Дәріс 1

1. АЛГОРИТМДЕУ ЖӘНЕ БАҒДАРЛАМАЛАУДЫҢ ТЕОРИЯЛЫҚ НЕГІЗДЕРІ

1.1. Алгоритм. Алгоритм қасиеттері. Алгоритмды сипаттау әдістері

Егер біз кез-келген бағдарламалау тілінде бағдарлама жазғыңыз келсе, алдымен мәселені шешу алгоритмін жасауыңыз керек.
Алгоритм - бұл мәселені шешуге арналған әрекеттер тізбегін дәл және қарапайым сипаттауы. Алгоритмде белгілі бір ретпен орындалуы керек бірнеше қадамдар бар. Алгоритмнің әр қадамы бір немесе бірнеше қарапайым операциялардан тұруы мүмкін.
Әрқайсымыз күн сайын әртүрлі алгоритмдерді қолданамыз: нұсқаулар, ережелер, рецепттер және т.б. Әдетте біз оны ойланбастан жасаймыз. Мысалы, есікті кілтпен ашқанда, ешкім қандай ретпен әрекет ету керектігі туралы ойламайды. Алайда, біреуге есікті ашуға үйрету үшін Сіз іс-әрекеттерді және оларды орындау тәртібін нақты көрсетуіңіз керек.
Алгоритм үшін тек әрекеттер жиынтығы ғана емес, сонымен қатар олар қандай тәртіппен орындалады.

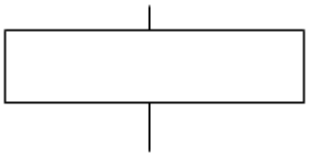
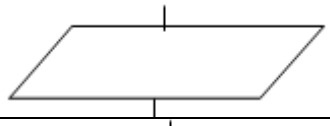
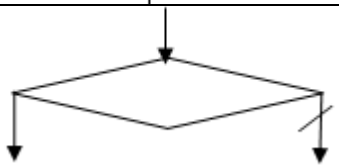
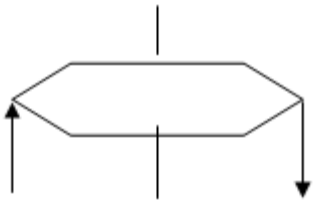
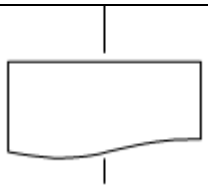
Алгоритм қасиеттері:
1. дискреттілік (үздіктілік,бөлектілік(даралық)) - алгоритм есепті шешу процесін қарапайым қадамдарды (кезеңдерді) рет-ретімен орындау ретінде көрсетілуі тиіс;
2. анықтылық(сенімділік)-алгоритмнің әр қадамы айқын және бір мәнді болуы керек. Алгоритмнің орындалуы механикалық және шешілетін мәселе туралы қосымша ақпаратты қажет етпейді;
3. нәтижелік-алгоритм соңғы қадамдар саны үшін есепті шешуге әкелуі керек;
4. жалпылық-шешім алгоритмі жалпы түрде жасалады, яғни ол тек бастапқы деректермен ерекшеленетін кейбір есептер класын шешуге қолданылуы керек.

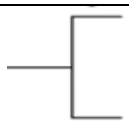
Алгоритмдерді сипаттау тәсілдері
сөздік;
графикалық;
кестелік;
формулалық.

Әрқайсымыз күнделікті ауызша әдісті қолданамыз, мысалы, әңгімелесуде әртүрлі нұсқаулар, ережелер, аспаздық рецепттер, яғни түпкілікті нәтижеге әкелетін қандай да бір реттілік.
Алгоритмдерді ұсынудың графикалық әдісі ауызша нұсқамен салыстырғанда ықшам және көрнекі болып табылады. Белгілі бір жағдайды жақсы түсіну үшін біз қандай-да бір схеманы, жоспарды, соған сәйкес әрекет етуді жеңілдетеміз.

Бағдарламалауда бұл әдіс ең қолайлы, өйткені ол әрқайсысы бір немесе бірнеше әрекеттерді орындауға сәйкес келетін функционалды блоктардың тізбегін қолдана отырып, белгілі бір мәселені шешудің барысын ұсынуға мүмкіндік береді. Алгоритмнің бұл көрінісі алгоритмнің құрылымдық диаграммасы немесе блок-схема деп аталады.
Кестелік әдіс, мысалы, бухгалтерлік есепте жылдық есептерді, мәліметтерді және т. б. жасау кезінде қолданылады.
Формула әдісі математика, физика және т.б. саласындағы есептерді шешуде өзінің қолданылуын табады, мысалы, квадрат теңдеуді шешкен кезде біз теңдеудің дискриминантын табуға кірісеміз, содан кейін алынған нәтижеге байланысты барлық белгілі формулалардан теңдеудің түбірлерін табамыз.

1.2. Функционалдық блоктардың мақсаты

Атауы	Блок схема түрі	Негізгі әрекеттер
Басы,соңы		Алгоритмнің басы немесе соңы
Процесс		Математикалық өрнектерді есептеу(Амалдарды орындау)
Енгізу және шығару		Мәлеметтерді енгізу және шығару
Таңдау		Шартқа байланысты алгоритмнің орындалу бағатың таңдау
Модификация		Циклды ұйымдастыру
Шығару		Қорытынды нәтижесін қағазға басып шығару
Қосалқы бағдарлама		Қосалқы бағдарламаны шақыру

Түсіндірме		Бағдарламма үзіндісіне түсініктеме жазу

1.3. Мәселелерді шешудің негізгі кезеңдері

Бағдарламалау ортасында белгілі бір мәселені шешу процесі, әдетте, бірнеше кезеңнен тұрады, олардың бір бөлігін пайдаланушы, ал бір бөлігін компьютер орындайды.

І-ші кезеңі. Міндеттің қойылуы .

Бұл кезеңде есептің мақсаты сипатталады ,бастапқы деректер тізімі жасалады.

ІІ-ші кезеңде. Математикалық модельді әзірлеу.

Бұл кезеңнің мақсаты-деректер мен ізделетін нәтижелер арасындағы формалды байланыстарды орнату. Кезең есептеу формулаларын немесе функционалдық тәуелділіктерді жазудан тұрады.

ІІІ-ші кезеңде. Алгоритмді әзірлеу.

Кезең іс-әрекеттердің реттілігін сипаттаудан тұрады, нәтижесінде есептің шешуі табылады.

IV ші кезеңде. Бағдарламаны әзірлеу.

Бағдарлама есепті шешудің әзірленген алгоритміне толық сәйкес жасалады.

V-ші кезеңде. Бағдарламаны ретке келтіру.

Бағдарламадағы қателерді табу және оларды жою процесі.

VI- кезеңде. Нәтижелерді талдау.

Бағдарламаға өзгерістер енгізу, қосымша есептеулер жүргізу немесе оларды аяқтау қажеттілігі туралы шешім қабылдауға мүмкіндік береді.

1.4. Python тілінің алфавиты

Кез - келген тілді үйрену алфавитті үйренуден басталады, әріптерден сөздер, сөздерден-сөйлемдер тұрады. Бағдарламалау тілін үйрену кезінде де солай болады. Алдымен белгілі бір конструкцияларды құруға болатын тіл сөздерін жазу үшін қандай символдарды қолдануға болатындығын түсінуіміз керек. Сонымен, Python алфавитіне мыналар кіреді:

A-дан z-ге және A-дан Z-ге дейінгі латын әріптері.

Python - да алфавиттің бас және кіші әріптерінің арасында айырмашылықтар бар, мысалы, chislo, CHISLO, Chislo-әр түрлі айнымалы атаулар.

Сандар 0-ден 9-ға дейін.

Арнайы таңбалар, мысалы+, -,*,/.

Сақталған (қызметтік) сөздер: for, if, class, def және т. б.

1.5. Идентификаторлар және оларды жазудың жалпы ережелері

Мәселені шешу бағдарламасы жалпылық қасиетке ие болуы үшін

шамалардың нақты мәндерін қолданбау керек, бірақ бағдарламаны орындау барысында олардың мәндерін өзгерту мүмкіндігі үшін олардың белгілерін қолдану керек. Бағдарламада айнымалылар мен тұрақты шамаларды белгілеу үшін идентификатор **атаулары қолданылады** (identification - объектінің белгілі бір таңбалар жиынтығына сәйкестігін белгілеу).
Python бағдарламасы-бұл операторлар деп аталатын нұсқаулар тізбегінен тұрады. Келесілерді ескеру қажет:

- идентификаторға **бос орындар**, арнайы **alfa** таңбалары кіре алмайды.
- идентификатор **тек әріптен немесе астын сызу белгісінен басталады**;
- идентификатор әріптерден, сандардан және астын сызу белгісінен тұруы мүмкін;
- идентификаторларды жазу кезінде латын алфавитінің бас әріптерін де, кіші әріптерін де қолдануға болады;
- идентификатор **қызмет сөзі** болмауы керек.

Мысалы:
summal дұрыс
2delta қате
B1оск_35 дұрыс
Nomer.doma қате
Сумма қате

1.6 Меншіктеу операторы

Мәселені шешу барысында компьютермен орындалатын әрекеттер алгоритмдік тілдің операторлары түрінде жазылады. Айнымалы мәнін өзгертуді меншіктеу операторы арқылы жүзеге асырады. Python-да меншіктеу мәнді белгілі бір айнымалы атпен байланыстыруды білдіреді.
Осы оператор орындайтын әрекет "="белгісімен белгіленеді.
Осы белгінің сол жағында айнымалының аты жазылады (мысалы, x=). Оң жақта сан болуы мүмкін:
Мысалы , сан x=5.
Мысалы арифметикалық өрнек x=(3*a)/2 немесе логикалық x=a>1.
Басқа айнымалы мысалы x=a.
Операторлар бағдарламада жазылған ретпен орындалады. Мысалы, арифметикалық ортаны табу бағдарламасының үзіндісі келесі операторлардан тұрады: a=5 b = 10 s = a+b v=s/2

Меншіктеу операторын келесі түрде жазуға болады:
* = меншіктеу операторы көбейту арқылы, мысалы, x * = 5 операторы x=x*5-ке сәйкес.
/ = меншіктеу операторы бөлу арқылы , мысалы, x/=5 x=x/5 операторымен бірдей.
% = меншіктеу операторы бөлуден қалған қалдық, мысалы, x%=5 сәйкес x = x % 5.

+ = меншіктеу операторы қосу арқылы, мысалы, x+=5 x=x+5 операторына сәйкес.
-= меншіктеу операторы алу арқылы , мысалы, x - =5 x=x-5 операторына ұқсас.

1.7. Мәліметтер типтері

Деректер түрлері кез-келген бағдарламалау тілінің ең негізгі ұғымдарына жатады. *Айнымалыларды анықтау (жариялау)* үшін аудармашы немесе компилятор келесі ақпаратты қажет етеді:
айнымалы атауы бойынша бағдарламадағы айнымалы компьютердің жедел жадымен байланысады;
айнымалы түрі-компиляторға айнымалы түрінде қандай ақпарат сақталатынын анықтауға мүмкіндік береді;
айнымалы мәні-айнымалыға орналасатын ақпараттың құрамын анықтайды.
Python-да жазылған бағдарламаларда қайта сипаттаудың мұндай бөлімі жоқ. Салыстырыңыз: Delphi, Pascal бағдарламалау тілдерінде, Lazarus бағдарламалау ортасында сіз Var сипаттамасы бөлімінде өз бағдарламаңызда қолданылатын айнымалыларды жариялауға міндеттісіз. Әйтпесе, олар жұмыс істемейді
Microsoft Visual Basic бағдарламалау тілінде сіз Dim сипаттамасы бөлімінде айнымалыларды жарияламай - ақ жасай аласыз, бірақ содан кейін пайдаланушы Variant деректер түрінің ерекшеліктері туралы білуі керек және айнымалылар әлі де жақсы сипатталған деген қорытындыға келеді. Visual C # бағдарламалау тілінде сіз айнымалы мәнді бағдарламашы оны қолдана бастаған сәтте сипаттай аласыз. Сонымен, Python-да тізімделгендердің ешқайсысы жоқ, дәлірек айтсақ, мәліметтер түрлері бар, бірақ айнымалы жариялау қажеті жоқ.
Python бағдарламаны орындау кезінде динамикалық типтерді пайдаланамыз, яғни мәліметтер типтері бағдарламаның орындау барысында анықталады. Айнымалы, объектінің өзін емес , белгілі бір типтегі объектіге сілтемені сақтайды,. Меншіктеу кезінде айнымалы басқа объектіге сілтеме жасайды, сол кезде айнымалы түрі де өзгереді. Python сіздің бағдарламаңыздағы айнымалы түрін сол немесе басқа айнымалыға "жіберетін" мәліметтерге сәйкес өзгертеді.
Айнымалы қандай деректер түріне сілтеме жасайтынын **type** функциясы көмектеседі , ол келесі синтаксистен тұрады:
type (айнымалы атауы) немесе **type** (мәні)
Мысалы, бағдарламада сіз type(a) немесе type(196228) мәлімдемесін жаза аласыз, ал егер A айнымалысы бірінші жағдайда бүтінмәнді типы болса, онда жауап келесідей болады: **<class 'int'>.** Екінші жағдайда, 196228 бүтін сан екені анық, сондықтан нәтиже бірдей болады. Int сөзі бүтін санға жататындығын көрсетеді.
Python-да деректердің негізгі түрлерін сипаттаймыз.
Бүтін типты мәліметтер. Бүтін сандарды көрсету үшін қолданылады. Санның мөлшері қол жетімді жедел жадтың көлемімен шектеледі. Жоғарыда айтылғандай, type функциясын type(10)
нәтижесі
<class 'int’>
Келесі арифметикалық амалдар бүтін типты мәліметтерде орындалады:

+ қосу
- азайту(алу)
* көбейту
/ бөлу
// бүтін бөлу
% бүтін санның бөлінуінен қалған қалдық
% -операциясының нәтижесін келесі формула бойынша есептеуге болады:
a%b=a - (a/b)*b.

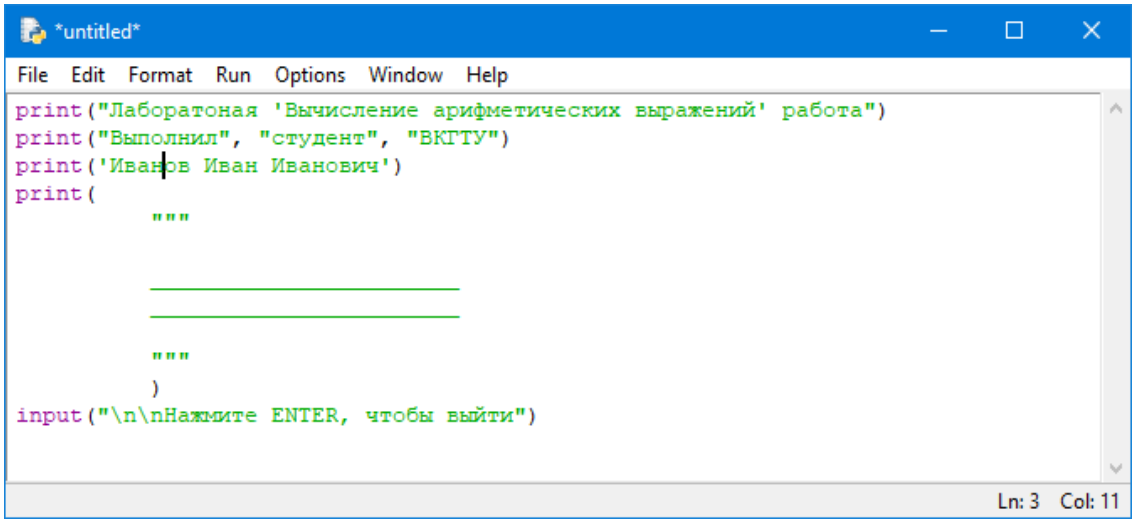
Төменде **%** және // операциялары арқылы мысалдарды орындау нәтижелері көрсетілген
Берілген: 36% 6 = 0 5 % 2 = 1 2 % 5 = 2 20 // 3 = 6 0 % 5 = 0

Нақты деректер түрлері
Python тілінде нақты (бөлшек) мәндерді өзгермелі(жылжымалы) және бекітілген нүкте түрінде ұсынуға рұқсат етіледі.
Бекітілген нүктесі бар нысанда санда "қосу" немесе "алу"белгісі бар Ондық сандар ретімен беріледі. Мысалы,
7.32; 456.721; 0.015; 192.0; -15.0 '
Жылжымалы нүктелі санда(экспоненциалды формат) қолданылады, өте үлкен немесе өте кішкентай сандарды көрсету үшін . Бұл формада сан келесі түрде жазылады:
± m E ± p
Мұнда **m** санның мантиссасы; **E** - ондық бөлшектердің негізін білдіретін символ ; **p** – санның дәрежесі;
Мысалы:
7.32E+00 4.56721E+02 1.5E-02 1.92E+02 -1.5E+01 .
Нақты деректер түрлеріне **float** типі бекітілген. type(1.5E-02)
Type функциясының нәтижесі <class 'float' >болады

Жолдық деректер түрлері

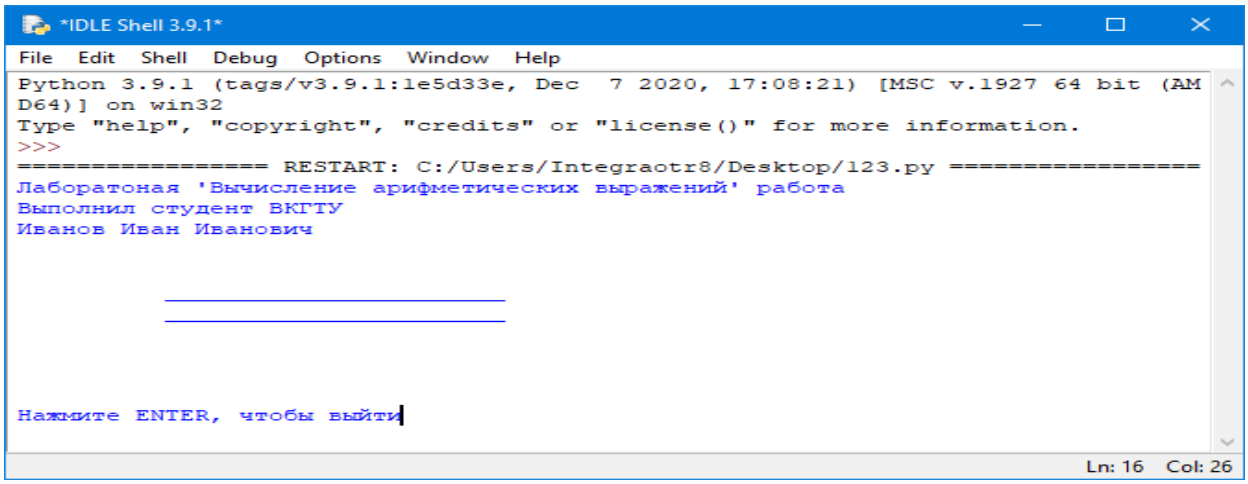
Жол айнымалыларының мәндері жол тұрақтылары (жолдар) болып табылады. Python тілінде жол типті объектілер **str** ретінде белгіленеді.
type ("зертханалық жұмыс")
Type функциясының нәтижесі <class ' str ' >болады
Келесі мысалда жолдармен жұмыс істеудің кейбір әдістері көрсетілген. Python - дағы ең танымал операторлардың бірі- **print** ол экранға мәтінді шығарады, ал мәтін қос тырнақшада орналасуы керек. Python регистрге сезімтал болғандықтан, **prin**t операторы кіші әріптермен жазылады, әйтпесе (Print немесе PRINT) жазылса ол жұмыс істемейді.
Әрі қарай, тырнақшаға салынған және **prin**t операторына берілген мән жол екенін атап өткен жөн. Жолдық тұрақты сақталатын тырнақша , жалғыз немесе қосарланған болу керек. Қос тырнақшада орналасқан ,жолдық өрнек ішінде , бір тырнақшада орналасқан жолдық өрнек орналасуы мүмкін.

Соңында, үш тырнақшаға орналасқан жол бірнеше жолға орналастырылған жол өрнегін көрсетеді.



Осы кодтағы қорытынды **input** операторы іске қосылған бағдарламаны тоқтатуға мүмкіндік береді және "шығу үшін ENTER пернесін басыңыз"хабарын көрсетеді. Бұл бағдарламалық жасақтама жолында Escape деп аталатын - кері слештан(сызықтан) және бірнеше рет қайталанатын(\n\n) **n** таңбасынан тұратын тізбек пайдаланады. Айта кету керек, көптеген бағдарламалау тілдерінде консоль режимінде Escape тізбегі қолданылады және Python ерекшелік емес. Олардың көмегімен сіз экранда көрсетілген жолдардың көрнекілігін арттыра аласыз, мысалы:
\n-курсорды келесі жолға аударады;
\t-Tab пернесін пайдалануға тең;
\\ - кері слешті көрсетеді \;
\ '- бір тырнақшаны көрсетеді;
\" - кеі тырнақшаны көрсетеді.

Python тіліндегі жолдарды байланыстыру (біріктіру) + таңбасы көмегімен орындалады. Мысалы:
print ("Қазақстан" + " 2020" \+ "ШҚМТУ")
Біріктіру операциясымен байланысты сөздер біріктірілмеуі(қосылып жазылмау) үшін тырнақшаны жаппас бұрын немесе мысалда көрсетілгендей тырнақшадан кейін бірден бос орын (бос орын) қосу керек. Кері слешті бірнеше жолды қосқан кезде пайдалануға болады.
Бір жолды бірнеше рет қайталау үшін Python тілінде келесі мысалда көрсетілген жолдың үстінен көбейту әрекетін орындау бос болады: басып шығару **print** ("ШҚМТУ"*5).
Жоғарыда қарастырылған мысалдардың қорытынды нәтижесі суретте көрсетілген. 1.



Сурет1- - Жолдармен операцияларды орындау нәтижесі

Логикалық типтер (Бульдік типі)

Осы типтегі айнымалылар екі мәннің біреуін қабылдай алады: **True** (Ақиқат) немесе **False**(жалған). Мысалы, 5>3 Логикалық өрнегінің мәні - ақиқат, ал 5=3 мәні-жалған. A>5 шартының мәні a айнымалысының мәніне байланысты және A мәнінің мәні белгісіз болғанға дейін анықталмайды.
Type функциясын логикалық типтегі объетіге пайдалану
type(True) немесе type (False) жауап береді
<class 'bool'>, int(True) нәтижесі 1, ал int(False) - 0 болады.
Логикалық типтегі мәліметтер логикалық операцияларды орындай алады. Олардың кейбірлері кестеде1 келтірілген.

Таблица 1. **Логические операции**

Операция	Действие	Выражение
and (конъюнкция)	Логическое И	A **and** B
or (дизъюнкция)	Логическое ИЛИ	A **or** B
not (отрицание)	Логическое НЕ	**not** A

Python-да, басқа бағдарламалау тілдерінен айырмашылығы, осы формада жазылған шартты тексеруге болады: 1<a<5. (1<a) and (a<5) шартын тексеруге де рұқсат етіледі, және бұл шарттың мәні, егер оған кіретін қарапайым шарттардың екеуі де ақиқат болса ғана ақиқат болады.
Біз Python бағдарламалау тілін үйрену кезінде болашақта қолданатын мәліметтердің негізгі түрлерін қарастырдық. Динамикалық типизацияға сәйкес айнымалы түрін оған басқа мән беру арқылы өзгертуге болатындығын тағы бір

рет атап өтеміз. Мысалы, операторларды орындау барысында a = 2017 a = "Жаңа жыл" a = 3.14 **a** айнымалысы деректер түрін бүтіннен жолға, содан кейін нақты түрге ауыстырады.

1.8. Типтерді келтіру функциялары

Өздеріңіз білетіндей, компилятор әр айнымалының түрін анықтай алуы керек. Сонымен қатар, егер бір айнымалыға екіншісінің мәні берілсе және түрлендіруді қажет етсе (мысалы, float-тан int-ке дейін), мұндай түрлендіру айқын болуы керек.
Түрлерді нақты түрлендіруге мүмкіндік беретін Python тілінің функцияларының тізімін қарастырайық:
y=**boo1**(объект) - объектіні логикалық типке келтіреді. У айнымалысында объектінің өзі емес, логикалық типтегі объектіге сілтеме сақталады, дегенмен меншіктеу операторы орындалғанға дейін у айнымалысында басқа типтегі объектіге сілтеме болуы мүмкін. Дәл осындай жағдай деректер түрлерін түрлендірудің басқа функцияларымен де болады;
y = int (объект) - объектіні бүтін түрге келтіреді;
y = float (объект) - объектіні нақты түрге келтіреді;
y = str (объект) - объектіні жол түріне келтіреді ;
y = тізім (реттілік) - реттілік элементтерін тізімге түрлендіреді;
y = tuple (реттілік) - реттілік элементтерін кортежге айналдырады.
Комментариялар
Бағдарламаны жақсы түсіну үшін түсіндірме мәтін - түсініктеме жиі жазылады. Комментариялар бірнеше маңызды функцияларды орындайды:
бағдарламаны жеңіл оқылатынғып жасайды, жолдарда түсініктемесін жазып;
бағдарламаны жөндеу кезінде оның үзіндісін уақытша өшіріп қояды
Python тілінде бағдарламалық жолға түсініктеме # таңбасын қолдану арқылы мүмкін болады. Мысалы,
 # Екі санның қосындысы есептеледі
sum=a+b

1.9. Математикалық функцияларды жазу

Жеке компьютердің пернетақтасынан кейбір стандартты математикалық функцияларды жазудың мүмкін еместігіне байланысты Python тілінде қолданушы арифметикалық өрнектерді жазатын кірістірілген функциялар бар.
Кестеде2 Python тілінің кейбір математикалық функциялары келтірілген. Математикалық функцияларды пайдаланбас бұрын, бағдарламаның басында **import math** нұсқаулығын жазу керек, бірақ содан кейін әр функцияны айтпас бұрын модульдің атын қосу керек - math, мысалы, y=math.sin(x). Math модулін бірнеше рет шақырмас үшін бір тәсілді қолдануға болады - бағдарламаның басында келесі жазбаны жазу арқылы: **math import *.**

Кесте 2 – **Math** модулінің жалпы математикалық функциялары

Функция	**math** модулінде Python тілінде функцияның жазылуы	Орындалатын әрекеттер
cos X	**math.cos(X)**	Вычисляет значение косинуса X (X указывается в радианах)
sin X	**math.sin(X)**	Вычисляет значение синуса X (X указывается в радианах)
tg X	**math.tan(X)**	Вычисляет значение тангенса X (X указывается в радианах)
ctg X	**math.cos(X)/ math.sin(X)**	Вычисляет значение котангенса X (X указывается в радианах)
$\|X\|$	**math.fabs(X)**	Вычисляет абсолютное значение числа X
e^X	**math.exp(X)**	Возвращает результат возведения числа **e** в степень X
$\log_{base} X$	**math.log(X, [base])**	Возвращает логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм.
ln X	**math.log1p(X)**	Возвращает натуральный логарифм от числа (1 + X). При X → 0 точнее, чем math.log(1+X)
lg X	**math.log10(X)**	Возвращает логарифм X по основанию 10.
mod(x,y)	**math.fmod(X, Y)**	Возвращает остаток от деления X на Y.
$\sqrt{X}$	**math.sqrt(X)**	Возвращает квадратный корень из числа X
acos X	**math.acos(X)**	Возвращает значение арккосинуса от числа X в радианах.
asin X	**math.asin(X)**	Возвращает значение арксинуса от числа X в радианах.
atg X	**math.atan(X)**	Возвращает значение арктангенса от числа X в радианах.
π	**math.pi**	Возвращает значение числа π.
e	**math.e**	Возвращает значение числа e.
degrees(X)	**math.degrees(X)**	Конвертирует радианы в градусы.
radians(X)	**math.radians(X)**	Конвертирует градусы в радианы.
floor(X)	**math.floor(X)**	Возвращает число округленное до ближайшего меньшего целого числа к X.
ceil(X)	**math.ceil(X)**	Возвращает число округленное до ближайшего большего целого числа к X.
X!	**math.factorial(X)**	Возвращает факториал числа X

		(например 3!=3*2*1).

Кестеде. 3 **math** модулін қосуды қажет етпейтін сандармен жұмыс істеуге арналған кейбір кірістірілген функцияларды ұсынады.

Кесте 3 – **Math** модулінің байланысын қажет етпейтін сандармен жұмыс істеуге арналған функциялар

Функция	Запись функции на языке Python	Выполняемое действие
X ≈	**round(X)**	Возвращает результат округления числа X до ближайшего меньшего целого значения для чисел с дробной частью меньше 0,5 или результат округления числа X до ближайшего большего целого значения для чисел с дробной частью больше 0,5. В случае если дробная часть числа X ровно 0,5, то результат округления числа будет к ближайшему целому четному числу к X.
$\mathbf{X^y}$	**pow(X,Y)** или **X**Y**	Возвращает результат возведения числа X в степеньY.
max(X,…, Z)	**max(**список чисел через запятую**)**	Возвращает большее из списка чисел.
min(X,…, Z)	**min(**список чисел через запятую**)**	Возвращает меньшее из списка чисел.
X+…+Z	**sum(**список чисел через запятую**)**	Возвращает сумму всех чисел из списка.
Int(объект**)**	**int(**объект**)**	Преобразует объект (например, строковое или допустим дробное значение) в целое число
Float(объект**)**	**float(**объект**)**	Преобразует объект (например, строковое или допустим целое значение) в вещественное число

Python егер бағдарламалаушы бастапқы деректерді(операндтарды) және математикалық әрекеттерді **print** операторында көрсетсе, мысалы,
print(2016+40-20) онда 2036 санын экранға шығарады.

1.10. Қатынастар операциялары

Қатынастар операциялары кесте 4 көрсетілген

Таблица 4. **Операции отношения**

Операция	Описание
Операнд1 < Операнд2	Меньше
Операнд1 > Операнд2	Больше
Операнд1 <= Операнд2	Меньше или равно
Операнд1 >= Операнд2	Больше или равно
Операнд1 != Операнд2	Не равно
Операнд1 == Операнд2	Равно

Бақылау сұрақтары

1. Алгоритм деген не? Оның қандай қасиеттері бар?
2. Алгоритмдерді сипаттау тәсілдерін атаңыз және түсіндіріңіз.
3. Блок схемаларында қолданылатын функционалды блоктарды сызыңыз. Олардың мақсатын түсіндіріңіз.
4. Бағдарламалау барысында орындалатын мәселені шешу кезеңдерін атаңыз.
5. Python алфавитіне не кіреді? "Идентификатор" түсінігін түсіндіріп, идентификаторларды жазудың жалпы ережелері туралы айтып беріңіз.
6. Меншіктеу операторының әрекеті қандай? Бөлшек сандарды жазудың екі формасы қандай?
7. Python тілінде қолданылатын динамикалық типтеудің ерекшелігі неде?
8. Python тілінің әр түрін сипаттаңыз.
9. Бүтін типті деректермен қандай операциялар анықталады?
10. Print пен Input операторларының мақсаты қандай? Мұндай операторларды пайдалануға мысалдар келтіріңіз.
11. Логикалық типті деректермен қандай логикалық операциялар орындалуы мүмкін?
12. Типтерді келтіру функциялары қандай? Мысалдар келтіріңіз.
13. Бағдарламалардағы комментариялар қандай мақсаттарда қолданылады? Python-да бағдарламалық кодтың бөлімін қалай түсіндіруге болады?
14. Математикалық функцияларды қолдану үшін Python тілінде жазылған бағдарламаларда қандай нұсқаулар жазылуы керек?