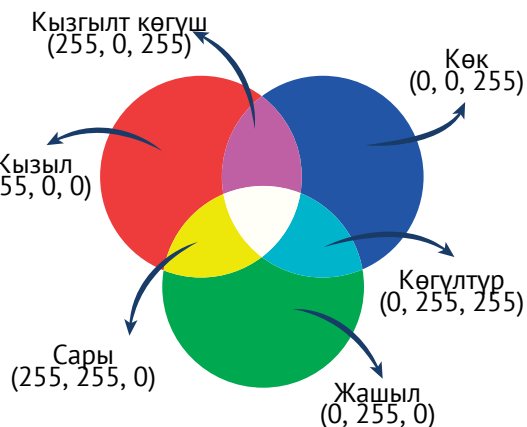
Сүрөттөлүштөр аналогдук формада санариптик (дискреттик) формага дискреттөө жолу менен – мисалы, сканерлөө жолу менен өзгөртүлүп түзүлөт.

Растрдык сүрөттөлүштөр үчүн коддоо жана түстү берүүнүн негизги үч системасын карап чыгалы: HSB, RGB жана CMYK.

HSB модели үч компоненттен турат: түстүн өңү (Hue), түстүн каныгуусу (Saturation) жана түстүн жарыктыгы (Brightness). Айлананын борборунан чыккан вектор түстүн маанисин берет. Кошумча түстөр вектордун багыты менен аныкталат жана бурчтук градустар менен берилет. Түстүн каныгуусу вектордун узундугу менен аныкталат, ал эми түстүн жарыктыгы өзүнчө окто берилип, анын нөлдүк чекити кара түстү берет. Борборундагы чекит ак түскө (бейтарап), ал эми периметри боюнча жайгашкан чекиттер – таза түстөргө дал келет.



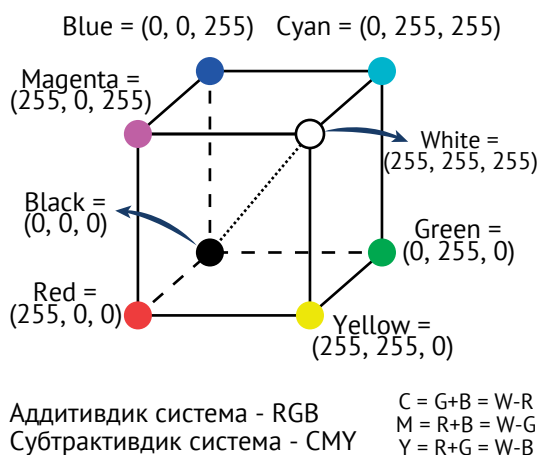
RGB модели: каалагандай түс үч түстүн айкалышынан түзүлөт: кызыл (Red, R), жашыл (Green, G), көк (Blue, B). Калган бардык түстөр жана кошумча түстөрү бул түзүүчүлөрдүн канчалык деңгээлде болгондугунан же жоктугунан алынат.



Мисалы, 256 градациялык түстө ар бир чекит 3 байт менен коддолот, анткени бир түстү коддоо үчүн 1 байт – 8 бит бөлүнөт ($2^8=256$). Модель өзү үч түстү колдонгондуктан, түстөрдүн мүмкүн болгон айкалыштарын коддоо үчүн 3 байт керектелет.

RGBнын (0, 0, 0) минималдык мааниси кара түскө туура келет (экрандын минималдык жарыктыгы), ак түскө – максималдык мааниси RGB (255, 255, 255), ал эми боз түскө – ар бир түстүн барабар мааниси туура келет (мисалы, RGB (35, 35, 35)).

СМҮК модели басмаканада басып чыгарууларда колдонулат. Кошумча түстү калган эки түстү кошуу менен же ак түстөн кемитүү менен алат. Кызыл түс үчүн кошумча түстөр көгүш (Cyan, C) = жашыл + көк = ак - кызыл, жашыл түс үчүн – кочкул кызыл (Magenta, M) = кызыл + көк = ак – жашыл, көк түс үчүн – сары (Yellow, Y) = кызыл + жашыл = ак – көк. СМҮК моделиндеги түстөрдүн аралашмасы накта кара түстү бербейт, ошондуктан К (Black) тамгасы менен белгиленген кошумча кара түстүн компоненти колдонулат.



Түстүү графиканы берүүнүн бир нече режимдерин айырмалашат:

а) толук түстүү (True color) – ар бир үч негизги түстүн жарыктыгын коддоо үчүн 256 кошумча түс колдонулат ($2^8=256$, сегиз экилик разряд), башкача айтканда бир пикселдин түсүн коддоо үчүн (RGB системасында), $8*3=24$ разрядды коротуш керек. Бул 16,5 млн түстү аныктоого мүмкүндүк берет. Түстүү графиканы берүү үчүн СМҮК системасынын жардамы менен коддоодо $8*4=32$ экилик разрядга ээ болуу керек ($2^{32}=4,3$ млн.).

б) High Color – ар бир чекитти коддоодо экилик разряддардын санын эки эсеге азайтуудан келип чыккан режим. Бул 16 разряддуу экилик сандардын жардамында коддолот. Бирок бул учурда түстөрдүн диапозону да ошончолук азаят.

Чагылдырылган түстөрдүн саны (K) менен аларды коддоочу биттердин санынын (a) дал келүүсү төмөнкү формула менен аныкталат:

$$K = 2^a$$

Түстөр экилик коддо коддолот. 16 түстүү сүрөттөлүштү коддоо үчүн ар бир пикселге 4 бит ($16=2^4$) керектелет. Ал эми 16 бит (2 байт) $2^{16}=65536$ түрдүү түстөрдү кодго айландыруу үчүн колдонулат. Бир чекиттин түсүн коддоо үчүн 3 байтты (2^4 бит) колдонуу 16777216 (же 17 миллионго жакын) түстүн ар кандай өндөрүн чагылдырууга мүмкүндүк берет.

Экрандын мониторундагы графикалык маалымат

Сүрөттөлүштүн сапаты монитордун чечүүчү жөндөмдүүлүгү менен аныкталат, б.а. сүрөт түзүлгөн чекиттердин саны менен. Чечүүчү жөндөмдүүлүгү канчалык чоң болсо, сүрөттөлүш дагы ошончолук сапаттуу.

Монитордун экраны негизги эки режимде иштей алат: тексттик жана графикалык.

Графикалык режимдер төмөнкүдөй көрсөткүчтөрү менен мүнөздөлөт:

- 1 Чечүүчү жөндөмдүүлүгү** – чекиттердин саны, алардын жардамы менен экранда сүрөттөлүштү алууга мүмкүн болот: мисалы, 1920 x 1080.
- 2 Түстүн тереңдиги** – чекиттин түсүн коддоо үчүн колдонулган биттердин саны, мисалы, 8, 16, 24, 32 бит. Монитордун экранында чагылдырылган түстөрдүн саны төмөнкү формула менен аныкталат:

$$K=2^I,$$

мында K – түстөрдүн саны, I – түстүн тереңдиги же биттик тереңдик.

Таблица. Түстүн тереңдиги (I) жана чагылдырылган түстөрдүн саны (K)

Түстүн тереңдиги (I)	Чагылдырылган түстөрдүн саны (K)
1	2 (кара жана ак)
2	$2^2 = 4$
4	$2^4 = 16$
8	$2^8 = 256$
16 (HighColor)	$2^{16} = 65\,536$
24 (True Color)	$2^{24} = 16\,777\,216$
32 (True Color)	$2^{32} = 4\,294\,967\,296$



АНЫКТАМА

Чечүү – сүрөттөлүштүн бир дюйм өлчөмүнө туура келүүчү пикселдердин саны.

Түстүн тереңдиги – бул пикселдин түсүн коддоо үчүн колдонулган биттердин саны.

Сүрөттөлүштү коддоонун сапаты төмөнкүлөргө көз каранды:

1 Дискреттөө жыштыгына, б.а. сүрөттөлүш бөлүнгөн фрагменттердин өлчөмүнө.

2 Коддоонун тереңдигине, б.а. бир чекит үчүн керек болгон түстүн санына.



АНЫКТАМА

Видеоэс – бул графикалык сүрөттөлүш түзүлгөн оперативдүү эстин бөлүгү

Монитордун экранында сүрөттөлүштү чыгаруу үчүн сүрөттөлүштүн бардык чекиттеринин түстөрү сактала турган компьютердин видеоэси керек.

Видеоэстин көлөмү төмөнкү формула менен эсептелет:

$$V = I * X * Y,$$

мында I – өзүнчө чекиттин түсүнүн тереңдиги,
 X , Y – экрандын горизонталдык жана вертикалык өлчөмдөрү (X тин Y ке көбөйтүндүсү экрандын чечүүчү жөндөмдүүлүгүн берет).

Мисалдар:

1-маселе. Ак-кара (боз градациясыз) растрдык графикалык сүрөттөлүш $10*10$ чекиттик өлчөмгө ээ. Бул сүрөттөлүш эстин кандай көлөмүн ээлейт?

Чыгаруу:

Чекиттердин саны – 100

Болгону 2 түс болгондуктан: ак жана кара, түстүн тереңдиги 1ге ($2^1=2$) барабар.

Видеоэстин көлөмү $100*1 = 100$ бит болот.

Жообу: 100 бит.

2-маселе. Өлчөмү 128×128 пиксель болгон растрдык сүрөттөлүштү сактоо үчүн 4 Кб эс бөлүнгөн. Сүрөттөлүштүн палитрасындагы түстөрдүн мүмкүн болгон максималдык саны канчага барабар?

Чыгаруу:

Сүрөттөлүштү түзгөн чекиттердин санын аныктайбыз. $128 * 128 = 16384$ чекит же пикселдер. Сүрөттөлүш үчүн берилген эстин 4 Кб көлөмүн бит менен туюнталы, анткени $V = I * X * Y$ бит менен эсептелет. $4 \text{ Кб} = 4 * 1024 = 4096$ байт = $4096 * 8$ бит = 32768 бит. Түстүн тереңдигин табабыз:

$$I = V / (X * Y) = 32768 : 16384 = 2$$

$N = 2^I$, мында N – палитрадагы түстөрдүн саны. $N = 2^2 = 4$.

Жообу: 4.

3-маселе. Чечүүчү жөндөмдүүлүгү 1024 x 700 чекиттен жана түстүн палитрасы 65536 түстөн турган монитордун High Color режимин берүүгө керек болгон компьютердин видеоэсинин көлөмүн килобайт менен аныктагыла.

Чыгаруу:

1. $K = 2^I$ формуласы боюнча, K – түстүн саны, I – түстүн тереңдиги. $2^I = 65536$ болгондуктан таблица боюнча түстүн тереңдиги $I = 16$ битке ($2^{16} = 65536$) барабар болот.
2. Сүрөттөлүштүн чекиттеринин саны $1024 * 700 = 716\,800$ гө барабар
3. Видеоэстин талап кылынуучу көлөмү $16 \text{ бит} * 716\,800 = 11\,468\,800$ битке барабар болот.

Эми муну килобайтка которобуз:

11 468 800 бит: 8 = 1 433 600 байт

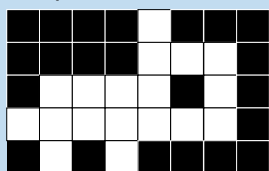
1 433 600 байт: 1024 = 1400 Кб

Жообу: 1400 Кб

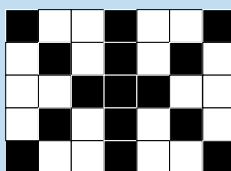
СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

1) Берилген ак-кара сүрөттөр үчүн экилик коддорду түзгүлө жана аларды он алтылык эсептөө ситемасында жазгыла:

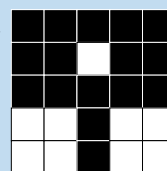
а),



б),



в),



2) Түстүн тереңдиги менен файлдын көлөмү кандай байланышта?

3) Эгерде сүрөттө 65536 түс колдонулса, анын түсүнүн тереңдиги кандай? 256 түстүкүчү? 16 түстүкүчү?

4) 64 түс колдонулган палитра файлда канча орунду (көлөм) ээлейт? 128 түс колдонулсачы?

5) Өлчөмү 800x1000 пиксель болгон жана 1 млн түстү камтыган сүрөт 1 Гб көлөмдөгү флешкага батабы?

6) Монитордун экраны – 1024x768 чекиттен турат, түстүн тереңдиги – 16 бит. Бул графикалык режим үчүн керек болгон видеоэстин көлөмү канчага барабар?



- бөлүм



Компьютер жана ПК

2.1-тема:

Компьютердик графика

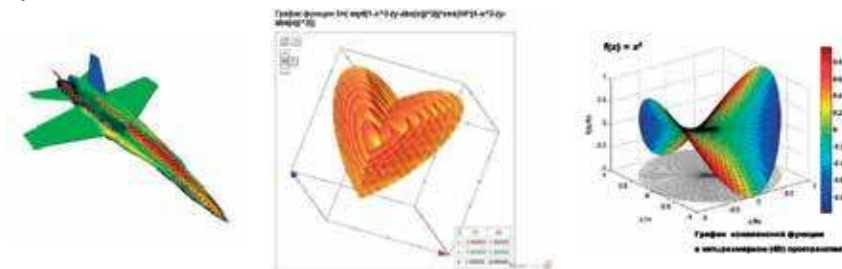
Компьютердик графика – андан ары сактоо жана иштетүү максатында реалдуу дүйнөдөн алынган визуалдык маалыматты санариптештирүүгө, түзүүгө, редакциялоого мүмкүндүк берүүчү компьютердик технологиялардын бөлүмү.

Компьютердик графиканы ар кандай кесиптин ээлери колдонушат: инженерлер жана конструкторлор, архитекторлор жана астрономдор, дизайнерлер жана модельерлер, сүрөтчүлөр жана окумуштуулар. Алардын ар бири үчүн графикалык программалар же графикалык пакеттер деп аталган атайын программалык камсыздоо түзүлөт.

Негизги колдонуу аймактары

1 Иш жана илимий графика илимий изилдөө объекттерин графикалык иштетүү үчүн, андан тышкары сандык берилиштерди график, маалымдама, иллюстрация түрүндө көрсөтмөлүү берүү үчүн кызмат кылат.

Мисалдар:



Инженердик графика менен иштөөчү программалардын түрлөрү:

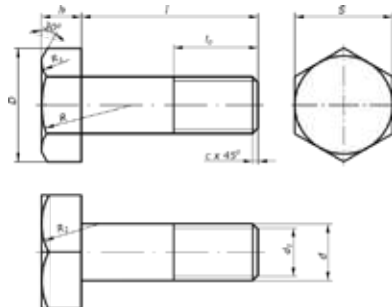
Gnuplot – эки жана үч өлчөмдүү графиканы түзүү үчүн эркин таралуучу программа.

Zhu3D – математикалык функцияларды жана жыйынтыктарды эсептөөчү OpenGLге негизделген интерактивдүү пакет. Анимация эффекти, текстураны өзгөртүү, сүрөттөлүштү айландыруу ж.б. функциялары бар.

SciDAVis – илимий берилиштерге талдоо жүргүзүү жана визуалдаштыруу үчүн эркин ПК. Бир нече графикалык форматтарда, андан тышкары PDF, EPS же SVG форматтарында сактала алган 2D жана 3D-графиктерди түзө алат. Python тили үчүн скрипттік интерфейсти да камтыйт.

MapViewer масштабын, координаталарды ж.б. өзгөртүү менен картаны түзүүгө жана корректирлөөгө мүмкүндүк берүүчү программа. Ал демо-графиялык берилиштерди иштетүүгө жана визуалдаштырууга мүмкүндүк берет.

2 Инженердик графика инженер-конструкторлордун, архитекторлордун, жаңы техниканы ойлоп табуучулардын ишинде кеңири колдонулат. Компьютердик графиканын бул түрү ДСАнын (долбоорлоо системасын автоматташтыруу) негизги элементи болуп саналат. Конструктордук графика тегиздиктеги сүрөттөлүштөр (проекция, кесилиш) жана мейкиндиктеги үч өлчөмдүү сүрөттөлүштөрдү да алууга мүмкүндүк берет.



3 Иллюстративдик графика – иллюстративдик графиканын пакеттери жалпы колдонуудагы ПКга кирет. Мисал – графикалык редакторлор (Gimp, Adobe Photoshop, Adobe Illustrator, CorelDraw ж.б.).

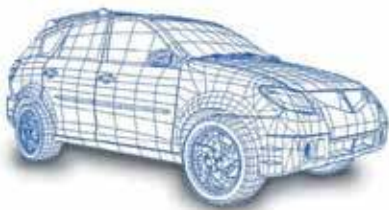
4 Көркөм жана жарнамалык графика – анын жардамы менен жарнамалык роликтер, мультфильмдер, компьютердик оюндар, видеосабактар, видеопрезентациялар жасалат. Бул максаттар үчүн графикалык пакеттер компьютердин көп ресурсун талап кылат: ылдамдыкты жана эсти. Башкалардан айырмаланган өзгөчөлүгү болуп реалдуулукка жакын жана «кыймылдуу» сүрөттөлүштөрдү түзүү мүмкүнчүлүгү эсептелет. Үч өлчөмдүү объекттердин сүрөттөрүн алуу, аларды айландыруу, жакындатуу, алыстатуу, деформациялоо, көп көлөмдөгү эсептөөлөр менен байланышкан. Жарыктын булагына жараша, көлөкөнүн жайгашуусуна жараша, беттин фактурасына жараша объекттин жарыктандыруусун берүү оптиканын мыйзамдарын эске алып эсептөөнү талап кылат.

5 Компьютердик анимация – дисплейдин экранында кыймылдуу сүрөттөлүштөрдү алуу. Сүрөтчү экранда кыймылдагы объекттин баштапкы жана акыркы абалын гана түзөт, ал эми эки абалдын арасындагы калган бардык абалдарды компьютер математикалык эсептөөлөргө таянып аткарат. Белгилүү жыштыкта экранга ырааттуу чыгарылуучу алынган сүрөттөлүштөр кыймылдын иллюзиясын берет.

Сүрөттөлүштөр берүү ыкмасы боюнча экиге бөлүнөт:

- Эки өлчөмдүү графика (**2D**) – тегиздиктеги сүрөттөлүш же видео;
- Үч өлчөмдүү графика (**3D**) – үч өлчөмдүү мейкиндикте моделденген көлөмдүү объекттер, сүрөттөлүш же видео.

3D-моделдөө – бул реалдуу объекттин (автомобиль, имарат, чагылган, астероид) же абстракттуу объекттердин (бир эле сүрөттөлүш кичирейтилген же чоңойтулган масштабда кайталана берген геометриялык фигуралардын фракталынын проекциясы) маалыматтык моделинин көлөмдүү образын иштеп чыгуу.



Үч өлчөмдүү сүрөттөлүштү түзүүнүн ирети:

- 1 Моделдөө – объекттердин үч өлчөмдүү математикалык моделин түзүү.
- 2 Текстуралоо – материалдын касиеттерин жана текстураларын иштеп чыгуу (мисалы, тунуктугу), растрдык, вектордук же процедуралык текстураларды көчүрүп көбөйтүү.
- 3 Жарыктандыруу – виртуалдык жарык булактарын моделдөө (жарыктын багыты, жарыктандыруу даражасы).
- 4 Анимация – объекттердин кыймылын түзүү.
- 5 Динамикалык симуляция – ар түрдүү элементтердин бири-бири менен жана моделденип жаткан процесстер менен өз ара аракеттешүүсүн автоматтык түрдө эсептөө (мисалы, шамал, гравитация ж.б.).
- 6 Визуалдаштыруу (рендеринг) – 3 өлчөмдүү моделди 2 өлчөмдүү мейкиндикте (монитордун экранында) көрсөтүү.

Колдонулган программалар: Autodesk 3ds Max, Autodesk Maya, Autodesk Softimage, Blender, Cinema, 4D Houdini, LightWave 3D.

Мындагы программалардын ичинен үч өлчөмдүү графиканы түзүү үчүн Blender эркин ПК болуп саналат.

Blender татаал үч өлчөмдүү моделдерди жана интерактивдүү оюндарды түзүү үчүн колдонулушу мүмкүн жана өзүнө төмөнкүлөрдү камтыйт:

- моделдөө каражаттарын;
- анимацияларды;
- рендерингди;
- кийинки иштетүүнү;
- видеону үнү менен монтаждоону;
- «түйүндөрдүн» жардамы менен чогултууну.

Blender чөйрөсүндө программалоо Phyton тилинде ишке ашат.

3D-графиканын өзгөчөлөнгөн артыкчылыгы болуп «кошумчаланган реалдуулук» (augmented reality, AR) эсептелет. Бардык айлана-чөйрө компьютердик графикадан түзүлгөн «виртуалдык реалдуулуктан» айырмаланып «кошумчаланган реалдуулукта» компьютердик графика кээ бир бөлүктөрүндө гана колдонулат. Мисалга алсак Инстаграм тиркемесиндеги «Маскалар» ушул технологиянын негизинде түзүлгөн: сиз камераны өзүңүздүн бетинизге багыттайсыз ал эми программа өзү эле беттин сүрөтүнө мультфильмдеги күчүктүкүндөй кулак жана мурунду, көз айнек же болбосо жасалгаларды «жабыштырып» коёт.



Башкача айтканда реалдуу табигый чөйрөдө сүрөттөлүштөрдү таануу технологиясы (маркерлер) колдонулат, ал эми ага кошумчаланган реалдуулуктун программасы виртуалдуу 3D-объекттерди кошуп көрсөтөт. Сиз маркерди өзгөртсөңүз болот: ар кандай багыттарга айланып, ар түрдүүчө жарыктандырып, кээ бир бөлүктөрүн жаап ж.б. Бул учурда экрандагы 3D-объекттердин абалы да ошого жараша өзгөрөт.

СУРОЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Адамдын күзгүдөгү сүрөттөлүшүн визуалдаштыруу кайсы компьютердик графикага кирет?
- 2) 3 өлчөмдүү графика 3D-моделдөөдөн эмнеси менен айырмаланат?
- 3) «Кошумчаланган реалдуулукту» дагы кайсы жерлерде колдонсо боло тургандыгына мисал келтиргиле.

2.2-тема:

Робототехникага киришүү

Биз азыр көптөгөн үйдөгү жана жумуштагы майда-барат иштерди интеллектуалдуу жардамчылардын мойнуна илип койсо боло тургандай заманда жашап жатабыз. Мындай учурларда бизге жардамга роботтор келет.

Робот – бул программалануучу жана адамдын жардамысыз эле сырткы чөйрө менен аракеттешип, андан ары биз киргизген маселелерди аткаруучу механикалык түзүлүш.

Робототехника – программалануучу механикалык түзүлүштөрдү – роботторду долбоорлоо, иштеп чыгуу жана пайдалануу менен байланышкан иштердин чөйрөсү.



БУЛ КЫЗЫКТУУ

«Робот» деген сөздү 1921-жылы өзүнүн эркиндиги үчүн күрөшкөн, адам түспөлдүү жасалма кулдардын классы сүрөттөлгөн «Россумдун универсалдуу роботтору» деген чыгармасында чехиялык драматург Карл Чапек эң биринчи колдонгон. «Robota» деген чех сөзү «аргасыз кулчулук» дегенди түшүндүрөт. Ал эми «робототехника» сөзү 1941-жылы биринчи жолу илимий фантастиканын автору Айзек Азимов тарабынан колдонулган.



Бүгүнкү күндө роботтор көп милдеттерди аткарышат жана көптөгөн колдонулуштарга ээ. Роботтордун колдонулушу төмөнкүдөй аймактарга бөлүнөт:

- **өндүрүштүк роботтор** – аларды колдонуу көптөгөн өндүрүштүк процесстердин (жыйноо, чогултуу, боёо жана таңгактоо) тактыгын жогорулатууга жана ылдамдатууга мүмкүндүк берди. Роботтун мындай түрлөрү техникалык көрүү системалары менен камсыз болгон жана билдиргичтерден (датчиктерден) келген татаал жооптун негизинде чечим чыгара алышат. Алар татаал, коркунучтуу тапшырмаларды аткаруу үчүн колдонулушу мүмкүн, андан тышкары адам аткара албаган тапшырмаларда (бомбаны зыянсыздандырууда, ядролук реакторлорду тейлөөдө, океандын тереңдиктерин изилдөөдө жана космосту изилдөөдө да колдонулат);

- **изилдөөчү роботтор** – опурталдуу жана татаал чөйрөлөрдө (космос мейкиндигин жана күн системасынын планеталарын изилдөөдө ж.б.) тапшырмаларды аткаруу үчүн колдонулат;

- **билим берүүчү роботтор** – окуу маселелерин аткаруу жана долбоорлоо үчүн колдонулат;

Заманбап роботтордун негизги компоненттери:

- **механикалык түзүүчүсү (тулкусу/алкагы)** – бул түздөн түз роботтун конструкциясы, ал ар кандай формада жана өлчөмдө болушу мүмкүн. Роботтор адам түспөлдүү болушу да мүмкүн, ошол эле учурда адамга таптакыр окшошпошу да мүмкүн, анткени роботтун долбоорунда сырткы келбетке эң аз, ал эми анын түрдүү милдеттерди аткарышына көп көңүл бурулат.

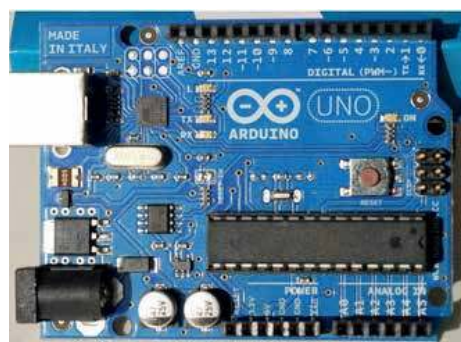
- **Программалык түзүүчүсү (башкаруу системасы)** – роботтун бардык элементтерин башкарат. Мисалы, роботтун билдиргичтери сырткы чөйрө менен аракеттенишкенде борбордук процессорго сигналды жөнөтүшөт, ал болсо программалык камсыздоонун жардамы менен берилиштерди иштетет да логиканын негизинде чечим кабыл алат. Ушундай эле көрүнүш колдонуучу тарабынан командаларды киргизүүдө да аткарылат.



Arduino

Arduino – бул өзүндө механикалык жана программалык түзүүчүлөрдү камтыган бир платалуу микрокомпьютер. Бул окуучулар арасында эң кеңири тараган аппараттык платформа болуп эсептелет, ал ар түрдүү роботторду курууга окутуу үчүн колдонулат. Arduino автономдуу интерактивдүү роботторду түзүүдө да жана ошону менен бирге компьютерде аткарылып жаткан программалык камсыздоого кошулууда да колдонула берет.

Робототехникада Arduino көп учурда **роботтун мээси** катары колдонулат.



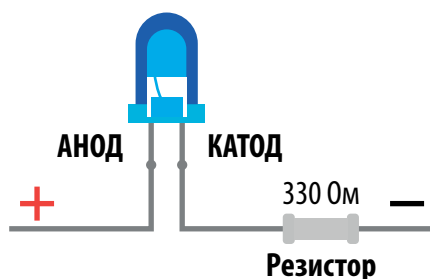
Плата каалагандай жөнөкөй түзүлүштөрдү: кнопка (баскыч), датчик (билдиргич), мотор, экран сыяктуу кошууга арналган көптөгөн аналогдук жана санып-риптик портторго ээ.

Arduinonун ичинде жүктөгүч (загрузчик, BootLoader) – сиз ар дайым аппаратты иштеткенде кошо ишке кирген программа орнотулган болот. Arduinoну программалоо үчүн платаны персоналдык компьютерге USB порт аркылуу туташтырып коюу жетиштүү болот.

Программаны жазуу үчүн эркин таралган Arduino IDE редактору колдонулат. Аны төмөнкү шилтеме аркылуу көчүрүп алууга болот:

<http://www.arduino.cc/en/Main/Software>.

1-маселе. Arduinoго жарык диодун кошобуз жана аны өчүп, күйүп тургандай кылабыз. Бул робототехникадагы бирден-бир базалык схема болуп саналат. Платага жарык диодун кошуудан мурда анын түзүлүшүн карайлычы. Жарык диоду бири экинчисинен узун келген эки чыгуучу зымдан (бутчалардан) турат. Узун чыгуучусу ток булагынын (мисалы батарейка) оң уюлуна туташкан – **анод** болуп саналат.

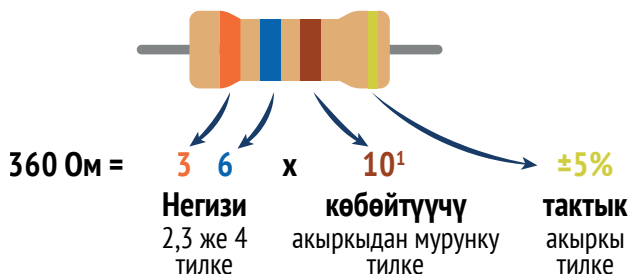


Катод – кыска чыгуучусу, минус менен же жалпы өткөргүч менен бириктирилет.

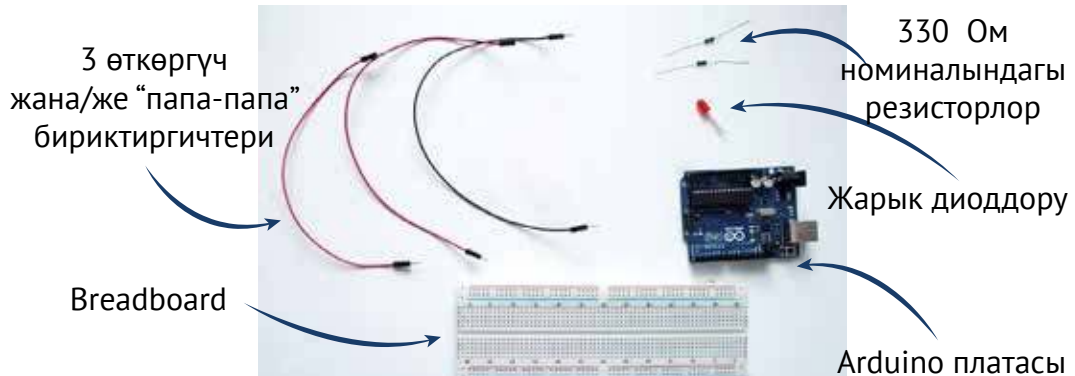
Резистор – бул ток үчүн жасалма «тоскоолдук». Ал электр энергиянын бөлүгүн жылуулукка айландыруу менен токтун күчүн чектейт. Резистордун чыгуучулары плюс да, минус да болуп эсептелбейт, ошондуктан ал уюлдуу эмес болуп саналат.

Резистордун эң негизги мүнөздөмөсү болуп каршылыгы саналат. Каршылыктын өлчөө бирдиги – **Ом**. Каршылык канчалык көп болсо, ошончолук көп ток жылуулукка айланат.

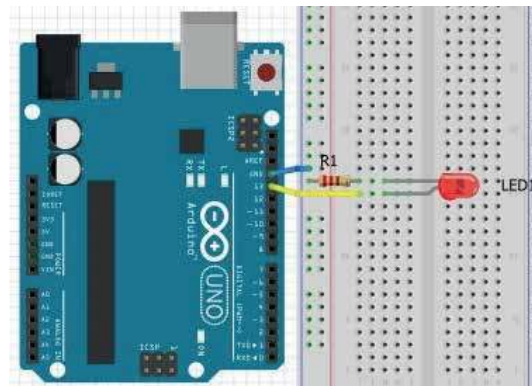
Каршылык деңгээлин же **резистордун номиналын** резисторду курчаган түстүү тилкелер менен аныкташат:



Моделди чогултуу үчүн бизге төмөнкүлөр керек болот:



Жарык диодун туташтыруу схемасы:



Бул схеманын иштеши үчүн төмөнкү программа керек болот (программаны силер Arduino IDE редакторуна көчүрүп алгыла)

```
void setup() { //программаны баштоо үчүн setup милдеттүү функциясы
pinMode (13, OUTPUT); //13-пинди чыгууга (OUTPUTка) орнотуу.
}
void loop() { //loop функциясы программаны циклда кайталайт
digitalWrite (13, HIGH); //жарык диодун жандыруу үчүн команда
delay (1000); //1000 миллисекундага (1 сек.) кармалуу
digitalWrite (13, LOW); //жарык диодун өчүрүүгө команда
delay (1000); //1000 миллисекундага (1 сек.) кармалуу
}
```

Жыйынтыгында биз жарык диодунун 1 секунда сайын өчүп, күйгөнүн байкайбыз.

Arduino программалоо чөйрөсүндө функцияны чакыруу **void** сөзүнөн башталат.

Void setup() функциясы милдеттүү функция болуп, программанын башында ишке кирет. Бул функция менен аткарылган команданын блоктору фигуралуу кашаанын ичинде жазылат. Бул командаларды ал бир жолу гана – программанын башталыш учурунда аткарат.

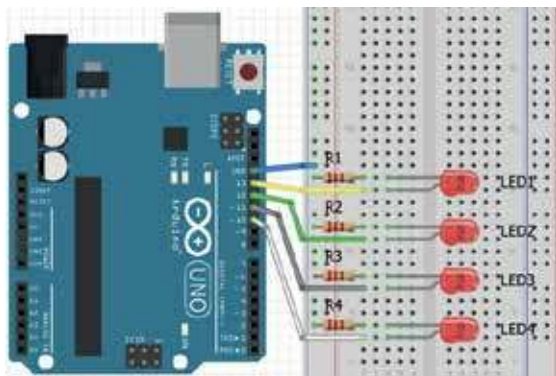
Биздин мисалда **Void setup**тын ичинде **pinMode** нускамасы 13-пинди «чыгуу» (OUTPUT) режимине орнотот.

Void loop функциясы **Void setup** функциясынан кийин чакырылат жана командаларды Arduino платасы иштеп турган (электр тогу берилип турганда) бардык убакытта (циклдик түрдө) аткарыла берет. Биздин мисалда **delay** (1 секундага кармалуу) функциясы жарык диодунун өчүп-күйгөнүн ажыратып байкоого мүмкүндүк берет. Ансыз өчүрүп/күйгүзүү сигналын ажыратуу мүмкүн эмес.

2-маселе. Эми биздин схемага дагы 3 жарык диодун кошулу. Бул схема «жалпы катоду менен кошулуу схемасы» деп аталат.

```
void setup() {
  pinMode (13, OUTPUT);
  //колдонулуп жаткан пиндерди чыгууга орнотолу
  pinMode (12, OUTPUT);
  pinMode (11, OUTPUT);
  pinMode (10, OUTPUT);
}

void loop() {
  digitalWrite (13, 1);      //жарык диодун күйгүзөбүз
  delay (1000);
  digitalWrite (13, 0);     //жарык диодун өчүрөбүз
  delay (1000);             //1000 миллисекундага (1 секунда) кармалуу
  digitalWrite (12, 1);
  delay (1000);
  digitalWrite (12, 0);
  delay (1000);
  digitalWrite (11, 1);
  delay (1000);
```

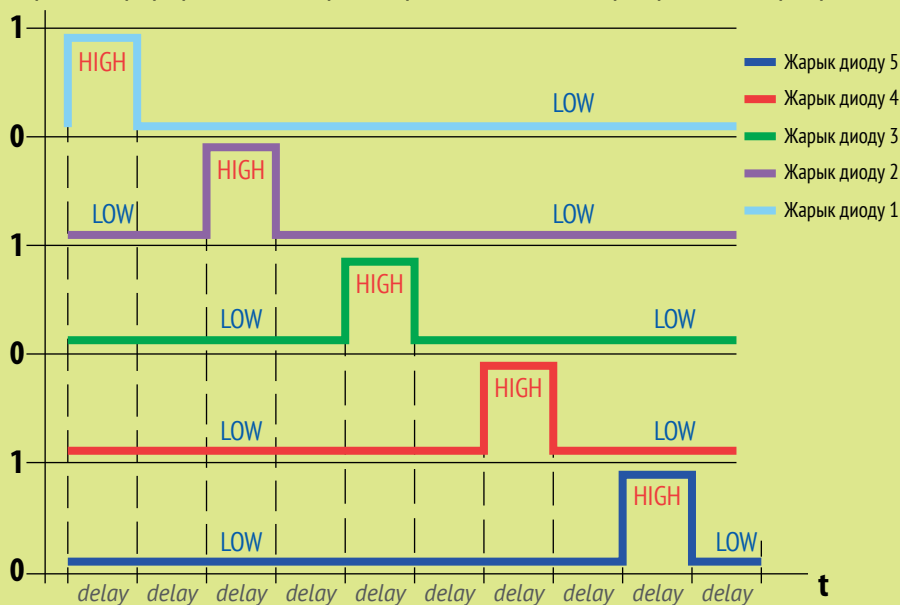


```
digitalWrite (11, 0);
delay (1000);
digitalWrite (10, 1);
delay (1000);
digitalWrite (10, 0);
delay (1000);
}
```

Силер байкагандай, биз HIGH жана LOW командаларын 1 жана 0гө алмаштырдык, бул силердин программаңардын иштешине тоскоолдук кылбайт.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Роботтун аяктары кандай аталат?
- 2) Интернеттен робототехниканын 3 мыйзамы боюнча маалымат тапкыла. «Терминатор» фильминдеги башкы каарман Арнольд Шварценеггер ушул мыйзамдардын чегинде иш алып бардыбы? Айтып бергиле.
- 3) Бардык жарык диоддору бир убакта күйүп, кайра бир убакта өчүшү үчүн кандай кылуу керек?
- 4) Беш жарык диоду кошулган схеманы чогулткула жана чуркоочу жарыктар үчүн төмөнкү алгоритм боюнча программа түзгүлө.



3

- бөлүм



Программалоо

3.1-тема:

Рекурсия

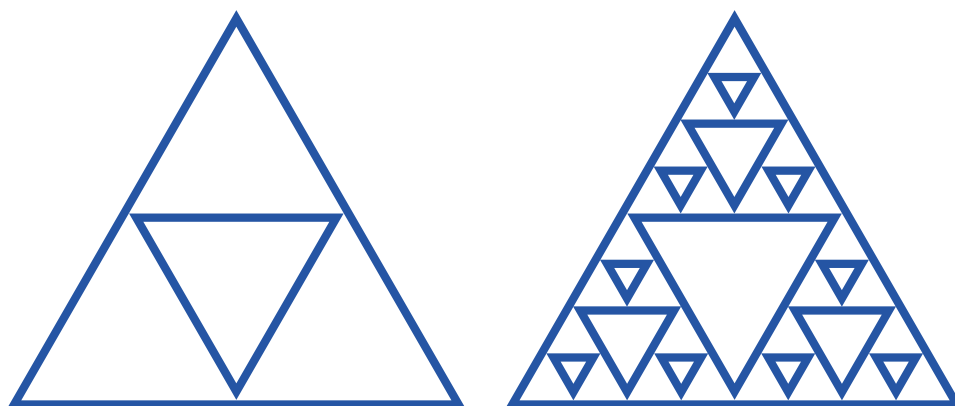
8-класстын курсунан силер билесиңер, көпчүлүк учурда бир функция башка функцияны чакырышы мүмкүн. Бирок ошол эле функция өзүн өзү да чакырышы мүмкүн. Программа түздөн түз өзүн чакырган (жөнөкөй рекурсия) же кыйыр (башка функциялар аркылуу), мисалы, А функциясы В функциясын чакырат, ал эми В функциясы А ны чакырган кырдаалдар **рекурсия** деп аталат.



Байкагандай, кыйыр кайрылууда чынжырдагы бардык функциялар – рекурсивдүү.

Рекурсивдүү объекттердин популярдуу мисалдары болуп – **фракталдар** эсептелет. Мындай ат менен математикадагы өзүнө **окшоштукка ээ** фигураларды аташат. Бул демек, алар ар бири бүтүн фигурага окшош болгон кичинекей өлчөмдөгү фигуралар менен куралган дегенди түшүндүрөт.

1-сүрөт. Серпинскийдин үч бурчтугу (1915-жылы польшалык математик В. Серпинский тарабынан сунушталган эң биринчи фракталдардан болуп саналат).



Тең жактуу үч бурчтук кичирээк өлчөмдөгү 4 барабар үч бурчтукка бөлүнөт (сол жактагы сүрөт), андан кийин борбордогудан башка ар бир алынган үч бурчтук андан ары да майда 4 барабар үч бурчтукка бөлүнүшөт ж.б.

1-маселе. $n!$ саны үчүн факториалды эсептөө мисалында рекурсия кантип иштээрин карайлы. Мисалы, 3 санынын факториалын эсептөө үчүн, $3*2*1$ ге көбөйтүү керек, б.а. $n*(n-1)*((n-1)-1)$ амалын аткаруу керек.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(3))
>>>
6 #ЖЫЙЫНТЫК
```



ЭСИҢЕ ТУТ

n санынын факториалы деп, 1ден n ге чейинки бардык натуралдык сандардын көбөйтүндүсүн айтабыз:

$$n! = 1 * 2 * 3 * 4 * \dots * n$$

Мисалы, 5 санын факториалы 120га барабар $120 (5! = 1 * 2 * 3 * 4 * 5)$.

Рекурсивдүү функциялар программалоодо көптөгөн маселелерди чечишет, бирок тилекке каршы, аларды жазууда көп учурда каталар кетет. Эң көп таралган ката – бул чексиз рекурсия, бул учурда функцияны чакыруу чынжыры эч качан бүтпөйт жана компьютердин эси толуп калмайынча улана берет.

Көптөгөн чексиз рекурсиянын негизги эки себеби:

- 1** Рекурсиядан чыгуунун туура эмес жазуусу. Мисалы, эгер биз факториалды эсептөө программасында `if n==0` деген текшерүүнү унутуп койсок, анда `factorial(0)`, `factorial(-1)` ди чакырат, ал болсо `factorial(-2)`-ни ж.у.с.
- 2** Функциянын параметрлеринин туура эмес жазылышы. Мисалы, эгер `factorial(n)` функциясы кайрадан эле `factorial(n)` функциясын чакырса да чексиз чынжыр алынат.

Ошондуктан, рекурсивдүү функцияны иштеп чыгууда эң биринчи кезекте **рекурсиянын базалык шарты** деп аталган рекурсияны аяктоо шартын туура түзүү керек.

Рекурсиянын түз жана тескери жүрүшү

Рекурсиянын кийинки деңгээлине кирүүгө чейин функция тарабынан аткарылган аракеттер рекурсиянын түз жүрүшүндө аткарылгандар деп аталат. Ал эми тереңирээк деңгээлден учурдагы деңгээлге кайткандагы аткарылган аракеттер – рекурсиянын тескери жүрүшүндөгү аткарылгандар деп аталат.

2-маселе. Натуралдык санды экилик системага которо турган функцияны түзүп көрөлү. Санды экилик системага которуунун стандарттык алгоритмин мындай жазса болот:

```
def printBin (n):
    while n!= 0:
        print (n % 2, end = '')
        n = n // 2
printBin(14) #функцияны чакыруу үчүн мисалы 14 санын киргизели
>>>
0111 #жыйынтык
```

Негизги проблема бул, экилик сандар тескери, б.а. биринчи кезекте акыркы разряд чыгарылгандыгында.

Бул маселени чыгаруунун ар кандай ыкмалары бар. Алардын бардыгы бөлүүдөн кийин калдыктарын эстеп калып (мисалы, символдук сапка), андан соң бардык жыйынтык алынгандан кийин гана аны экранга чыгарууга негизделген.

Рекурсияны колдонуп көрөлү. Идея мындай: n санынын экилик жазуусун чыгаруу үчүн, алгач $n//2$ санынын экилик жазуусун чыгаруу керек, андан кийин анын калдыгын бөлүп чыгуу керек ($n\%2$). Эгерде алынган сан (параметр) нөлгө барабар болсо, анда функциядан чыгуу керек. Мисал үчүн 14 санын алалы:

```
14 // 2 = 7, 7 % 2 = 1
7 // 2 = 3, 3 % 2 = 1
3 // 2 = 1, 3 % 2 = 1
1 // 2 = 0, чыгуу
```

Мындай алгоритм эң жөнөкөй программаланат. `print_bin` аталыштагы функцияны жазалы:

```
def print_bin (n):
    if n == 0: return 0
    print_bin (n // 2)
    print (n % 2, end = '') #калдыктарды бир сапка жазуу
printBin (14) #функцияны чакыруу үчүн мисал катары 14 санын
киргизебиз
>>>
1110 #жыйынтык
```


Ушуну эле циклдин жардамында да жасаса болот. Мындан маанилүү корутунду чыгат: **рекурсия циклди алмаштырат**. Ошону менен бирге программа көпчүлүк учурда түшүнүктүүрөөк болуп калат.

3-маселе. а санын b даражасына көтөрүү (a^b) үчүн рекурсивдүү функцияны жазалы.

Санды даражага көтөрүүнүн алгоритми төмөнкүдөй аткарылат:

$(a \cdot \dots (a \cdot (a \cdot (a \cdot (a \cdot 1))))))$

Математика курсунан биз $a^{**0} = 1$ экенин билебиз.

Маселени чыгаруу:

```
def func_step(a, b):
    if b == 0:
        return 1
    else:
        return a * func_step(a, b-1)
print(func_step(2, 4)) #мисалы, 2нин 4-даражасы
>>>
16 #жыйынтык
```



ЭСИҢЕ ТУТ

Чексиз рекурсиянын жыйынтыгын чыгарууну токтотуу үчүн **Ctrl+C** баскычтарынын комбинациясын консолдо басуу керек.



БУЛ КЫЗЫКТУУ!

Рекурсиянын адамдын жашоосундагы мисалдары:

- Сиз банкка процентке акча салдыңыз жана эсептелген проценттер сиздин эсебиңизде банкта калат жана аларга да проценттер эсептелет. Сиз качан банктан чогулган акчаларды алып кеткенче, процесс улана берет.
- Бири-бирине караган күзгүлөрдөгү объекттин чагылышы да чексиз рекурсиянын мисалы болот.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) n натуралдык саны берилген. 1ден n ге чейинки бардык натуралдык сандарды чыгаргыла.
- 2) n берилген саны үчүн 1ден n ге чейинки бардык сандардын суммасын чыгаруучу рекурсивдүү функцияны жазгыла.

3.2-тема:

Массивдерди иштетүү алгоритмдери

7-, 8-класстарда силер тизме, кортеж жана сөздүк түрүндөгү массивдерди үйрөн баштадыңар, алар менен ар кандай амалдарды жасадыңар жана массивдерге маалыматты киргизип жана чыгарганды билдиңер.

Бул темада биз массивдеги берилиштерди иштетүүнүн негизги стандарттык алгоритмдерин тереңирээк карайбыз:

- издөө
- модификациялоо
- сорттоо

ОШОНДОЙ ЭЛЕ КАРА

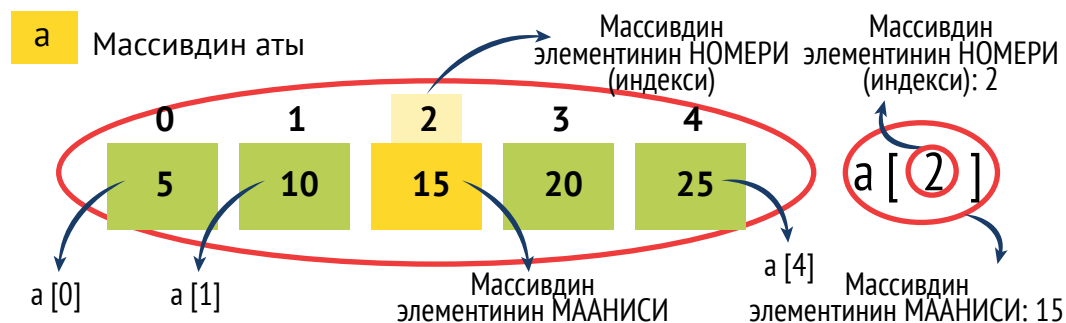
3.6-тема 8-класс

Массивдер



ЭСИҢЕ ТУТ

Массив – бул жалпы атка ээ болгон эсте жакын жайгашкан (кошуна уячаларда) өзгөрмөлөрдүн тобу. Массивдеги ар бир уяча уникалдуу номерге (индекске) ээ.



Массивден издөө

Издөө алгоритмин маселенин мисалында карап көрөлү. **a** – массивдин аты, **n** – массивдеги элементтердин саны (бүтүн сан), ал эми **i** – өзгөрмөсү тизменин элементтеринин индексин билдирет.

1-маселе. Массивден **x** өзгөрмөсүнүн маанисине барабар болгон элементти табуу керек же ал элемент жок деген билдирүүнү чыгаруу керек. Чыгаруу алгоритми – бул массивдин биринчиден аягына чейинки бардык элементтерин карап чыгуу. Качан гана **x** ке барабар элемент табылганда, циклден чыгып, жыйынтыкты көрсөтүш керек.

Ал үчүн **nx** өзгөрмөсү менен циклди колдонобуз, массивдин бардык элементтерин иргейбиз жана керек болгон маани табылса эле циклди аяктайбыз.

```
a = [1, 5, 4, 31, 10] #5 элементи бар массив берилген
x = int(input('Введите x: ')) #Изделген x маанисин киргизүү
nx = 0 #nx өзгөрмөсү табылган элементтин номерин сактайт
for item in a:          #a массивинин элементтери үчүн
    if item == x:        #эгерде элемент x ке барабар болсо
        nx = item       #анда ал nx өзгөрмөсүнө жазылат
        break           #циклди аяктайт
if nx >= 1:              #nx - өзгөрмөсү өзгөрдү
    print ('Табылды!')  #изделген маани табылды деп чыгарат
else:
    print ('Табылган жок')
```

Циклден чыгуу үчүн **break** оператору колдонулат, табылган элементтин номери «**nx**» өзгөрмөсүндө сакталат. Эгерде анын мааниси 0 боюнча калса (циклдин аткарылышында өзгөрбөсө), анда массивде **x** ке барабар элемент жок.

Бул маселени дагы бир жол менен чыгарса болот. Ал үчүн массивде **x** ке барабар биринчи табылган элементтин индексин кайтарган **index** функциясын колдонуу керек:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18] #массивдин мисалы
print (a.index (23))
>>>
4
```

Эгерде массивде изделген элемент жок болсо, программа катаны кайтарат жана ишти токтотот.

2-маселе. Массивде **m** өзгөрмөсүнө жазылып кала турган максималдуу элементти табабыз. Ал үчүн массивдеги бардык элементти биринин артынан экинчисин карап чыгуу керек. Эгерде массивдин берилген элементи мурунку максималдуу элементтен чоң болсо (**m** өзгөрмөсүндө жайгашкан), анда **m** өзгөрмөсүндө максималдуу элементтин жаңы маанисин эстеп калабыз.

Массивдин элементтери бизге белгисиз болгондугуна байланыштуу **m** ге нөл же терс санды жазышыбыз керек. Эгерде ал **m = a[0]** болсо, анда иргөө цикли экинчи элементтен башталат, башкача айтканда **a[1]**ден:

```

a = range (1, 20)
m = a[0]
for i in range (1, 20):
    if a[i] > m:
        m = a[i]
print (m)

```

Мына, дагы бир варианты:

```

a = range (1, 20)
m = a[0]
for x in a:
    if x > m:
        m = x
print (m)

```

Мунун айырмасы, ал индекс-өзгөрмөнү колдонбойт, бирок **a[0]** элементин эки жолу карайт (2-жолу – бардык элементтерди иргөө циклинде).

Максималдык жана минималдык элементтерди издөө көп учурда колдонулгандыктан, Python тилинде тиешелүү **max()**, **min()** камтылган функциялары каралган.

```

a = range (1,20)

m = max (a)
print (m)

```

Массивди модификациялоо

Массивдер менен иштөө үчүн жана андагы берилиштерди өзгөртүү үчүн, Python тилинде көптөгөн ар кандай функциялар бар, мисалы:

ФУНКЦИЯ	МААНИСИ	МИСАЛ
print (a)	а тизмесин экранга чыгарат	<pre> a = [16, 'b', 34, 'c'] #бардык мисалдар үчүн базалык тизме print (a) >>> [16, 'b', 34, 'c'] </pre>
append ()	Тизменин артына бир элемент кошот	<pre> a.append (18) print (a) >>> [16, 'b', 34, 'c', 18] </pre>
clear ()	Тизменин бардык элементин өчүрөт	<pre> a.clear () print (a) >>> [] </pre>
count ()	Берилген мааниси менен элементтердин санын кайтарат	<pre> print (a.count (16)) #тизмеде мааниси 16га барабар канча элемент бар экендигин эсептейт >>> 1 </pre>

ФУНКЦИЯ	МААНИСИ	МИСАЛ
extend ()	Базалык тизменин аягына башка тизмени кошот	<pre>B = [18, 'h'] #экинчи тизме a.extend (b) print (a) >>> [16, 'b', 34, 'c', 18, 'h']</pre>
index ()	Биринчи табылган окшош элементтин индексинин номерин кайтарат	<pre>print (a.index (34)) >>> 2</pre>
insert ()	Элементтерди индекси боюнча коюп чыгат	<pre>a.insert (1, 22) #индексти эсептөө 0ден башталгандыктан, базалык тизмеде 1 индексинде 'b' турат, демек анын ордуна 22 цифрасы коюлуп, калгандары оңго жылат. print (a) >>> [16, 22, 'b', 34, 'c']</pre>
pop ()	Берилген индекстеги элементти өчүрөт	<pre>a.pop (0) #0 индексиндеги маанини өчүрүү print (a) >>> ['b', 34, 'c'] a.pop () print (a) >>> [16,'b', 34] #эгерде маанилери берилбесе, анда акыркы элемент өчүрүлөт</pre>
remove ()	Берилген мааниси менен 1-элементти өчүрөт, эгерде элемент табылбаса ValueError билдирүүсү чыгат	<pre>a.remove (34) print (a) >>> [16, 'b', 'c']</pre>
reverse ()	Тизменин элементтерин тескери иретте жайгаштырат	<pre>a.reverse () print (a) >>> ['c', 34, 'b', 16]</pre>
sort ()	Тизмени сорттойт (бир типтеги элементтүү тизме үчүн гана)	<pre>a = [16, 8, 34, 3] #тизмеде жалаң бүтүн (int) сандар a.sort () print (a) >>> [3, 8, 16, 34] #өсүү тартибинде сорттойт a = ['m', 'b', 'o', 'c'] # тизмеде жалаң символдор (str) a.sort () print (a) #алфавит боюнча сорттойт >>> ['b', 'c', 'm', 'o']</pre>

Буллардын кээ бирөөлөрүн тереңирээк карайлы: массивдин реверси, массивдин элементтерин жылдыруу жана керектүү элементтерди тандоо.

Массивдин реверси

Массивдин реверси (*reverse*) – бул анын элементтерин тескери тартипте жайгаштыруу: биринчи элемент акыркысы болуп, ал эми акыркы элемент биринчи жайгашып калат.

Python тилинде массивдер Одөн башталат. Ошондуктан эгерде элементтердин жалпы саны N болсо, анда i үчүн каршысындагы элемент төмөнкү формула менен аныкталат: $N - i - 1$. Мисалы (төмөнкү таблицаны карагыла), 2 индексиндеги элементти алмаштыра турган элементтин индексин табуу үчүн, төмөнкү формула менен чыгарабыз:

$$N - i \\ 9 - 2 - 1 = 6$$

Демек, 2 индексиндеги элемент 6 индексиндеги элемент менен алмашат.

Массивдин элементтери	16	29	-5	-11	23	14	-7	23	18
Индекстер	0	1	2	3	4	5	6	7 же (n-2)	8 же (n-1)

Элементтин экинчи түгөйүн табуу үчүн, биз массивдин биринчи жарымындагы эле индекстерди карайбыз. Бул болсо циклди массивдин ортосунан токтотуу керек дегенди түшүндүрөт:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a) #массивдеги элементтердин санын эсептейбиз
for i in range(n//2): #массивдин 1-жарымынын индекстери
    a[i], a[n-i-1] = a[n-i-1], a[i] #элементтердин ордун ал-
    маштырабыз
print(a)
>>>
[18, 23, -7, 14, 23, -11, -5, 29, 16]
```

Массивди реверстөө амалын стандарттык **reverse()** методунун жардамында да аткарса болот:

```
a.reverse ()
```

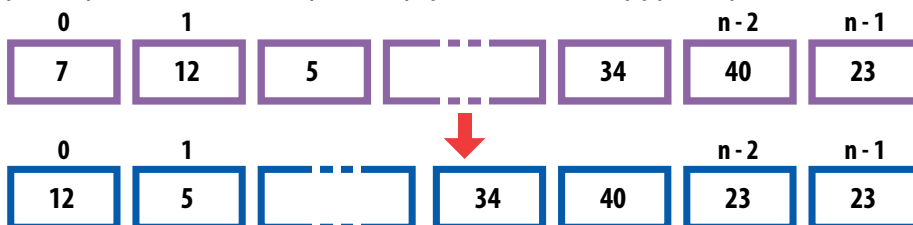


ЭСИЦЕ ТУТ

Python тилинде индекстердин маанилери терс болушу да мүмкүн. Бул учурда номерлөө аягынан баштап жүрөт. Мисалы, массивдин эң акыркы элементинин индекси -1, андан беркисиники -2 ж.б.

Массивдин элементтерин жылдыруу

Массивдин элементтерин коюуда же өчүрүүдө алардын бардыгын же бөлүгүн тиги же бул жакка жылдыруу керек болот. Массивди көп учурда 1-элементи сол жагында жайгашкандай, таблица түрүндө көрсөтүшөт. Ошондуктан солго жылдыруу – бул **a[1]** **a[0]**дун ордуна, **a[2]** **a[1]**дин ордуна ж.у.с. бардык элементтерди бир уячага жылдыруу болуп саналат.



Акыркы элемент өзүнүн ордунда эле калат, б.а. кайталанат, анткени ал жаңы маанини эч кайдан ала албайт, массивдин элементтери түгөндү.

Алгоритмин жазалы:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a)
for i in range (n-1):
    a[i] = a[i+1]
print(a)
>>>
[29, -5, -11, 23, 14, -7, 23, 18, 18]
```

Көңүл бурсаңар мында массивдин чегинен чыгып кетпеш, б.а. жок элемент $a[n]$ ге кайрылбашы үчүн цикл $a=[n-1]$ де аяктап жатат.

Биз көрүп тургандай, биринчи элемент жок болуп кетти, ал эми акыркы элемент эки жолу кайталанды. Негизи биринчи элементтин мурунку маанисин акыркынын ордуна жазып койсо деле болот. Мындай жылдыруу циклдик деп аталат. Ал үчүн алдын ала (цикл башталганга чейин) биринчи элементти **c** кошумча өзгөрмөсүнө сактап коюу керек, ал эми цикл аяктагандан кийин аны массивдин акыркы уячасына жазып салуу керек:

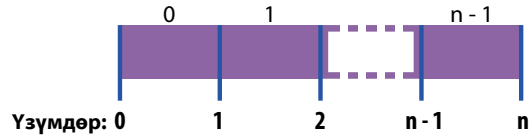
```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a)
c = a[0]
for i in range(n-1):
    a[i] = a[i+1]
a[n-1] = c
print(a)
>>>
[29, -5, -11, 23, 14, -7, 23, 18, 16]
```

Pythonдун камтылган мүмкүнчүлүктөрүн колдонуп жылышууну оңой аткарсак болот:

$$a = a[1:n] + [a[0]]$$

Бул жерде $a[0]$ биринчи элементи $a[1:n]$ кесилип алынган массивдин артына коюлат. Эми ал нөлдөн эмес бирден башталып калат.

Оң жактагы сүрөттөн биз $a[0:2]$ үзүмү Одөн 2ге чейинки бардык элементтерди камтыйт, б.а.: $a[0]$ жана $a[1]$ ди.



Керектүү элементтерди тандоо

Кайсы бир шартты канааттандырган **a** массивинин бардык элементтерин **b** массивине чогултуу керек. Ал үчүн баштапкы массивдин элементтерин иргеп, улам кийинки элемент бизге туура келсе, анда экинчи эсептегичти колдонуп, аны жаңы массивге кошобуз. Мисалы, 2-тизмеге так сандарды чогулталы:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
b = []
for x in a:
    if x % 2 == 0:
        b.append(x)
print(b)
>>>
[16, 14, 18]
```

Чыгаруунун экинчи варианты – шарты менен генераторду колдонуу:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
b = [x for x in a if x % 2 == 0]
print(b)
```

Бул жерде **b** тизмесине 2ге гана бөлүнгөн элементтер тандалды.

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) Массивди $(0, 20)$ интервалындагы кокустук сандар менен толтургула. X санын киргизгиле жана X ке барабар бардык маанилерди тапкыла.

2) N элементтен турган сан маанисиндеги бир өлчөмдүү массив берилген. Массивдин элементтерин оң жакка айланма жылдырууну аткар, б.а. $a(1) \rightarrow a(2)$; $a(2) \rightarrow a(3)$; ... $a(n) \rightarrow a(1)$.

3.3-тема:

Тизмелерди сорттоо

Көпчүлүк учурда керектүү маалыматты издөөнү жеңилдетүү үчүн биз сорттоону колдонобуз. Мисалы, сөздүктө алфавит боюнча сөздөрдү сорттоо издөөнү жеңилдетет.

Программалоодо **сорттоо** – бул массивдин элементтерин берилген иретте жайгаштыруу болуп саналат.

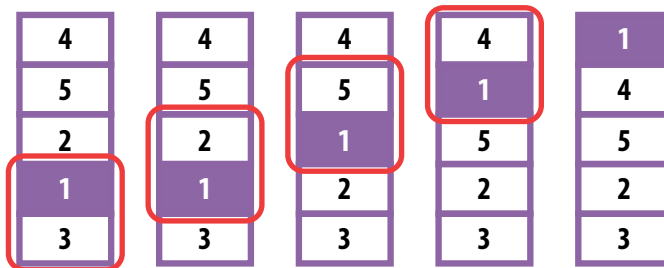
Сорттоо ирети ар кандай болушу мүмкүн: сандар үчүн адатта маанилеринин өсүү (кемүү) тартибинде сорттоо каралат. Мисалы, [3 1 0 5 2 7] бир өлчөмдүү массивин өсүү тартибинде сорттоодо [0 1 2 3 5 7] массиви алынат. Символдук берилиштер адатта алфавиттик тартипте сорттолушат.

Негизи сорттоонун көптөгөн ыкмалары ойлоп табылган. Жалпысынан аларды эки топко бөлүп караса болот: 1) жөнөкөй, бирок өтө жай иштөөчү (чоң массивдер үчүн), 2) татаал, бирок тез иштөөчү.

Сорттоонун эң эле көрсөтмөлүү методдорунун бири болгон – «көбүкчө методун» карайлы.

Көбүкчө методу (алмаштыруу менен сорттоо)

Бул методдун аты белгилүү физикалык кубулуштан алынган: суудагы аба көбүкчөсү өйдө көтөрүлөт. Массивди бул ыкма менен сорттоо үчүн, массивден солдон оңду көздөй коңшу элементтерди экиден салыштырып өтүү керек. Качан сол жактагы элемент оң жактагысынан чоң болуп калса, анда алардын ордун алмаштыруу керек. Ошентип 1-өтүштөн кийин эң чоң элемент тизменин акырына келип калат. Эми тизмени кайрадан иргеп чыгуу керек. 2-өтүштө эң чоң элемент тизменин аягында тургандыктан аны салыштырбайбыз. Б.а. салыштыруулардын саны 1ге азаят. Демек, циклдеги өтүштөрүнүн саны тизменин элементтеринин санынан 1ге аз болот.



Өзүнүн ордуна биринчи (минималдык) элементти орноткон биринчи өтүштү псевдокоддо мындай жазса болот:

```
i   үчүн n-2ден 0гө чейин, кадам -1
    if (a[j] > a[j+1])
        ордун алмаштыр a[j] жана a[j+1]
```

Мында *i* өзгөрмөсү минималдык элемент жазылган уячанын индексин сактайт. Алгач бул биринчи уяча болот. *j* өзгөрмөсү учурда каралып жаткан уячаны билдирет.

Бир өтүштө мындай цикл бир элементти гана орунга коёт. Экинчи элементти «тартыш» үчүн, циклдин башындагы нин акыркы мааниси гана айырмалуу болгон ушул сыяктуу эле дагы бир циклди жазуу керек. Эң чоң элемент өзүнүн ордунда тургандыктан ага тийишпишибиз керек:

```
for i in range(n-1): #өтүүлөрдүн саны
    for j in range(n-i-1): #акыркы элемент каралбайт
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
```

Ушундай циклдерден $n-1$ санда жасоо керек, б.а. тизмедеги элементтердин санынан 1ге кем. Эмне үчүн n эмес? Анткени эгерде $n-1$ элемент өздөрүнүн ордуна коюлса, анда калганы автоматтык түрдө өзүнүн ордуна турат – башка орун жок.

Көбүкчө методу боюнча толук программаны жазалы:

```
from random import randint
n = 10
a = [randint(1,99) for n in range(n)]
print(a)
for i in range(n-1): #тизмеден өтүүлөрдүн саны
    for j in range(n-i-1): #салыштыруулардын саны iге азаят
        if a[j] > a[j+1]: #j жана j+1 элементтерин салыштырат
            a[j], a[j+1] = a[j+1], a[j] #керек болсо элементтердин ордун алмаштырат
print(a)
>>>
[5, 84, 90, 37, 30, 32, 29, 62, 17, 99]
[5, 17, 29, 30, 32, 37, 62, 84, 90, 99]
```

Тандоо методу

Дагы бир популярдуу сорттоо методу – бул тандоо методу. Алгач бардык массив каралып чыгат, минималдык элемент табылганда ал массивдин эң башына коюлат. Экинчи жүрүштө калган бардык, б.а. экинчиден баштап акыркы элементке чейин каралат, кайрадан эң кичине элемент табылат жана ал 2-орунга жылдырылат. Андан ары 3-элементтен баштап дагы эң кичине элементи табылат жана 3-орунга жылдырылат, ж.б.у.с. $(n-1)$ -элементке чейин.

Тизмени тандоо методу боюнча сорттоонун алгоритмин жазалы. Мында i өзгөрмөсү минималдык элемент жазылган уячанын индексин сактайт, ал эми j өзгөрмөсү каралып жаткан элементти сактайт.

```
for i in range(n-1):
    n_min = i
    for j in range(i+1, n):
        if a[j] < a[n_min]:
            n_min = j
    if i != n_min:
        a[i], a[n_min] = a[n_min], a[i]
```

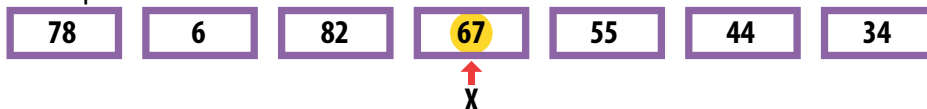
Бул жерде табылган минималдык элемент өзүнүн ордунда турбаса гана орун алмашуу жүрөт, б.а. $i! = n_{min}$ болгондо. Минималдык элементтин номерин издөө циклда аткарылгандыктан, бул сорттоо алгоритми да камтылган циклди түзөт.

«Тез сорттоо» же quicksort

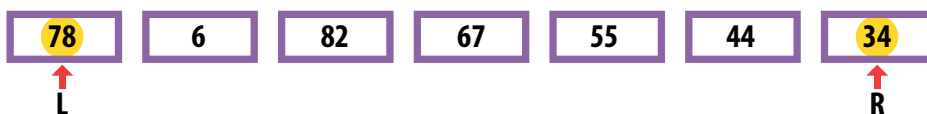
Көбүкчөлүү сорттоо менен тандоо методу чоң берилиштеги массивдер үчүн (10000дей элементтер) жай иштейт. Ошондуктан чоң массивдерди сорттоо үчүн атайын «тез сорттоонун» рекурсивдүү алгоритми колдонулат (англ. quicksort).

Мында сорттоонун идеясы мындай: алгач массив болжол менен ортосунан тең экиге бөлүнөт, андан соң сол жак бөлүгүндөгү элементтерден «ортодо турган» элементтен чоң же ага барабар элементтер тандалып алынат жана оң бөлүгүнө орун алмаштырылат. Андан ары оң жагы жана сол жагы өзүнчө массивдер катары каралат да кайрадан экиге бөлүнүшөт.

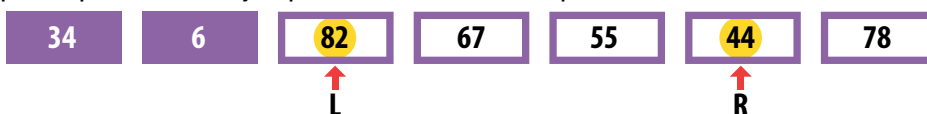
Бул методдун маңызын түшүнүш үчүн мисал карайлы. Бизге төмөндөгүдөй массив берилсин:



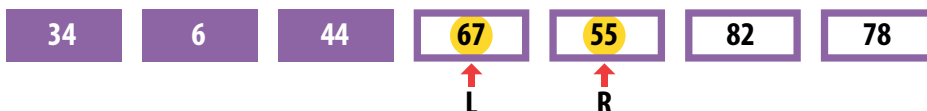
Х деп массивдин ортоңку элементин карайлы, б.а. 67. Массивдин сол жагындагы **Х**тен чоң же ага барабар элементти табабыз. Бул элемент 78 саны. Бул элементтин индексин **L** менен белгилейли. Эми **Х**тен оң жакта турган эң кичине элементти табалы. Бул 34 саны. Анын индексин **R** тамгасы менен белгилейбиз.



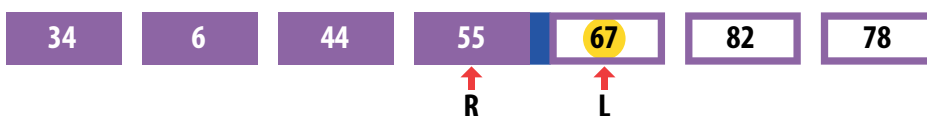
Эми ушул эки элементтин ордун алмаштырабыз. **L** өзгөрмөсүн оңго, ал эми **R** өзгөрмөсүн солго жылдыруу менен биз ордун алмаштыра турган кийинки түгөйдү табабыз. Булар 82 жана 44 сандары.



Орундары алмаша турган кийинки түгөйлөр – 67 жана 55 сандары.



Бул орун алмашуудан кийин, андан аркы издөө **L** өзгөрмөсү **R** өзгөрмөсүнөн чоң болуп калгандыгына алып келет, б.а. массив экиге бөлүнүп калды. Жыйынтыгында **A[L]** дин сол жагында жайгашкан массивдин бардык элементтери **X** тен кичине же барабар, ал эми **A[R]**дин оң жагында жайгашкан бардык элементтер **X** тен чоң же барабар.



Ошентип, баштапкы массивди сорттоо, массивдин эки бөлүгүн сорттоого, б.а. ошол эле типтеги эки маселени (бирок кичинекей өлчөмдөгү) чечүүгө алып келди. Эми ушул алгоритмди массивдин алынган эки бөлүгүнө колдонуу керек: биринчи бөлүгү – 1ден $R_{ге}$ чейинки элементтер, экинчи бөлүгү – $L_{ден}$ акыркы элементке чейин.

Биз айтып кеткендей сорттоонун ылдамдыгы чоң массивдер менен иштөөдө маанилүү. Төмөндөгү таблицада кокустук маанилер менен толтурулган ар кандай өлчөмдөгү массивдердин биз үйрөнгөн үч алгоритмди колдонуп сорттоо убактысы (секунда менен) көрсөтүлгөн.

№	Көбүкчө методу	Тандоо методу	Тез сорттоо
1000	0,09 с	0,05 с	0,002 с
5000	2,4 с	1,2 с	0,014 с
15000	22 с	11 с	0,046 с

Таблицада көрүнүп тургандай, мисалы 15 миң элементи менен массивди тез сорттоо, көбүкчө методуна караганда 500 эсе тез иштейт экен.

Мурунку темадан Python тилинде массивди сорттоо үчүн **sorted** деген атайын камтылган функция бар экенин билебиз. Ал көптөгөн белгилүү берилиштер менен оңой иштеген **timesort** гибриддик алгоритмин колдонот.

Андан тышкары аны жазууда төмөнкүлөрдү эске тутуш керек:

```
a.sort () #a тизмесинин өзү сорттолот
b = sorted (a) #b массивине өсүү тартибинде сорттолгон a массиви жайгаштырылат.
```

Адатта сорттоо өсүү тартибинде же тагыраак айтканда, ар бир кийинки элемент мурункусунан чоң же барабар болушу керек. Массивди кемүү тартибинде сорттоо үчүн (ар бир кийинки элемент мурункусунан кичине же ага барабар) сорттоо функциясына `reverse = True` деп көрсөтүш керек:

```
b = sorted (a, reverse = True) #же
a.sort (reverse = True)
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

- 1) N элементтен турган бир өлчөмдүү сандык маанидеги массив берилген. Баштапкы массивдин элементтеринен эки жаңы массив тургузула. Биринчисине 3кө бөлүнгөн гана сандар, ал эми экинчисине 5ке бөлүнгөн гана сандар кирсин.
- 2) Биринчи суроодогу программаны уланткыла жана 3кө бөлүнгөн сандарды өсүү тартибинде, ал эми 5ке бөлүнгөн сандарды кемүү тартибинде жайгаштыра тургандай кылып программаны толуктагыла.

3.4-тема:

Матрицалар

Матрица – бул таблицалык структурага ээ болгон эки өлчөмдүү массив. Эки өлчөмдүү массивдер деле бир өлчөмдүү массивдердей баяндалат. Айырмасы эки өлчөмдүү массивдин элементинде эки координата (эки индекс) – элемент жайгашкан саптын жана мамычанын номерлери бар.

Мисалы, «Х жана О» оюну үчүн программаны түзүүдө бош клеткаларга «-1» кодун, нөл бар клеткага 0 кодун, ал эми Хтер бар клеткага 1 кодун ыйгарса болот:

		0	1	2
	0	-1	0	1
	1	-1	0	1
	2	0	1	-1

Python тилинде таблицалар менен иштөө үчүн тизмелерди колдонушат. Эки өлчөмдүү таблица – бул ар бир элементи тизме болуп сакталган тизме («тизмелердин тизмеси»). Мисалы, сүрөттө көрсөтүлгөн таблицаны мындай жазсак болот:

```
a = [[-1, 0, 1],
      [-1, 0, 1],
      [0, 1, -1]]
```

Бул тизменин бир сапка да жазса болот:

```
a = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
```

Бирок адам матрицаны таблица катары кабыл алгандыктан, аны экранда да таблица түрүндө чыгарган жакшы. Ал үчүн матрицаны мындай жазсак болот:

```
a = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
for i in range ( len(a) ):
    for j in range ( len(a[i]) ):
        print ( '{:4d}'.format(a[i][j]), end = ' ' )
    print ()
>>>
```

{:4d} чыгаруу форматы – мамычалардын арасындагы талааны кеңейтет: биздин мисалда ар бир элементтин алдында 4 бош орун коюлат.

```
-1  0  1
-1  0  1
 0  1 -1
```

Мында i – камтылган тизменин индекси (саптын санын аныктайт), ал эми j – камтылган тизменин ичиндеги элементтин индекси (мамычалардын санын аныктайт); $\text{len}(a)$ – бул чоң тизмедеги камтылган тизмелердин саны (бул жерде алар 3), $\text{len}(a[i])$ – мамычалардын саны менен дал келген камтылган тизмедеги элементтердин саны.

$a[i][j]$ – бул j индекси менен i камтылган тизмесиндеги элемент:

$a[0][0]==-1$, $a[0][1]==0$, $a[0][2]==1$, $a[1][0]==-1$, ж. у. с.

Бул мисалды мындай жазса да болот:

```
for row in a:      #a сабында
    for elem in row: #саптагы элемент үчүн
        print(elem, end=' ') #элементтерди чыгар
    print()
```

1-маселе. Матрицаны кокустук сандар менен толтурабыз. Саптын жана мамычанын санын клавиатурадан киргизебиз.

Матрицанын ар бир элементине каалагандай маанини ыйгарууга болот. Индекстер экөө болгондуктан матрицаны толтуруу үчүн камтылган циклди колдонобуз. Мындан ары n сабынан жана m мамычасынан турган a матрицасы бар деп, ал эми i жана j – саптын жана мамычанын индексин билдирген бүтүн сандуу өзгөрмөлөр деп эсептейли. Бул мисалда матрица кокустук сандар менен толтурулат жана экранга чыгарылат:

```
import random
n = int(input ('Саптын санын киргизгиле: '))
m = int(input ('Мамычанын санын киргизгиле: '))
a = []
for i in range (n):
    a.append([])
    for j in range (m):
        a[i].append (random.randint (10,40)) #ар бир элемент-
#ке 10дон 40ка чейинки кокустук сан ыйгарылат
for i in a: #ар бир камтылган тизме жаңы саптан чыгарылат
    print (i) #бирок чарчы кашаада
>>>
[33, 16, 31, 33] #n=2, m=4 болгондогу кокустук сандар
[39, 35, 11, 15]
```

Ушул эле маселени жыйынтыгында кашаалар жок болгондой кылып кыскача жазалы:

```
import random
n = int(input ('Саптын санын киргизгиле: '))
m = int(input ('Мамычанын санын киргизгиле: '))
a = [[random.randint(10, 40) for i in range(m)] for j in
range(n)]
print(' '.join([str(elem) for elem in row])) #бардык эле-
менттер биригишет, ортосунда бош орун аркылуу тизмектелет
```

Эки өлчөмдүү массивди иштетүү

Матрицанын бардык элементтерин иргеп чыгуу үчүн дал ушундай эле камтылган эки циклди колдонуу керек. Биринчи цикл саптын номерлерин иргейт, экинчи цикл болсо саптын ичиндеги элементтерди иргейт. Мисалы, бардык элементтердин суммасы (s) мындай эсептелет:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s) #жыйынтыгы 45
```

Бул жазуулар үчүн даяр функция **sum** ду колдонсо да болот:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    s += sum(row)
print (s)
```

Матрицанын кээ бир элементтерин иштетүүнү маселенин мисалында карап көрөлү.

2-маселе. n саптан жана n мамычадан турган квадраттык массив берилсин дейли. Диагоналды бирлер менен, анын сол жагындагы аймакты 2лер менен, ал эми оң жагындагы аймакты нөлдөр менен толтуруу керек.

Негизги диагональ – бул $a[0,0]$, $a[1,1]$, ..., $a[n-1,n-1]$ элементтери, б.а. саптын номери мамычанын номерине барабар.

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Негизги диагоналды бирлер менен толтуруу үчүн бир цикл керек:

```
for i in range(n):    #A[i][i] менен иштейбиз
```



```
a[i][i] = 1          #аны бирлер менен толтурабыз
```

Диагоналдан оң жактагы элементтерди нөлдөр менен толтурушубуз керек, ал үчүн i номериндеги ар бир саптагы $a[i][j]$ элементтерине $j=i+1, \dots, n-1$ үчүн 0 маанисин ыйгарышыбыз керек. Бул жерден бизге камтылган цикл керек болот:

```
for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0
```

Ушундай жол менен эле $j=0, \dots, i-1$ үчүн $a[i][j]$ элементине 2 маанисин ыйгарабыз:

```
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
```

Эгерде сырткы циклдерди бирөөгө чогултсак, анда мындай болгон бир чыгарылышын алабыз:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

КОМПЬЮТЕРДИК ПРАКТИКУМ:

1) n саптан жана m мамычадан турган тик бурчтуу a матрицасын кокустук сандар менен толтургула. Массивдин элементтеринин орточо арифметикалык маанисин тапкыла.

2) Матрицанын ар бир сабы үчүн табылган орточо маанилеринин эң чоңун аныктагыла.

4

- бөлүм



Компьютердик тармактар жана интернет

4.1-тема:

Келечектин технологиялары

Интернет технологиялардын өнүгүшү дүйнөнүн артка кайткыс өзгөрүүсүнө алып келди. Компьютерлер жана девайстар учурдун бөлүнгүс бөлүгү болуп, бул түзүлүштөр бизди келечекте да жандап жүрө тургандыгы шексиз.



БУЛ КЫЗЫКТУУ

Футуролог (келечекти айтуучу), ойлоп табуучу, технологиялардын өнүгүшүн изилдөөчү жана Googleдун машиналык окутуу тармагынын техникалык директору **Рэй Курцвейл** бир нече жыл сайын технологиялык божомолдорун жарыялап турат. Анын көп айтуулары чындыкка айланды. Мисалы ал шахматтык партияда дүйнөдөгү белгилүү шахматчыны компьютер жеңе алат деп, ошондой эле кошумчаланган жана виртуалдык реалдуулук жөнүндө да алдын ала айткан.



Бул темада биз «жирөө» же «секирик» деп аталган технологиялары (англ. Disruptive technologies) жөнүндө айткыбыз келип турат. Алар кийинки жылдарда биздин дүйнө жөнүндөгү көз карашыбызды түп тамырынан бери өзгөртөт.

1 Жасалма интеллект

Жасалма интеллект (ЖИ, англ. Artificial intelligence, AI) – бул адамдын мээсинин когнитивдүү функцияларын көчүргөн, компьютерлерге жана машиналарга гана тиешелүү болгон интеллекттин өзгөчө тиби. ЖИ адамдын тиги же бул чечимдерди кантип кабыл ала тургандыгын түшүнүүгө багытталган.

Интеллектуалдык компьютердик программалардын башка программалардан болгон эң негизги артыкчылыгы – бул алардын өз алдынча үйрөнүүгө жөндөмдүүлүгү жана өзүнүн ичиндеги каталарды оңдоо мүмкүнчүлүгү. ЖИ адам менен жана башка программалар менен канчалык көбүрөөк аракеттенишсе, ал ошончолук көп маалыматты сактап алат жана ал анын «жашоо тажрыйбасынын» бир бөлүгү болуп калат.

ЖИ технологиялары учурда нейрондук тармак жана булуттук эсептөөлөрдүн кеңири таралышынын негизинде интенсивдүү өнүгүп жатат. Курцвейлдин божомолдору боюнча 2030-жылы ЖИ толугу менен адам сыяктуу ойлонууга жөндөмдүү болот.

2 Интернет буюмдар (англ. *Internet of things, IoT*)

Компьютерлер ушунчалык кичирейтилди, эми аларды кийимдерге кошуп тигүүгө, тиш щёткага, саатка, лампага орнотуп койсо болот. Биз буюмдарыбызды аралыктан – интернет аркылуу башкарууга мүмкүндүк алдык. Учурда ар бир девайс жана кийим бири-бири менен аракеттенишип тургандай өзүнүн тармактагы IP-дарегине ээ болушат.



Ден соолугубузга кам көргөн «акылдуу приборлор» бизди боло турган сасык тумоодон алдын алышы мүмкүн. Атайын билдиргич менен камсыз болгон ар бир троллейбусту жана автобусту ушул замат шаардын интернет-картасынан көрүүгө болот. «Акылдуу» көчө чырактары күндүн жарыгынын деңгээли төмөндөгөндө автоматтык түрдө жанат, ал эми жол чырактын жашыл жарыгы жолдун кайсы тарабында автомобилдердин агымы көп болсо ошол тарапка көбүрөөк күйөт.

Бир эле машиналар эле эмес бүткүл шаар, а түгүл өлкө да акылдуу боло баштады. Силер мүмкүн фильмдерден көргөнсүңөр, полиция кызматкери бир баскычты басып – өлкөдөгү каалаган видеобайкоо камераларына кошулгандыгын. Ушул бардык артыкчылыктарга карабастан, өзүңөрдүн ар бир девайсыңарды интернетке кошоордон мурда ойлонуңуз, сиз чындап эле бардык жерде өзүңүздүн санариптик изиңизди калтыргыңыз келеби?

3 Робот техникасы

Роботтор көптөн бери эле биздин арабызда. Албетте, жасалма интеллект менен камсыз болуп алар толук бойдон адамдын түспөлүндөй болбосо да, кайсы бир учурда адамдарга гана тиешелүү ролдорду аткарууда. Футурологдордун божомолдору боюнча жакынкы 10 жылда роботтор биздин үйүбүздө муздаткыч, кир жуугуч машина сыяктуу эле кадимки буюмдардын катарына кирет.



Роботтор адамдардын бөлүгү боло башташты. Мисалга, роботтоштурулган же «бионикалык» деп аталган протездерди адамга өөрчүтүү. Азыркы учурда мындай протездерди башкаруу үчүн жакын жайгашкан булчуңдардан алынган сигналдарды окуучу атайын системалар колдонулат.

Ал эми мындай протездин ээси түзүлүшү менен тийишкен объекттин касиети (ысык, бодуракай ж.б.) жөнүндө маалыматты түздөн түз мээге ала турган мезгил алыс эмес. Эң жөнөкөй протездер азыр жеткиликтүү, аларды оңой эле 3D-принтерде басып чыгарып алса болот.

4 3D-басып чыгаруу

Эгерде биз каалагандай нерсени, анын ичинде автомобилди да принтерде басып чыгарып алсак эмне болот эле? Муну элестетүү кыйын, бирок бул чындап эле болуп жатат: адамдар өздөрүнө үйдү, эмеректерди, идиш-аяктарды, транспортту, кийимди, дененин бөлүктөрүн, ал эмес тамакты да басып чыгарып калды, анткени кондитердик 3D-принтерлер пайда болду! Элестеткиле, силер алманы принтерден басып чыгардыңар, бирок картриджде шекер түгөнгөндүктөн аныңар ачуу болуп калгандыгын.

3D-басып чыгаруунун өнүгүшү менен товарлары бар ири контейнерлерди ар жактан ташып келүүнүн кажети жок болот. Болгону керектүү товардын чиймеси менен файлды жөнөтүү гана жетиштүү болот. Мүмкүн ушунун негизинде биз зыяндуу өндүрүштү, глобалдуу транспорттук системанын натыйжасында болуп жаткан абанын булганышын токтото алаарбыз.

Биотехнология жана гендик инженерия

5 Окуу китебин жазып жаткан учурда ар кандай жигердүү биотехнологияларды иштеп чыгуу боюнча дүйнөдө миңдеген медициналык сыноолор өтүп жатат. Жаңы вакциналар иштелип чыгып жатат, CRISPR технологиясын колдонуу менен гендерди редакциялоо (биологиялык организмдердин генотиптерин жасалма жана максаттуу өзгөртүү) боюнча клиникалык сыноолор жүргүзүлүп жатат. Гендик инженериянын негизги ыкмасы – тиешелүү гендерди бөлүп алуу жана аларды модификациялоо: мутацияланган гендерди оңдоо, жоголгон гендерди калыбына келтирүү ж.б. Бул иммундук системасы бузулгандагы, кандын уюшу, онкология менен байланышкан ооруларды дарылоодо өзгөчө актуалдуу.

Экинчи өтө тездик менен өнүгүп жаткан багыт – бул наномедицина. Ал наноматериалдарды иштеп чыгуудан башталып, наноэлектрондук биосенсорлор менен бүтөт. Мисалы, организмдин клеткасына чейин кирип, аларды «тамактандырып» жана ашыкчаларын жок кылуучу нанороботтор иштелип чыгууда.



6 Виртуалдуу реалдуулук (VR)

2030-жылдын аягына чейин VR ушунчалык жогорку сапатка жеткендиктен, аны чыныгы реалдуулуктан айырмалоо мүмкүн болбой калат. Интернет тармагынын өнүгүшү бир убакта бир нече колдонуучулар колдоно ала тургандай виртуалдуу дүйнөнү курууга мүмкүндүк берет.

Азыркы күндө VR бир эле оюн-зоок индустриясында эле эмес, автомобиль куруу өндүрүшүндө, аэрокосмостук өндүрүштө жана кеме курууларда кеңири колдонулууда. VR технологиясы иштеп чыгуу убактысын кыскартып жана азырынча чыга элек продуктуна да сыноого мүмкүнчүлүк берет.

7 Калыбына келүүчү энергия жана жашыл технологиялар

Калыбына келүүчү же «жашыл» энергия – бул адамдык масштабда түгөнбөй турган булактардан алынган энергия. Эксперттердин ою боюнча, калыбына келүүчү энергиянын булактары климаттын өзгөрүүсүн жана планетабыздын булганышын кескин азайтат.

Калыбына келүүчү энергия шамал турбиналары, фотоэлектрдик элементтер, күн панелдери, геотермалдык энергия, океан толкундарынын энергиясы ж.б. ойлоп табуулардын негизинде колдонууга мүмкүн болууда.

Андан тышкары өзүнүн габариттерин сактоо менен энергияны чогултуучулардын кубаттуулугунун өсүшү да калыбына келүүчү энергиянын булактарына болгон суроо-талаптарды олуттуу өстүрүүдө.

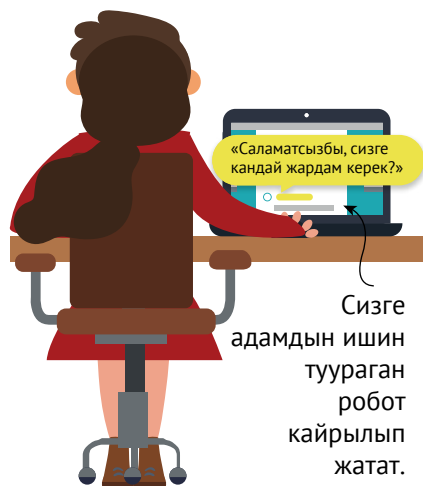
**СУРОЛОР ЖАНА ТАПШЫРМАЛАР:**

- 1) Кандай калыбына келүүчү энергиянын булактарын билесиңер? Силердин оюңарча «жашыл» технологияларды колдонуу эмнеликтен биздин планета үчүн маанилүү?*
- 2) Гендик модификациялоону этикага жатат деп эсептейсиңерби?*
- 3) Силер 3D-принтерлерден, ЖИ, виртуалдуу реалдуулуктан коркунучтарды көрө алдыңызбы? Бул технологиялар адамзат үчүн эмнеси менен коркунучтуу болушу мүмкүн?*

4.2-тема:

Санариптик дүйнөдөгү коопсуздук

Санариптик дүйнө же анын дагы бир аты болгон – кибермейкиндик – бул компьютерлер жана компьютердик тармактар, өзүнүн мыйзамдары жана эрежелери бар параллель аалам; чексиз мүмкүнчүлүктөрдү ачкан жана коркунучтарды камтыган мейкиндик.



Виртуалдык мейкиндикте жагымсыз окуялардын болбошу үчүн кээ бир эрежелерди сактоого туура келет.

Биринчиден интернетке кирип жатканда эле – бул ачык жана көп кырдуу дүйнө жана биз ал жерде жалгыз эмес экендигибизди унутпашыбыз керек. Ар бир профилдин, аккаунттун артында тирүү адам, адамдардын тобу же программалык бот жайгашат.

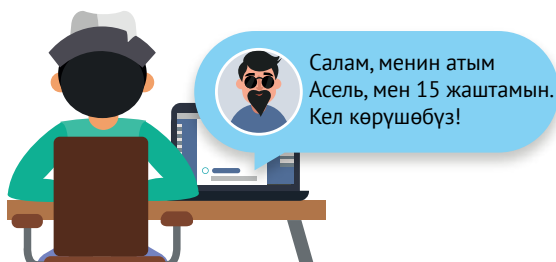
Интернетте иштеп жатып, бир эле боттор өздөрүн адамдар катарында көрсөтпөшүн унутпашыбыз керек. Адамдар да өздөрүн башка адамдардай көрсөтүшү мүмкүн.

Мындай адамдар ар түрдүү максаттарды көздөшү мүмкүн – сиздин аккаунтуңузду бузуудан (сиздин паролюңузду уурдоо, профилиңизди ар кандай максатта колдонуу) баштап, физикалык жактан аңдуу, шантаж жана опурталдуу аракеттерге тартууга чейин.

Эсиңерде болсун, силер интернетте бөлүшкөн бардык маалыматтар шылуундар тарабынын колдонулушу мүмкүн. Тармакта эмне жөнүндө сүйлөштү, кандай маалыматтарды жайгаштырууну жана кимди достукка кабыл алууну биринчи ойлонгула.

**АНЫКТАМА**

«Бот» («робот» сөзүнүн кыскартылышы) – адамдын ишин туураган программа. Чат-бот чаттагы маектешүүчүнү туурайт.



Биз башында айтып кеткендей интернетте өзүн башка адам кылып көрсөтүү үчүн өтө көп мүмкүнчүлүктөр бар. Бир эле адамдын бир учурда бир нече аккаунту болушу мүмкүн, бир жерде ал өспүрүм болсо, башкасында аскер кызматкери, мамлекеттик чиновник болот.

Эгерде досуңардан анын телефонуна акча салып коюуну суранган же белгиленген жерге жолугууга кел деген же башка бир күтүүсүз билдирүү келсе, аны аткаруудан мурда эң биринчи досуңузга байланышып, чындап эле ал жөнөткөнүн тактап алыңыз. Мүмкүн анын аккаунтун бузуп, анын атынан башкалар билдирүү жөнөтүшү мүмкүн.

Андан тышкары сизге шилтеме менен барууну же тиркелген документтерди ачууну сунуштаган электрондук каттар келиши мүмкүн. Келген маалыматтар сизди кызыктырышы (сиз миллион утуп алдыңыз!) же коркутушу (биз сиздин сырыңызды бардыгына айтабыз!) мүмкүн. Мындай билдирүүлөр көпчүлүк учурда бир эле максатта келиши мүмкүн – вирус жуктуруу же аккаунтуңузду уурдоо. Андыктан шылуундардын кайырмагына илинбеңиздер!

Биздин аккаунттар – бул биз, биздин жакындарыбыз жана биздин мамиле-биз жөнүндө маалыматтар. Башка адамдардын колуна өтсө алар бир эле бизге эмес, биз менен байланышта болгондордун бардыгына зыян алып келиши мүмкүн. Мындай окуя болбош үчүн аккаунтту ишенимдүү пароль менен коргоо керек.

Ишенимдүү пароль ондон кем эмес, баш жана кичине тамгаларды, цифраларды жана атайын символдорду (тор белгиси, жылдызча, төмөнкү сызык ж.б.) өзүнө камтыйт.

Ишенимдүү паролдо өздүк маалымат (ат, фамилия, туулган датасы, үй жаныбарынын аты, кызыгуулар жана сиз жөнүндө социалдык тармактардан билип ала турган маалыматтар) камтылбашы керек.

Ишенимдүү паролду эстеп калуу албетте кыйын. Ал үчүн эстеп калуунун ар түрдүү технологиялары колдонулат. Мисалы, силер өзүңөргө гана түшүнүктүү болгон, логикалык жактан байланышкан үч-төрт сөздү курап алсаңар болот.



Андан да жакшысы бириктирилген сөздөрдү цифралар же символдор менен ажыратып койсоңор болот. Мисалы: **чымын1кыям2ыр3**

Андан да сонуну: **muh@1v@renie2stihotvorenie3**

Окуу китебинде берилген паролду эч качан колдонбогула!

Эч качан түрдүү сайттарда бир эле паролду колдонбогула!

Ар түрдүү паролдорду эстеп калууда кыйналбаш үчүн атайын программаларды – пароль менеджерлерин колдонууга болот.

Пароль менеджерлери паролду түзүүгө, сактоого жана колдонууга жардам берет. Мындай программалардын ичинен эң кеңири тараганы болуп KeePass жана LastPass эсептелет.



Сиздин паролуңуз абдан ишенимдүү болгон күндө да, баары бир ал башкалардын колуна түшүп калуу ыктымалдуулугу бар. Мисалы, кимдир бирөө сиздин паролду терип жатканыңызды көрүп калышы мүмкүн же аны фишинг аркылуу алып алышы мүмкүн.

Өзүңүздүн коопсуздугуңузду камсыз кылуу үчүн сиз интернетте макулдашкан жазылуу же жаңы кызмат тейлөөсүнүн шарттарын абдан көңүл бөлүп окуп чыгуу керек. Мисалы, мобилдик операторлордун сайтында сунушталган «жаңы рингтон» же «анекдоттор» деген сыяктуу типтеги ар түрдүү тейлөөлөргө жазылуу. Көпчүлүк учурда сиз «Окуу» деген чоң, кызыктыруучу баскычты көрөсүз жана ага бир басуу менен сизден абоненттик акы алып тура турган тейлөөгө жазылып каласыз.

Коргонуш үчүн ар дайым шартын (алар адатта майда шрифт менен жазылат) окуп чыккыла жана операторлордон келген шектүү тейлөө кызматтарына жазылуу жөнүндөгү СМС билдирүүлөрдү байкабай койбоңуз. Эгерде сиз өз алдыңызча жазылуудан баш тарта албай жатсаңыз, өзүңүздүн операторуңуздун байланыш салонуна кайрылып, тейлөө кызматын өчүртүп жана жаңы жазылууларды блоктоп (тосмолоп) салуу мүмкүнчүлүгүн сураңыз.



Фишинг (*fishing «балык кармоо, колго түшүрүү»*) – бул башка бирөөлөрдүн аккаунтуна жана маалыматына уруксат алууга багытталаган тармактык алдамчылык. Көпчүлүк учурда бул сырткы келбети боюнча түп нускадан айырмасы жок болгон сайтты колдонуучуга сунуштоо менен ишке ашырылат. Ал жерде колдонуучуга пароль жана логинди киргизүү сунушталып, кийин пароль алдамчылардын колуна түшөт да колдонуучу өзүнүн аккаунтуна кире албай калат.

Колдонуучуларды аккаунтун жоготуп алуудан коргоо максатында азыркы сервистер эки факторлуу аутентификацияны колдонууну сунуш кылышат.

Эки факторлуу аутентификация – бул паролдон тышкары, телефондун жардамында аккаунтка кирүүгө мүмкүн кылган кошумча коргоо деңгээли. Анда колдонуучу телефонуна СМС билдирүү катары келген же телефондо атайын тиркеменин жардамында генерацияланган кодду да киргизүүсү керек болот.

Ошентип, кылмышкер сиздин аккаунтуңуздун паролун билген күндө да сиздин телефонуңузга кире албай туруп, аккаунтуңузду колдоно албайт.

Телефонуңузга да пароль коюуну унутпаңыз!

Ушул жөнөкөй эрежелерди сактоо менен сиз өзүңүздүн маалыматыңызды күтүүсүз чабуулдардан коргоп, өзүңүздү ишенимдүү сезе аласыз.

Баардык жогоруда айтылгандар — булар санариптик мейкиндикте коопсуздукту камсыз кылуу сунуштарынын бир гана бөлүгү. Буга антивирустук коргоо, маалыматтын резервдик көчүрмөлөрүн түзүү жана башка көптөгөн сунуштар да кирет.

Коопсуздуктун эң негизги эрежесин дагы унутпаңыз – ойлонгон адамды алдоо алда канча кыйынга турат.

СУРООЛОР ЖАНА ТАПШЫРМАЛАР:

- 1) Вирустар кантип силердин компютериңерге кирет деген темада ойлонгула. Алардан кантип коргонуш керек?
- 2) Силердин санариптик маалыматыңарга болгон кандай коркунучтарды (вирустардан тышкары) айтып бере аласыңар?
- 3) Өзүңөр жана жакындарыңар жөнүндөгү кандай маалыматтарды интернетке жарыялоого болбойт жана эмне үчүн?



Тиркемелер

№1 тиркеме

Тексттин стандарттык коддоолору (кодировкалары) CP-1251:

Á	à	,	è	”	...	†	‡	€	% ₀	É	<	и	И	Ó	Ý
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
á	‘	,	“	”	•	–	—	ë	™	é	>	ò	и	ó	ý
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
nbsp	ÿ	Ы	Э	Ѡ	ы	ı	§	Ë	©	Ю	«	¬	shy	®	Я
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
°	±	ы	э	’	μ	¶	•	ë	№	ю	»	э	ю	я	я
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Тексттин стандарттык коддоолору (кодировкалары) КОИ-8:

–		Г	Г	Л	Л	Т	Т	Т	Т	Т	■	■	■	■	■
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
■	■	■	Г	■	•	√	≈	≤	≥	nbsp	J	°	²	•	÷
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
=		F	ë	П	F	Ɔ	П	П	П	П	П	П	П	П	П
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
		Ɔ	Ë			Ɔ	П	П	П	П	П	П	П	П	П
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

№2 тиркеме

Электрондук таблицанын элементтери

Бөлүп алуу ыкмалары

Уяча

Курсорду уячага келтирүү жана ЧСБны басуу, бул учурда мамычанын атынын жана бир сызыкка жаткан саптын номеринин түсү өзгөрөт (уяча активдүү болуп калат).

Жанаша жайгашкан уячалардын диапозону

1-ыкма: курсорду диапозондун биринчи бурчтагы уячасына келтирүү, shift баскычын басып, коё бербей туруп, курсорду диапозондун карама-каршы бурчуна жылдыруу, ЧСБны басуу. 2-ыкма: курсорду биринчи бурчтагы уячага келтирүү, ЧСБны басып, коё бербестен диапозондун карама-каршы бурчуна жылдыруу.

Жанаша жатпаган уячалардын диапозондору

Ctrl баскычын басып, коё бербестен, жанаша жайгашкан уячаларды бөлүп алуу ыкмасын колдонуп, өзүнчө диапозондорду бөлүп алуу.

Сап

Сол жактагы сызгычтагы саптын номерин басуу керек.

Мамыча

Жогорку сызгычтагы мамычанын атына басып коюу керек.

Барак

Мамычанын баш сөзү А менен саптын баш сөзү 1дин ортосундагы тик бурчтукту басып коюу керек.

№3 тиркеме

Арифметикалык операторлор

Оператор

Мааниси

Мисалдар

+

Кошуу – оператордон сол жана оң жактагы маанилерин суммалайт

10 + 5 жыйынтыгында 15 болот
27 + -3 жыйынтыгында 24 болот
9,4 + 7 жыйынтыгында 16,4 болот

-

Кемитүү – сол операнддан оң операндды кемитет

15 - 5 жыйынтыгында 10 болот
20 - -3 жыйынтыгында 23 болот
13,4 - 7 жыйынтыгында 6,4 болот

*

Көбөйтүү – операнддарды көбөйтөт

5 * 5 жыйынтыгында 25 болот
7 * 3,2 жыйынтыгында 22,4 болот
-3 * 12 жыйынтыгында -36 болот

/

Бөлүү – сол жактагы операндды оң жактагысына бөлөт

15 / 5 жыйынтыгында 3 болот
5 / 2 жыйынтыгында 2,5 болот

%

Модуль боюнча бөлүү – сол жактагы операндды оң жактагысына бөлөт жана калдыгын чыгарып берет

6 % 2 жыйынтыгында 0 болот
7 % 2 жыйынтыгында 1 болот
13,2 % 5 жыйынтыгында 3,19999999 болот

**

Даражага көтөрүү – сол операндды оң жактагы даражага көтөрөт

5 ** 2 жыйынтыгында 25 болот
2 ** 3 жыйынтыгында 8 болот
-3 ** 2 жыйынтыгында -9 болот

//

Бүтүн бөлүктүү бөлүү – бөлүүнүн жыйынтыгында бүтүн бөлүгү гана чыгарылат. Үтүрдөн кийинки бөлүгү алып салынат

12 // 5 жыйынтыгында 2 болот
4 // 3 жыйынтыгында 1 болот
25 // 6 жыйынтыгында 4 болот

Салыштыруу операторлору

==	Эки операнд тең барабар экенин салыштырат, эгерде барабар болсо, анда шарт чындык болот.	$5 == 5$ жыйынтыгында <i>True</i> болот $True == False$ жыйынтыгында <i>False</i> болот $"hello" == "hello"$ жыйынтыгында <i>True</i> болот
!=	Эки операнд тең барабар экенин салыштырат, эгерде барабар эмес болсо, анда шарт чындык болот.	$12 != 5$ жыйынтыгында <i>True</i> болот $False != False$ жыйынтыгында <i>False</i> болот $"hi" != "Hi"$ жыйынтыгында <i>True</i> болот
<>	Эки операнд тең барабар экенин салыштырат, эгерде барабар эмес болсо, анда шарт чындык болот.	$12 <> 5$ жыйынтыгында <i>True</i> болот $!=$ операторуна окшош
>	Сол операнд оң жактагыга караганда чоң экенин салыштырат, эгер чоң болсо, анда шарт чындык болот.	$5 > 2$ жыйынтыгында <i>True</i> болот $True > False$ жыйынтыгында <i>True</i> болот $"A" > "B"$ жыйынтыгында <i>False</i> болот
<	Сол операнд оң жактагыга караганда кичине экенин салыштырат, эгер кичине болсо, анда шарт чындык болот.	$3 < 5$ жыйынтыгында <i>True</i> болот $True < False$ жыйынтыгында <i>False</i> болот $"A" < "B"$ жыйынтыгында <i>True</i> болот
>=	Сол операнд оң жактагыга караганда чоң же барабар экенин салыштырат, эгер чоң же барабар болсо, анда шарт чындык болот.	$1 >= 1$ жыйынтыгында <i>True</i> болот $23 >= 3,2$ жыйынтыгында <i>True</i> болот $"C" >= "D"$ жыйынтыгында <i>False</i> болот
<=	Сол операнд оң жактагыга караганда кичине же барабар экенин салыштырат, эгер кичине же барабар болсо, анда шарт чындык болот.	$4 <= 5$ жыйынтыгында <i>True</i> болот $0 <= 0,0$ жыйынтыгында <i>True</i> болот $-0?001 <= -36$ жыйынтыгында <i>False</i> болот

Ыйгаруу операторлору

=	Сол жактагы операндга оң жактагы операнддын маанисин ыйгарат.	$b = 23$ <i>b өзгөрмөсүнө 23 маанисин ыйгарат</i>
+=	Оң жактагы операнддын маанисине сол жактагынын маанисин кошуп, аны сол жактагы операндга ыйгарат.	$b = 5$ $a = 2$ $b += a$ барабар: $b = b + a, b = 7$
-=	Сол жактагы операнддын маанисинен оң жактагынын маанисин кемитип, аны сол жактагы операндга ыйгарат.	$b = 5$ $a = 2$ $b -= a$ барабар: $b = b - a, b = 3$
*=	Оң жактагы операнддын маанисине сол жактагынын маанисин көбөйтүп, аны сол жактагы операндга ыйгарат.	$b = 5$ $a = 2$ $b *= a$ барабар: $b = b * a, b = 10$
/=	Сол жактагы операнддын маанисин оң жактагынын маанисине бөлүп, аны сол жактагы операндга ыйгарат.	$b = 10$ $a = 2$ $b /= a$ барабар: $b = b / a, b = 5$
%=	Операнддарды модулу боюнча бөлөт жана жыйынтыгын сол жактагыга ыйгарат.	$b = 5$ $a = 2$ $b \% = a$ барабар: $b = b \% a, b = 1$
**=	Сол жактагы операндды оң жактагынын маанисиндей даражага көтөрөт жана жыйынтыгын сол жактагыга ыйгарат.	$b = 3$ $a = 2$ $b ** = a$ барабар: $b = b ** a, b = 9$
//=	Сол жактагы операндды оң жактагыга бүтүн бөлүктүү бөлөт жана жыйынтыгын сол жактагыга ыйгарат.	$b = 11$ $a = 2$ $b //= a$ барабар: $b = b // a, b = 5$

Логикалык операторлор

Оператор	Мааниси	Мисалдар
and	Логикалык оператор “ЖАНА”, эгерде эки операнд тең чындык болсо шарт чындык болот.	<i>True and True барабар True True and False барабар False False and True барабар False False and False барабар False</i>
or	Логикалык оператор “ЖЕ”, эгерде жок дегенде бир операнд чын болсо, анда бардык сүйлөм чындык болот.	<i>True or True барабар True True or False барабар True False or True барабар True False or False барабар False</i>
not	Логикалык оператор “ЭМЕС”, операнддын логикалык маанисин карама-каршысына өзгөртөт.	<i>not True барабар False not False барабар True</i>

Мүчөлүк операторлор

Мүчөлүк операторлор сап, тизме, кортеж же сөздүктөргө окшогон курама маалыматтардын тибинде элементтердин бар же жок экенин текшерет.

Оператор	Мааниси	Мисалдар
in	Эгерде элемент удаалаштыкта бар болсо анда чындыкты, ал эми жок болсо анда жалганды кайтарып берет.	<i>“cad” in “cadillac” кайтарам True 1 in[2,3,1,6] кайтарам True “hi” in {“hi”:2, “bye”:1} кайтарам True 2 in {“hi”:2, “bye”:1} кайтарам False (сөздүктөрдө маанисиндеги эмес, ачыктардагы элементтин бардыгы текшерилет)</i>
not in	Эгерде элемент удаалаштыкта жок болсо анда чындыкты кайтарып берет.	<i>in операторунун жыйынтыгына карама-каршы жыйынтыктар.</i>

Тендештиктер операторлору

Окшоштуктар операторлору компьютердин эсиндеги эки объекттин жайгашышын салыштырат.

Оператор	Мааниси	Мисалдар
is	Эгерде эки операнд тең бир объектти көрсөтсө, анда чындыкты кайтарат.	<i>x is y чындыкты кайтарып берет, эгерде id(x) барабар id(y) болсо</i>
is not	Эгерде эки операнд тең бир объектти көрсөтсө, анда жалганды кайтарат.	<i>x is y чындыкты кайтарып берет, эгерде id(x) барабар эмес id(y) болсо.</i>

Экилик эсептөө системасындагы берилиштер менен иштеген операторлор

Операторлор	Мааниси
**	Даражага көтөрүү
~, +, -	Комплиментардык операторлор
*, /, %, //	Көбөйтүү, бөлүү, модулу боюнча бөлүү, бүтүн бөлүктүү бөлүү
+, -	Кошуу жана кемитүү
>>, <<	Оңго жана солго биттик жылышуу
&	Бинардык “ЖАНА”
^,	Бинардык “Артыкча ЖЕ” жана бинардык “ЖЕ”
<=, < >, >=	Салыштыруу операторлору
<>, ==, !=	Барабардыктар операторлору
=, %=, /=, //, -=, +=, *=, **=	Ыйгаруу операторлору
is, is not	Теңдештиктер операторлору
in, not in	Мүчөлүк операторлор
not, or, and	Логикалык операторлор

№4 тиркеме

Растрдык сүрөттөрдүн форматтарынын мисалдары

ФОРМАТ	АРТЫКЧЫЛЫГЫ	КЕМЧИЛИГИ
BMP (.bmp) – стандарттык формат, ал	- палитра менен коддоону жана чыныгы түстүү режимди да колдойт	- көп орунду ээлейт; - BMPны колдонуу Windows жана OS/2 платформалары менен чектелген. Бул болсо форматтын тармакта таралышына чек коёт
JPEG (.jpg .jpeg) – фотографияларды коддоо үчүн атайын иштелип чыккан формат	- жарыктыгы жана түсү тегиз өткөн реалдуу көрүнүштөрдү камтыган сүрөттөрдү жана фотографияларды башка форматтарга караганда эң сонун кысат	- көп орунду ээлейт; - BMPны колдонуу Windows жана OS/2 платформалары менен чектелген. Бул болсо форматтын тармакта таралышына чек коёт
GIF (.gif) – мурдагы форматтардан айырмаланып палитраны (2ден 256 түскө чейинки) коддоону гана колдогон формат	- сүрөттүн кээ бир бөлүгү тунук болушу мүмкүн, б.а. веб-баракта алардын артынан фон көрүнүп турат; - анимацияны колдойт	- түстүн аз санда болушу; - бир нече статикалык кадрлардын удаалаштыгынан турган анимациялык сүрөттөрдү колдойт, андан тышкары ар бир кадр экранда канча убакыт көрсөтүлүшү жөнүндөгү маалыматты да колдойт
PNG (.png) – чыныгы түстүү режимди да, палитра менен коддоонуда колдогон формат; сүрөттүн кээ бир бөлүгү тунук же жарым тунук болушу мүмкүн.	- кысууда сапат жоголбойт; - калыбына келтирүү жана сүрөттү кайра сактоо сапатты жоготуусуз ишке ашат.	- бул формат башында GIF эскирген форматты алмаштыруу үчүн долбоорлонгонунга карабастан PNG бир файлда бир эле сүрөттү сактай алат.

Сөздүк (Глоссарий)

CMS – мазмунун (контентин) башкаруу системалары (*англ.* content management system, CMS) – бул сайттын мазмунун башкаруу үчүн пайдаланылуучу программалык камсыздоо.

Аутентификация – бул кайсы бир нерсенин аныктыгын текшерүү процесси. Аутентификациянын мисалы катары колдонуучу тарабынан киргизилген пароль менен сервердин маалыматтар базасында сакталган паролду салыштыруу болушу мүмкүн.

«Бот» («робот» сөзүнүн кыскартылышы) – адамдын иштерин туураган программа. Чат-бот чаттагы маек-тешти туурайт.

Видеоэс – бул графикалык сүрөттөлүш түзүлгөн оперативдүү эстин бөлүгү.

Гиподинамия – организмдин төмөнкү кыймыл активдүүлүгү, жалпы физикалык активдүүлүктүн төмөндөшү.

Дискреттөө – бул графикалык маалыматты аналогдук формадан дискреттик формага өргөртүп түзүү, башкача айтканда үзгүлтүксүз графикалык сүрөттү өзүнчө элементтерге бөлүп чыгуу.

Жөнөкөй логикалык айтым – бул маанисин жоготпостон туруп кичирейтүүгө же бөлүүгө мүмкүн болбогон айтым.

Интерпретатор – бул сиздин программаңызды окуп, андагы камтылган нускаманы аткаруучу программа.

Контент (*англ.* мазмуну) – бул колдонуучуга арналган маалымат жана тажрыйба. Сайттын контенти – бул анда камтылган текст, сүрөт, видео, аудио.

Код – бул маалыматты чагылдыруу үчүн шарттуу белгилердин жана эрежелердин системасы.

Коддоо – бул берилген коддун жардамы менен маалыматты чагылдыруу.

Логикалык туюнтма – бул логикалык амалдар менен бириктирилген логикалык айтымдар.

Лэндинг (*англ.* landing page) – эң негизги максаты сайтка кирүүчүнү кандайдыр бир конкреттүү (максаттуу) аракетти жасатуу болгон веб-баракча: мисалы, бир нерсе үчүн добуш берүү, кайсы бир товар менен таанышуу жана аны сатып алуу, китепке заказ берүү ж.б.

Лонгрид (*англ.* Longread) – ар кандай мультимедиялык элементтердин (фотография, сүрөттөр, видео, диаграмма ж.б.) жардамында блокторго бөлүнгөн узун текст (макала, баян).

Маалыматтар базасы – бул эсептөө системасында алардын эффективдүү издөө жана иштетүү максатында логикалык системалаштырылган маалыматтар топтому.

Маалыматтык сабаттуулук – бул адамдын маалыматты аны издөө, тандоо, баалоо жана колдонуу көнүмдөрүнүн керектигин

андап билүү жөндөмдүүлүгү.

Өзгөрмө – бул аты, тиби жана мааниси бар чоңдук. Өзгөрмөнүн мааниси программанын аткарылышында өзгөрүп турушу мүмкүн.

Протокол – бул тармак аркылуу маалыматты берүүнү жүргүзгөн тиешелүү эрежелер.

Табигый тил – адамдардын баарлашуусу үчүн колдонулат (орус, кыргыз, англис тил ж.б.).

Татаал айтым логикалык амалдарга дал келген логикалык байламталар менен бириккен жөнөкөй айтымдардан турат.

Түстүн тереңдиги – бул пикселдин түсүн коддоо үчүн колдонулган биттердин саны.

Формалдаштыруу – бул символдордун жардамында конкреттүү мазмундан (айтымдан) формалдуу жазууга өтүү.

Формалдуу тил – бул сүйлөм түзүүнүн так эрежелери менен мүнөздөлгөн жасалма тил: ноталар, Морзе алиппеси, химиялык элементтер символдору, программалоо тилдери.

Фэйк (*англ.* fake – «жасалма, жалган») маалыматты берүүдө «жалган маалымат» деген маанисин түшүндүрөт.

Чечүү – сүрөттөлүштүн бир дюйм өлчөмүнө туура келүүчү пикселдердин саны.

