Комерциялық емес акционерлік қоғамы

ҒҰМАРБЕК ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ

«Телекоммуникация және инновациялық технологиялар» кафедрасы

ТАРТ 2209 ТЕЛЕКОММУНИКАЦИЯДАҒЫ АЛГОРИТМДІК БАҒДАРЛАМАЛАУ ТІЛДЕРІ

"6В06201–Радиотехника, электроника және телекоммуникациялар" мамандығы бойынша дәрістер жинағы (Бакалавр)

Алматы 2021

ҚҰРАСТЫРЫШЫЛАР: аға оқытушы Г. К. Касымова, PhD, доцент Қадылбекқызы Эльвира, оқытушы Шенер А.В. Дәрістер жинағы "6B06201– Радиотехника, электроника және телекоммуникациялар" мамандық студенттеріне арналған. Алматы: АЭжБУ, 2021 – 76 бет.

Дәрістер жинағында заманауи технологиялар мен бағдарламалау әдістерінің теориялық негіздері, бағдарламаларды құрудың практикалық мәселелері, сондай-ақ негізгі алгоритмдік конструкциялар мен оларды Python жоғары деңгейлі тілінде жүзеге асыру қарастырылған.

Рецензент: к.т.н., доц. Тусупова Б.Б.

2021 жылға арналған Ғ. Дәукеев атындағы "Алматы энергетика және байланыс университеті" коммерциялық емес акционерлік қоғамының жоспардан тыс басылымы бойынша басылып шығарылады.

Мазмұны

Кіріспе

Дәріс жазбалары «Телекоммуникациядағы алгоритмдік бағдарламалау тілдері» пәнінде Python алгоритмдік программалау тілін зерттеуге арналған (дәріс - 15 сағат.) «6В06201 - Радиотехника, электроника және телекоммуникация» мамандығының бірінші курс бакалаврлары үшін.

Дәріс жазбаларында бағдарламалық қамтамасыз ету әдістерінің заманауи технологияларының теориялық негіздері, сонымен қатар негізгі алгоритмдік конструкциялары мен жоғары деңгейдегі Python тілінде енгізу қарастырылған. Бағдарламалаудың теориялық негіздерін қарастыру студенттерге бағдарламалаудың практикалық дағдыларын қалыптастыруға мүмкіндік беретін, құру әдістерін көрсететін бағдарламалардың мысалдарымен, сонымен қатар өз бетінше орындауға арналған тапсырмалармен бірге жүреді.

Білім алушылар нұсқаулық материалдарын зерделеу нәтижесінде:

Білу:

- бағдарламаның құрылымы;
- негізгі мәліметтер мен олардың ерекшеліктері;
-стандартты тіл модульдері.

Істей білу:

-әр түрлі мәліметтер бойынша стандартты операцияларды орындау;

-Мәліметтерді өңдеуге арналған стандартты алгоритмдік құрылымдарды қолдану.

Игеру:

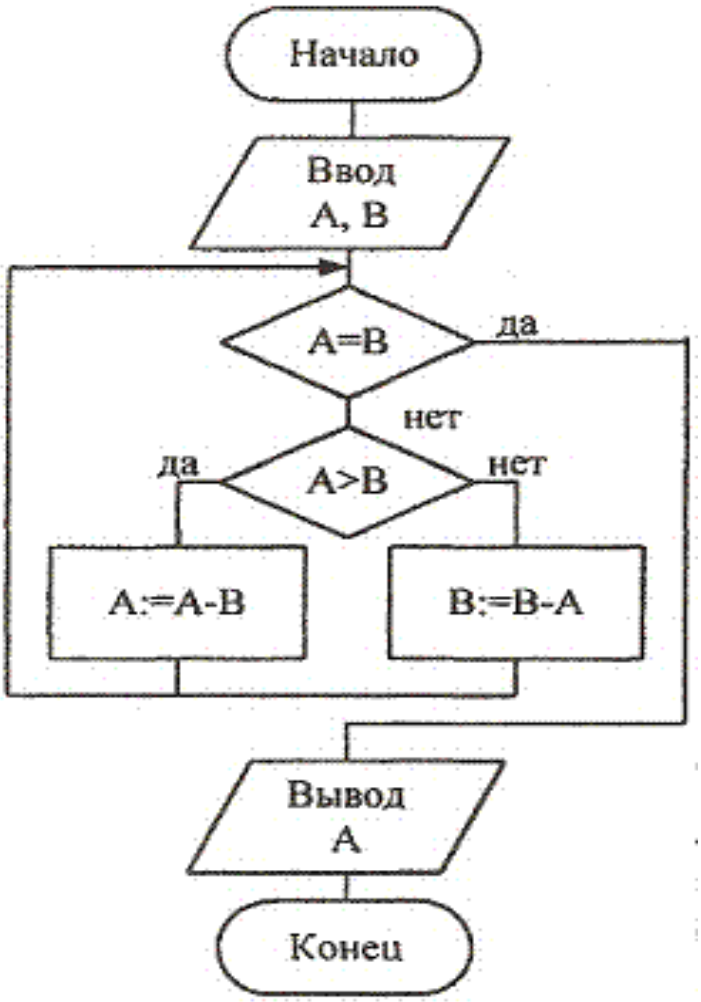
-бағдарламаларды жазу кезінде объектіге бағытталған тәсіл принциптері;

-мәліметтер типінің айнымалыларымен жұмыс істеу ерекшеліктері.

1 дәріс. Алгоритмдеудің негізгі түсінігі.

Алгоритм - бұл нақты тәсіл, қажетті нәтижеге жеткізілетін бастапқы мәліметтерден басталатын логикалық және есептеу процесін анықтайтын тәсіл.

Алгоритмдеу мен алгоритм ұғымы олардың жалпы пәні - кибернетикаға жататындығын көрсететін белгілі бір алгоритмдерді енгізуге



дейін тек компьютерлерді ғана емес, басқа да барлық техникалық құрылғылар мен жүйелерді басқару процестерімен бірге жүреді.

Алгоритмнің қасиеттері:

1. детерминизм - нұсқаулардың өз бетінше түсіндірілуін болдырмайтын дәлдігі;
2. дискреттілік - есептеу процесін жеке элементарлы операцияларға бөлу мүмкіндігі, орындалу мүмкіндігі күмән тудырмайды;
3. тиімділік - қажетті нәтижелерді немесе есептік процесті жалғастырудың мүмкін еместігі туралы хабарламаны шығарумен белгілі бір қадамдардан кейін процесті тоқтату;

4. бұқаралық сипат - берілген класстың барлық есептерін шешуге алгоритмнің жарамдылығы.

Алгоритмдеу процесі алдымен жалпы форманы - алгоритмдік тілді қолдана отырып, жалпыға ортақ жеке түрге ауысады, компьютерлік бағдарлама түрінде жоспарланған әрекеттерді орындайтын компьютер түсінетін бағдарламалау тілі түрінде аяқталады.

Алгоритмдік тіл- алгоритмдерді жазу үшін осы таңбалардың конструкцияларын қалыптастыру және түсіндіру үшін символдар мен ережелер жиынтығы.

Бағдарламалау тілі- компьютерлік бағдарламаларды жазуға арналған ресми белгі жүйесі. Бағдарламалау тілі бағдарламаның сыртқы түрін және орындаушы (әдетте компьютер) басқаратын әрекеттерді атап айтқанда лексикалық, синтаксистік және семантикалық ережелер жиынтығын анықтайды.

Компьютер тілі- бағдарламалау тіліне жақын, бірақ адамға емес, компьютерге бағытталған ұғым компьютерде бағдарламаларды жүзеге асыруға арналған. Шын мәнінде, компьютерлік тіл - бұл хаттама, адамның компьютермен және компьютерлік бағдарламаның басқа компьютерлік бағдарламамен ақпарат алмасу ережелері.

Компьютерлік бағдарлама бұл машина қабылдаған формада жазылған алгоритм. Бағдарламада команданың мәліметтерін сипаттаумен қатар, машина қандай ретпен, қандай деректермен және қандай операцияларды орындау керек, сонымен қатар нәтижені қандай формада алу керек екендігін әртүрлі операторлар қамтамасыз етеді.

Оператор- бағдарламаның нақты әрекетінің көрсеткішін білдіретін компьютер тілінің бір қарапайым өрнегі; әдетте, бірдеңе жасаңыз.

Бұл пән студентке компьютерлік бағдарламалаудың тәжірибесі мен дағдыларын меңгеруге арналған - электронды компьютерлер (ЭВМ). Бағдарламалаудың негізі - алгоритмдеу процесі және компьютерлік программалау тілдерін білу.

Аспаптық бағдарламалық жасақтаманы дамытуда бағдарламалау тілдерінің бес буыны қарастырылады (АЯП).

1. Машинаға бағытталған, игеру қиын, компьютерді жақсы білу қажет (ЭВМ)
2. Ассемблер, макроассемблер - сонымен қатар машинаға тәуелді
3. Жоғары мобильді, адамға бағытталған тілдер, үйренуге оңай. 1956 жылы бірінші деңгейлі тіл - Фортран, IBM -де Дж.Беккус басшылығымен дамыған . Қысқа уақыт ішінде Фортран инженерлік, техникалық және ғылыми мәселелерді шешудің негізгі тіліне айналды. Бастапқыда Фортранның кейіпкерлер туралы ақпаратты өңдеуге мүмкіндігі шектеулі болды.
4. Процедуралық емес, объектіге бағытталған, сұрау тілдері, параллель қарапайым қолданушыға және параллель сәулеті бар компьютерлерге бағытталған. Мұндай Algol, COBOL, Basic, PL / 1, Паскаль,

APL, ADA, C және т.б. Төртінші, Lisp, Modula, сондай-ақ танымал тілдері Қазіргі 2000 түрлі жоғары деңгейлі тілден астам бар.

5. Жасанды интеллект тілдері, сараптамалық жүйелер мен білім базалары, табиғи тілдер компьютерлердің интеллектуалды деңгейін және тілдермен интерфейсті арттыруға бағытталған. Жасанды интеллект, сараптамалық жүйелер, білім базалары тілдері (InterLisp, ExpertLisp, IQLisp, SAIL және т.б.), сондай-ақ қандай да бір арнайы синтаксисті игеруді талап етпейтін табиғи тілдер (қазіргі уақытта мүмкіндіктері шектеулі табиғи ЯП — Clout, Q&A, HAL және т. б.) сәтті пайдаланылуда.

Төртінші буын тілдері процедуралық емес сипатқа ие, мұндай тілдердегі бағдарламалар оны қалай жасау керектігін емес, нені сипаттайтындығымен анықталады. Бағдарламаларда коэффициенттер алгоритмдерді орындау қадамдарын орындаудан құрылады. Процедуралық емес тілдерге мысал ретінде жасанды интеллект тапсырмалары үшін қолданылатын тілдерді алуға болады (мысалы, Пролог, Лангин). Процедуралық емес тілдерде синтаксистік ережелер минималды болғандықтан, олар бағдарламалаушы емес мамандардың қолдануына әлдеқайда қолайлы. Төртінші буын БТ дамуының екінші бағыты-бұл Simula-67 тілінде алғаш рет қолданылатын және кейіннен танымал SmallTalk тілінің негізін құрайтын бағдарламалық объект ұғымына негізделген объектіге бағытталған тілдер.

Бағдарламалық қамтамасыз ету объектісі мәліметтер құрылымы мен алгоритмдерден тұрады, олардың әрқайсысы өз деректерінде операцияларды қалай орындау керектігін біледі. Іс жүзінде әр түрлі объектілерді әр түрлі алгоритмдер қолдана алады, сол 3 кілт сөзімен (полиморфизмнің қасиеті деп аталады). Мысалы, деректер ретінде күрделі сандары мен массивтері бар объект көбейту операциясын орындау үшін әр түрлі алгоритмдерді қолданады. Мұндай қасиеттерге объектіге бағытталған Паскаль, Basic, C ++, SmallTalk, Simula, Actor және басқа да бірқатар бағдарламалау тілдері ие. Төртінші буын тілдерін дамытудың үшінші бағыты пайдаланушыға мәліметтер қорынан ақпарат алуға мүмкіндік беретін тілдерді қарастыруға болады. Сұрау тілдерінің өзіндік арнайы синтаксисі бар, олар БТ үшінші буынындағыдай орындалуы керек, бірақ сонымен бірге қолдану оңай. Сұранысқа ие тілдерінің арасында SQL (StructuredQueryLanguage) іс жүзінде стандартқа айналды. Сонымен, дамудың төртінші бағыты- параллельді бағдарламалау тілдері (Fortran IWU, Occam, SISAL, FP және т.б. тілдерінің модификациясы), Үшінші буын тілдерден айырмашылығы дәстүрлі процессорлық энергияға бағытталған.

Бағдарламалау тілдерінің элементтері

Бағдарламалау тілдерінің элементтерін келесі деңгейлерде қарастыруға болады: алфавит - баспа және экрандық құрылғыларда көрсетілетін және / немесе терминал пернетақтасынан енгізілген таңбалар жиыны.

лексика - тілді субстанцияларды (айнымалылар мен белгілерді), операторларды, операцияларды және басқа да лексикалық компоненттерді құрайтын символдар тізбегін (лексемаларды) құруға арналған ережелер жиынтығы. Бұған операторлардың сөздерін, кіріктірілген функцияларды және т.б. синтаксисті белгілеуге арналған резервтелген (тыйым салынған, кілт) БТ кіреді. Синтаксис - тілдік конструкцияларды немесе БТ сөйлемдерін құруға арналған ережелер жиынтығы - блоктар, процедуралар, құрама операторлар, шартты операторлар, цикл және т.б.

Объектіге бағытталған бағдарламалау (ОББ)

Объектіге бағытталған программалау (ОББ)-бұл объектілер бағдарламаның негізгі элементтері болып табылатын бағдарламалау техникасы. Бағдарламалау тілінде объект қасиеттер жиынтығы (объектінің сипаттамалық деректер құрылымы), оларды өңдеу әдістері (олардың қасиеттерін өзгерту тәртібі) және объект жауап беруі мүмкін оқиғалар жиынтығы ретінде жүзеге асады. объектінің қасиеттерінің өзгеруі. Деректер мен өзінің өңдеу процедураларын бір объектіде біріктіру инкапсуляция деп аталады және ОББ -ның маңызды принциптерінің бірі болып табылады.

Тағы бір негізгі ұғым - класс.

Класс бұл нақты бағдарлама объектісін құруға болатын үлгі; бұл осы объектілердің мінез -құлқын анықтайтын көрсетілген қасиеттер мен әдістер. Бұл сыныпқа ие әрбір нақты объект сынып экземпляры деп аталады.

ОББ келесі маңызды принциптері мұрагерлік және полиморфизм болып табылады.

Мұрагерлік- қолдану негізінде жаңа класстар құруды қарастырады және ұрпақ класына ата -аналық класстың барлық қасиеттеріне ие болуға (мұрагерлікке) мүмкіндік береді.

Полиморфизм- туылған объектілерде тізбектің қай жерінде орналасқанына байланысты қандай әдістерді қолдану керектігі туралы ақпарат бар екенін білдіреді.

Модульділік- бір объектінің ресурстарын басқа объектілерде қолдану мүмкіндігі.

Қазіргі объектіге бағытталған бағдарламалық тілдері C ++және Java... Объектіге бағытталған жүйелік визуалды агент Visual Basic, Delphi, C ++ құрастырушысы, Visual C ++. VBA (Visual Basic for Applications) тілі - Microsoft Office қосымшаларының тілі (Excel, Word,Қол жеткізу, Power Point және т. VBA негізгі тіл синтаксисі мен бағдарламалау ережелерін ұстанады негізгі тілдер- диалектілер, пайдаланушының кейбір және графикалық интерфейстерін орындау үшін тапсырмалар құруға, әр түрлі бағдарламалық өнімдер арасында өзара әрекеттесуге мүмкіндік береді.

№2 дәріс. Тілдің негізгі элементтері. Айнымалылар

Негізгі түрлері:
char short int long float double
Операциялар:
(қосу)
(ЖӘНЕ)
(эксклюзивті
НЕМЕСЕ)
(қамтитын
НЕМЕСЕ)
(логикалық солға
< жылжу)
(логикалық оңға
> жылжу)

Бағдарламалау тілінің айнымалысы - бұл мәндерді сақтауға арналған компьютер жадындағы резервтелген кеңістіктің атауы. Бұл айнымалы мәнді құрған кезде сіз шынымен компьютер жадында бос орын сақтайсыз дегенді білдіреді. Мәліметтер түріне сүйене отырып, аудармашы жадтың қажетті көлемін бөледі және жадтың резервтелген аймағында не болуы мүмкін екенін шешеді, кез келген нәрсе болуы мүмкін, бірақ тек белгілі бір мөлшерде. Бұл мысалдағы өлшем модельдің түрі болады. Мәндерді айнымалыларға тағайындау: Python сізге C ++ сияқты модель түрін қолмен жариялаудың қажеті жоқ. Айнымалыға мән тағайындаған кезде декларация автоматты түрде болады. Мәндерге мәндерді тағайындау үшін теңдік белгісі (=) қолданылады.

Мысалға:

country = «Klad» # Ел деп аталатын айнымалыларға Klad мәнін тағайындаңыз

жас = 55 # 55 айнымалы мәніне 55 тағайындаңыз

баспасөз елі

басып шығару жасы

Орындалған кезде бұл код шығады: **Klad 55**

Мәндерді бірнеше тағайындау: Python -да бір мәнді бірнеше айнымалыға бірден тағайындауға болады.

Мысалы: a = b = c = 88

Стандартты мәліметтер типіне мыналар жатады:

- Сандар (сандар)
- Жол (жол)
- тізім (тізім)
- Жұптау (қосылу)
- Сөздік (сөздік)
- Орнату (орнату)

Сандық объектілер оларға мән берген кезде.

Мысалға:

n1 = 23

n2 = 42

Сондай -ақ del кілт сөзімен сандық нысанды жоюға болады.

del n1 # n1 айнымалысын жояды

del n2, n3 # бірден n2, n3 екеуін де жояды

Python -да сандық мәліметтердің төрт түрі бар:

- int (бүтін сан)

- ұзын (ұзын бүтін сан [сегіздік немесе он алтылық жүйеде болуы мүмкін])

- өзгермелі (өзгермелі нүкте нөмірі: -0.2, 0.0, 3.14159265 және т.б.)

- сандық деректер түріне түрлерінің кешені (кешенді саны) мысалдары: INT ұзақ қалтқы l 51924361L 0.0 3.14j 102 кешенді -0x19323L 15.20 45.j -786 0122L -21,9 9.322e-36j 0 0xDEFABCECBDAECBFBAEl 32,3 + E18 0b10 535633645298 .876j . + 0J -0x260 -052318172735L -32.54e100 3e + 26J 0x69 -4721885298529L 70.2 -E12 4.53e -7j

№3 дәріс. Python if else - шартты мәлімдеме

Қарапайым мысал («True» бассақ, 1-шындық екенін білуге болады):

```
>>> if 1:
...     print('true')
... else:
...     print('false')
...
true
```

Күрделі мысал

```
a = int(input())
if a < -5:
    print('Low')
elif -5 <= a <= 5:
    print('Mid')
else:
    print('High')
```

Python -да шындықты тексеру
Логикалық операторлар:

```
X and Y
```

Егер X пен Y екеуі де дұрыс болса.

```
X and Y
```

Егер X немесе Y мәндерінің кем дегенде біреуі дұрыс болса, бұл дұрыс.

```
not X
```

Егер X жалған болса, дұрыс.

№ 4 дәріс. Python тілінде while операторы

while шарт:
 нұсқаулар
Мұнда

- *шарт*- нұсқаулықтың жолдарын орындау шарты;
- *нұсқаулар*- егер шарт = True болса орындалатын бір оператор немесе операторлар жиынтығы. Егер шарт False -ге тең болса, онда while циклінің операторының орындалуы аяқталады.

Тізімдегі элементтердің қосындысын есептеу

```
# тізім құрып, мәндері бар айнымалыларды инициализациялаңыз
A = [1,3,5,8, -3,10]
i = 0
summ = 0 # қажетті сома
while i<len(A):        #   қосынды есептеуге арналған цикл
   summ=summ+A[i]
   i=i+1
print("summ = ", summ)    # summ = 24
```

Python тілінде for операторы

```
for <variable> in <object>:
   <statements1>
   if <condition1>: break
   if <condition2>: continue
else:
   <statements2>
```

Мұнда

- *Айнымалы (variable*) - объектіден элементтердің мәнін кезекпен алатын цикл айнымалысы;
- *Объект* (*object*) - айналып өтуге арналған көптеген элементтер бар кейбір тізім, түйін, жол немесе басқа нысан (объект);
- *Мәлімдемелер1 (statements1)* - берілген айнымалы үшін орындалатын бір немесе бірнеше операторлар (нұсқаулар);
- *мәлімдемелер2 ((statements2)*- егер цикл денесінде нұсқаулық бір рет те орындалмаған болса, орындалатын бір немесе бірнеше операторлар (командалар).

Операторды басқа блоксыз қысқартылған түрде қолдануға болады. Бұл жағдайда оператордың қысқартылған түрі:

```
for <variable> in <object>:
   <statements>
```

Мұнда

- *Айнымалы (variable)*- объектіден элементтердің мәнін кезекпен алатын цикл айнымалысы;
- *Мәлімдемелер (statements)*- берілген мән бойынша орындалатын бір немесе бірнеше операторлар (нұсқаулар).

Тізім элементтерінің қосындысын есептеуге мысал

```
# тізім сомасын есептеу
# Көрсетілген тізім
T = [2.8, 3.5, 4.9, 0.01, 2.34]
s = 0
for t in T:
 s = s + t
print("s = ", s)          # s = 13.549999999999999
```

Жоғарыда көрсетілген кодты орындау нәтижесінде келесі нәтиже көрсетіледі

```
s = 13.549999999999999
```

№5 дәріс. Питондағы жол операторлары

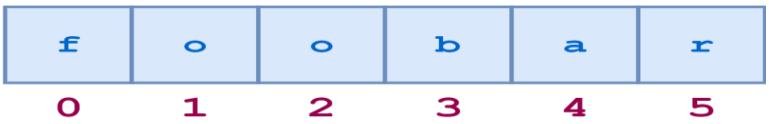
Жол қосу операторы + - жолдарды біріктіру операторы.
Жолды көбейту операторы *
s * n
n * s Жолдарды көбейтудің мысалдары:
>>> s = 'ру'.
>>> в * 4
'ру.ру.ру.ру.'
Қосалқы жолдың тиістілік операторы in
Python сонымен қатар жолдарды басқару үшін қолдануға болатын байланыс операторын ұсынады.. Ішкі жол жолға қосылған болса, in операторы True мәнін қайтарады, ал егер жоқ болса, False:
>>> s = 'Python'
>>> s in 'I love Python.'
True
>>> s in 'I love Java.'
False
Python аудармашыға енгізілген көптеген мүмкіндіктерді ұсынады.
ФункцияСипаттамасы
chr () Бүтін санды символға айналдырады
ord () Таңбаны бүтін санға түрлендіреді
len () Жолдың ұзындығын қайтарады
str () Объектінің түрін өзгертеді string

>>> ord ('a')
97
>>> ord ('#')
35
•
Ord () функциясы сонымен қатар Юникод таңбалары үшін сандық мәндерді қайтарады:
>>> ord ('€')
8364
>>> ord ('∑')
8721
>>> chr (97)
'a'
chr () Юникод таңбаларын да өңдейді:
>>> chr(8364)
'€'
>>> chr(8721)
'∑'

```
>>> s = 'Простоя строка.'
>>> len(s)
15
>>> str(49.2)
'49.2'
>>> str(3+4j)
'(3+4j)'
>>> str(3 + 29)
'32'
>>> str('py')
'py'
```

Жолдарды индекстеу

Көбінесе бағдарламалау тілдерінде реттелген деректер жиынтығындағы жеке элементтерге сандық индекс немесе кілт арқылы қол жеткізуге болады. Бұл процесс индекстеу деп аталады.



Жеке таңбалар индекс бойынша келесідей қол жетімді:

```
>>> s = 'foobar'

>>> s[0]
'f'
>>> s[1]
'o'
>>> s[3]
'b'
>>> s[5]
'r'
```

Міне теріс индекстеудің бірнеше мысалдары:

```
>>> s = 'foobar'
>>> s[-1]
'r'
>>> s[-2]
'a'
>>> len(s)
6
>>> s [-len (s)] # теріс индекстеу -1 -ден басталады
'f'
```

Жолдарды кесу

«‘‘string slice’’» (жол бөлігі) деп аталатын жолдан ішкі жолды шығару. Егер s - жол болса, s [m: n] түрінің өрнегі m позициядан басталатын және n позициясын қоспағанда, s бөлігін қайтарады:

>>> s = 'python'
>>> s[2:5]
'tho' 'дегенмен'

Сол сияқты, егер сіз екінші [s: n] индексін жібермесеңіз, онда кесінді бірінші индекстен жолдың соңына дейін созылады. Бұл неғұрлым күрделі s [n: len (s)] -ге жақсы, қысқа балама:

>>> s = 'python'
>>> s[2:]
'thon'
>>> s[2:len(s)]
'thon'

Теріс индекстерді тілімдермен де қолдануға болады.

>>> s = 'питон'
>>> c [-5: -2]
'yth'
>>> c [1: 4]
'yth'
>>> s [-5: -2] == s [1: 4] True (Дұрыс)

Python кірістірілген жол әдістері

Жоқ	Әдісі мен сипаттамасы		Мысал

	string.capitalize()	бірінші әріпті бас әріпке, қалғандарын кіші әріпке түрлендіреді.	>>> s = 'everyTHing yoU Can IMaGine is rEAl'
			>>> s.capitalize()
			'Everything you can imagine is real'

	string.lower() .	барлық алфавиттік таңбаларды кіші әріпке түрлендіреді	'everyTHing yoU Can IMaGine is rEAl'.lower() 'everything you can imagine is real'
	string.swapcase()	алфавиттік таңбалардың жағдайын өзгертеді.	>> 'everyTHing yoU Can IMaGine is rEAl'.swapcase() 'EVERYthING YOu cAN imAgINE IS ReaL'
	string.title()	барлық сөздердің бірінші әріптерін бас әріпке түрлендіреді	>>> 'follow us @PYTHON'.title() 'Follow Us @Python'
	string.upper().	барлық әріптік әріптерді бас әріпке түрлендіреді	'follow us @PYTHON'.upper() 'FOLLOW US @PYTHON'

	string.count(<sub>[, <start>[, <end>]])	жолдағы ішкі жолдың пайда болу санын есептейді.	>>> 'foo goo moo'.count('oo') 3
	string.endswith(<suffix>[, <start>[, <end>]]) .	жол берілген ішкі жолмен аяқталатынын анықтайды	>> 'python'.endswith('on') True >>> 'python'.endswith('or') False
	string.find(<sub>[, <start>[, <end>]])	берілген жолды іздейді.	'Follow Us @Python'.find('Us') 7
	string.index(<sub>[, <start>[, <end>]]).	берілген жолды іздейді	'Follow Us @Python'.index('you') Traceback (most recent call last): File "<pyshell#0>", line 1, in <module> 'Follow Us @Python'.index('you') ValueError: substring not found
	string.isalnum()	жолда әріптер мен сандар бар -жоғын анықтаңыз.	'abc123'.isalnum() True >>> ''.isalnum() False

	string.isalpha() .	жолда тек әріптер бар -жоғын анықтаңыз	>>> 'ABCabc'.isalpha()
			True
			>>> 'abc123'.isalpha()
			False

	string.isdigit()	санның цифрларды қамтитынын анықтаңыз (қатысуын тексеріңіз).	>>> '123'.isdigit()
			True
			>>> '123abc'.isdigit()
			False

	string.isidentifier() идентификатором Python.	жолдың жарамдылығын анықтайды	'foo32'.isidentifier() True >>> '32foo'.isidentifier() False

	string.islower().	жолдың әріптік әріптері кіші әріп екенін анықтаңыз	>>> 'abc'.islower()
			True
			>>> 'Abc1$D'.islower()
			False

	string.isprintable()	жолда тек басып шығарылатын таңбалар бар -жоғын анықтаңыз.	>>> 'a\tb'.isprintable() # \t - символ табуляции
			False
			>>> 'a b'.isprintable()

			True

	string.isspace()	жолда тек бос орын таңбалары бар -жоғын анықтаңыз.	>>> ' \t \n '.isspace()
			True
			>>> ' a '.isspace()
			False

	string.istitle().	жолдың сөздері бас әріптен басталатынын анықтаңыз	'This Is A Title'.istitle() True >>> 'This is a title'.istitle() False
	string.isupper()	жолдың алфавиттік таңбалары бас әріп екенін анықтаңыз.	>> 'ABC'.isupper() True >>> 'Abc1$D'.isupper() False
	string.center(<width>[, <fill>])	жүйені орталыққа туралайды.	>>> 'py'.center(10) ' py '

	tring.expandtabs(tabsize=8)	қойындыларды бос орынмен ауыстырады	>>> 'a\tb\tc'.expandtabs()
			'a b c'
			>>> 'aaa\tbbb\tc'.expandtabs()
			'aaa bbb c'

	string.ljust(<width>[, <fill>])	өрісте сызықтың сол жақ туралануы.	>> 'python'.ljust(10) 'python '
	string.lstrip([<chars>])	сол жақта бос орын таңбаларын қысқартады	foo bar baz '.lstrip() 'foo bar baz '\t\nfoo\t\nbar\t\nbaz'.lstrip() 'foo\t\nbar\t\nbaz'
	string.replace(<old>, <new>[, <count>])	жолдағы ішкі жолдың пайда болуын ауыстырады.	>>> 'I hate python! I hate python!
	string.rjust(<width>[, <fill>])	өрістегі сызықты оң жаққа туралаңыз.	>>> 'python'.rjust(10) ' python'
	string.rstrip([<chars>])	оң жақтағы бос орын таңбаларын қиып алады	' foo bar baz '.rstrip() ' foo bar baz' >>> 'foo\t\nbar\t\nbaz\t\n'.rstrip() 'foo\t\nbar\t\nbaz'

	string.strip([<chars>])	жолдың сол және оң жақ шеттеріндегі таңбаларды жояды.	>>> s = ' foo bar baz\t\t\t'
			>>> s = s.lstrip()
			>>> s = s.rstrip()
			>>> s

			'foo bar baz'
	string.zfill(<width>)	сол жақта нөлдері бар жастықтар.	>>> '42'.zfill(5) '00042'
	string.join(<iterable>)	тізімді жолға біріктіреді.	>>> ', '.join(['foo', 'bar', 'baz', 'qux']) 'foo, bar, baz, qux'
	string.partition(<sep>)	бөлгішке байланысты пішінді бөледі.	>> 'foo.bar'.partition('.') ('foo', '.', 'bar')
	string.rsplit(sep=None, maxsplit=-1)	жолды ішкі жолдар тізіміне бөледі.	> 'foo bar baz qux'.rsplit() ['foo', 'bar', 'baz', 'qux']
	string.splitlines([<keepends>])	мәтінді жолдар тізіміне бөледі.	s.splitlines()

Бөлгіш Мағынасы
\ n Жаңа сызық
\ r Арбаның қайтарылуы
\ r \ n Арбаны қайтару + жолды беру
\ v немесе \ x0b Жол кестелері
\ f немесе \ x0c Пішінді жіберу
\ x1c Файлды бөлгіш
\ x1d Топтық бөлгіш
\ x1e Жазба бөлгіш
\ x85 Келесі жол
\ u2028 Жаңа жол Unicode (Юникод)
\ u2029 Жаңа параграф Unicode (Юникод)

№6 дәріс. Python тіліндегі массивтер

Python -да массивтерді қолдану үшін стандартты массив модулін импорттау қажет. Бұл массив жолдар, бүтін сандар және т.б.

from array import * (массив импортынан)

arrayIdentifierName = array(typecode, [Initializers]) - массив хабарландыруы

python массив хабарландыруының мысалы:

my_array = массив ('i', [1,2,3,4])

5 бүтін саннан тұратын Массив мысалы

```
from array import *
my_array = array('i', [1,2,3,4,5])
for i in my_array:
    print(i)
# 1
# 2
# 3
# 4
# 5
```

Python массивтері нөлдік индекстелген. my_array = array('i', [1,2,3,4,5])

```
print(my_array[1])
# 2
print(my_array[2])
# 3
print(my_array[0])
# 1
```

Append () әдісін қолданып массивке кез келген мәнді қосыңыз

```
my_array = array('i', [1,2,3,4,5])
my_array.append(6)
```

массив ('i', [1, 2, 3, 4, 5, 6]) 6 мәні бар массив мәндеріне қосылғанын ескеріңіз.

insert () әдісі кез келген массивке мән енгізу үшін.

```
my_array = array('i', [1,2,3,4,5])
my_array.insert(0,0)
```

#array ('i', [0, 1, 2, 3, 4, 5]) 0 мәні 0 индексіне енгізілді.

Python массивін extend () әдісі арқылы бірнеше мәнмен кеңейтуге болады.

```
my_array = array('i', [1,2,3,4,5])
my_extnd_array = array('i', [7,8,9,10])
my_array.extend(my_extnd_array)
# array('i', [1, 2, 3, 4, 5, 7, 8, 9, 10])
```

Fromlist () әдісі арқылы тізімнен элементтерді массивке қосыңыз
my_array = array('i', [1,2,3,4,5])
my_array.remove(4)
# array('i', [1, 2, 3, 5])
с массивінен # массив ('i', [1, 2, 3, 4, 5, 11, 12, 13]) 11,12 және 13 қосылды.

Remove () әдісі арқылы кез келген элементті алып тастаңыз
my_array = массив ('i', [1,2,3,4,5])
my_array.remove (4)
массив ('i', [1, 2, 3, 5])

Pop () әдісі
pop массивтен соңғы элементті жояды.
my_array = array('i', [1,2,3,4,5])
my_array.pop()
# array('i', [1, 2, 3, 4])

Index () әдісі
index () сәйкестік мәнінің бірінші индексін қайтарады. my_array = array('i', [1,2,3,4,5])
print(my_array.index(5))
# 5
my_array = array('i', [1,2,3,3,5])
print(my_array.index(3))
# 3

(Кері) reverse () әдісі
Reverse () әдісі массивті өзгертеді.
my_array = array('i', [1,2,3,4,5])
my_array.reverse()
# array('i', [5, 4, 3, 2, 1])

Buffer_info () әдісі
Жадтағы массив буферінің бастапқы адресін және массив элементтерінің санын қамтамасыз етеді.
my_array = array('i', [1,2,3,4,5])
my_array.buffer_info()
(33881712, 5)
Count () әдісі
count () массивтегі элементтің пайда болу санын қайтарады.
my_array = array('i', [1,2,3,3,5])
my_array.count(3)

2
Tostring () әдісі
tostring () массивті жолға түрлендіреді.

```
my_char_array = array('c', ['g','e','e','k'])
# array('c', 'geek')
print(my_char_array.tostring())
# geek
```

Tolist () әдісі массивті тізімге түрлендіреді.

```
my_array = array('i', [1,2,3,4,5])
c = my_array.tolist()
# [1, 2, 3, 4, 5]
```

Fromstring () әдісі
Fromstring () көмегімен таңбалар массивіне жолды қосуға болады

```
my_char_array = array('c', ['g','e','e','k'])
my_char_array.fromstring("stuff")
print(my_char_array)
#array('c', 'geekstuff')
```

№7 дәріс. Python тіліндегі сөздіктер

Сөздіктер-бұл объектілерді сақтауға арналған басқа кіріктірілген деректер түрі. Олар мәнді объектіні кілтпен байланыстыру үшін қолданады. Бұл байланыстыру картография деп аталады. Дисплейде кілт-мән жұбы пайда болады. Сөздікке кілт-мән жұптары қосылады. Содан кейін сіз оны кілт форматында іздеп, сәйкес мәнді ала аласыз. Алайда, сіз, керісінше, кілтті табу үшін мәнді пайдалана алмайсыз.

Сөздіктер, тізімдер сияқты, өзгермейтін. Яғни, олар жаңа кілт-мән жұптарын қамтуы мүмкін. Олардың пайдалылығы кілттер мен құндылықтар арасындағы байланыста. Мысалы, мүмкіндікте сіз достарыңыздың телефон нөмірлері туралы ақпаратты сақтай аласыз:

```
phones = {
 «Айнұр»: «+7123456789»,
 «Ғабит»: «+37520123456»
}
print(phones)
```

Сөздік құрған кезде бұйра жақшаларды қолдану керек, кілтті нүктеден қос нүктемен ажырату керек және кілт-мән жұптарын үтірмен ажырату керек. Бумалардан айырмашылығы, егер сізде тек бір ғана кілт-мән жұбы болса, одан кейін үтір қажет емес.

Сөздікті жасағаннан кейін, сіз онда dictionary_name [key] = мән синтаксисін қолдана отырып, кілт-мән жұптарын қолдана аласыз, сонымен қатар Dictionary_name [key] синтаксисінің көмегімен кілт арқылы мәнді іздеуге болады.

```
phones = {
 «Айнұр»: «+7123456789»,
 «Ғабит»: «+37520123456»
}
phones['Галым'] = 1234567890
print(phones['Малик'])
```

Мүмкіндік мәні кез келген объект болуы мүмкін. Мысалда алғашқы екі мән жолдар болды, ал соңғы мән 1234567890 бүтін сан болды.

Сөздік мәнінен айырмашылығы, сөздік кілті өзгермейтін болуы керек. Сөздік кілті жол немесе кортеж болуы мүмкін, бірақ тізім немесе сөздік емес. Кілттің тестте тұрғанын анықтау үшін кілттің жоқтығын тексеру үшін сөзді енгізіңіз, кірмеңіз.

```
phones = {
    " Малик ": "+7123456789",
    "Надир": "+37520123456"
}
print("Надир " in phones)  # True
```

Del кілт сөзінің көмегімен сөздіктен кілт-мән жұбын жоюға болады.

```
phones = {
    "Малик": "+7123456789",
    " Надир ": "+37520123456"
}
del phones["Надир "]
```

Басқа контейнерлердегі контейнерлерді блоктауға болады. Мысалы, сіз тізімдерді тізім ішінде сақтай аласыз.

```
music = {
    "rap": ["Бакыт", "Кут", "Береке"],
    "rock": ["Нур", "Казына", "Адам"],
```

Бұл мысалда музыканың үш кілті бар.

```
music = {
    "rap": ["Бакыт", "Кут", "Береке"],
    "rock": ["Нур", "Казына", "Адам"],
}
print(music['rap'])  # ["Бакыт", "Кут", "Береке"]
print(music['rock'][-1])  # ["Нур", "Казына", "Адам"]
```

№8 дәріс. Ерекшеліктерді өңдеу.

try-except операторы

Егер тестілеу кодын жазу процесінде бағдарламада қателер пайда болса, оны бағдарламашы қателіктер болмайтындай етіп кодтайды. Дегенмен, пайдаланушының әрекеттері көбінесе бағдарламада ерекше жағдайға әкеледі. Мысалы, бағдарлама нөмірді енгізуді күтеді, бірақ адам хат енгізді. Оны ValueError ерекшелігіне шығаруға түрлендіру әрекеті және бағдарлама бұзылады.

Бұл бағдарламалау тілінде, соның ішінде Python -да, арнайы оператор бар, ол сізге ерекшеліктерді анықтауға және оларды өңдеуге мүмкіндік береді, осылайша бағдарлама жұмысын жалғастырады немесе сәтті шығады.

Python -да мұндай ұстау операторын ұстау үшін try-except көріңіз. Бұл жағдай «Әрекет ету» «тырысу», «қоспағанда» дегенді білдіреді. Сөздер оның жұмысын былай суреттей алады: «Мұны істе, бұлай істе, егер ерекше жағдай орын алса, мынаны істе». Else шартты операторға ұқсайдйы

Мысал қарастырайық:

```
n = input («Бүтін сан енгізіңіз:»)
try:
    n = int(n)
    print("Удачно")
except:
    print («Бірдеңе дұрыс болмады»)
```

N мәні бүтін санға түрлендірілгенде кодтың үшінші жолында ерекше жағдай болуы мүмкін. Егер бұл мүмкін болмаса, тоқтатуды көріңіз. Бұл жағдайда print («Сәтті») мәлімдемесі орындалмайды. Бұл жағдайда бағдарламаны орындау ағыны оның негізгі бөлігін қоспағанда except.

Егер денеде try болмаса, онда бұтақтың денесі except орындалмайды.

Пайдаланушы бүтін санды енгізгенде бағдарлама шығысының мысалы:

```
Бүтін санды енгізіңіз: 100
Іске сәт
```

Ал мұнда - ол күткен нәрсеге кірмеген кезде:

```
Бүтін санды енгізіңіз: АА
Бірдеңе дұрыс болмады
```

Бір мәселе бар. Жоғарыдағы код кез келген ерекшелікке жауап береді. Дегенмен, арбада әртүрлі ерекшеліктерді қолдануға тырысыңыз және олардың әрқайсысының өз өңдеушісі болуы керек. Демек, except сөзінен кейін ерекшелік түрін көрсету дұрысырақ.

```
try:
 n = input ('Бүтін санды енгізіңіз:')
 n = int (n)
 print («Жарайды. Сіз нөмірді енгіздіңіз», n)
except ValueError:
 print («Сіз бүтін емес санды енгіздіңіз»)
```

Енді егер дене жұмыс жасаса, біз қатенің неден болғанын білеміз. Бірақ егер сіз арбада басқа ерекшелікті байқасаңыз, ол өңделмейді. Ол үшін бөлек тармақты жазу қажет, қоспағанда. Бағдарламаны қарастырыңыз:

```
try:
 a = float (енгізу («Дивидендті енгізіңіз:»))
 b = float (енгізу («Бөлгішті енгізіңіз:»))
 c = a / б
 print ("Жеке:% .2f"% c)
ValueError қоспағанда:
 print («Жолды енгізу мүмкін емес»)
except (ZeroDivisionError):
print («Нөлге бөлуге болмайды»)
```

Ол орындалған кезде сәйкестікті кодтың үш жолына енгізуге болады: мұнда мәндер дұрыс сандарға түрлендіріледі және бөлу орын алатын жерде. Бірінші жағдайда ValueError, екіншісінде ZeroDivisionError орын алуы мүмкін. Ерекшеліктің әр түрі тармақтан басқа жеке өңделеді.

Бірнеше ерекшеліктерді бір филиалға топтастыруға және бірге өңдеуге болады:

```
тырысу:
 a = float (енгізу («Дивидендті енгізіңіз:»))
 b = float (енгізу («Бөлгішті енгізіңіз:»))
 c = a / б
 басып шығару («Жеке:% .2f»% c)
қоспағанда (ValueError, ZeroDivisionError):
 басып шығару («Жолды енгізуге немесе нөлге бөлуге болмайды»)
```

Ерекшеліктерді өңдеу операторы, қоспағанда, ақырында және басқа тармақтарға ие болуы мүмкін (міндетті түрде екеуі де емес). Ақырында, блок әрқашан орындалғанына қарамастан орындалады, тек көтерілген ерекшеліктерге жауап ретінде. Көріңіз жоқ ерекше болған жағдайда басқа органы болып табылады, бұл іске қосылады , блоктарды қоюды жоқ өткелдер болды.

```
try:
    n = input('Введите целое число: ')
    n = int(n)
except ValueError:
    print("Вы что-то попутали с вводом")
else: # когда в блоке try не возникло исключения
    print("Все нормально. Вы ввели число", n)
finally: # выполняется в любом случае
    print("Конец программы")
```

Бұл код түсініктемелерді қолданады. Python -да олардың алдында хэш белгісі болады

Бұл сабақ ерекшеліктердің барлық ерекшеліктерін қамтымайды. Кодтың, модульдердің және бағдарламалардың бағдарламалық деңгейлерін қамтамасыз ету.

Сондай -ақ, ерекше жағдай қоспағанда, else немесе ақырында блокта болуы мүмкін, содан кейін оларға жеке өңдеуші қажет. Алдыңғы бағдарламаны сәл өзгертеміз және денеде ерекше жағдайды шығарамыз, тек қана:

```
try:
    n = input('Введите целое число: ')
    n = int(n)
except ValueError:
    print("Вы что-то попутали с вводом")
    3 / 0
except ZeroDivisionError:
    print("Деление на ноль")
else:
    print("Все нормально. Вы ввели число", n)
finally:
    print("Конец программы")
```

№ 9 дәріс. Сұрыптау әдістері

Сұрыптау алгоритмдері, көпіршікті сұрыптау немесе көпіршікті сұрыптау, элементтерді бірте -бірте жылжытады, сондықтан ең кішісі ең басында болады, суда ауа көпіршігі қалқып тұрғандай. Ал ең үлкен элемент, керісінше, төмендейді - яғни массивтің соңына дейін.

Бұл әдіс әрбір екі іргелес элементтерді кезекпен салыстырады және егер тапсырыс дұрыс болмаса, оларды қайта реттейді. Келесі сандарды алайық: 57380. Алдымен, көршілес алғашқы екі санды салыстырыңыз: 5 7 - ден кіші, олар өз орындарында қалады. Содан кейін келесі екі сан - 7 3 -тен үлкен, яғни олар орын ауыстырады. Содан кейін ол 7 мен 8 -ді салыстырады, оларды да ауыстырады және т.б. Сандар қатары арқылы бірінші рет өткеннен кейін, соңғы сан - ең үлкені - 8. Содан кейін алгоритм барлық сандар арқылы қайтадан өтеді және сан орнына түскенше - ең кішісінен үлкеніне дейін бірнеше рет өтеді.

Кездейсоқ ретпен сандары бар тізім бар, оларды сұрыптау қажет. Алдымен, алғашқы екі санды салыстырып, егер қате болса, оларды ауыстырайық - яғни, екінші сан біріншіден үлкен.Нәтижедегі тізімді басып шығарыңыз.

```
my_list=[5,7,3,8,0]
if my_list[0]>my_list[1]:
    my_list[0],my_list[1]=my_list[1],my_list[0]
print(my_list)
[5, 7, 3, 8, 0]
```

Бұл код 5 -тен 7 -ге дейін салыстырды және оларды сол жерде қалдырды. Енді келесі екі санды салыстырайық: кодтағы индекстерді 0 мен 2 -ге өзгертіңіз. Ал біз оның 3 пен 7 -ді салыстырып, ауыстырғанын көреміз.

```
if my_list[1]>my_list[2]:
    my_list[1],my_list[2]=my_list[2],my_list[1]
print(my_list)
[5, 3, 7, 8, 0]
```

Біз олардың әрқайсысы өз орнын табуы үшін сандар тізімінде қанша рет болса да осы операцияны орындауымыз керек. Сондықтан, әр жұпқа бірдей операцияны белгілемеу үшін, біз 1 массивінің ұзындығына тең цикл құрамыз, себебі тізімдегі соңғы санның оң жағында көршісі жоқ, яғни бұл салыстыру операциясы қажет емес. Және біз кодтың әр қадамын көру үшін алынған тізімді цикл ішінде басып шығаруды сұраймыз. my_list=[5,7,3,8,0]

```
for i in range(len(my_list)-1):
    print (f"сравниваем {my_list[i]} с {my_list[i+1]}")
    if my_list[i]>my_list[i+1]:
        my_list[i],my_list[i+1]=my_list[i+1],my_list[i]
    print(my_list)
```

```
сравниваем 5 с 7
[5, 7, 3, 8, 0]
сравниваем 7 с 3
[5, 3, 7, 8, 0]
сравниваем 7 с 8
[5, 3, 7, 8, 0]
сравниваем 8 с 0
[5, 3, 7, 0, 8]
```

Бұл код біртіндеп 8 -ге дейін жылжыды, бір рет өткеннен кейін бір сан өзінің дұрыс орнын тапты. Бірақ сұрыптау мұнымен бітпейді. Бізге массивтің әрбір элементі өз орнына өту үшін қажет - бұл бізде қанша сандар болса, сонша қайталау жасау керек дегенді білдіреді. Сондықтан біз жаңа цикл жасаймыз және оны қайталау. Бірақ бұл жерде біз соңғы айналымда ештеңе өзгермейтінін көреміз, себебі 3 өз орнын тапқанда, ең кіші 0 элементі де орнына түсті, демек, соңғы айналдыруды жасамау үшін 1 -ді алып тастай аламыз. Бұл кодты тағы қалай оңтайландыруға болады? Әр айналымнан кейін тізім соңында ең үлкен сан пайда болады. Сондықтан циклдің қайталануына сәйкес келетін цикл ішіндегі қайталау санын алып тастауға болады - старт.

```
my_list=[5,7,3,8,0]
for run in range (len(my_list)-1):
    for i in range(len(my_list)-1-run):
        print (f"сравниваем {my_list[i]} с {my_list[i+1]}")
        if my_list[i]>my_list[i+1]:
            my_list[i],my_list[i+1]=my_list[i+1],my_list[i]
    print(my_list)
```

```
сравниваем 5 с 7
сравниваем 7 с 3
сравниваем 7 с 8
сравниваем 8 с 0
[5, 3, 7, 0, 8]
сравниваем 5 с 3
сравниваем 5 с 7
сравниваем 7 с 0
[3, 5, 0, 7, 8]
сравниваем 3 с 5
сравниваем 5 с 0
[3, 0, 5, 7, 8]
сравниваем 3 с 0
[0, 3, 5, 7, 8]
```

Біз алгоритмді кез-келген басқа массивтерді жіберуге болатын функциямен қарастырамыз және оны жаңа тізімде тексереміз.

```
def bubble_sort(my_list):
    for run in range (len(my_list)-1):
        for i in range(len(my_list)-1-run):
            #print (f"сравниваем {my_list[i]} с {my_list[i+1]}")
            if my_list[i]>my_list[i+1]:
                my_list[i],my_list[i+1]=my_list[i+1],my_list[i]
    return my_list
my_list=[0,6,8,3,2,57,76,35,0]
bubble_sort(my_list)
 [0, 0, 2, 3, 6, 8, 35, 57, 76]
```

Кірістірілген сұрыптау әдістері мен функциялары

Python-да кірістірілген сұрыптау функциялары мен әдістері бар, олар бір кодты бір жолмен орындауға мүмкіндік береді.

.Sort () әдісі тізімді сұрыптайды және оны сұрыптайды. Сұрыпталған функция (түпнұсқаны өзгертпестен жаңа сұрыпталған тізім жасайды.

```
my_list=[0,6,8,3,2,57,96,45]
my_list.sort()
my_list
 [0, 2, 3, 6, 8, 45, 57, 96]
```

Тағы бір айырмашылығы - сұрыпталған функция тек тізімдермен ғана емес, басқа объектілермен де жұмыс істейді. Мысалы, егер біз жолды сұрыптасақ осылай болады.

```
tring='Важные истории'
sorted(string)
 [' ', 'В', 'а', 'е', 'ж', 'и', 'и', 'и', 'н', 'о', 'р', 'с', 'т', 'ы']
```

Бұл функция тізімнің қай түріне берілгеніне қарамастан, әр уақытта қайтарады. Сөздіктер жағдайында ол сөздік кілттерінің сұрыпталған тізімін қайтарады.

```
fsb_dict={'na':'Бактияров А.В.','inc':15879555.13,'wi_income':261956.95}
sorted(fsb_dict)
 ['inc', 'na', 'wi_income']
```

Әдепкі бойынша сұрыптау өсу ретімен болады. Бұған кері параметр жауап береді. Егер оған True параметрін берсек, сұрыптау кему ретімен болады.

```
my_list=[1,6,8,3,2,57,76,35,0]
sorted(my_list,reverse=True)
 [76, 57, 35, 8, 6, 3, 2, 1, 0
```

Сонымен қатар, сұрыпталған және сұрыпталған параметрлерде салыстыру функциясын көрсететін параметр бар. Мысалы, егер біз мәндерді регистрге қарамай сұрыптағымыз келсе, біз мұны істей аламыз:

```
string = 'Маңызды оқиғалар'
sorted(string,key=str.lower)
 [' ', 'а', 'В', 'е', 'ж', 'и', 'и', 'и', 'н', 'о', 'р', 'с', 'т', 'ы']
```

Параметрлер тек кіріктірілген функцияларды ғана емес, сонымен қатар біз жазған функцияларды да қамтуы мүмкін. Олар әдетте күрделі нысанды сұрыптағымыз келгенде қажет болады. Мысалы, мұндай мәселе бар: егер сіз күрделі нысанды сұрыптауға тырыссаңыз, мысалы, тізім тізімінен), сұрыптау бірінші элемент бойынша жұмыс істейді.

```
fsb_list = [['Бақтияров А.В.', 15879555.13,261956.95], ['Сачков С.М.', 11882496.91,593918.65], ['Кулманов В.Г.', 6795228.14,861350.27]]
сұрыпталған (fsb_list)
 [['Бақтияров А.В.', 15879555.13, 261956.95],
 ['Кулманов В.Г.', 6795228.14, 861350.27],
 ['Сачков С.М.', 11882496.91, 593918.65]]
```

Біз реттелетін функцияны өзіміз жасай аламыз.

```
def fsb_income(i):
    return i[1]
sorted(fsb_list,key=fsb_income)
 [['Кулманов В.Г.', 6795228.14, 861350.27],
 ['Сачков С.М.', 11882496.91, 593918.65],
 ['Бақтияров А.В.', 15879555.13, 261956.95]]
```

№10 дәріс. Міндеттемелер

Бағдарламалаудағы функция - бұл аталатын аты бойынша оған сілтеме жасау арқылы шақыруға болатын бөлек код бөлігі. Шақыру кезінде функция денесінің командалары орындалады.

Функциялар шағын бағдарламалар салыстыруға болады өздері, яғни бұл , дербес, орындалған жоқ, бірақ тұрақты бағдарламасына енгізілген. Көбінесе оларды кіші программалар деп атайды. Функциялар мен бағдарламалар арасында басқа маңызды айырмашылықтар жоқ. Функциялар қажет болған жағдайда қоңырау шалушы бағдарламадан деректерді шақырады және қайтарады. Және олар өз жұмысының нәтижесін қайтарады.

Python бағдарламалау тілінде көптеген кіріктірілген функциялар бар. Бұл print (), input (), int (), float (), str (), type (). Олардың денесінің коды бізге көрінбейді, ол бір жерде «тілдің ішінде жасырылған». Бізге тек интерфейс беріледі - функция атауы.

Бағдарламашы әрқашан өзінің жеке функцияларын анықтай алады. Бірнеше сандарды енгізуді және оларды қосуды олардан үш рет сұрау керек делік. Осы мақсатта циклды қолдануға болады:

```
i = 0
while i < 3:
    a = int(input())
    b = int(input())
    print(a+b)
    i += 1
```

Алайда, егер сандарға әр сұраудан бұрын жазуды көрсету қажет болса, олар не үшін қажет және әр жолы бұл жазу әр түрлі болады. Біз циклды үзе алмаймыз, содан кейін сол циклге қайта ораламыз. Сіз оны таңдауыңыз керек, содан кейін сіз бірдей бөлімдері бар әр түрлі жерде берілген ұзын кодты аласыз:

```
print("Сколько еды для обезьян?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")

print("Сколько  конфеток   для ежей?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")

print("Сколько рыб и моллюсков для выдр?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

Пример исполнения программы:

```
Сколько еды для обезьян?
15
5
Всего 20 шт.
Сколько  конфеток   для ежей?
50
12
Всего 62 шт.
Сколько рыб и моллюсков для выдр?
16
8
Барлығы 24 дана.
```

Функцияны енгізу бағдарламаның әр жерінде кодты қайталау мәселесін шешуге мүмкіндік береді. Олардың арқасында сіз кодтың бір бөлігін бірден емес, қажет болғанда ғана орындай аласыз.

Функцияның анықтамасы. Def мәлімдемесі

Python бағдарламалау тілінде функциялар def операторының көмегімен анықталады. Кодты қарастырыңыз:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Бұл функцияның анықтамасының мысалы. Шарттар мен ілмектер сияқты басқа күрделі мәлімдемелер сияқты, функция тақырып пен денеден тұрады. Тақырып қос нүктемен және жаңа жолмен аяқталады. Дене шегініспен бекітілген.

Def кілт сөзі аудармашыға оның алдында функцияның анықтамасы бар екенін айтады. Def соңынан функцияның аты шығады. Бұл кез келген нәрсе болуы мүмкін, сонымен қатар кез келген идентификатор, мысалы, айнымалы. Бағдарламалауда барлығына мағыналы атаулар берген жөн. Бұл жағдайда бұл функция орыс тіліне аударғанда «тағамды сана» деп аталады.

Жақшалар функция атауынан кейін қойылады. Көрсетілген мысалда олар бос. Бұл функция шақырушы бағдарламадан ешбір мәліметті қабылдамайтынын білдіреді. Параметрлер деп аталатын, ол оларды қабылдай алады, содан кейін жақшада параметрлер деп аталатындар көрсетіледі.

Көп нүктеден кейін дене жазылады, функция шақырылған кезде командалар орындалады. Функция мен оның шақыруын ажырату қажет. Бағдарлама кодында олар жақын емес және бірге емес. Сіз функцияны анықтай аласыз, бірақ оны ешқашан шақырмаңыз. Сіз анықталмаған функцияны шақыра алмайсыз. Функцияны анықтай отырып, бірақ оны ешқашан шақырмай, сіз оның денесін ешқашан орындамайсыз.

Функционалды қоңырау

Бағдарламаның толық нұсқасын мына функциямен қарастырыңыз:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
 print("Сколько еды для обезьян?")
countFood()
 print("Сколько  конфеток   для ежей?")
countFood()
print("Сколько рыб и моллюсков для выдр?")
countFood()
```

Әрбір ақпараттық хабарлама экранда көрсетілгеннен кейін, функционалды қоңырау шалынады, ол жақшаның көмегімен оның аты туралы ғана айтылады. Функция функциясында жақшаға ештеңе берілмейді, қайтадан бос. Жоғарыда келтірілген кодта функция үш рет шақырылады.

Функция шақырылғанда, бағдарлама ағыны оның анықтамасына сәйкес орындалады. Функцияның негізгі бөлігі орындалғаннан кейін, орындалу ағыны функция шақырылған жерде негізгі кодқа оралады. Әрі қарай, қоңыраудан кейінгі өрнек орындалады.

Python -да функцияның анықтамасы оның шақыруларынан бұрын болуы керек. Бұл аудармашының кодты жолдан -жолға оқып шығуына және төменде не екенін әлі білмеуіне байланысты. Сондықтан, егер функцияға шақыру оның анықтамасында қате тудырса, ол жіберіледі (NameError ерекшелігі шығарылады):

```
print("Сколько еды для обезьян?")
countFood()

print("Сколько  конфеток   для ежей?")
countFood()
 print("Сколько рыб и моллюсков для выдр?")
countFood()
 def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Результат:

```
Сколько еды для обезьян?
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    countFood()
NameError: name 'countFood' is not defined
```

Көптеген құрастырылған тілдер үшін бұл қажет емес. Онда сіз функцияны бағдарламаның кез келген жерінде шақыруға және шақыруға болады. Алайда, кодтың оқылуы үшін бағдарламашылар бұл жағдайда да ережелерді енгізеді.

Функциялар тұрақтылық береді

Функцияларды қолдану бағдарламаның әр жерінен бір кодты бірнеше рет шақыру мүмкіндігінде ғана емес. Олардың арқасында бағдарлама шынайы тұрақтылыққа ие болуы маңызды. Функциялар, қалай болса да, оны бөлек бөліктерге бөледі, олардың әрқайсысы өзінің нақты тапсырмасын орындайды.

Әр түрлі пішіндердің ауданын есептейтін бағдарлама жазу керек делік. Пайдаланушы қай фигураның аймағын есептегісі келетінін көрсетеді. Осыдан кейін ол бастапқы деректерді енгізеді. Мысалы, тіктөртбұрыш жағдайында ұзындығы мен ені. Орындалу ағынын бірнеше тармақтарға бөлу үшін if-elif-else операторын қолданыңыз:

```
figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")
 if figure == '1':
  a = float(input("Ширина: "))
  b = float(input("Высота: "))
  print("Площадь: %.2f" % (a*b))
elif figure == '2':
  a = float(input("Основание: "))
  h = float(input("Высота: "))
  print("Площадь: %.2f" % (0.5 * a * h))
elif figure == '3':
  r = float(input("Радиус: "))
  print("Площадь: %.2f" % (3.14 * r**2))
else:
  print("Ошибка ввода")
```

Мұнда ешқандай функциялар жоқ және бәрі жақсы. Бірақ функциялары бар нұсқаны жазайық:

```
def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    print("Площадь: %.2f" % (a*b))
 def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print("Площадь: %.2f" % (0.5 * a * h))
 def circle():
    r = float(input("Радиус: "))
    print("Площадь: %.2f" % (3.14 * r**2))
 figure = input("1-прямоугольник,
2-треугольник, 3-круг: ")
if figure == '1':
  rectangle()
elif figure == '2':
  triangle()
```

```
elif figure == '3':
  circle()
else:
  print("Ошибка ввода")
```

Үш функцияның әрқайсысы бір рет шақырылады. Дегенмен, бағдарламаның жалпы логикасынан аумақтарды табуға арналған нұсқаулар алынып, оқшауланған. Бағдарлама енді бөлек «Лего кірпіштерінен» тұрады. Филиалда біз оларды қалағандай біріктіре аламыз. Ол басқару механизмінің рөлін атқарады.

Егер біз үшбұрыштың ауданын биіктік бойынша емес, Герон формуласы бойынша есептегіміз келсе, онда бағдарлама бойынша код іздеуге тура келеді. Біз функциялар анықталған жерге барып, олардың біреуінің денесін өзгертеміз.

Біз бұл функцияларды басқа бағдарламада қолдана аламыз, сонда біз оларды кодпен осы файлға сілтеме жасай отырып импорттай аламыз (бұл Python -да жасалғандай, кейінірек талқыланатын болады).

Дәріс №11. Модульдер

Бағдарламалау тіліне кіріктірілген функциялар бірден қол жетімді. Біз модульдерді, пакеттерді және кітапханаларды қолданамыз. Әрбір модульде белгілі бір саладағы мәселелерді шешуге арналған функциялар мен сыныптар жинағы бар. Python модулінде математикалық функциялар математикалық функцияларды қамтамасыз етеді, модуль жалған кездейсоқ сандарды кездейсоқ құруға мүмкіндік береді, күндізгі уақыт модулінде күндер мен уақыттармен жұмыс жасау үшін сыныптар бар, sys модулі жүйелік айнымалыларға қатынауды қамтамасыз етеді және т.

Кейбір модульдер стандартты кітапхана деп аталады. Бұл стандартты, себебі ол орнату пакетімен бірге келеді. Дегенмен, үшінші тарап кітапханалары бар. Олар бөлек жүктеледі және орнатылады.

Модульдің функционалдылығына қол жеткізу үшін оны бағдарламаға импорттау қажет. Импортталғаннан кейін, ол қосымша сыныптар мен функциялардың бар екенін «біледі» және оларды пайдалануға мүмкіндік береді.

Python -да импорттау пәрмені арқылы жүзеге асады. Дегенмен, импорттаудың бірнеше жолы бар. Математика мысалын қолдана отырып, модульмен жұмыс жасауды қарастырайық. Сонымен,

```
>>> import math
>>> math
<module 'math' (built-in)>
```

Бағдарламада модуль класына жататын математикалық объект бар.

Бұл модульге кіретін функциялар тізімін көру үшін біз Python кіріктірілген dir () функциясын қолданамыз, оған модуль атауын аргумент ретінде жібереміз:

```
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite',
'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Қос асты сызығы бар есімдерді елемейік. Қалғанның бәрі математикалық модульге қосылған функциялар мен тұрақтылар. Модульден функцияға қоңырау шалу үшін модульдің атын алдына жазу керек, нүкте қою керек, функцияның атын көрсету керек, содан кейін аргументтерді жақшаға беру қажет. Мысалы, математикадан по шақыру үшін сіз мынаны жазасыз:

```
>>> math.pow(2, 2) 4.0
```

Назар аударыңыз, бұл тілдің өзіне кіріктірілгеннен басқа pow () функциясы. «Кәдімгі» pow () функциясы бүтін санды қайтарады, егер аргументтер бүтін сандар болса:

```
>>> pow(2, 2)
4
```

Тұрақтыға сілтеме жасау үшін жақша қажет емес:

```
>>> math.pi
3.141592653589793
```

Егер біз бұл немесе басқа функцияның не істейтінін білмесек, кіріктірілген Python help () функциясын қолдана отырып, бұл туралы анықтама алуға болады:

```
>>> help(math.gcd)
Help on built-in function gcd in module math:
gcd(...)
    gcd(x, y) -> int
    greatest common divisor of x and y
```

Онлайн анықтамадан шығу үшін q қосу керек. Бұл жағдайда функция x пен y сандарының ең үлкен ортақ бөлгіші болып табылатын бүтін санды қайтаратынын айтады. Модульдердің сипаттамасы мен олардың мазмұнын python.org сайтының ресми құжаттамасынан да табуға болады.

Екінші импорт әдісі - модульдің өзі импортталған кезде.

```
>>> from math import gcd, sqrt, hypot
```

Оны «математикалық модульден gcd, sqrt және hypot функцияларын импорттау» деп аударуға болады.

Бұл жағдайда оларды шақырған кезде функция атауының алдында модуль атауын көрсетудің қажеті жоқ:

```
>>> gcd(100, 150)
50
>>> sqrt(16)
4.0
>>> hypot(3, 4)
5.0
```

Модульден барлық функцияларды бірден импорттау үшін:

```
>>> from math import *
```

Арқылы импорттау кемшіліксіз болмайды. Бағдарлама импортталған функциялардың немесе тұрақтылардың бірінің атымен бірдей идентификаторға ие болуы мүмкін. Қателер болмайды, бірақ олардың біреуі «өшіріледі»:

```
>>> pi = 3.14
>>> from math import pi
>>> pi
3.141592653589793
```

Дәріс нөмірі 12. Python бағдарламалауға арналған 6 маңызды кітапхана

Python (python)-жоғары деңгейлі жалпы мақсаттағы бағдарламалау тілі, ол бағдарламалау қоғамдастығының жетекші және танымал бірі болды. Мүмкіндіктеріне қарай, ол жеңілдетілген қосымшаларды әзірлеумен күрделілік деңгейі бірдей күрделі математикалық есептеулерге дейін жіктеледі.

Жетекші программалау тілдерінің бірі ретінде сізде қолдануға болатын көптеген құрылымдар (қосымшалар құруға арналған платформалар) мен кітапханалар бар. Бағдарламалау тілінің кітапханасы - бұл бағдарламалау тілінің көмегімен белгілі бір операцияларды орындайтын модульдер мен функциялардың жиынтығы.

Сонымен, мұнда әрбір Python әзірлеушісі білуі керек 6 Python бағдарламалау кітапханасы:

1. Керас кітапханасы

Түрі - нейрондық желі кітапханасы.

Алғашқы шығарылым наурыз 2015 ж.

Keras - бұл Python -да жазылған ашық коды бар нейрондық желі кітапханасы. Ықшам, модульдік және кеңейтілетін етіп жасалған кезде, терең оқу желілерімен тез жұмыс жасауды мақсат етеді.

Нейрондық желінің экспресс -қозғалтқышынан басқа, Keras сонымен қатар модельдерді жинауға, деректер жиынтығын өңдеуге және графиктерді визуализациялауға арналған ең жақсы мүмкіндіктерді ұсынады. Артқы жағында (серверде) Keras Theano немесе TensorFlow пайдаланады.

Керас серверлік инфрақұрылымды қолдана отырып есептеу графигін құрады, содан кейін оны операцияларды орындау үшін қолданады, сондықтан ол машиналық оқыту кітапханаларына қарағанда баяу. Алайда, Керастағы барлық модельдер портативті.

Ерекшеліктер:

- Python -да жазылғандықтан, оны жөндеу және зерттеу оңай.
- активтендіру функциялары, деңгейлері, мақсаттары мен оңтайландырушылары сияқты қолданылатын нейрондық желінің құрылыс блоктарының кеңінен енгізілуін қамтиды.
- Керемет экспрессивтілік пен икемділік оны инновациялық зерттеулер үшін тамаша етеді.
- Inception, MNIST, ResNet, SqueezeNet және VGG сияқты бірнеше құрылған деректер жиынтығы мен дайындалған модельдерді ұсынады.
- Нейрондық желілердің барлық дерлік модельдеріне қолдау көрсетеді, соның ішінде конвульсиялық, ендірілген, толық қосылған, бассейндік және қайталанатын. Сонымен қатар, бұл модельдерді одан да күрделі модельдерді жасау үшін біріктіруге болады.

- CPU (орталық процессор) мен GPU (процессор ядросы) екеуінде де жұмыс істейді

Қолдану саласы:

- Netflix, Square, Uber және Yelp қолданған.
- терең зерттеу үшін. CERN мен NASA зерттеушілері қабылдады.
- Стартаптар арасында танымал, терең білімге негізделген жеке өнімдер.

2 .. NumPy кітапханасы

Тип - техническая вычислительная библиотека.

Начальная версия - 1995 (как Numeric).

2006 (Как NumPy).

NumPy-ді 2005 жылы Травис Олифант бәсекелес Numarray кітапханасын кітапханаға қосу арқылы жасаған.сандық және кең модификациялар. Тегін, ашық бастапқы кітапханада әлемнің бірнеше авторлары бар.

Python-дағы ең танымал машиналарды оқыту кітапханаларының бірі, TensorFlow және басқа да бірнеше кітапханалар тензорларға бірнеше операция жасау үшін ішіндегі NumPy Python кітапханасын пайдаланады.

Ерекшеліктері:

- Белсенді қоғамдық қолдау
- Толығымен тегін және ашық бастапқы код
- Матрицалық көбейту сияқты күрделі матрицалық амалдар
- Интерактивті және өте қарапайым
- Күрделі математикалық іске асыруды жеңілдетеді
- Оңай оқылатын тұжырымдамалармен кодтау

Қолдану саласы:

- Күрделі математикалық есептеулерді орындау үшін

-Суреттерді, дыбыстық толқындарды және басқа да ресми өңделмеген ағындарды n өлшемді нақты сандар массиві ретінде көрсету үшін

- Машиналық оқыту жобалары үшін

3. Pillow кітапханасы

Түрі-суретті өңдеу кітапханасы

Бастапқы нұсқасы - 1995 (Как Python Imaging Library или PIL)

2011 (Как Pillow)

Жастық-бұл Python кітапханасы, ол берілген бағдарламалау тілі сияқты ескі. Шын мәнінде, Pillow - бұл pil шанышқысы (Python Imaging Library). Еркін пайдаланылатын Python кітапханасы әртүрлі кескін файлдарын ашу, басқару және сақтау үшін қажет.

Pillow бірнеше Linux дистрибутивтерінде, атап айтқанда Debian және Ubuntu-да түпнұсқа PIL-ді ауыстыру ретінде қабылданды. Алайда, ол MacOS және Windows үшін де қол жетімді.

Ерекшеліктері:

- Суреттерге мәтін қосады
- Бұлыңғырлықты, жарықтылықты реттеуді, контурды және айқындықты қоса алғанда, кескінді жақсарту және сүзу
- Маска және мөлдірлік
- Пиксельдік манипуляциялар
- BMP, GIF, JPEG, PNG, PPM және TIFF сияқты көптеген кескін файл форматтарына қолдау көрсетеді. Қол жетімді файл пішімдерінің кітапханасын кеңейтумен жаңа файл декодерлерін құруға қолдау көрсетеді.

Қолдану саласы:

- Суреттерді өңдеу үшін

4. PYGLET кітапханасы

Түрі-ойын дамыту кітапханасы

Бастапқы нұсқасы-Сәуір 2015

Python үшін көп платформалы рамка мен мультимедиялық кітапхана, PYGLET-Python әзірлеуінің танымал атауы. Ойындардан басқа, визуалды бай қосымшалар жасауға арналған кітапхана.

PYGLET кесуге қолдау көрсетуден басқа, сурет пен бейнені жүктеуді, дыбысты және музыканы ойнатуды, OpenGL графикасын және UI оқиғаларын өңдеуге қолдау көрсетеді.

Ерекшеліктер:

- мониторлар арқылы бірнеше терезелер мен жұмыс үстелдерін қолдану
- Барлық форматтағы суреттерді, дыбысты және бейнені жүктеңіз
- Сыртқы тәуелділіктер мен орнату талаптары жоқ
- BSD ашық бастапқы кодымен берілген, сондықтан жеке және коммерциялық мақсатта еркін сөйлесе алады
- Python 2 мен Python 3 үшін қолдау көрсетеді

Қолдану саласы:

- Көрнекі бай қосымшаларды әзірлеу үшін
- Ойын дамыту үшін

5. Requests .Кітапханасы

Түрі - HTTP кітапханасы

Алғашқы шығарылым - ақпан 2011 ж

Сұраныстар - бұл HTTP сұрауларын қарапайым және ыңғайлы ету үшін Python HTTP кітапханасы. Кеннет Рейтц және басқалар жасаған, сұраныстар HTTP / 1.1 сұрауларын адамның қатысуынсыз жіберуге мүмкіндік береді.

Nike пен Spotify -тен Amazon мен Microsoft -қа дейін HTTP -мен жұмыс істеу үшін ондаған пайдаланушыларды ішкі сұраулар арқылы пайдалануға болады. Толық Python тілінде жазылған, сұраулар Apache2 лицензиясы бойынша ақысыз және ашық бастапқы кітапхана ретінде қол жетімді.

Ерекшеліктер:

- мазмұнды автоматты түрде декодтау

- Негізгі / дайджест аутентификациясы
- шолғыш стиліндегі SSL-ді тексеру
- Частичные запросы и время ожидания соединения
- Обеспечивает поддержку прокси-серверов netrc и HTTP (S)
- Сеансы с сохранением cookie
- Ответное тело Unicode

Қолдану саласы:

- Python көмегімен HTTP / 1.1 сұрауларын және тақырыптар, форма деректері және көп компонентті файлдар сияқты мазмұнды жіберуге мүмкіндік береді
- URL мекен-жайына сұрау жолдарын автоматты түрде қосу үшін
- POST деректерін автоматты түрде кодтау үшін

6. TensorFlow кітапханасы

Түрі-Машиналық оқыту кітапханасы.

Бастапқы нұсқасы-Қараша 2015

TensorFlow - бұл мәліметтер ағыны мен бағдарламалаудың дифференциалды мәселелерін шешуге арналған ақысыз, ашық көзі бар Python кітапханасы. Алайда, TensorFlow математикалық кітапханасы - Python машиналық оқыту кітапханаларының бірі.

Google Brain -тің ішінде әзірленген кітапхана коммерциялық және зерттеу мақсатында қолданылады.

Тензорлар-бұл мәліметтерді білдіретін N өлшемді матрицалар. TensorFlow кітапханасы көптеген тензорлы операцияларды қамтитын жаңа алгоритмдерді жазуға мүмкіндік береді.

Оларды тензорлық операциялар ретінде TensorFlow кітапханасының көмегімен оңай енгізуге болады.

Ерекшеліктер:

- Графиктің әрбір бөлігін визуализациялауға мүмкіндік береді
- Полностью бесплатный и открытый исходный код
- Таратылған есептеу үшін CPU (CPU) және GPU (CPU ядросы) оңай оқытылады
- Қоғамдастықтың үлкен қолдауы
- Жұмыс қабілеттілігінде икемділікті қамтамасыз етеді. Ең көп қажет ететін бөліктерді автономды түрде жасауға болады
- Бірнеше нейрондық желілерді оқытуды қолдайды.
- Сызықтық алгебраның операцияларын жеделдету үшін XLA сияқты әдістерді қолданады

Қолдану саласы:

- Машиналық оқыту жобалары үшін
- Нейрондық желілер жобалары үшін
- DeepDream сияқты тақырыптарды құруға арналған автоматтандырылған бағдарламалық бөлімде

- Google фотосуреттері және Google Voice Search сияқты Google өнімдерінде Машиналық оқыту

№13 лекция. Файлы

Большие объемы данных имеет смысл хранить не в списках или словарях, а в файлах. . В Python файлы рассматриваются как объекты файловых классов, то есть, например, текстовый файл - это тип данных с типами списка, словаря, целого числа и др.

Обычно файлы делят на текстовые и байтовые (бинарные). Первые рассматриваются как символьные данные, строки. Вторые - как поток байтов. Побайтово считываются, например, файлы изображений.

Работа с бинарными силами несколько сложнее. Нередко их обрабатывают с помощью специальных модулей Python (pickle, struct). В этом уроке рассмотрены базовые приемы чтения текстовых файлов и записи в них.

Функция open () - файлды ашу

Файлды ашу Python-дің кіріктірілген open () функциясы арқылы жүзеге асады. Әдетте оған бір немесе екі аргумент беріледі. Біріншісі - файл аты немесе адресі бар атау, егер файл сценарий орналасқан каталогта болмаса. Екінші аргумент - бұл файл ашылатын режим.

Оқу ('r') және жазу ('w') режимдері жиі қолданылады. Егер файл оқу режимінде ашық болса, онда оған жазу мүмкін емес. Сіз одан тек деректерді оқи аласыз. Егер файл жазу режимінде ашық болса, онда оған тек деректерді жазуға болады, оны оқу мүмкін емес.

Егер файл 'w' режимінде ашылса, онда оған дейінгі барлық деректер жойылады. Файл бос болады. Егер файлда деректерді қолдану қажет болмаса, онда жазу режимінің орнына қосымша жазу режимін ('a') қолданыңыз.

Егер файл жоқ болса, оны 'w' режимінде ашу жаңа файл жасайды. Жаңа файлды жасауға кепілдік беру қажет болған жағдайлар бар, бар файлды кездейсоқ қайта жазудан аулақ болыңыз. Бұл жағдайда 'w' режимінің орнына 'x' режимі қолданылады. Ол әрдайым жазу үшін жаңа файл жасайды. Егер бар файлдың аты көрсетілсе, онда ерекшелік тасталады. Қолда бар файлдағы деректер жоғалмайды.

Егер open () шақыру кезінде екінші аргумент көрсетілмесе, онда файл мәтіндік файл ретінде оқу режимінде ашылады. Файлды байт ретінде ашу үшін оқу / жазу әрпіне 'b' белгісі де қосылады. 'T' әрпі мәтіндік файлды білдіреді. Бұл әдепкі файл түрі, әдетте оны көрсетпейсіз.

Сіз тек файл түрін көрсете алмайсыз, яғни ашық («file_name», 'b'), егер файл оқу үшін ашылса да қате бар. Дұрыс - ашық («файл атауы», 'rb'). Тек мәтіндік файлдар ғана open («file_name») пәрменін аша аламыз, себебі әдепкі бойынша «r» мен «t» екеуі де айтылады.

Open () функциясы файл түріндегі нысанды қайтарады. Оны жоғалтпау үшін бірден байлау керек немесе бірден оқу керек.

Файлды оқу

Read () әдісінің көмегімен сіз бүкіл файлды немесе белгілі бір байт санын оқи аласыз. Келесі мазмұны бар data.txt файлы бар делік:

one - 1 - I
two - 2 - II
three - 3 - III
four - 4 - IV
five - 5 - V
Откроем его и почитаем:

```
>>> f1 = open('data.txt')
>>> f1.read(10)
'one - 1 - '
>>> f1.read()
'I\ntwo - 2 - II\nthree - 3 - III\n
four - 4 - IV\nfive - 5 - V\n'
>>> f1.read()
''
>>> type(f1.read())
<class 'str'>
```

Біріншіден, он таңбаға тең болатын алғашқы он байт оқылады. Бұл екілік файл емес, бірақ біз байттарды оқуды жалғастырамыз. Read () келесі шақыруы қалған мәтіннің барлығын оқиды. Осыдан кейін f1 файл түрінің объектісі бос болады.

Read () әдісі санды қайтаратынын және жолдың соңы '\ n' деп оқылатынын ескеріңіз.

Файлды жолдан -жолға оқу үшін readline () әдісі бар:

```
>>> f1 = open('data.txt')
>>> f1.readline()
'one - 1 - I\n'
>>> f1.readline()
'two - 2 - II\n'
>>> f1.readline()
'three - 3 — III\n'
```

Readlines () әдісі барлық жолдарды бірден оқиды және тізім жасайды:

```
>>> f1 = open('data.txt')
>>> f1.readlines()
['one - 1 - I\n', 'two - 2 - II\n',
'three - 3 - III\n',
'four - 4 - IV\n', 'five - 5 - V\n']
```

Файл типті объект-бұл итератор. Элементтер осындай объектілерден дәйекті түрде алынады. Сондықтан сіз олардан деректерді оқу әдістерін қолданбай бірден циклде оқи аласыз:

```
>>> for i in open('data.txt'):
...     print(i)
...
one - 1 - I
 two - 2 - II
 three - 3 - III
 four - 4 - IV
 five - 5 - V
```

Мұнда қосымша бос жолдар шығуда көрінеді. Print () функциясы '\ n' жаңа жолға түрлендіреді. Бұл сипаттамаға сіздің жаңа жолыңыз. Файл жолдарының тізімін '\ n' жоқ құрайық:

```
>>> nums = []
>>> for i in open('data.txt'):
...     nums.append(i[:-1])
...
>>> nums
['one - 1 - I', 'two - 2 - II',
'three - 3 - III',
'four - 4 - IV', 'five - 5 - V']
```

Файлдың келесі жолы і айнымалысына тағайындалады. Біз оның бір бөлігін басынан бастап соңғы таңбаға дейін аламыз, оны қоспағанда. '\ N' екі емес, бір таңба екенін ескеріңіз.

Файлға жазыңыз

Файлға жазу write () және Writelines () әдістерінің көмегімен жүзеге асады. Екіншісінде деректерді тасымалдауға болады:

```
>>> l = ['tree', 'four']
>>> f2 = open('newdata.txt', 'w')
>>> f2.write('one')
3
>>> f2.write(' two')
4
>>> f2.writelines(l)
```

Write () әдісі жазылған таңбалар санын қайтарады.

Файлды жабу

Файлмен жұмыс аяқталғаннан кейін жадта бос орын босату үшін оны жабуды ұмытпаған жөн. Бұл көмегімен жүзеге асады

файл әдісін close(). Жабық нысан файлдарының сипаты файлдың жабылғанын тексеруге мүмкіндік береді.>>> f1.close()

```
>>> f1.closed
True
>>> f2.closed
False
```

Егер файл цикл тақырыбында ашылса (I in open ('fname')), онда аудармашы цикл аяқталған кезде немесе біраз уақыттан кейін оны жабады.

Практикалық жұмыс

1. Оқу үлгісінің негізінде data.txt файлын жасаңыз. Бұл файлды оқуға ашатын, жазуды деректер жолынан қатарынан оқитын және жолдарды басқа файлға (dataRu.txt) жазатын, ағылшын цифрларын тізімге енгізілген орысшаға ауыстыратын бағдарламаны жазыңыз ([«бір», «екі» , «», «төрт», «бес»]) файлдарды ашпас бұрын анықталған. 2. Бос орынмен бөлінген бірнеше сандарды пайдаланып nums.txt файлын жасаңыз. Осы файлда сақталған сандардың жалпы сомасын есептейтін және көрсететін бағдарлама жазыңыз.

Қолданбаның мысалдары мен курстың Android және pdf нұсқаларындағы қосымша сабақтар

Ыңғайлы ақпарат түрінде сақталатын қарапайым файл форматтарының бірі-csv пішімі. Csv файлындағы әр жол жеке жазба немесе үтірмен бөлінген жеке бағандардан тұратын жол. Сондықтан формат үтірмен бөлінген мәндер деп аталады. Бірақ csv пішімі мәтіндік файл пішімі болса да, Python онымен жұмыс істеуді жеңілдету үшін арнайы кірістірілген csv модулін қамтиды.

Мысал арқылы модульдің жұмысын қарастырыңыз:

```
import csv
FILENAME = "users.csv"
 users = [
    ["Темирлан", 28],
    ["Alice", 23],
    ["Bob", 34]
]
with open(FILENAME, "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerows(users)
  with open(FILENAME, "a", newline="") as file:
    user = ["Самат", 31]
    writer = csv.writer(file)
    writer.writerow(user)
```

Файлда екі өлшемді тізім жазылады-іс жүзінде әр жол бір пайдаланушыны білдіреді. Әр пайдаланушының екі өрісі бар - аты мен жасы. Яғни, іс жүзінде үш жол мен екі бағаннан тұратын кесте.

Жазу үшін файлды ашқан кезде үшінші параметр ретінде newline = "" мәні көрсетіледі - бос жол амалдық жүйеге қарамастан кестеден жолдарды дұрыс оқуға мүмкіндік береді.

Жазу үшін csv функциясына оралған жазушы нысанын алу csv.writer(file). Бұл функцияда ашық файл жіберіледі. Ал жазбаның өзі writer әдісін қолдану арқылы writer.writerows(users) бұл әдіс жолдар жиынтығын қабылдайды. Біздің жағдайда бұл екі өлшемді тізім.

Егер бір өлшемді тізім болып табылатын бір жазбаны қосу қажет болса, мысалы ["Самат", 31], онда бұл жағдайда writer әдісін шақыруға болады. writer.writerow(user)

Нәтижесінде users.csv:

Темирлан, 28

Алиса, 23 года

Боб, 34 года

Самат, 31

Файлдан оқу үшін, керісінше, reader нысанын жасау керек:

```
import csv
FILENAME = "users.csv"
with open(FILENAME, "r", newline="") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row[0], " - ", row[1])
```

Нысанды алған кезде оқырман циклде оның барлық жолдарын сұрыптай алады:

Темирлан - 28

Алиса - 23

Боб - 34

Самат - 31

Сөздіктермен жұмыс

Жоғарыдағы мысалда әр жазба немесе жол бөлек тізім болды, мысалы ["Самат", 31]. Сонымен қатар, csv модулінде сөздіктермен жұмыс істеу үшін арнайы қосымша мүмкіндіктер бар. Атап айтқанда, Атап айтқанда, csv функциясы.DictWriter () файлға жазуға мүмкіндік беретін writer нысанын қайтарады.. Ал csv.DictReader() функциясы нысанын Файлдан оқу үшін қайтарады.

Мысалы

```
import csv
FILENAME = "users.csv"
users = [
    {"name": "Темирлан", "age": 28},
    {"name": "Alice", "age": 23},
    {"name": "Bob", "age": 34}
]
with open(FILENAME, "w", newline="") as file:
    columns = ["name", "age"]
    writer = csv.DictWriter(file, fieldnames=columns)
    writer.writeheader()
    # запись нескольких строк
    writer.writerows(users)
    user = {"name" : "Самат", "age": 41}
```

```
    # запись одной строки
    writer.writerow(user)
  with open(FILENAME, "r", newline="") as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row["name"], "-", row["age"])
```

Жазу жолдары жазушы () және жазушы () әдістерінің көмегімен де жасалады. Бірақ қазір әр жол бөлек сөздік болып табылады, сонымен қатар жазу мен баған тақырыбы writeheader () әдісі арқылы құрылады, ал бағандар жиыны csv.DictWriter әдісінде екінші ретінде беріледі.

Оқу кезінде баған атауларын қолдана отырып, біз жол ішіндегі мәндерді қолдана аламыз: row [«name»].

Дәріс 14. Класстар

Python объектіге бағытталған бағдарламалау парадигмасын қолдайды.

Класс-бұл шаблон немесе объектінің ресми сипаттамасы, ал объект осы сыныптың данасын, оның нақты бейнесін білдіреді. Келесі аналогияны жасауға болады: бізде адам туралы түсінік бар - екі қолдың, аяқтың, бастың, ас қорыту жүйесінің, мидың және т.б. кейбір шаблондар бар - бұл үлгіні сынып деп атауға болады. Шын мәнінде бар адам (іс жүзінде осы сыныптың данасы).

Класс коды нүктесінен функциялар жиынтығы мен белгілі бір тапсырманы орындайтын Сан біріктіріледі. Класс функциялары әлі де әдістерді қолданады. Олар сыныптың мінез-құлқын анықтайды. Класс айнымалылары атрибуттар деп аталады-олар сынып күйін сақтайды

Класс class кілт сөзімен анықталады:

класс аты_класс:

класса_әдісі

Класс объектісін құру үшін келесі синтаксис қолданылады:

object_name = class_name ([параметрлер])

Мысалы, ер адамды білдіретін ең қарапайым Man класын анықтайық:

```
class Person:
    name = "Темирлан"

    def display_info(self):
        print("Привет, меня зовут", self.name)

person1 = Person()
person1.display_info()        # Привет, меня зовут Темирлан

person2 = Person()
person2.name = "Самат"
person2.display_info()        # Привет, меня зовут Самат
```

Person класы адамның атын сақтайтын атрибуттың атын және адам туралы ақпаратты көрсететін display_info әдісін анықтайды.

Кез келген кластың әдістері ретінде әр түрлі программалау тілдерінде қолданылатын бағдарламалау шарттарына сәйкес қабылдануы тиіс тілдің түрін қолдану керек. Сынып ішіндегі осы сілтеме арқылы біз сол сыныптың әдістеріне немесе атрибуттарына сілтеме жасай аламыз. Атап айтқанда, self.name өрнегі пайдаланушының атын алу үшін пайдаланылуы мүмкін.

Person класын анықтағаннан кейін біз оның объектілерінің жұбын құрамыз - person1 және person2. Нысан атауын қолдана отырып, біз оның әдістері мен атрибуттарына жүгіне аламыз. Бұл жағдайда әрбір объект үшін

консольге шығатын display_info () әдісін шақырамыз, ал екінші объект үшін атрибуттың атын да өзгертеміз. Бұл жағдайда, әдісті шақырған кезде display_info өзіне берілмеуі керек.

Құрастырушылар

Класс объектісін құру үшін конструктор қолданылады. Сонымен, жоғарыда, біз Адам сыныбының объектілерін жасаған кезде, барлық сыныптарда жанама түрде болатын әдепкі конструкторды қолдандық:

```
person1 = Person()
person2 = Person()
```

Дегенмен, біз __init () деп аталатын арнайы әдісті қолданып конструкторды анықтаймыз. Мысалы, конструкторды қосу арқылы Face класын өзгертейік:

```
    # конструктор
    def __init__(self, name):
        self.name = name  # устанавливаем имя
    def display_info(self):
        print("Привет, меня зовут", self.name)

person1 = Person("Темирлан")
person1.display_info()        # Привет, меня зовут Темирлан
person2 = Person("Самат")
person2.display_info()        # Привет, меня зовут Самат
```

Бірінші орындаушы ретінде конструктор ағымдағы объектіге - өзіне сілтеме қабылдайды. Конструкторларда класс атрибуттарын орнату сирек емес. Сонымен, бұл жағдайда реттелетін атау екінші конструктор ретінде орнатылады. Сонымен қатар, атрибут үшін Human сыныбының алдыңғы нұсқасындағыдай, сыныпта айнымалы атауды анықтау қажет емес. Self.name = name мәнін орнату қазірдің өзінде ат төлсипатын жасырын түрде жасайды.

```
person1 = Person("Темирлан")
person2 = Person("Самат")
```

Нәтижесінде келесі консоль шығысын аламыз:

```
Сәлем, менің атым Темірлан
Сәлем менің атым Самат
```

Деструктор

Нысанмен жұмыс аяқталғаннан кейін біз оны del жадынан жою үшін del операторын қолдана аламыз:

```
person1 = Person("Темирлан")
del person1     # удаление из памяти
```

person1.display_info () # Бұл әдіс жұмыс істемейді, өйткені person1 жадтан жойылған

Айта кету керек, бұл іс жүзінде қажет емес, өйткені сценарий аяқталғаннан кейін барлық нысандар автоматты түрде жадтан жойылады.

Сонымен қатар, біз del операторына қоңырау шалу арқылы немесе нысанды автоматты түрде жою арқылы шақырылатын __del__ функциясын орындау арқылы деструкторды сыныпта анықтай аламыз.

Мысалы:

класс Адам:

құрастырушы

def __init __ (я, имя):

self.name = name # устанавливаем имя

def __del __ (сам):

print (self.name, "удален из памяти")

def display_info (self):

print ("Привет, меня зовут", self.name)

person1 = Человек ("Темирлан")

person1.display_info () # Привет, меня зовут Темирлан

del person1 # удаление из памяти

адам2 = Адам («Самат»)

person2.display_info () # Сәлеметсіз бе, менің атым Самат

Консоль шығысы:

Сәлем, менің атым Темірлан

Темірлан жадынан өшірілді

Сәлем менің атым Самат

Самат жадыдан алынып тасталды

Модульдерде Python сыныптарын анықтау және қосылу

Class.py файлында біз екі сыныпты анықтаймыз:

сынып адамы: # конструктор

def __init __ (мен, есімім):

self.name = name # атауды орнатыңыз

def display_info (өзінше):

басып шығару («Сәлем, менің атым», self.name)

Авто класс:

def __init __ (мен, есімім):

self.name = аты

Қозғалыс (өздігінен, жылдамдықпен):

басып шығару (өзіндік аты, «жылдамдықпен жүру», жылдамдық, «км / сағ»)

Адам сыныбынан басқа, ол автокөлікті бейнелейтін, жүру әдісі мен атрибуты бар Auto класын да анықтайды. Бұл сыныптарды байланыстырып, оларды main.py сценарийінде қолданайық:

from classes import Person, Auto

```
 Темирлан = Person("Темирлан")
tom.display_info()
 bmw = Auto("BMW")
bmw.move(65)
```

Подключение классов происходит точно также, как и функции из модуля. Мы можем подключить весь модуль выражением:

```
import classes
```

15 лекция. Наследование классов

Мұрагерлік-объектіге бағытталған бағдарламалаудың маңызды құрамдас бөлігі. Өз сыныптарының атрибуттары. Алайда, әдетте, ООР-тегі мұрагерлік дегеніміз-сыныптар мен ішкі сыныптардың болуы. Оларды супер немесе супер сыныптар мен сыныптар, сондай-ақ ата-аналар мен балалар сыныптары деп атайды.

Мұндағы мұрагерліктің мәні кластардан алынған объектілердің мұрасына ұқсас. Бала сыныптары ата-ана атрибуттарын мұра етеді.

Мысал ретінде біз кестелер класының және оның екі қосалқы сыныптарының - ас үй мен үстелдердің мысалын қарастырамыз. Барлық үстелдер, түріне қарамастан, ұзындығы, ені және биіктігі бар. Үстелдердің беткі ауданы маңызды, ал ас үй үшін - орын саны маңызды болсын. Жалпы сыныпқа, жеке - ішкі сыныптарға шығарылады.

Ата-ана мен бала класы арасындағы байланыс бала арқылы орнатылады: ата-ана сыныптары оның атауынан кейін жақшада тізімделеді.

Класс кестесі:

```
 def __init __ (self, l, w, h):
 self.length = l
 self.width = w
 self.height = h
 класс KitchenTable (Стол):
 def setPlaces (self, p):
 self.places = p
 class DeskTable (Таблица):
 def квадрат (self):
 вернуть self.width * self.length
```

Бұл класта KitchenTable мен DeskTable өздерінің конструкторлары жоқ. Бұл кестелерді құру кезінде __init __ () -ге берілген аргументтер қажет, әйтпесе қате жіберіледі:

```
>>> from test import *
>>> t1 = KitchenTable()
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
TypeError: __init__() missing 3 required
positional arguments: 'l', 'w', and 'h'
>>> t1 = KitchenTable(2, 2, 0.7)
>>> t2 = DeskTable(1.5, 0.8, 0.75)
>>> t3 = KitchenTable(1, 1.2, 0.8)
```

Әрине жасауға болады үстелдер мен ата-аналар сынып Table. Алайда, ол белгілі бір туыстық қатынастарға сәйкес setPlaces () және square () әдістеріне қол жеткізе алмайды. Сондай-ақ, KitchenTable класының нысаны жұмыс үстелінің мейірбикелік класының жеке атрибуттарына қол жеткізе алмайды.

```
>>> t4 = Table(1, 1, 0.5)
>>> t2.square()
1.2000000000000002
>>> t4.square()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Table' object
has no attribute 'square'
>>> t3.square()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'KitchenTable'
object has no attribute 'square'
```

Бұл тұрғыда "ата-ана мен бала класы" терминологиясы мүлдем дұрыс емес. ООР-тегі мұрагерлік-бұл тірі табиғаттағы сияқты жүйелеу мен жіктеудің аналогы. Барлық сүтқоректілердің төрт камералы жүрегі бар, бірақ тек мүйізтұмсық - мүйіз.

Қосымша класс әдісін толығымен қайта анықтау

Егер ішкі сыныпта оның жоғарғы класының коды бізге сәйкес келмесе ше? Айталық, біз DeskTable-қа қатысты басқа сыныптарды енгіземіз. Бұл компьютерлік үстелдер болсын, олардың жұмыс бетін есептеу кезінде берілген жұмыс бетін алып тастау керек. Бұл жаңа ішкі сыныпқа өзінің square () әдісін енгізу мағынасы бар:

```
class ComputerTable(DeskTable):
    def square(self, e):
        return self.width * self.length - e
```

ComputerTable типті нысанды құру кезінде параметрлерді көрсету қажет, өйткені аудармашы конструкторды іздеуде мұрагерлік ағашына алдымен ата-анасына, содан кейін ата-анасына барып, сол жерден __init _ _ () әдісін табады.

Алайда, square () әдісі шақырылған кезде, ол ComputerTable-дан табылғандықтан, Жұмыс үстеліндегі square () әдісі көрінбейтін болып қалады, яғни ComputerTable класының объектілері үшін ол қайта анықталған.

```
>>> from test import ComputerTable
>>> ct = ComputerTable(2, 1, 1)
>>> ct.square(0.3) 1.7
```

Қосымша, бұл кеңейту, әдіс

Егер сіз ауданды есептеуді қарасаңыз, онда Ішкі сынып кодының бір бөлігі ішкі сыныпта қайталанады. Егер сіз ата-ана әдісін шақырып, содан кейін оны толықтырсаңыз, бұған жол бермеуге болады:

```
class ComputerTable(DeskTable):
    def square(self, e):
        return DeskTable.square(self) - e
```

Бұл жерде басқа сыныптың әдісі шақырылады. Бұл жағдайда азайту.

Мысалы, KitchenTable сыныбында бізге әдіс қажет емес, конструкторда объект құру кезінде орындар өрісін орнату керек. Сыныпта сіз өзіңіздің конструкторыңызды ата -ананың орнына қоймай, нөлден жасай аласыз:

```
class KitchenTable(Table):
    def __init__(self, l, w, h, p):
        self.length = l
        self.width = w
        self.height = h
        self.places = p
```

Алайда, егер суперкласстың барлық конструкторы қайталанса, бұл ең жақсы әдіс емес. Ата -аналық конструкторды шақыру оңай, содан кейін өзіңіздің кодты қосыңыз:

```
class KitchenTable(Table):
    def __init__(self, l, w, h, p):
        Table.__init__(self, l, w, h)
        self.places = p
```

Енді KitchenTable сияқты нысанды құру кезінде конструкторда төрт аргумент көрсетілуі керек. Олардың үшеуі мұрагерлік баспалдақтан жоғары, ал төртіншісі сол жерде қабылданады.

```
>>> tk = KitchenTable(2, 1.5, 0.7, 10)
>>> tk.places
10
>>> tk.width
1.5
```

Ата-аналар класындағы әдепкі мәндері бар параметрлер

Ата - ана класында әдепкі мәні бар параметрлер бар, ал бала жоқ болған жағдайды қарастырыңыз:

```
class Table:
    def __init__(self, l=1, w=1, h=1):
        self.length = l
        self.width = w
        self.height = h
```

```
class KitchenTable(Table):
  def __init__(self, p, l, w, h):
     Table.__init__(self, l, w, h)
     self.places = p
```

Класстардың осы анықтамасымен сіз конструкторға дәлелдер келтірместен кестеден көшірме жасай аласыз:t = Table()

KitchenTable данасын тек p параметрінің мәнін беру арқылы жасай аламыз ба? Мысалы, мына сияқты:

k = KitchenTable(10)

P -ге 10 саны тағайындалуы мүмкін бе, ал l, w және h ата -аналық сыныптан алады? Мүмкін емес, конструктор талап ететін санға берілген аргументтер санының сәйкес келмеуіне байланысты ерекшелік шығарылады:

```
k = KitchenTable(10)
TypeError: __init__() missing 3 required
 positional arguments: 'l', 'w', and 'h'
```

Нысан бала класынан жасалған кезде, егер бар болса, алдымен конструктор шақырылады. Аудармашы бұл конструктордың денесінде ата-ана класының конструкторы шақырылатынын әлі білмейді. Өйткені, бұл қажет емес. Сонымен, егер бала конструкторының барлық параметрлерінде әдепкі мәндер болмаса, нысанды құру кезінде барлық мәндер беріледі.

Сондықтан, егер қажет болса жол объектілерін құру еншілес сынып бермей дәлелдер қажет болып тағайындалсын маңызы бар әдепкі бойынша, сондай-ақ құрушы еншілес сынып оқушысы.

```
class Table:
   def __init__(self, l=1, w=1, h=1):
      self.length = l
      self.width = w
      self.height = h
 class KitchenTable(Table):
   def __init__(self, l=1, w=1, h=0.7, p=4):
      Table.__init__(self, l, w, h)
      self.places = p
```

Ата-ана класында жоқ p параметрі, біз жай ғана емес, соңғы жасаймыз. Әр түрлі байланысты кластардың объектілері бір циклде, яғни бір алгоритмде өңделеді. Сонымен қатар, оларда бірдей" интерфейстер " болуы керек, яғни конструкторға берілетін дәлелдер саны бірдей.

Екіншісі-балалар класындағы конструктордан бас тарту, ал әдісті шақыру арқылы орнату опциясының өрісі үшін мән:

```
class Table:
   def __init__(self, l=1, w=1, h=1):
      self.length = l
      self.width = w
      self.height = h
 class KitchenTable(Table):
```

```
    places = 4
     def set_places(self, p):
        self.places = p
```

Мұнда барлық ас үй үстелдері әдепкі бойынша 4 орынға ие болады. Егер біз орындар өрісінің мәнін өзгерткіміз келсе, set_places () әдісін шақыра аламыз. Python жағдайында бұл тіректерді тағайындау арқылы тікелей жасай алады. Бұл дананың өзіндік орындар өрісіне ие болуына әкеледі.

```
k = KitchenTable()
k.places = 6
```

Сондықтан set_places () әдісі әдетте қажет емес.

Кез келген жағдайда орындардың ерікті саны конструкторда емес, бөлек орнатылады. Егер сіз әлі де объект жасау кезінде орындарды көрсетуіңіз керек болса, мұны ата -ананың конструкторында жасауға болады:

```
class Table:
    def __init__(self, l=1, w=1, h=1):
        self.length = l
        self.width = w
        self.height = h
        if isinstance(self, KitchenTable):
            p = int(input("Сколько мест: "))
            self.places = p
```

Isinstance () функциясының көмегімен жасалатын нысанның KitchenTable типі бар екендігі тексеріледі. Егер солай болса, онда ол орын өрісі бар.

Біз конструктор тақырыбында әдепкі мәні бар p параметрін қолданбаймыз, өйткені егер басқа байланысты сыныптар қажет болмаса, кодты құжаттаумен шатасулар мен қиындықтар болмайды.

Алгоритмдеу және бағдарламалау тілдері курсы бойынша жаттығулар

1Алгоритмдік өрнек тілінің ережелеріне сәйкес жазыңыз:

) $\frac{x+y}{x-1/2}-\frac{x-z}{xy};$

) $\frac{\sqrt{|\sin^2 x|}}{3{,}01x-e^{2x}};$

) $(1+z)\frac{x+\frac{y}{z}}{a-\frac{1}{1+x^2}};$

) $\frac{|\cos x^2-\sin^2 y|}{\sqrt[4]{|\ln x|}+xy};$

) $(x^n)^{m+2}+x^{n^m};$

) $\ln\left(y^{-\sqrt{|x+1|}}\right)\cdot\sin^2 \operatorname{arctg} z;$

) $\frac{(a+b)^n}{1+\frac{a}{a^m-b^{m-n}}};$

) $r_{ij}^{|x-y|}-0.15|\sin e^{-z^2}|;$

) $\frac{\left(a_i^{2l}+b_{j+1}^{2k}\right)\cdot\left(3^n-x\right.}{z-\frac{d_{i,j+1}+1}{z+\frac{y}{\sqrt{t^2+xyz}}}}$) $a^{(x+y)/2}-\sqrt[3]{\frac{x-1}{|y|+1}}\cdot e^{-(y+}$

[Ответ**]**

2. Арифметикалық өрнектерді қалыпты математикалық формада жазыңыз:

а) *a / b ** 2;* **л)** *5*arctg(x)-arctg(y)/4;*
б) *a+b/c+1;* **м)** *lg(u*(1/3)+sqrt(v)+z);*
в) *1/a*b/c;* **н)** *ln(y*(-sqrt(abs(x))));*
г) *a**b**c/2;* **о)** *abs(x**(y/x)-(y/x)**(1/3));*
д) *(a**b)**c/2;* **п)** *sqrt((x1-x2)**2+(y1-y2)**2);*
е) *a/b/c/d*p*q;* **р)** *exp(abs(x-y))*(tg(z)**2+1)**x;*
ж) *x**y**z/a/b;* **с)** *lg(sqrt(exp(x-y))+x**abs(y)+z);*
з) *4/3*3.14*r**3;* **т)** *sqrt(exp(a*x)*sin(x)**n)/cos(x)**2;*
и) *b/sqrt(a*a+b);* **у)** *sqrt(sin(arctg(u))**2+abs(cos(v)));*
к) *d*c/2/R+a**3;* **ф)** *abs(cos(x)+cos(y))**(1+sin(y)**2);*

[Жауап**]**

3. X = 1 арифметикалық өрнектердің мәндерін есептеңіз:
а) *abs(x-3)/ln(exp(3))*2/lg(10000);*
Шешім: *abs(1-3)=2; ln(exp(3))=3; lg(10000)=4; 2/3*2/4=0.33;*

б) *sign(sqrt(sqrt(x+15)))*2**2**2;*
в) *int(-2.1)*int(-2.9)/int(2.9)+x;*
г) *-sqrt(x+3)**2**(sign(x+0.5)*3)+tg(0);*
д) *lg(x)+cos(x**2-1)*sqrt(x+8)-div(2,5);*
е) *sign(x-2)*sqrt(int(4.3))/abs(min(2,-1));*
ж) *div(10,x+2)*mod(10,x+6)/max(10,x)*mod(2,5).*

[Жауап**]**

4.Мәндері болатын арифметикалық өрнектерді жазыңыз:
а)a, b, c (a, b, c> 0) және p семипериметрінен үшбұрыштың ауданы;
Жауап: sqrt(p*(pa)*(pb)*(pc));

б) a, b, c, d сандарының орташа арифметикалық және геометриялық ортасы;
в) (x, y) координаттары бар нүктеден (0,0) нүктесіне дейінгі қашықтық;
г) синус x градустан;
д) текше бетінің ауданы (жиектің ұзындығы a -ға тең);
е) текше шеңберінің шеңберінің радиусы (жиектің ұзындығы a -ға тең);
ж) a1x + b1y + c1 = 0 және a2x + b2y + c2 = 0 теңдеулерімен берілген екі

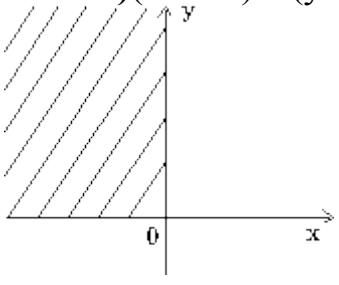
түзудің қиылысу нүктелерінің координаттары (түзулер параллель емес). **[**Жауап**]**

5.Логикалық өрнектердің мәндерін есептеңіз:
а) x * x + y * y <= 9 x = 1 үшін, y = -2 Жауабы: иә;
б) b * b -4 * a * c <0 a = 2, b = 1, c = -2 үшін;
в) (a> = 1) және (a <= 2) a = 1.5 үшін;
г) (a <1) немесе (a> 1.2) a = 1.5 үшін;
д) (mod (a, 7) = 1) және (div (a, 7) = 1) a = 8 үшін;
е) a (5), b = 4 үшін ((a> b) және (a <9) немесе (a * a = 4)) емес. **[**Жауап**]**

6.Логикалық өрнектерді тек көрсетілген шарттарда жазыңыз:
а) *X*[a, b] сегментіне жатады Жауабы: (x> = a) және (x <= b);
б) *X*[a, b] сегментінің сыртында жатыр;
в) *X*[a, b] сегментіне немесе [c, d] кесіндісіне жатады;
г) *X*[a, b] және [c, d] сегменттерінің сыртында жатыр;
д) бүтін сан k - тақ;
е) бүтін сан k-бес таңбалы санның үш таңбалы еселігі;
ж) екі өлшемді массивтің ai, j элементі тақ жол мен жұп бағанның қиылысында;
з) a1x + b1y + c1 = 0 және a2x + b2y + c2 = 0 түзулері параллель;
және) a, b, c сандарының ең кішісі c, үлкені b;
Кімге) a, b, c, d сандарының арасында өзара қарама -қарсы орналасқан;
и) a, b, c бүтін сандарының арасында кем дегенде екі жұп бар;
к) из отрезков с длиной a, b, c можно построить треугольник;
л) треугольники со стороны a1, b1, c1 и a2, b2, c2 подобны;
м) точка с координатами (x, y) принадлежит внутренней области треугольника с вершинами A (0,5), B (5,0) и C (1,0);
н) точка с координатами (x, y) принадлежит области, внешней по отношению к треугольнику с вершинами A (0,5), B (1,0) и C (5,0);
о) четырехугольник со стороны a, b, c и d является ромбом. **[**Ответ**]**

7. Жазықтықта (x, y) нақты өрнек көрсетілген және тек көрсетілген аймақты сызыңыз. Осы аймаққа жатпайтын шекараны нүктелі сызықпен сызыңыз.

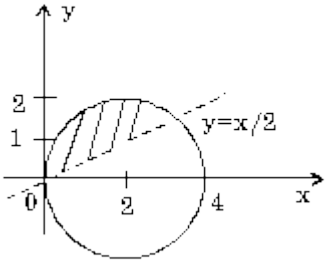
а)(x <= 0) и (y> = 0) Ответ:



б) (x> = 0) или (y <= 0)

е) ((x-2) ** 2 + y * y <= 4) и (y> x / 2)

Ответ:

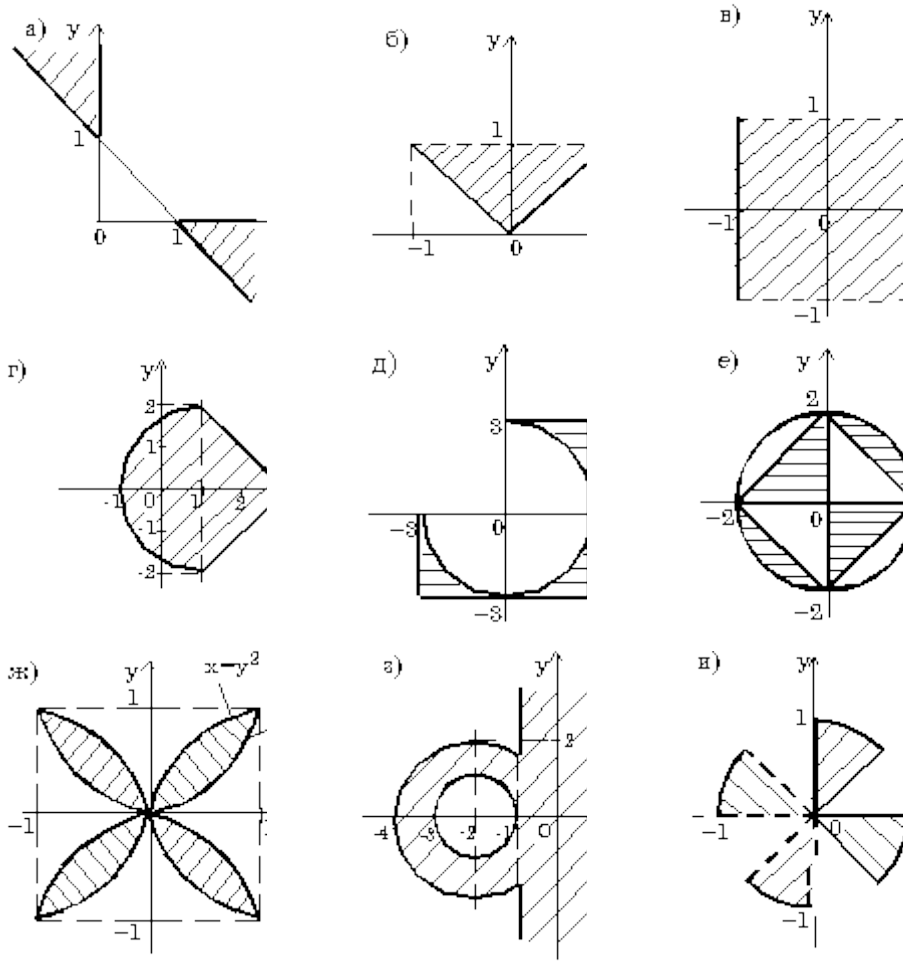


ж) (x * x + y * y <1) и (y> x * x);

в) x + y> = 0 **з)** (y> = x) и (y + x> = 0) и (y <= 1);
г) (x + y> 0) и (y <0) **и)** (абс (x) <= 1) и (y <2);
д)абс (x) + абс (y)> = 1 **к)** (x ** 2 + y ** 2 <4) и (x ** 2 + y ** 2> 1);

[Ответ**]**

8. координаттары бар нүкте (x, y) көлеңкеленген аймаққа тиесілі болған кезде ғана "Ақиқат" мәнін алатын логикалық өрнекті жазыңыз



[Ответ**]**

9. a = 3, b = 5, c = 7 болсын. Бұл айнымалылар операторларды орындау нәтижесінде қандай мәндерге ие болады:
а) *a: = a + 1; b: = a + b; c: = a + b; a: = sqrt (a)*
Решение: a = 3 + 1 = 4, b = 4 + 5 = 9, c = 4 + 9 = 13, a = {корень квадратный из} 4 = 2. Ответ: a = 2, b = 9, c = 13;
б) *c: = a * b + 2; б: = б + 1; a: = cb ** 2; b: = b * a;*
в) *b: = b + a; c: = c + b; b: = 1 / b * c;*
г) *p: = c; c: = b; б: = a; a: = p; c: = a * b * c * p;*
д) *c: = a ** (b-3); b: = b-3; a: = (c + 1) / 2 * б; c: = (a + б) * a;*
е) *x: = a; a: = b; b: = c; c: = x; a: = sqrt (a + b + c + x-2);*
ж) *b: = (a + c) ** 2; a: = lg (b ** 2) ** 2; c: = c * a * b.*
[Ответ**]**

10 Келесі әрекеттерді тағайындау операторларының көмегімен орнатыңыз: а) массив $x = (x_1, x_2)$ ережелерге сәйкес түрлендіріңіз: x_1 ретінде соманы алыңыз, ал x_2 ретінде-бастапқы компоненттердің көбейтіндісі; Жауап: c: = x [1]; x [1]: = x [1] + x [2]; x [2]: = c * x [2] **б)** массив элементтерінің мәндерін ауыстырыңыз $X = (x_1, x_2)$; **в)** A (N) массивінде i ($1 < i < N$) нөмірі бар компонент оған іргелес бастапқы компоненттің жартылай қосындысымен алмастырылады, көрші оң жақ компонент нөлге ауыстырылады, сол жақ компонент 0.5-ке көбейтіледі; **г)** u = max (x, y, z) + min (xz, y + z, y, z); **[**Ответ**]**

11. Формулалар бойынша есептеулерді таңдау немесе таңдау пәрмендерін қолданыңыз:

)
$$y = \begin{cases} \sqrt[7]{x} & \text{, если } x \le -100, \\ \sqrt[3]{x} & \text{, если } -100 < x < 100, \\ \sqrt{x} & \text{, если } x \ge 100; \end{cases}$$

)
$$z = \begin{cases} x^2 + y^2 & \text{, если } x^2 + y^2 \le 1, \\ x + y & \text{, если } x^2 + y^2 > 1 \text{ и } y \ge x, \\ 0{,}5 & \text{, если } x^2 + y^2 > 1 \text{ и } y < x; \end{cases}$$

)
$$F(z) = \begin{cases} 2z + 1, & \text{если } z \ge 0, \\ \sin z, & \text{если } z < 0, \end{cases}$$ где

)
$$z = \begin{cases} \lg(-x), & \text{если } x < 0, \\ \sqrt{x + 1}, & \text{если } x \ge 0. \end{cases}$$

)
$$z = \begin{cases} 1 & \text{, если } c = 0, \\ x & \text{, если } c = 1, \\ 3x^2 - 1/2 & \text{, если } c = 2, \\ x^3 - 3x/2 & \text{, если } c = 3, \\ 2x^4 - 3x/2 & \text{, в противном случае} \end{cases}$$

)
$$z = \begin{cases} \sqrt{x^2 + y^2} & \text{, если } |x| + |y| < r, \\ \max(|x|, |y|) & \text{, если } |x| + |y| \ge r; \end{cases}$$

)
$$v = \begin{cases} x + y & \text{, если } x > 1 \text{ и } y > 1, \\ x - y & \text{, если } x > 1 \text{ и } y \le 1, \\ -x + y & \text{, если } x \le 1 \text{ и } y > 0, \\ -x - y & \text{, если } x \le 1 \text{ и } y \le 0. \end{cases}$$

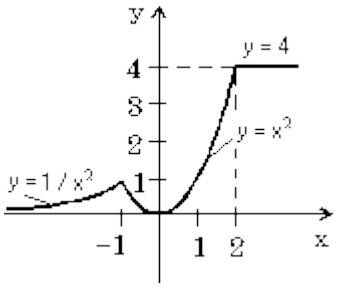
)
$$z = \begin{cases} |x| & \text{если точка лежит внутри круга радиусом r (r> 0) с точкой в точке (a, b)} \\ x & \text{в противном случае} \end{cases}$$

[Ответ**]**

12. Командалар анықтаған y (x) функциясының графигін жасаңыз, егер:

а) если x <= - 1
то y: = 1 / x ** (x)
2
иначе
если x <= 2
то y: = x * x
иначе y: = 4
все
все

Решение



б) если x <-5
то y: = - 5
иначе
если x <0
то y: = x
иначе
если x <3
то y: = 2 * x
иначе y: = 6
все
все
все

в) если x <-0,5
то y: = 1 / абс (x-0,5)
все
все

г) если x <0
то y: = 1
иначе
если x <3,14
то y: = соз (x)
иначе y: = - 1
все
все

д) если abs (x)> 2
то y: = x * x
иначе
если x <0
то y: = - 2 * x
иначе
если x> = 1
то y: = 4
иначе y: = 4 * x * x
все
все
все

[Жауап**]**

13.Орындалғаннан кейін S бүтін моделін анықтаңыз:

а) S: = 128
nts 1 -ден 4 -ке дейін
S: = div (S, 2)
кті

Шешім

ен	С.
	12 8
	12

G) S: = 0
үшін nts мен 1 ден 2 ге дейінмін
үшін nts j 2 -ден 3 -ке дейін
S: = S + i + j
кті

Шешім

ен		С.
		0
		0 + 1 + 2 = 3

	8/2 = 64
	64 /2 = 32
	32 /2 = 16
	16 /2 = 8

Жауап: S = 8

кте

		3 + 1 + 3 = 7
		7 + 2 + 2 = 11
		11 + 2 + 3 = 16

Жауап:S = 16

б) S: = 1; a: = 1
nts үшін 1 -ден 3 -ке дейін
S: = S + i * (i + 1) * a
a: = a + 2
кте
кте

в) S: = 1; a: = 1
nts үшін 1 -ден 3 -ке дейін
S: = S + i
nts 2 -ден 3 -ке дейін
S: = S + j
кте
кте

г) nts үшін 1 -ден 3 -ке дейін
S: = 0
nts 2 -ден 3 -ке дейін
S: = S + i + j
кте
кте

д) nts үшін 1 -ден 2 -ге дейін
S: = 0
nts 2 -ден 3 -ке дейін
n үшін 1 -ден 2 -ге дейін
S: = S + i + j + k
кте
кте
кте

[Жауап**]**

он төрт. Мәліметтерді орындағаннан кейін S мәнін анықтаңыз:

а)мен: = 0; S: = 0
nc кезінде i <3
i: = i + 1;
S: = S + i * i
кте

Шешім

Шарт <3	ен	С.
		0
0 <3? Иә		0 + 12 = 1

G)S: = 0; N: = 125
nc, ал N> 0
S: = S + mod (N, 10) | S - цифрлардың қосындысы
N: = div (N, 10) | саны N
кте

Решение

Услови е N> 0	S	
	0	25
125> 0?	0	

1 <3? да		1 + 22 = 5
2 <3? да		5 + 32 = 14
3 <3? нет (кц)		

Ответ:S = 14

да	+ 5 = 5	2
12> 0? да	5 + 2 = 7	
1> 0? да	7 + 1 = 8	
0> 0? нет (кц)		

Ответ:S = 8

б)S: = 0; я: = 1
нц пока i> 1
S: = S + 1 / я
я: = я-1
кц

в)S: = 0; я: = 1; j: = 5
нц пока i <j
S: = S + я * j
я: = я + 1
j: = j-1
кц

д)a: = 1; b: = 1; S: = 0;
нц пока a <= 5
a: = a + b; b: = b + a;
S: = S + a + b
кц

е)a: = 1; b: = 1
нц пока a + b <10
a: = a + 1
б: = б + a
кц
S: = a + Ь

[Ответ**]**

15. Сызықтық құрылымды есептерді шешудің алгоритмдерін жасаңыз (бұл мәселелердің шарттары в. М. Заварыкин, в. г. Житомирский және М. П. Лапчиктің "информатика және есептеу негіздері" Оқулығынан алынған, 1989):

а) үшбұрышта a, b және c үш жағы белгілі; формулаларды қолдана отырып, осы үшбұрыштың бұрыштарын (градуспен) табыңыз:

$$\cos A = \frac{b^2 + c^2 -}{2bc} \qquad \sin B = \frac{b \sin}{}$$ C = 180^0-(A + B).

Түсініктеме. Arccos және arcsin стандартты тригонометриялық функциялары есептелген мәнді радиалды шамада қайтаратынына назар аударыңыз.

Шешім:

Альг Углы треугольника (арг вещ a, b, c, рез вещ UgolA, UgolB, UgolC)
нач вещ RadGr, UgolARad
| RadGr - коэф. перевода угла из радианной меры в градусную
| УголАРад - угол А (в радианах)
RadGr: = 180 / 3,14
UgolARad: = ArcCos ((b * b + c * ca * a) / (2 * b * c))
УголА: = УголАРад * РадГр

УголБ: = ArcSin (b * sin (UgolARad) / a) * RadGr
УголС: = 180- (УголА + УголБ)
кон

б) үшбұрышта A, b екі жағы және олардың арасындағы с бұрышы (радиандарда) белгілі; формулаларды қолдана отырып, С жағын, А және В бұрыштарын (радиандарда) және үшбұрыштың ауданын табыңыз:

$$\sin A = \frac{a \sin C}{c}; \sin B = \frac{b \sin C}{c};$$

$$S = \frac{bc \sin A}{2};$$ c2 = a2 + b2 - 2ab cos C.

Түсініктеме. Алдымен с жағын , содан кейін қалған қажетті мәндерді табу керек;

в) үшбұрышта a, b және с үш жағы белгілі; формулаларды қолдана отырып, сипатталған шеңбердің радиусы мен А бұрышын (градуспен) табыңыз:

$$S = \frac{bc \sin A}{2}; \operatorname{tg} \frac{A}{2} = \sqrt{\frac{(p-b)(p-c)}{p(p-a)}},$$ где $$p = \frac{a+b+c}{2}.$$

г) дұрыс үшбұрышты Пирамидада базаның жағы мен бүйір бетінің негіз жазықтығына қарай қисаюының а бұрышы (градуспен) белгілі; формулаларды қолдана отырып, пирамиданың толық бетінің көлемі мен ауданын табыңыз:

$$V = S_{осн} \cdot H / 2;$$ $$S_п = S_{осн} \left(1 + \frac{}{c}\right.$$

де $$S_{осн} = \frac{a}{}$$ $$H = \frac{a\sqrt{3}}{6} \operatorname{tg}\alpha.$$

д) кесілген конуста R және r негіздерінің радиусы және генератрицаның үлкен базаның бетіне көлбеу бұрышы белгілі; формулаларды қолдана отырып, конустың бүйір бетінің көлемі мен ауданын табыңыз:

$$V = \frac{1}{3} \pi H (r^2 + R^2$$ $$S_{бок} = \pi l ($$

де $$H = (R-r) \operatorname{tg}($$ $$l = \frac{R-r}{\cos\alpha}.$$

д) дұрыс төртбұрышты Пирамидада базаның жағы А-ға тең, бүйір шеті базаның жазықтығына а бұрышымен қисайған; пирамиданың толық бетінің көлемі мен ауданын және пирамиданың шыңы мен базаның диагоналы арқылы өтетін қиманың ауданын табыңыз d; формулаларды қолданыңыз:

$$H = \frac{a\sqrt{2}}{2} \cdot \operatorname{tg}\alpha;$$ $$V = \frac{1}{3} \cdot S_{осн} \cdot H;$$

$$S_{сеч} = \frac{1}{2} H a \sqrt{2} = \frac{a}{}$$ $$S_{полн} = a^2 + a^2 \sqrt{2 \operatorname{tg}^2}$$

[Ответ**]**

16. Көңіл көтеретін құрылым есептерін шешудің алгоритмін жасаңыз:

а) берілген A, b, c жақтары бар үшбұрыштың изоссельдер екенін анықтаңыз;

Шешім:

Альг Треугольник (арг вещ a, b, c, рез лог Ответ)
дано| a> 0, b> 0, c> 0, a + b> c, a + c> b, b + c> a
надо| Ответ = да, если треугольник равнобедренный
| Ответ = нет, если треугольник не равноведренный
нач
если (a = b) или (a = c) или (b = c)
то Ответ: = да
иначе Ответ: = нет
все
кон

б) берілген A, b және c сандары арасындағы оң сандар санын анықтаңыз;

в) берілген екі тең емес санның кішісін екі есе көбейтіңіз, үлкенірек өзгеріссіз қалдырыңыз;

г) A және b сандары - бір тік бұрышты үшбұрыштың, ал c және d-басқасының катеттері; осы үшбұрыштардың ұқсас екендігін анықтау;

д) жазықтықта үш нүкте берілген; олардың қайсысы координаталар басына жақын екенін анықтау;

е) берілген нүкте (x, y) координатаның басында центрі бар, ішкі радиусы r1 және сыртқы радиусы R2 болатын жазықтық фигураға жататындығын анықтаңыз;

ж) A, b және c үш санының тізбегін көбейту арқылы реттеңіз.

Билеттер

№ 1 БИЛЕТ
1. Алгоритмдік және ауызша алгоритмдерді жазудың бағдарламалық әдісінің айырмашылығы неде?
2. Машина тілдерінің артықшылықтары мен кемшіліктері қандай?
3. Псевдокод тілінің сипаттамасы мен мақсатын беріңіз.
№ 2 БИЛЕТ
1. Параллелограмм түріндегі блок-схема фигурасының функционалдық мақсаты қандай?
2. Стандартты функция дегеніміз не?
3. Ассемблер тілі дегеніміз не? Тілдің сипаттамасы мен мақсатын беріңіз.
№ 3 БИЛЕТ
1. Кіші бағдарламалар қандай мақсаттарда қолданылады?
2. Логикалық өрнектер қалай жазылады?
3. LISP тілінің сипаттамасы мен мақсатын беріңіз.
N 4 БИЛЕТ
1. Процедуралар дегеніміз не және оларды қандай жағдайларда қолдану керек?
2. Таңба константасын орнатудың қандай әдістері бар?
3. FORTRAN тілін сипаттаңыз және тағайындаңыз.
№ 5 БИЛЕТ
1. Деректер түрлері дегеніміз не және олар қандай?
2. Бағдарламаларға қандай талаптар қойылады?
3. Бағдарламалау тілдерінің жіктелуін қандай білесіз?
№ 6 БИЛЕТ
1. Массив дегеніміз не? Массивтердің қандай түрлерін білесіз?
2. Алфавит, синтаксис, семантика ұғымдарына анықтама беріңіз.
3. Сіз қандай тілдер мен бағдарламалау жүйелерін білесіз және олардың ерекшеліктері қандай?
№ 7 БИЛЕТ
1. Нысанға бағытталған бағдарламалау термині нені білдіреді?
2. BASIC тілінің сипаттамасы мен мақсатын беріңіз.
3. Компиляция түсіндіруден несімен ерекшеленеді?
№ 8 БИЛЕТ
1. С тілінің сипаттамасы мен мақсатын беріңіз.
2. Арифметикалық өрнектер қалай жазылады?
3. Функция процедурадан несімен ерекшеленеді?
№ 9 БИЛЕТ
1. Алгоритм ұғымын сипаттаңыз және оның құрылымын анықтаңыз.
2. Логикалық операторлар дегеніміз не? Беріңіз, олардың сипаттамасы.
3. SQL тілін сипаттаңыз және тағайындаңыз.
№ 10 БИЛЕТ

1. Алгоритмді Орындаушы терминін беріңіз және оның негізгі сипаттамаларын тізімдеңіз.
2. Бағдарламалау тілдерінің жіктелуін қандай білесіз?
3. Java тілінің сипаттамасы мен мақсатын беріңіз-сценарий.

№ 11 БИЛЕТ

1. Ассемблер тілі дегеніміз не? Тілдің сипаттамасы мен мақсатын беріңіз.
2. Жол операторлары дегеніміз не? Беріңіз, олардың сипаттамасы.
3. Java тілін сипаттаңыз және тағайындаңыз.

№ 12 БИЛЕТ

1. Бағдарламаның сипаттамалық бөлігі дегеніміз не? Қандай, оның мақсаты не?
2. Массив дегеніміз не? Массивтердің қандай түрлерін білесіз?
3. PHP тілінің сипаттамасы мен мақсатын беріңіз.

№ 13 БИЛЕТ

1. Математикалық операторлар дегеніміз не? Беріңіз, олардың сипаттамасы.
2. Алфавит, синтаксис, семантика ұғымдарына анықтама беріңіз.
3. Clarion тілінің сипаттамасы мен мақсатын беріңіз.

№ 14 БИЛЕТ

1. Негізгі алгоритмдік құрылымдарды тізімдеп, оларға сипаттама беріңіз.
2. Функция процедурадан несімен ерекшеленеді?
3. Паскаль тілін сипаттаңыз және тағайындаңыз.

№ 15 БИЛЕТ

1. Компилятор мен аудармашының айырмашылығы неде? Екеуінің де жалпы жұмыс принциптерін сипаттаңыз.
2. Арифметикалық өрнектер қалай жазылады?
3. ActionScript тілін сипаттаңыз және тағайындаңыз.

№ 16 БИЛЕТ

1. Итерациялық циклдердің негізгі ерекшеліктерін және олардың ерекшеліктерін сипаттаңыз.
2. Стандартты тіл функциясы дегеніміз не?
3. Ada тілін сипаттаңыз және тағайындаңыз.

№ 17 БИЛЕТ

1. Басқару операторлары дегеніміз не? Беріңіз, олардың сипаттамасы.
2. Бағдарламаларға қандай талаптар қойылады?
3. FORTRAN тілін сипаттаңыз және тағайындаңыз.

№ 18 БИЛЕТ

1. Сіз қандай тілдер мен бағдарламалау жүйелерін білесіз және олардың ерекшеліктері қандай?
2. Таңба константасын орнатудың қандай әдістері бар?
3. С тілінің сипаттамасы мен мақсатын беріңіз.

№ 19 БИЛЕТ

1. Компиляция түсіндіруден несімен ерекшеленеді?
2. Нысанға бағытталған бағдарламалау термині нені білдіреді?
3. DBase тілін сипаттаңыз және тағайындаңыз.

№ 20 БИЛЕТ
1. Деректер түрлері дегеніміз не және олар қандай?
2. Желілік бағдарламалау және желілік тілдер дегеніміз не?
3. COBOL тілінің сипаттамасы мен мақсатын беріңіз.
№ 21 БИЛЕТ
1. Алгоритмнің қандай түрі блоктардың дәйекті орындалуына негізделген?
2. Декларативті бағдарламалау тілдері дегеніміз не?
3. Assembler тілін сипаттаңыз және тағайындаңыз.
№ 22 БИЛЕТ
1. Кірістірілген циклдердің негізгі ерекшеліктерін және олардың ерекшеліктерін сипаттаңыз.
2. Объектіге бағытталған бағдарламалау және ООР тілдері дегеніміз не?
3. С ++тілінің сипаттамасы мен мақсатын беріңіз.
№ 23 БИЛЕТ
1. Бағдарламаның алгоритмдік бөлігі дегеніміз не? Қандай, оның мақсаты не?
2. Логикалық өрнектер қалай жазылады?
3. Визуалды бағдарламалау дегеніміз не?
№ 24 БИЛЕТ
1. Ромбус түріндегі блок-схема фигурасының функционалдық мақсаты қандай?
2. Бағдарламалау тілінің деңгейі дегеніміз не?
3. Процедуралар дегеніміз не және оларды қандай жағдайларда қолдану керек?
№ 25 БИЛЕТ
1. Кіші бағдарламалар қандай мақсаттарда қолданылады?
2. Алгоритмнің қай түрі блоктардың қайталануына негізделген?
3. HTML тілін сипаттаңыз және тағайындаңыз.
№ 26 БИЛЕТ
1. Бағдарламалау тілдерінің классификациясының түрлерін келтіріңіз.
2. Ішкі бағдарламада сипатталған параметрлер қалай аталады?
3. DDL DML тілдері және олардың компоненттері дегеніміз не?
№ 27 БИЛЕТ
1. Процедуралық бағдарламалау және процедуралық тілдер дегеніміз не?
2. Ішкі тақырып атауындағы параметрлер қалай аталады?
3. Аудармашы дегеніміз не? Сізге қандай аудармашылар белгілі?
№ 28 БИЛЕТ
1. Тіктөртбұрыш түріндегі блок-схема фигурасының функционалдық мақсаты қандай?
2. Ішкі операторлардағы параметрлер қалай аталады?
3. Бағдарламалау тілінен процессор командаларына аударатын бағдарлама қалай аталады?

Әдебиеттер тізімі

1. Ашарина И.В. Основы программирования на языках С и С ++. - М .: Горячая линия - Телеком, 2002.

2. Марченко А.Л. С ++. Бархатный путь. - М .: Горячая линия - Телеком, 2002.

3. Дейтел Х.М., Дейтел П.Дж. Как программировать на С ++. - М .: БИНОМ, 1999.

4. Страуструп Б. Язык программирования С ++. - М .: Радио и связь, 1991.

5. Культин Н.Б. Самоучитель С ++ Builder. - СПб .: БХВ-Петербург, 2004.

6. Архангельский А.Я. С ++ Builder 6. Справочное пособие. Книга 1. Язык С ++. - М .: Бином-Пресс, 2002.

7. Вирт Н. Алгоритмы и структуры данных. - М .: Мир, 1989.

8. Культин Н. С / С ++ в задачах и примерах. - СПб .: Питер, 2002.

9. Аляев Ю.А., Козлов О.А. Алгоритмизация и языки программирования Pascal, C ++, Visual Basic: Учебно-справочное пособие. - М .: Финансы и статистика, 2004.

10. Липпман С., Лажойе Ж. Весь С ++ от азов до совершенства. - СПб .: Невский диалект. - М .: ДМК Пресс, 2007.

11. Давыдов В.Г. Технологии программирования С ++. - СПб., 2005.

12. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. - Томск: МП «РАСКО», 1991.

13. Красикова И.Е. С ++ просто как. - М., 2005.

14. Сябина Н.В. Технологии программирования. Конспект лекций (для студентов всех форм обучения спец. 050702, 050703). - Алматы: АИЭС, 2008.

15. Л.К.Ибраева, Н.В.Сябина. Информатика. Основы С ++. Часть 5. Методические указания к выполнению лабораторных работ (для студентов всех специальностей). - Алматы: АИЭС, 2006.

Жиынтық жоспары 2021 ж. реті

Қадылбекқызы Эльвира
Касымова Галия Кадылбековна
Шенер Анар Викторовна

ТАРТ 2209 ТЕЛЕКОММУНИКАЦИЯДАҒЫ АЛГОРИТМДІК БАҒДАРЛАМАЛАУ ТІЛДЕРІ

"6В06201–Радиотехника, электроника және телекоммуникациялар" мамандығы бойынша дәріс конспектісі (Бакалавриат)

Редактор: Изтелеуова Ж.Н.

Стандарттау бойынша маман: Ануарбек Ж.А.

Басуға қол қойылды
Таралымы 50 экз
Көлемі 4.6 оқу баспа

Пішіні 60х84 1/16
№1 типографиялық қағаз
Тапсырыс бағасы _2150_теңге

«Ғұмарбек Дәукеев атындағы Алматы энергетика және байланыс университеті» коммерциялық емес акционерлік қоғамының көшірмелі-көбейткіш бюросы
050013, Алматы, Байтұрсынұлы көшесі, 126/1