Цифрлық әдіскер

ақпараттық-білім беру ортасы

ПРАКТИКАЛЫҚ САБАҚТАР

Алгоритмдеу және бағдарламалау негіздері

Алгоритмдік ойлаудың негізі: ақпараттық жүйелер, деректерді өңдеу кезеңдері және бағдарламалау парадигмалары.

- Ақпараттық жүйелер
- Алгоритм құру принциптері
- Блок-схемалар
- Басқару құрылымдары

ӘДІСТЕМЕЛІК НҰСҚАУ

№1 ПРАКТИКАЛЫҚ САБАҚ

АРИФМЕТИКАЛЫҚ ОПЕРАЦИЯЛАР

1.1 тақырып: Python бағдарламалау тілі. Бағдарламалардың жалпы құрылымы. Сызықтық бағдарламалар.

> Жұмыстың мақсаты: бағдарламаны енгізу мен өңдеу үшін Python біріктірілген қабықшасын қолдануды үйрену, сызықтық бағдарламаларды құру үшін тілдің негізгі құрылымдарын қолдану.

Қысқаша теориялық мәлімет Интерпретаторды іске қосу үшін Іске қосу батырма мәзірін ашыңыз және Python 3.5 таңдаңыз. Бұл топтан IDLE Python 3.5 таңдаңыз. Терезе ашылады (1 сурет)

1 сурет. IDLE ортасының жұмыс терезесі

Алдымызда интерпретатордың командалық қабықшасы. Бұл терезеде бірнеше мәзір мен үлкен жұмыс облысы бар, онда әрбір үш >>> бағыттауыштан кейін курсор белгі береді – бұл командалық қатар. Бұл жерде <Enter> пернесін басқаннан соң орындалатын команда енгізіледі. 1 мысал. Біз команданы орындағымыз келеді:

```
print(" Академиның барлық студенттеріне – жалынды сәлем!!")
```

бұл команданы командалық қатарға енгізуіміз (яғни, >>> символдан соң, курсор тұрған жерге) және < Enter> пернесін басуымыз керек. Нәтижесінде, команда орындалады, ал оның орындалу нәтижесі төменде, командалық қатардан кейін көрсетіледі.

2 сурет. Бірінші мысалды орындау нәтижесі.

Сонымен қатар команданы орындау нәтижесінің астында тағы да үш > > > бағыттауыш пайда болады және бұл жерге жаңа команданы енгізуге болады. 2 мысал. 3 суреттегідей алгебралық өрнек енгіземіз: 15*4 + 20/4.

3 сурет. Екінші мысалды орындау нәтижесі.

Бағдарламаның файлын сол ортаның қабықшасының көмегімен құрамыз. Егер File мәзірін шертетін болсақ, командалар мен ішкі мәзірлер тізімі ашылады, оның ішінен New - File таңдаймыз. Осы команданы таңдаған соң, 4 суретте көрсетілгендей кодтар редакторы бірден ашылады.

4 сурет. Кодтар редакторын бағдарлама файлын құру үшін қолданамыз

Редактор терезесіне бағдарламалық кодты енгізуге болады. 3 мысал. 1 листингте көрсетілген, бірнеше командалардан тұратын қарапайым бағдарламаны кодтар редакторына енгізу.

Листинг 1.

```
print( " Начинаем вычисления ! " )
a = 4
print( " Значение переменной a= " , a )
b= 12
print( " Значение переменной b= " , b )
c=b / a
```

```
print( " Результат деления b/ a= " , c )
print( " Вычисления закончены ! " )
```

Дәл осы бағдарламалық кодты кодтар редакторы терезесіне енгіземіз. Бағдарламалық коды бар редактор терезесі 5 суретте көрсетілген. Егер сіз командаларды синтаксистік қателіксіз, дұрыс енгізсеңіз, олар күлгін түспен белгіленеді. Егер команда қара түсте болса – онда қателік бар деген сөз. 5 сурет. 3 мысалдың бағдарламалар коды бар редактор терезесі

Бағдарламалық кодты терген соң, оны бірден орындауға болады. Ол үшін Run мәзірінен Run Module командасын таңдаймыз. Егер сіз бағдарламаны сақтамасаңыз, онда оны сақтау ұсынысы бар терезе ашылады. Бағдарламаның мәтінін әрбір өзгерткен сайын оны сақтап отыру қажет. Бағдарламаны сақтау үшін өз компьютеріңіздің Жұмыс үстеліне орналасқан Студенттер бумасын құрыңыз. Бағдарламаны орындау нәтижесінде командалық қабықша терезесінде 6 суретте көрсетілген нәтиже пайда болады.

6 сурет. 3 мысалдағы бағдарламаны орындау нәтижесі

Жұмысты орындау жоспары: Әрбір тапсырмаға шешім алгоритмінің блок-схемасын құрыңыз, сосын Питон бағдарламалау тілінде бағдарламасын жазыңыз. Бағдарламаны енгізіп, іске қосып, нәтижесін алыңыз. 1 тапсырма. x және y сандары берілген. Олардың қосындысын, айырмасын және көбейтіндісін есептеңіз. Нұсқау: x және y сандары пернетақтадан енгізіледі. 2 тапсырма. Сыныпта N оқушылар бар. Бақылаудан соң: А - бес, В –төрт, С –үш деген бағаларды алды. Үштің, төрттің және бестіктің пайыздарын табыңыз. 3 тапсырма. Екі параллель қосылған резисторлардың R1 және R2 кедергілерінен тұратын, бөліктегі I күш тогының мәнін есептейтін бағдарлама құрыңыз, егер осы бөліктегі кернеу U тең болса.

Жұмысты орындау нәтижесі: Есепте үш бағдарламаның блок-схемасын және тестілеу нәтижелерін басып шығарыңыз. Бақылау сұрақтары

1. Python тілінде айнымалылардың атауларын тұрғызу ережелерін сипаттаңыз. Python тілінде түсіндірме сөздер қалай жазылады? Алгоритмде қателікті іздеу кезінде түсіндірме сөздерді қалай қолдануға болатынын ойлаңыз? Python тілінде шығару операторының жұмысы туралы айтыңыз. Айнымалы дегеніміз не? Трансплятор айнымалының типін қалай анықтайды? Айнымалының типі не үшін қажет? Айнымалының мәнін қалай өзгертуге болады? Меншіктеу операторы дегеніміз не? Мәліметтерді енгізудің алдында көмекші сөзді шығару не үшін дұрыс? Қашан дәл шығару операторында нәтижені есептеуге болатынын, ал қашан жеке айнымалыны іске қосу керектігін ойланыңыз. Форматтық шығару дегеніміз не?

1.2 тақырып.Есептеулер

> Жұмыстың мақсаты: бағдарламаларда есептеулерге арналған арнайы операторларды, форматтық шығаруды, кездейсоқ сандардың генераторын, eval() функциясын қолдануды үйрену.

Қысқаша теориялық мәлімет Python тіліндегі бағдарламаларда қолданылатын мәліметтердің негізгі типтері:

- int – бүтін мәндер;
- float – нақты мәндер (бөлшек болуы да мүмкін); Осы типтердің айнымалылары мен сандарына келесі арифметикалық амалдар қолданылауы мүмкін:

1.3 КЕСТЕ. АРИФМЕТИКАЛЫҚ АМАЛДАР

Оператор	Сипаттамасы
+	Қосу операторы. Сандардың қосындысын есептейді
-	Азайту операторы. Сандардың айырмасын есептейді
*	Көбейту операторы. Сандардың көбейтіндісін есептейді
/	Бөлу операторы. Сандардың қатынасын есептейді
//	

Бүтін санды бөлу операторы. Бір санды екіншісіне бөлген кезде бүтін бөлігін есептейді. % Санды бөлуден қалатын қалдықты есептейтін оператор. Бір санды екіншісіне бөлген кезде қалдық бөлікті есептейді. ** Дәрежені есептеу операторы.

Арифетикалық өрнектер сандардан, айнымалылар атауларынан, арифметикалық амалдардың белгілерінен, дөңгелек жақшалардан (әрекеттер ретін өзгерту үшін) және функцияларды шақырудан тұруы мүмкін. Мысалы,

```
a =(c + 5 – 1) / 2*d
```

Егер ұзын өрнектің жазбасы экранның бір қатарына сыймаса, оны «\» (оны «кері слэш» деп атайды) белгісінің көмегімен келесі қатарға орналастыруға болады:

```
a =(c + 5 – 1) \
```

/ 2*d

```
Python тілінде (Си тілінде секілді) көптік меншіктеуді қолдануға болады.
a = b = 0
```

жазбасы

```
b = 0
a = b
```

операторлардың жұбына тең Сонымен қатар, Си тілінде секілді, арифметикалық амалдардың қысқартылған жазбасын жиі қолданылады: қысқартылған жазба толық жазба

```
a += b                a = a + b
a –= b                    a = a – b
a *= b                    a = a * b
a /= b                    a = a / b
```

Егер өрнекке әр түрлі типтердің айнымалылар кірса, кейбір жағдайларда автоматты түрде аса «үлкен» типке келеді. Мысалы, бүтін санның нақты санға көбейту нәтижесі – бұл нақты сан. «Кіші» типке өту автоматты түрде орындалмайды. Бөлу («/» амалының) нәтижесі – бұл нақты сан, тіпті бөлінді мен бөлгіш – бүтін сандар және бір біріне бүтін бөлінсе де. Бүтін сандарды бөлуден бүтін және қалдық сандарды алу жиі керек болады. Бұл жағдайда «//» және «%» операторларын қолданады:

1 мысал.

```
d = 85
a = d // 10         # = 8
b = d % 10         # = 5
```

Форматтық шығару жиі қолданылады: енгізуге қажетті барлық мәліметтерді алдымен format функциясының көмегімен символдық қатарға өзгертеді: 2 мысал.

```
a = 1/3
print ( "{:7.3f}".format(a) )
```

Бұл жағдайда «7.3f» форматы қолданылған, бұл бөлшек бөлікті 7 позицияда үш белгімен бекітілген үтір арқылы (f от англ. fixed – бекітілген) санның шығуын анықтайды: 0.333 Бұл жазба 5 позицияны (ал оған 7 белгіленген) алатындықтан, санның алдына «◦» белгісімен белгіленген екі бос орын қосылады. Бірден бірнеше мәндер үшін форматтық шығаруды қолдануға болады (бұл мәндердің тізімі дөңгелек жақшаларға жазылады): 3 мысал.

```
a = 1/3
b = 1/9
print ( "{:7.3f} {:7.3f}".format(a, b) )
```

«%e» жазбасы экспоненциалдық форматты білдіреді:

```
print ( "{:10.3e} {:10.3e}".format(a, b) )
```

мұндағы 10 мен 3 сандары – бұл белгіленген бөліктің жазбасындағы ондық нүктеден кейінгі белгілердің саны мен позициялардың жалпы саны: ◦3.333e-01 1.111e-01 Стандартты функциялар Python тілінің кітапханасы бағдарламадан шақыруға болатын дайын функциялардан тұрады. Кейбір функциялар тілдің ядросына енгізілген, мысалы, санның модулін есептеу үшін abs функциясы қолданылады:

```
print ( abs(–1.2) )
```

Нақты сандардан бүтін сандарға өтуге арналған енгізілген функциялар бар:

```
• int(x) –x нақты санды бүтін санға келтіру, бөлшек бөлікті алып тастау;
• round(x) – x нақты санды жақын бүтінге дөңгелектеу.
```

Python тілінің көптеген стандартты функциялары қызметі бойынша топтарға бөлінеді және әрбір топ модуль деп аталатын жеке файлға жазылады. Математикалық функциялар math модуліне жинақталған:

```
sqrt(x) – x санының квадраттық түбірі ;
sin(x) – радианмен берілген x бұрышының синусы;
cos(x) – радианмен берілген x бұрышының косинусы;
exp(x) – x санының экспоненті;
log(x) – x санының натурал логарифмі.
Осы модульді қосу үшін импорттау (модульді жүктеу) командасы қолданылады:
import  math
```

Сосын функцияларды шақыру үшін нүктелік жазба деп аталатын команда қолданылады: модульдің аты және сосын нүктеден кейін функцияның атауы көрсетіледі:

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

Басқаша орындауға да болады: модульдің барлық функцияларын жұмыс кеңістігіне жүктеуге болады:

```
from math import *
```

Енді math модулінің функцияларын енгізілген функциялардағы секілді шақыруға болады:

```
print ( sqrt(x) )
```

Бұл тәсілдің айтарлықтай кемшілігі бар: жұмыс алаңында көптеген қосымша атаулар пайда болады, яғни басқа модульдерде хабарланған функциялардың атауларымен сәйкес келуі мүмкін. Сондықан қажеттілік болмаса, олай істеудің керегі шамалы. Үшінші нұсқа – тек қажетті функцияларды жүктеу.

```
from math import sqrt, sin, cos
```

Бұл жағдайда аталғандардан басқа (sqrt, sin, cos), math модулінің барлық функциялары қолжетімсіз болады. Псевдокездейсоқ сандармен жұмыс істеу функциялары random модуліне жинақталған. Берілген аралықта псевдокейздейсоқ сандарды алу үшін функцияларды қолданады:

```
• random() – жартылай интервалдағы [0,1) кейдейсоқ нақты сан;
• randint(a,b) – [a,b] кездейсоқ бүтін сан.
```

4 мысал. 1 ден 6 дейінгі аралықта (кубикті лакқтыру нәтижесі) n айнымалыға кездейсоқ санды жазу үшін төмендегідей операторларды қолдануға болады:

```
from random  import  randint
n = randint(1,6)
print('Выпало число ',n)
```

5 мысал. 5 пен 12 (12 қоспағанда) жартылай интервалдағы нақты кездейсоқ сан мынандай болады:

```
from random import random
x = 7*random() + 5
print('x=  ', x)
```

eval ( ) функциясы мәтіндік форматта берілген өрнектерді есептеуге мүмкіндік береді. Мысалы, егер Python тілінің синтакисінің ережелеріне сәйкес жазылған кейбір алгебралық өрнекті екілік тырнақшаға алсақ, онда мәтін пайда болады. Егер осы мәтінді print ( ) функциясының аргументімен көрсететін болсақ, онда экранда сәйкес алгебралық өрнек пайда болады. Егер дәл осы мәтінді (алгебралық өрнекпен) eval () функциясының аргументімен беретін болсақ, онда сәйкес алгебралық өрнек есептелетін болады. 6 мысал.

```
a="(5+2)**2–3*2"
b=''6 – 5/2"
c="10//4+1 0%3"
print ("Результаты вычислений :")
print (a+" =" , eval (a) )
print (b+" =" , eval (b) )
print (c+" =" , eval (c) ).
```

Осы бағдарламалық кодты орындау нәтижесінде алатынымыз: Есептеулер нәтижелері:

```
(5+ 2) **2–3*2 = 43
```

6-5/2 = 3.5 10//4+10%3 = 3 Жұмысты орындау жоспары: 1 тапсырма. 1-6 мысалдардағы жұмыстарды енгізіп, нәтижелерін алыңыз. 2 тапсырма. Шеңбердің радиусын енгізетін және оның ауданы мен ұзындығын есептейтін бағдарламаны жазыңыз. π санына жуық тең болатын, math модулінен math.pi енгізілген константаны қолданыңыз. 3 тапсырма. Екі бүтін санды a және b (a < b) енгізетін, және [a,b] аралығынжа 5 кездейсоқ бүтін сандарды экранға шығаратын бағдарлама жазыңыз. 4 тапсырма. Екі ойын кубиктерді лақтыруды үлгілейтін бағдарлама жазыңыз: қосу кезінде 2 мен 12 аралығында кездейсоқ сан шығаратын. 5 тапсырма. Кездейсоқ тәсілмен кезекшілерді таңдайтын бағдарлама жазыңыз: 1 мен N аралығындағы жеке кездейсоқ сандарды екі рет шығаратын, мұндағы N – топтағы студенттердің саны. Бақылау сұрақтары

1. Мәліметтердің қандай типтерін білесіз?
2. Қандай мәліметтер логикалық айнымалыларға жазылады?
3. Python тіліндегі айнымалылардың ерекшеліктері туралы айтыңыз.
4. Амалдардың басымдылықтары дегеніміз не? Ол не үшін қажет?
5. Амалдар қандай ретпен орындалады, егер олардың басымдылықтары бірдей болса?
6. Жақшалар не үшін қолданылады?

7. Егер өрнектің құрамында әр түрлі сандық типтердің айнымалылары болса, не болады?
8. // және % амалдарын сипаттаңыз.
9. Қандай стандартты функцияларды білесіз? Тригонометриялық функциялардың аргументі қандай бірлікпен беріледі?
10. Нақты санды бүтін санға жуықтап дөңгелектеуді қалай орындауға болады?

№2-3 ПРАКТИКАЛЫҚ САБАҚ

ШАРТТЫ ОПЕРАТОР

2.1 тақырып: Python бағдарламалау тілі. Тармақталған құрылымдағы бағдарламалар.

> Жұмыстың мақсаты: Python бағдарламалау тілінің логикалық таңдау операторларын қолдануды үйрену.

Қысқаша теориялық мәлімет

1. Шартты оператор: if шарт: 1 нұсқаулар блогы else: 2 нұсқаулар блогы Шартты оператор келесі алгоритм бойынша жұмыс істейді: Алдымен IF қызметтік сөзінен кейін орналасқан логикалық өрнектің (шарттың) мәні есептеледі. Егер оның нәтижесі ақиқат болса, жаңа қатарда 4 бос орынмен басталатын 1 нұсқаулар блогы орындалады, егер нәтижесі жалған болса, онда ELSE сөзінен кейін орналасқан 2 нұсқаулар блогы орындалады. Нұсқаулар блогы қажетті операторлар санынан тұруы мүмкін, бірақ олардың барлығы міндетті түрде 4 бос орыннан кейін жазылуы керек. 1 мысал. Берілген екі санның үлкенін экранға шығару. x= int(input('Введите первое число')) y= int(input('Введите второе число')) if x > y: print('Большее число', x) else: print('Большее число', y)

Шартты нұсқауда else сөзі мен келесі блок болмауы да мүмкін. Мұндай нұсқау толық емес тармақталу деп аталады. Мысалы, егер: 2 мысал. Пернетақтадан x саны еншізіледі және біз оны абсолюттік өлшемге, осы санның модуліне алмастырғымыз келеді. Мұны былай жасауға болады: егер сан теріс болса, «унарлық минус» деген амалды қолдануымыз, ал егер сан оң болса, онда ештеңе өзгертпестен шығаруымыз керек.

```
x = int(input())
```

if x < 0:

```
    x = -x
print(x)
Бұл мысалда x айнымалыға –x мәні меншіктеледі, бірақ тек x<0 болғанда ғана. Ал print(x) нұсқауы тексерілетін
шартқа тәуелсіз, әрқашанда орындалады.
```

2 мысал: Пайдаланушы мектептегі сынып нөмірін енгізеді, бағдарлама оның қай сыныпқа жататыны туралы хабарлама шығарады.

```
n=int(input('Введите номер класса'))
```

if n<4:

```
    print('Младшие классы')
else:
```

if n<9:

```
        print('Средние классы')
    else:
        print('Старшие классы')
```

HYPERLINK "http://pythontutor.ru/lessons/ifelse/" \l "section_3"Салыстыру операторлары Әдетте, тексерілетін шарт ретінде төмендегі салыстыру операторларының бірін есептеу нәтижесі қолданылады: < кіші — шарт дұрыс, егер бірінші операнд екіншісінен кіші болса. > үлкен — шарт дұрыс, егер бірінші операнд екіншісінен үлкен болса.

```
<=   кіші немесе тең.
>=  үлкен немесе тең.
==   теңдік. Шарт дұрыс, егер екі операнд тең болса.
```

!= теңсіздік. Шарт дұрыс, егер екі операнд тең болмаса. Мысалы, (x*x < 1000) шарты «x*x мәні 1000 нан кіші» дегенді, ал (2 * x != y) шарты «x айнымалысының мәні y айнымалының мәніне тең емес» дегенді білдіреді. Питонда салыстыру операторларын тізбекке біріктіруге (көптеген басқа да бағдарламалау тілдеріне қарағанда, мұнда ол үшін логикалық байланыстарды қолдану керек) болады, мысалы, a == b == c немесе 1 <= x <= 10. Сонда 2 мысалды дұрысырақ жазуға болатын еді :

```
n=int(input('Введите номер класса'))
if 1<=n<=4:
    print('Младшие классы')
else:
     if 4<n<9:
         print('Средняя школа')
     else:
         print('Старшие классы')
```

Салыстыру операторлары арнайы bool логикалық типінің мәнін қайтарады. Логикалық типтің мәндері екі мәндердің бірін қабылдауы мүмкін: True (ақиқат) немесе False (жалған). Егер логикалық True мәнін int типіне өзгертсек, онда 1 болады, ал егер False болса, онда 0 береді. Кері жағдайда 0 саны False өзгереді, ал кез келген нөл емес сан True өзгереді. Str типін bool типіне кері өзгерту кезінде, бос қатар False, ал кез келген бос емес қатар True мәніне өзгереді. Жұмысты орындау жоспары 1 тапсырма. Пернетақтадан үш сан енгізіледі. Тек оң сандардың ғана қосындысын табу керек. 2 тапсырма. Сіздің енгізген саныңыз қалдықсыз 3ке бөлінетіндігін тексеретін бағдарлама құрыңыз. 3 тапсырма. Функцияныі графигі өтетіндігін немесе өтпейтіндігін анықтайтын бағдарлама құрыңыз. (a,b) координаттарымен берілген нүкте арқылы y = 5×2 – 7 x + 2. Координаттар пернетақтадан енгізіледі. 4 тапсырма. Адамды жасы бойынша талдайтын және мектеп жасына дейінгі, оқушы, жұмысшы, зейнеткер сияқты төрт топтардың біріне жатқызатын бағдарлама құрыңыз. Адамның жасы пернетақтадан енгізіледі. 5 тапсырма. HYPERLINK "http://pythontutor.ru/lessons/ifelse/problems/signum/"Санның белгісі

```
Математикада sign(x) (санның белгісі) функциясы төмендегідей анықталған:
sign(x) = 1, егер x > 0, □
sign(x) = −1, егер x < 0, □
sign(x) = 0, егер x = 0.
```

Берілген x саны үшін sign(x) мәні. Бұл есепті if... elif... else нұсқауларын қолдану арқылы шешу керек. 6 тапсырма. HYPERLINK "http://pythontutor.ru/lessons/ifelse/problems/chess_board/"Шахматтық тақта Шахматтық тақтаның екі торы берілген. Егер олар бір түспен боялса, онда YES сөзін, ал егер әр түрлі түске боялса — NO сөзін шығарыңыз. Бағдарлама бірінші тор үшін алдымен баған мен қатардың нөмірін беретін, сосын екінші тор үшін кіріске әрқайсысы 1 ден 8 дейінгі сандардың ішінен төртеуін алады. 7 тапсырма. HYPERLINK "http://pythontutor.ru/lessons/ifelse/problems/leap_year/"Кібісе жылы Натурал сан берілген. Нөмірмен берілген жыл кібісе немесе ондай еместігін анықтау керек. Егер жыл кібісе болса, онда YES сөзін, кері жағдайда NO сөзін шығарыңыз. Есте сала кетсек, григориан күнтізбесіне сәйкес, жыл кібісе болып табылады, егер нөмір 4ке еселі, бірақ 100 еслі болмаса, сондай ақ егер ол 400 еселі болса. Задание 8. Корольдің жүрісі Шахматтық король тек 1 тор бойынша ғана көлденең, тігінен және диогональ бойынша жүреді. Шахматтық тақтаның екі түрлі торлары берілген, корольдің бірінші тордан екіншісіне бір жүріспен түсе алатындығын анықтаңыз. Бағдарлама бірінші тор үшін алдымен баған мен қатардың нөмірін беретін, сосын екінші тор үшін кіріске әрқайсысы 1 ден 8 дейінгі

сандардың ішінен төртеуін алады. Бағдарлама YES сөзін, егер корольдің жүрісімен екіншісіне түсе алса немесе кері жағдайда NO сөзін шығарыңыз. Жұмысты орындау нәтижесі: 1-6 тапсырмаларының бағдарламалар мәтіндерін, алгоритмдер блок-схемаларын және тестілеу нәтижелерін есепте басып шығарыңыз.

2.2 тақырып: Логикалық операторлар мен логикалық өрнектер

> Жұмыстың мақсаты: Бағдарламаларда күрделі шарттарды, логикалық операторларды қолдана білуді үйрену.

Қысқаша теориялық мәлімет Кейде бір уақытта бір емес, бірнеше шарттарды тексеру керек болады. Мысалы, берілген санды (n % 2 == 0) (n ді 2ге бөлгенде қалдық 0 тең) шартының көмегімен жұп екендігін/еместігін тексеру керек, ал егер берілген екі n және m бүтін сандардың жұп екендігін тексеру қажет, екі шарттың да әділдігін тексеру керек: n % 2 == 0 және m % 2 == 0, оларды and (логикалық ЖӘНЕ) операторының көмегімен біріктіру керек: n % 2 == 0 and m % 2 == 0. Питон тілінде стандартты логикалық операторлар бар: логикалық ЖӘНЕ; логикалық НЕМЕСЕ; логикалық терістеу. Логикалық ЖӘНЕ бинарлық операторлар (яғни сол және оң екі операнды бар оператор) болып табылады. and операторы True мәнін қайтарады, тек екі операндтың мәні True болғанда ғана. Логикалық НЕМЕСЕ бинарлық оператор болып табылады және ең болмағанда тек бір операнд True тең болғанда ғана True мәнін қайтарады. «Логикалық НЕМЕСЕ» операторы or түріне ие. Логикалық ЕМЕС (терістеу) унарлық (яғни бір ғана операндпен) оператор болып табылады және жалғыз операнды бар not түрінде болады. Логикалық ЕМЕС егер операнд False тең болса және керісінше болса, True қайтарады. Мысал 1. Кемінде a немесе b сандарының біреуі 0-де аяқталады:

```
a = int(input())
b = int(input())
if a % 10 == 0 or b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Жұмыс жоспары

Тапсырма 1. X нөмірін ескеріңіз. Бұл нөмір осы аралыққа [a, б] сәйкес келетінін анықтаңыз. Тапсырма 2. A нақты саны беріледі, егер f (x) = x2 + 4x + 5 болса, онда f (A)▒ x> 2; әйтпесе f (x) = 1 / (x2 + 4x + 5). Тапсырма 3. Патшаның жолы Шахмат патшасы көлденең, тігінен және диагональмен жүреді, бірақ тек 1 шаршы. Шахмат тақтасының екі түрлі жасушасын ескере отырып, корольдің бірінші кезектен екінші айналымға бір рет ауыса алатынын анықтаңыз. Бағдарлама әрқайсысы 1-ден 8-ге дейінгі төрт нөмірді қабылдайды, ал бірінші ұяшық үшін бағанның нөмірі мен жолдың нөмірін көрсетіп, екінші ұяшық үшін. Егер бағдарламаның бірінші ұяшығынан екіншісіне немесе '' NO '' деп жазылса, бағдарлама «YES» Жұмыстың нәтижесі: тестілеу нәтижелері бойынша 1-3 тапсырмалардың бағдарламасы туралы есеп.

Тақырып 2.3. Шартты каскадты нұсқаулар

> Жұмыстың мақсаты: кірістірілген жағдайларды, бағдарламалау тіліндегі каскадты шартты операторларды қолдану арқылы күрделі алгоритмдерді жазуды үйрену.

Қысқаша теориялық мәлімет

```
1-мысал. X және y нөл емес сандарды ескере отырып, координаталық жазықтықтың квадранттарының нүктесін (x, y)
анықтайтын бағдарламаны қарастырыңыз. Бағдарлама операциялардың каскадты реттілігін пайдаланады, егер ...
elif ... else:x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Первая четверть")
elif x > 0 and y < 0:
    print("Четвертая четверть")
elif y > 0:
    print("Вторая четверть")
else:
    print("Третья четверть")
```

Бұл құрылыста «if», ..., «elif» шарттары өз кезегінде тексеріледі, шындықтың бірінші шарттарына сәйкес блок орындалады. Егер барлық шарттар жалған болса, онда егер басқа болса, басқа блок орындалады. Мысал 2. Пайдаланушы енгізген нөмір үшін амалдық жүйені анықтау

```
print("""Какой операционной системой вы пользуетесь?
```

1 - Windows 7 2 - Windows XP 3 - Windows Vista 4 -Другая"""')

```
os = input ( "Введитете число, соответствующее ответу: ")
os = os.rstrip("\r")
```

if os = "1":

```
    print ("Вы выбрали –Windows 7")
```

e1if os = "2":

```
    print ("Вы выбрали – Windows XP")
elif os = "3":
     print ("Вы выбрали – Windows Vista")
elif os = "4":
    print ("Вы выбрали – Другая")
elif not os:
    print ("Вы не ввели число" )
else:
     print ("Мы не смогли определить вашу операционную систему")
```

Мысал 3. Алдыңғы бағдарламаны пайдаланушы ешқандай нөмірді енгізген кезде опцияны болдырмау үшін өзгертіңіз

```
print("""Какой операционной системой вы пользуетесь?
```

1 - Windows 7 2 - Windows XP 3 - Windows Vista 4 - Другая:""')

```
os = input ("Введите число, соответствующее ответу: ")
os = os.rstrip("\r")
```

if os != "": if os = "1":

```
         print ("Вы выбрали – Windows 7")
    e1if os == "2":
         print ("Вы выбрали – Windows XP")
    e1if os == "3":
         print("Вы выбрали– Windows Vista")
    e1if os == "4":
         print ("Вы выбрали– Другая")
```

e1se:

```
         print ("Мы не смогли определить вашу операционную систему")
else:
print ("Вы не ввели число")
```

Тапсырма 1. 1-3 мысалдар бағдарламаларын енгізіңіз. Нәтижені алыңыз Тапсырма 2. Пайдаланушы нөмірін болжайтын бағдарламаны жазыңыз. Бағдарлама хабарларды көрсетеді: Guessed!, Кішкентай, көптеген адамдар. Тапсырма 3 .. A нақты саны A беріледі f (A) f (x) = 0 егер x <= 0; f (x) = x2 - x үшін 0 <x <1, әйтпесе f (x) = x2 - sin (x2).

Бақылау сұрақтары

1. Сызықтық алгоритмдерден тармақталған алгоритмдер арасындағы айырмашылық қандай?
2. Неге барлық тапсырмаларды желілік алгоритмдер арқылы шешуге болады деп ойлайсыз?
3. Сіз қандай да бір бағдарламаны әзірлеу үшін сызықтық алгоритмдер мен филиалдар жеткілікті болады деп ойлайсыз ба?
4. Неге екі айнымалының мәндерін екі кезеңмен алмастыру мүмкін емес: a = b; b = a?
5. Шартты операторлардың толық және толық емес нысандардағы айырмашылығы қандай? Сіз қалай ойлайсыз? Сізге толық емес пішін қажет пе?
6. Сіз қандай қарым-қатынас білесіз? Қандай қатынастар «тең» және «тең емес» дегенді білдіреді?
7. Таңдауды бірнеше нұсқадан қалай ұйымдастыру керек?
8. Қандай қиын жағдай?
9. Есептеу процедурасы күрделі жағдайда қалай анықталады?

№4-5 ПРАКТИКАЛЫҚ САБАҚ

ЦИКЛДІК ЕСЕПТЕУ ПРОЦЕССТЕРІН ПРОГРАММАЛАУ ТАҚЫРЫП 3.1: ЦИКЛДІ АЛГОРИТМДЕР. FOR ПАРАМЕТРІ БАР ЦИКЛ

> Мақсаты: Циклдық алгоритмдерді енгізу үшін цикл операторын пайдалану параметрін үйрену.

Қысқаша теориялық мәлімет Параметрімен бірге цикл деп аталатын цикл үшін Python тіліндегі мүмкіндіктерге бай. For циклында айнымалыны іске қосатын айнымалыны және мәндердің жиынын көрсетесіз. Мәндер жиынтығы тізіммен, жолақпен, жолмен немесе ауқыммен көрсетілуі мүмкін. Міне, мәнерлер жиынтығы ретінде пайдаланылатын циклды пайдаланудың ең қарапайым мысалы:

```
i = 1
```

for color in 'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'violet':

```
    print('#', i, ' color of rainbow is ', color, sep = '')
```

i += 1 Бұл мысалда ауыспалы түс түсінде 'қызыл', 'апельсин' және т.б. мәндерді алады. Циклдің корпусында түстің атауын, яғни айнымалы түстің мәнін және циклдің иерархиялық нөмірін, 1-цифрды, содан кейін циклдің әр өтуімен біреуімен арта отырып, шығарылады. i = i + 1 (бұл жай ғана қысқартылған жазба) құрылысына тең. Мәндер тізімінде әртүрлі өрнектер болуы мүмкін, мысалы: for i in 1, 2, 3, 'one', 'two', 'three':

```
    print(i)
```

Алғашқы үштіктің итерациясы үшін i айнымалы i int типінің мәнін алады, ал келесі үшеуі - 'str' типі.

1. HYPERLINK "http://pythontutor.ru/lessons/for_loop/" \l "section_2"Функция range Әдетте, цикл үшін пайдаланылады немесе әрекеттер тізбегі бірнеше рет алдын ала анықталған саны қайталау, немесе финалға кейбір бастапқы құнынан циклде айнымалы мәнін өзгерту үшін. цикл уақытын алдын ала белгілі бір санын қайталау N бірге функциясымен, цикл ауқымы үшін пайдалануға болады: 0, 1, 2, 3 I үшін # балама нұсқаулар: диапазоны (4) Мен: # Мұнда циклдік

әрекеттерді орындауға болады басып шығару (i) басып шығару (i ** 2) # цикл аяқталды, себебі шегінісі бар блок аяқталды басып шығару («Циклдың соңы») сандық тұрақты ретінде N, айнымалыны немесе ерікті арифметикалық өрнекті (** 10 мысалы, 2) пайдаланылуы мүмкін. N құны нөлдік немесе теріс болса, цикл тіпті бір рет орындалмайды. Ауқым функциясы бір емес, екі параметрді қабылдай алады. Call диапазоны (а, б) индекстік айнымалы б а м ндерді өтеді дегенді білдіреді - 1 екі параметрлерімен деп аталатын, яғни бірінші параметр ауқымы функциясын, индексі айнымалы бастапқы мәні, екінші параметрі-орнатады - индекстік айнымалы болып табылады бірінші мән қабылданбайды. Егер a'b болса, цикл бір рет орындалмайды. Мысалы, 1-ден бастап n-ға дейінгі мәндерді есептеу үшін келесі бағдарламаны қолдануға болады: sum = 0 n = 5 for i in range(1, n + 1): sum += i print(sum) Бұл мысалда айнымалы i, 1, 2, ..., n мәндерін қабылдайды және айнымалы соманың мәні көрсетілген мәндер бойынша дәйекті түрде артады. Ақырында, индекстің айнымалысы төмендейтін циклді ұйымдастыру үшін ауқымды функцияны үш параметрмен қолдану қажет. Бірінші параметр индекс айнымалысының бастапқы мәнін анықтайды, екінші параметр - индекстік айнымалы өзгереді (оны қоспағанда!), Ал үшінші параметр - индекс айнымалы өзгерісінің сомасы. Мысалы, ауқымды (1, 100, 2) функциясын 1-ден 99-ға дейінгі барлық тақ сандар арқылы айналдыра аласыз және 100-ден 1-ге дейін барлық сандар бойынша циклды (100, 0, -1) қолдануға болады. I> a, i = a + d, i = a + 2 * d индекстік айнымалы мәндерінің мәндерін анықтайды және барлық мәндер үшін i < б. Егер d <0 болса, айнымалы айнымалы i> b барлық мәндерін қабылдайды.

Жұмыс жоспары Тапсырма 1. A және B екі бүтін сандары берілген (мұнда A ≤ B). Барлық нөмірлерді A-дан B-ге дейін басып шығарыңыз. Тапсырма 2. A және B екі бүтін сандар беріледі, A <B болса немесе басқа тәртіппен азайтылса, барлық сандарды A-дан B-ға дейін қоса алғанда шығарыңыз. Тапсырма 3. A және B екі бүтін сандары берілген, A> B. A-дан B-ге дейін барлық тақ нөмірлерді басып шығарыңыз. Бұл тапсырмада if if операторын пайдалана алмайсыз. Тапсырма 4. 10 бүтін сандар беріледі. Олардың сомасын есептеңіз. Тапсырма 5. Айнымалылардың ең аз санын қолданатын бағдарламаны жазыңыз. Тапсырма 6. Факторлық. N санының факторигі - бұл 1 × 2 × ... × n өнім. Белгілеу: n !. N бүтін сан үшін n мәнін есептеңіз. Бұл тапсырмада математиканы пайдалануға тыйым салынады.

Тақырып 3.2: Python программалау тілі. Циклдік құрылымның бағдарламаларын жасау.

> Мақсаты: While алгоритмін түрлі алгоритмдік тапсырмаларды қалай пайдалану керектігін үйрену.

Қысқаша теориялық мәлімет Уақыттық циклда сынақ шарты шындық болған кезде орындалатын әрекеттердің бірізділігін береді. Шарт конструкцияның корпусына жазылады және циклдің корпусының орындалуына дейін тексеріледі. Әдетте, цикл уақыт циклінің орындалу санын нақты мәнін анықтау мүмкін болмаған кезде қолданылады. Уақытша циклдың қарапайым жағдайда синтаксисі: ал жағдай: нұсқаулық блок Алғашқы цикл іске қосылғанда, алдымен шарт тексеріледі. Егер бұл жалған болса, онда цикл тоқтайды және басқару элементі келесі уақыт циклінің денесінің кейінгі нұсқасына ауысады. Егер шарты дұрыс болса, нұсқаулық орындалады, одан кейін шарт қайтадан тексеріледі, нұсқаулық орындалады. Бұл шарт шын болғанша жалғасады. Шарт жалған болып шыққаннан кейін, циклдің жұмысы аяқталып, циклнан кейін бақылау келесі нұсқаулыққа ауыстырылады. Мысал 1. 1-ден 10-ға дейінгі бүтін сандардың квадраттарын көрсету.

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

Бұл мысалда цикл ішіндегі i айнымалыны 1-ден 10-ға дейін өзгереді. Циклдің әрбір жаңа өтуі өзгеретін мәні осы контролер деп аталады. Айта кету керек, бұл фрагментті орындағаннан кейін i айнымалы мәні 11 болады, себебі i == 11 үшін i <= 10 шарты бірінші рет орындалуын тоқтатады.

```
2-мысал. N санының табиғи санының санын анықтаңыз:n = int(input())
kol = 0
```

while n > 0:

```
    n =n// 10   # это эквивалентно n = n // 10
    kol=ko l + 1
print(kol)
```

Бұл циклде біз бүтін бөлікке 10 (n = n // 10) тең деп саналатын, соңынан бастап санның бір санын алып тастаймыз, ал біз айнымалы борға қанша рет жасалды деп болжап отырмыз.

1. Циклді басқару нұсқаулары Циклдың корпусынан кейін басқа сөзді жаза аласыз: кейіннен цикл аяқталғаннан кейін бір рет орындалатын операциялар блогы, сынақ жағдайы дұрыс болмаған кезде: i = 1 while i <= 10: print(i) i += 1 else: print('Цикл окончен, i =', i) Бұл жерде ешқандай ұғым жоқ сияқты көрінеді, өйткені циклдің аяқталғаннан кейін ғана сол нұсқаулықты жазуға болады. Мән тек үзіліс үзіндісімен бірге пайда болады. Егер іске қосу уақытында, Python цикл ішінде үзіліс үзіндісін тапса, ол циклды бірден тоқтатады және оны шығарады. Бұл жағдайда басқа филиал орындалмайды. Әрине, үзіліс туралы мәлімдеме тек if операторында ғана айтылуы мүмкін, яғни, егер ол арнайы шарт орындалса ғана орындалуы керек. 3-мысал сандарды теріс санды тапқанша оқиды. Теріс сан пайда болған кезде бағдарлама тоқтайды. Бірінші нұсқада сандар тізбегі 0 санымен тоқтатылады (оны оқығанда тоқтату керек).a = int(input()) while a != 0: if a < 0: print('Встретилось отрицательное число', a) break a = int(input()) else: print('Ни одного отрицательного числа не встретилось') Бағдарламаның екінші нұсқасында бірізділіктегі элементтер саны алдымен кіріске енгізіледі, содан кейін элементтердің өзі. Бұл жағдайда циклды қолдану ыңғайлы. For циклында сондай-ақ, басқа тармақ болуы мүмкін және үзіліс туралы мәлімдемелер болуы мүмкін. n = int(input()) for i in range(n): a = int(input()) if a < 0: print('Встретилось отрицательное число', a) break else: print('Ни одного отрицательного числа не встретилось') Басқа циклді басқару бойынша нұсқаулық жалғасады. Егер бұл нұсқаулық циклдің ортасында орын алса, қалған нұсқаулар циклдің соңына өткізіп жіберіледі және цикл орындалуы келесі итерацияда жалғасады. Үзіліс және жалғастыру туралы мәлімдемелер бірнеше кірістірілген циклдарда болса, олар тек ішкі циклдің орындалуына әсер етеді. Бұл көрсететін ең ақылдылықтың үлгісі емес: for i in range(3): for j in range(5): if j > i: break print(i, j) Үзілістерге деген ұмтылыс және жалғастыру нұсқаулары оларды қолданбай-ақ жасалса, ынталандырылмайды. Міне, үзіліс операторының нашар қолдануының үлгі мысалы (бұл код сандағы таңбалардың санын есептейді). n = int(input()) length = 0 while True: length += 1 n //= 10 if n == 0: break print('Длина числа равна', length) Гораздо лучше переписать этот цикл так: n = int(input()) length = 0 while n != 0: length += 1 n //= 10 print('Длина числа равна', length) Дегенмен, Python-да сізге неғұрлым талғампаз шешім ұсынуға болады: n = int (енгізу ()) print ('Нөмірдің ұзындығы', len (str (n)))
2. Бірнеше тағайындау Python бағдарламасында бір тапсырма мәлімдемесі үшін бірнеше айнымалы мәнді бірден өзгертуге болады. Бұл келесідей: a, b = 0, 1 Бұл кодты да жазуға болады: a = 0 b = 1 Екі әдіс арасындағы айырмашылық, бірінші әдіске бірнеше тапсырма бір мезгілде екі айнымалы мәнін өзгертеді. Егер бірнеше тағайындағанда «=» белгісінің сол жағында үтірмен бөлінген айнымалы атаулар болуы керек болса, оң жақта үтірлермен бөлінген ерікті өрнектер болуы мүмкін. Ең бастысы, тапсырма белгісінің сол және оң жағында бірдей элементтер бар. Жұмыс жоспары Тапсырма 1. Квадраттардың тізімі. бүтін туралы N өсу тәртібімен, N-ден аспайтын оң бүтін сандар барлық квадрат басып шығарыңыз. Тапсырма 2. Таңертеңгі жүгіру. Бірінші күні спортшы туралы x шақырым жүгіріп, содан кейін ол алдыңғы құнынан 10% кем күн сайын іске арттырады. y санына іске спортшы y километрден кем емес, онда күн санын анықтау. Бағдарлама кірістегі нақты сандарды x және y алады және бір табиғи нөмірді шығару керек. Тапсырма 3. Банктің қызығушылығы. Банкке қосқан үлесі - рубль. Жыл сайын бұл көрсеткіш пайызбен өседі, содан кейін центтің бөлшек бөлігі жойылады. Анықтаңыз, қанша жылдан кейін жарна кем дегенде рубль болады. өрнек сіз 123.4567 рубль, яғни қосулы болса дегенді білдіреді «бөлшек бөлігі цент тасталады». Егер сіз 123 рубль және 45 копеек алуға дөңгелектеу кейін Е. 123 рубль және 45,67 цент, яғни 123.45 рубль болды. Бағдарлама үш табиғи сандарды алады: x, p, y және бір бүтінді шығару керек. Тапсырма 4. Тізбектің біркелкі элементтерінің саны. Түз элементтердің санын 0 санымен аяқталатын қатарда анықтаңыз. Тапсырма 5. Фибоначчи сандары. Фибоначчи реті келесідей анықталады: φ0 = 0, φ1 = 1, φn = φn-1 + φn-2. Берілген санға n, n-th Fibonaccі санын анықтаңыз. Бұл тапсырма цикл үшін шешілуі мүмкін.

№6-7 ПРАКТИКАЛЫҚ САБАҚ БІР ӨЛШЕМДІ ЖӘНЕ ЕКІ ӨЛШЕМДІ МАССИВТЕРДІ ӨҢДЕУ

Тақырып 4.1: тізімдер - айнымалы тізбектер

Мақсаты: тізімдерді жасауды үйрену, бірнеше тізімдерге қосылу, олардың индексі бойынша тізімдерге кіру, үзінділерді алу, тізім ұзындығын өлшеу.

Қысқаша теориялық мәлімет Периодты бағдарламалау тіліндегі тізімдер сияқты тізімдер реттеледі. Дегенмен, жолдарға қарағанда, тізімдер таңбалардан емес, түрлі нысандардан (мәндерден, деректерден) тұрады және тырнақша жақшаларымен [] тырнақшалармен бірге қойылады. Объектілер бір-бірінен үтірмен бөлінеді. Тізімдер түрлі нысандардан тұруы мүмкін: сандар, жолдар, тіпті басқа тізімдер. Соңғы жағдайда тізімдер ұя салған деп аталады.

Тізімге мысалдар:

```
[23, 656, -20, 67, -45]             # бүтін сандар тізімі
[4.15, 5.93, 6.45, 9.3, 10.0, 11.6]     # бөлшектік сандар тізімі
["Katy", "Sergei", "Oleg", "Dasha"]     # қатарлар тізімі
["Москва", "Титова", 12, 148]     # аралас тізім
[[0, 0, 0], [0, 0, 1], [0, 1, 0]]         # тізімдерден құралған тізім
```

Жұмысты орындау тәртібі Аудармашыдағы тапсырмаларды орындаңыз. Жоғарыда келтірілген командаларды енгізіп, жауапты жазыңыз. Тапсырма 1.

```
>>> c = [45, -12, 'april']
>>> d= [21, 48.5, 33]
>>> print(c+d)
```

Жауабы: _____

Тапсырма 2.

```
>>> spisok=[[0,0],[0,1],[1,1]]
>>>spisok=spisok*2
>>>print(spisok)
```

Жауабы: _____ Жол символдарымен ұқсастығымен тізімдердің нысандарына индекстер бойынша қол жеткізуге, үзінділерді шығаруға, тізім ұзындығын өлшеуге болады:

Тапсырма 3.

```
>>> li = ['a','b','c','d','e','f']
>>> len(li)
```

Жауап: ____

```
>>> li[0]
```

Жауап: ____

```
>>> li[4]
```

Жауап: ____

```
>>> li[0:3]
```

Жауап: _____

```
>>> li[3:]
```

Жауап: _____ Жолдардан айырмашылығы, тізімдер айнымалы тізбелер болып табылады. Егер сіз жадыдағы нысан ретінде жолды көрсетсеңіз, онда оған конкатенация және қайталау операциялары орындалса, бұл жол өзгермейді және операцияның нәтижесінде басқа жады басқа орынға жасалады. Жолға жаңа таңба қосу мүмкін емес немесе жаңа жолды жасамай, барын жоюға болмайды. Жағдай тізіммен ерекшеленеді. Операцияларды орындау барысында басқа тізімдер жасалмауы мүмкін, бірақ түпнұсқаны тікелей өзгертуге болады. Тізімдерден элементтерді жоюға, жаңадан қосуыңызға болады. Есіңізде болсын, айнымалы мәндерді басқаруға байланысты көп нәрсе. Тізімдер әлі көшірілсе, жағдайлар бар. Мысалы, операцияның нәтижесі басқа айнымалыға тағайындалады.

Тапсырма 4. Жолдағы таңбаны өзгерту мүмкін емес, тізім элементі болуы мүмкін:

```
>>> mystr = 'abrakadabra'             #Задана строка
>>> mylist = ['ab','ra','ka','da','bra']    # Задан список
>>> mystr[3] = '0'            #Пытаемся поменять строку, получаем ошибку
Traceback (most recent call last):
File "<pyshell#11>", line 1, in <module>
mystr[3] = '0'
```

TypeError: 'str' object does not support item assignment

```
>>> mylist[1] = 'ro'            # Меняем список
>>> mylist               #Выводим новый список
```

Ответ: _____ Тізімде сіз барлық тілдің орнын ауыстыра аласыз:

```
>>> mylist[0:2] = [10,20]
>>> mylist
```

Ответ: _____ Тапсырма 5. 4 тапсырмадан тізімдермен жұмыс істеуді жалғастырыңыз Неғұрлым күрделі жағдай:

```
>>> alist = mylist[0:2] + [100,'it is ',200] + mylist[2:]     # новый список
>>> a2list = mylist     # создается вторая ссылка–переменная на  первый список
>>> alist
```

Ответ: _____

```
>>> a2list
```

Ответ: _____

```
>>> a2list[0] = '!!!'         # изменяем список
>>> a2list
>>> mylist
```

Ответ: _____ Mylist және a2list тізімдерін жасаңыз Жұмыс жоспары: 1-5 тапсырмаларды орындаңыз. Екі тізімді жасаңыз және оларды айнымалылармен байланыстырыңыз. Бірінші тізімнен екінші тармақты шығарып алыңыз. Екінші тізімдегі соңғы нысанды өзгертіңіз. Тізімді экранда көрсету. Нәтижені жаңа айнымалыға тағайындау, екі тізімді біреуіне қосыңыз. Алынған тізімді экранға шығарыңыз. Қос тізімдердің кейбір бөліктеріне жету үшін байланыстырылған тізімнен б лікті «тазарту». Кесуді жаңа келесі айнымалыға қосыңыз. Бұл айнымалы мәнді басып шығарыңыз. Слайерлер тізіміне екі жаңа элемент қосып, оны қайтадан шығарыңыз.

Тақырып 4.2: Python программалау тілі. Тізімдермен жұмыс істеу.

Мақсаты: тізімдегі элементтерді ауыстыру үйрену түрлі жолдармен тізімін кіріс және шығыс элементтері әр түрлі жолдармен тізіміне жаңа элементтерді қосу, әдістерін пайдалана отырып және ыдырау joine, тізім түсіну пайдаланып тізімдерін жасау, кездейсоқ сандардың генераторы үшін.

Қысқаша теориялық мәлімет Көптеген бағдарламалар жеке айнымалы, және айнымалы жиынтығымен жұмыс істемейді. Мұндай деректерді сақтау үшін деректер құрылымын пайдалануға болады (пайдаланылатын басқа да көптеген, термин «жиым» бағдарламалау тілі) бір Python тізімін деп аталады. Тізімі, 0-ден нөмірленген, жолдың рәміздер элементтердің жүйелілігі болып табылады. Сіз төмендегідей мысалы, тізім орнатуға болады, тік жақшада элементтердің аудару тізіміне тізімін көрсетуге болады:

```
Primes = [2, 3, 5, 7, 11, 13]
Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']
```

В списке Primes — 6 элементов, а именно: Primes[0]==2, Primes[1] == 3, Primes[2] ==5, Primes[3] == 7, Primes[4] == 11, Primes[5] == 13. Rainbow тізімі 7 элементтен тұрады, олардың әрқайсысы жол. Жолдағы таңбалар сияқты, тізім элементтері соңынан теріс сандармен индекстелуі мүмкін, мысалы: Primes [-1] == 13, Primes [-6] == 2. Тізімнің ұзындығы, яғни оның элементтерінің саны len () функциясымен, мысалы, l en (Primes) == 6 арқылы білуге болады. Жолдардан айырмашылығы, тізім элементтерін өзгертуге болады, оларға жаңа мәндерді тағайындау. Мысал 1. Тізім элементтерін ауыстыруRainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']

```
print(Rainbow[0])
Rainbow[0] = 'красный'
print('Выведем радугу')
for i in range(len(Rainbow)):
    print(Rainbow[i])
```

Тізімдерді жасау мен оқудың бірнеше жолын қарастырайық. Ең алдымен, сіз бос тізім жасай аласыз (құрамында элементтер жоқ, ұзындығы 0) және тізімнің соңында қосымшаны қосу әдісі арқылы элементтерді қосуға болады. Мысалы, бағдарлама n тізіміндегі элементтер санының енгізілуін, ал содан кейін тізімдегі n элементтерін бөлек жолда бірдей алыңыз. Мұнда осы форматта кіріс деректерінің мысалы келтірілген 5 1809 1854 1860 1891 1925 Бұл жағдайда тізімді оқуды келесідей ұйымдастыруға болады: Мысал 2. Элементтерді тізімге қосу

```
a = [ ]                 # бос тізімді енгіземіз
n = int(input())              # тізімдегі элемент санын енгіземіз
for i in range(n):
    new_element = int(input())      # келесі элементті енгіземіз
    a.append(new_element)          # оны тізіме қосамыз
 print(a)
```

Бұл мысалда бос тізім жасалады, одан кейін тізімдегі элементтер саны оқылады, содан кейін бір тізім элементі бір уақытта оқылады және оның соңына қосылады. N айнымалысын сақтау арқылы жазылуы мүмкін:

```
a = [ ]
for i in range(int(input())):
    a.append(int(input()))
print(a)
```

Тізімдер үшін келесі әрекеттер толығымен анықталған: тізімдерді біріктіру (тізімдерді қосу, басқалардың тізімін тағайындау) және қайталанатын тізімдер (тізімді санға көбейту). 3-мысал. Тізімді біріктіру

```
a = [1, 2, 3]
b = [4, 5]
c = a + b
d = b * 3
```

```
print([7, 8] + [9])
print([0, 1] * 3)
```

Нәтижесінде c тізімі [1, 2, 3, 4, 5] болады және d тізімі [4, 5, 4, 5, 4, 5] болады. Бұл тізімдерді басқа жолмен оқу процесін ұйымдастыруға мүмкіндік береді: алдымен тізімнің өлшемін қарастырып, қажетті элементтердің тізімін жасаңыз, содан кейін 0 санынан бастап айнымалы i циклын ұйымдастырыңыз және тізбектің i-ші элементі цикл ішінде оқылады:

```
a = [0] * int(input())
for i in range(len(a)):
    a[i] = int(input())
```

Тізім элементін тізім элементінің төртбұрыш жақшаларымен және тізім элементтері арасында үтірлермен бірге басып шығаруға болады (a). Мұндай қорытынды ыңғайсыз, жиі бір жолда немесе тізімдегі бір элементтің барлық тізім элементтерін шығару қажет. Міне, екі мысал, сонымен қатар, цикл ұйымдастыру: Мысал 4. Тізім элементтерінің индексі бойынша тізімі

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

Мұнда i элементінің индексі циклде өзгереді, одан кейін i индексі бар тізім элементі шығады. Мысал 5. Элементтерді тізімге енгізіңіз

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

for elem in a:

```
    print(elem, end=' ')
```

Бұл мысалда тізім элементтері бос орынмен бөлінген бір жолда көрсетіледі. Бұл жағдайда тізбе элементінің индексі циклде өзгермейді және айнымалы мәннің мәні (мысалы, «қызыл», «жасыл», «көк» элемент үшін циклде айнымалы элемент дәйекті «қызыл» мәндерін қабылдайды, «жасыл», «көк». Соңғы мысалға ерекше назар аударыңыз! Python идеологиясының өте маңызды бөлігі - кез-келген жүйенің барлық элементтерін сұрыптаудың ыңғайлы әдісін беретін цикл. Бұл Python мен Pascal арасындағы айырмашылық, онда элементтердің өздері емес, элементтердің көрсеткіштері бойынша сұрыптау керек. Python жүйесінде тізімдер, тізімдер, функция ауқымының мәндері (бұл тізімдер емес) және кейбір басқа нысандар. Мұнда барлық сандар жолдан таңдалатын және массивке сандар ретінде қосылатын жағдайдағы 'for' циклын пайдалануды көрсететін мысал. Мысал 6. # дано: s = 'ab12c59p7dq' # надо: извлечь цифры в список digits, # чтобы стало так:

```
# digits == [1, 2, 5, 9, 7]

s = 'ab12c59p7dq'
digits = []
```

for sym in s:

```
    if  '1234567890'.find(symbol) != -1:
        digits.append(int(symbol))
print(digits)
```

1. «Бөлу» және «қосылу» әдістері Тізімнің элементтерін бір жолға бір жолмен енгізуге болады, бұл жағдайда бүкіл жол кіріс () функциясы ретінде қарастырылуы мүмкін. Содан кейін, split () әдісін қолдануға болады, ол бастапқы жолды бөліктерге бөлінгенде алынатын жолдардың тізімін қайтарады. Мысал: Мысал 7. Түпнұсқа жолды бөлек таңбаларға

бөліп, жаңа тізім жасаңыз
енгізу - бұл жол # 1 2 3 s = input() # s == '1 2 3' a = s.split() # a == ['1', '2', '3'] Егер сіз осы бағдарламаны бастаған кезде 1 2 3 жолын енгізсеңіз, тізім a ['1', '2', '3'] болады. Тізім сандар емес, жолдардан тұратынын ескеріңіз. Сандардың тізімін алғыңыз келсе, тізім элементтерін бір-бірлеп нөмірлендіруге болады: Мысал 8. Түпнұсқаны бөлек таңбаларға бөліп, сандардан тұратын жаңа тізім жасаңыз a = input().split() for i in range(len(a)): a[i] = int(a[i]) Python-генераторларының ерекше сиқырын пайдалану - бір жолмен жасалуы мүмкін: Мысал 9. 9-мысалға қысқаша жазба. a = [int(s) for s in input().split()] Егер нақты сандардың тізімін оқығыңыз келсе, int типін float түрімен ауыстырыңыз. Split () әдісі тізім элементі арасындағы қай бөлікті бөлгіш ретінде пайдаланылатындығын анықтайтын қосымша параметрге ие. Мысалы, Split («.») Деп аталатын әдіс бастапқы жолды '.' Арқылы кесу арқылы алынған тізімді қайтарады. Белгілер: a = '192.168.0.1'.split('.') Python бағдарламасында бір жолды пәрменді пайдалану арқылы жолдардың тізімін көрсетуге болады. Мұны істеу үшін біріктірілген жол әдісін қолданыңыз. Бұл әдіс бір параметрге ие: жолдардың тізімі. Нәтижесінде, жол қайтарылады, ол берілген тізімді элементтерді бір жолға қосу арқылы алынады, ал бөлгіш әдіс қолданылатын сызыққа тең тізім элементтері арасында енгізіледі. Келесі мысалдарды қарастырайық: Мысал 10. Саптарды сепараторды оса отырып біріктіруа = ['red', 'green', 'blue'] print(' '.join(a)) # вернёт red green blue print(''.join(a)) # вернёт redgreenblue print('***'.join(a)) # вернёт red***green***blue

1. Генераторлар тізімі Бірдей элементтермен толтырылған тізімді жасау үшін, тізімді қайталау операторын пайдалануға болады, мысалы: n = 5 a = [0] * n Күрделі формулалармен толтырылған тізімдерді жасау үшін сіз генераторларды пайдалана аласыз: формуламен тізімді толтыруға мүмкіндік беретін өрнектер. Генератордың жалпы көрінісі келесідей: [айнымалы мәндегі өрнек ретпен] айнымалы қайда - айнымалы идентификаторы, бір тізбегі - айнымалыны алады құндылықтар тізбегі, (бұл тізім, жол немесе нысан ауқымы функциясын пайдаланып алуға болады) өрнек - бір өрнек, оның айнымалы генератор, әдетте, тәуелді пайдаланылады толтырылған тізім элементтері. Мұнда генераторлар пайдалану бірнеше мысалдар болып табылады.
   N нөлдер тұратын тізімін жасау және сіз генераторын пайдалануға болады:a = [0 for i in range(5)] Мынадай бүтін сандардың квадраттарымен толтырылған тізімді жасаңыз: n = 5 a = [i ** 2 for i in range(n)] Если нужно заполнить список квадратами чисел от 1 до n, то можно изменить параметры функции range на range(1, n + 1): n = 5 a = [i ** 2 for i in range(1, n + 1)] Міне, сіз 1-ден 9-ға дейінгі кездейсоқ сандармен толтырылатын тізімді ала аласыз (кездейсоқ модульден randrange функциясын пайдалану арқылы): from random import randrange n = 10 a = [randrange(1, 10) for i in range(n)] Бұл мысалда тізім стандартты енгізуден оқылған жолдардан тұрады: алдымен тізім элементтерінің санын енгізу керек (бұл мән ауқым функциясына аргумент ретінде пайдаланылады), содан кейін көрсетілген жолдар саны: a = [input() for i in range(int(input()))]

Жұмыс жоспары Тапсырма 1. Тіпті индекстер. Тізім орнатылды. Тізімнің барлық элементтерін тіпті индекстермен шығару (яғни A [0], A [2], A [4], ...). Тапсырма 2. Тіпті элементтер Тізімнің барлық элементтерін шығарады. Бұл ретте тізімдегі элементтерді тізімдеу үшін, олардың индекстерін емес, циклін пайдаланыңыз! Тапсырма 3. Алдыңғыдан көп. Сандардың тізімін ескере отырып. Алдыңғы элементтен үлкен тізімнің барлық элементтерін шығарыңыз. Тапсырма 4. Сол белгінің көршілеріне. Сандардың тізімін ескере отырып. Егер сол белгісінің екі көршілес элементтері болса, осы нөмірлерді басып шығарыңыз. Егер көрші элементтердің бірдей белгісі болса - ештеңе шығармайды. Егер бірнеше жұптар көршілес болса, бірінші жұпты шығарыңыз. Қосымша: Тапсырма 5. Көршілерінен көп. Сандардың тізімін ескере отырып. Осы тізімдегі қанша адам өздерінің көршілерінен екіншісінен артық екенін анықтаңыз және осындай элементтердің санын шығарыңыз. Тізімнің аса маңызды элементтері ешқашан назарға алынбайды, өйткені олардың көршілері жеткілікті емес. Тапсырма 6. Ең үлкен элемент Сандардың тізімін ескеріңіз. Тізімдегі ең үлкен элементтің мәнін, одан кейін тізімдегі сол элементтің индексін шығарыңыз. Ең үлкен элементтерден артық болса, бірінші индексті шығарыңыз. Тапсырма 7. Петя басқа мектепті ауыстырды. Дене тәрбиесі сабағында ол өз орындарында өз орнын анықтауға тура келді. Оған көмектесу. Бағдарлама әр адамның қалыптасуы үшін өсуін білдіретін табиғи сандардың тұрақты емес реттілігін алады. Осыдан кейін X саны енгізілді - Petit өсуі. Кіріс деректеріндегі барлық сандар табиғи және 200-ден аспайды. Петя желісі бойынша нөмірді алу керек нөмірді шығарыңыз. Егер сатыларда Петя сияқты бірдей өсім бар адамдар болса, онда ол кейін олардан тұруы керек. Тапсырма 8. Ең мин және макс. Тізімде барлық элементтер әртүрлі. Осы тізімнің ең аз және ең жоғарғы элементтерін ауыстырыңыз.

Тақырып 4.3: Сөздіктермен жұмыс істеу

> Мақсаты: күрделі деректер құрылымымен жұмыс істеуді үйрену - сөздіктер: жасау, өңдеу, сүзу, сұрау бойынша деректерді таңдау.

Қысқаша теориялық мәлімет Жай тізімдер (массивтер) - нөмірленген элементтер жиынтығы, яғни тізімнің кез келген элементіне сілтеме жасау үшін оның нөмірін көрсету керек. Тізімдегі элемент нөмірі элементінің өзін анықтайды. Бірақ деректерді сандық нөмірлер бойынша анықтау әрдайым ыңғайлы емес. Мысалы, Ресейде бағыттар бойынша пойыздар рейстерді анықталған сандық-хат коды (нөмірі және бір нөмірі), және сан-хат коды анықталған, бұл мәтіндік жол ретінде сандарды қолдануға болмайды еді ыңғайлы идентификатор ретінде рейстер поездарда немесе ұшақтар туралы ақпаратты сақтау болып табылады. Деректер құрылымы, оның элементтерін сандық индекс бойынша емес, еркін индексі арқылы анықтауға мүмкіндік береді, ол сөздік немесе қауымдастық массив деп аталады. Python деректерінің құрылымы dict деп аталады.

1. Сөздікті пайдаланудың қарапайым мысалын қарастырайық. Біз астананың сөздіктерін бастайық, онда индекс елдің аты болып табылады және мағынасы - осы ел астанасының атауы. Бұл ел атымен астанасы бар желімен анықтауға мүмкіндік береді. Мысал 1. # Создадим пустой словать Capitals Capitals = dict() # Заполним его несколькими значениями Capitals['Russia'] = 'Moscow' Capitals['Ukraine'] = 'Kiev' Capitals['USA'] = 'Washington' Countries = ['Russia', 'France', 'USA', 'Russia'] for country in Countries: # Для каждой страны из списка проверим, есть ли она в словаре Capitals if country in Capitals: print('Столица страны ' + country + ': ' + Capitals[country]) else: print('В базе нет страны с названием ' + country) Осылайша, сөздің әрбір элементі екі нысаннан тұрады: кілт және мән. Біздің мысалда кілт - елдің аты, мағынасы - астананың атауы. Кілт сөздік элементін анықтайды, мән - бұл кілтке сәйкес келетін деректер. Негізгі мәндер бірегей, сөздікте екі бірдей кілттер болмайды. Мұндай әдеттегі қағаз сөздіктер (біртілді, емле, тіл) ретінде өмір таралған сөздіктер, жылы. Онда кілт - мақаланың сөздік тақырыбы және құндылығы мақаланың өзі. Мақалаға қол жеткізу үшін сөздік кілтін көрсету керек. Деректер құрылымы ретінде сөздіктің тағы бір мысалы - телефон анықтамалығы. Онда кілт аты болып табылады, ал мағынасы - телефон нөмірі. Ал сөздіктер, және телефон анықтамалығы ол белгілі негізгі лексиканы (жазбалар кілттер әліпбилік ретпен сақталады, егер, мысалы, сіз оңай мысалы сондай-ақ белгілі пернесін, екілік іздеу таба аласыз) элементін табу оңай, сондықтан сақталады, бірақ тек құнына белгілі болса neizvestven кілті, берілген мәнімен іздеу элемент сөздіктің барлық дәйекті сканерлеу элементтері талап етуі мүмкін. Ассоциативті массивтің ерекшелігі оның динамизмі: ол еркін кілттермен жаңа элементтерді қосып, бар элементтерді жоя алады. Қолданылатын жадтың өлшемі ассоциативті массаның өлшеміне пропорционалды. Ассоциативті алаптың элементтеріне қатынау қарапайым массивтерден гөрі баяу болса да, бірақ тұтастай алғанда тезірек орындалады. Python тілінде кілт еркін өзгермейтін деректер түрі болуы мүмкін: бүтін сандар және нақты сандар, жолдар, қапсырмалар. Сөздікдегі кілт жиынтық бола алмайды, бірақ ол frozenset түрінің элементі бола алады: құрастырылғаннан кейін өзгертілмейтін түрдегі жиынтықтың аналогы болып табылатын арнайы деректер түрі. Сөздік элементінің мәні айнымалы қоса алғанда, кез келген деректер түрі болуы мүмкін.
2. Сөздіктерді қашан қолдану керек? Сөздіктер келесі жағдайларда пайдаланылуы керек:
3. Объектілер санын есептеу. Бұл жағдайда сізде сөздердің болуы керек, онда кілттер нысан болып табылады, ал мәндер - олардың саны.
4. Объектіге қатысты кез-келген деректерді сақтау. Кілттер - нысандар, мәндер олармен байланысты деректер. Мысалы, айдың атауымен оның сериялық нөмірін анықтағыңыз келсе, оны Num ['January'] = 1 сөздік арқылы орындауға болады; Сан ['ақпан'] = 2; ....
5. Нысандар арасында сәйкестікті орнатыңыз (мысалы, «ата-бала»). Кілт объект болып табылады, мән оған сәйкес келетін нысан.
6. Егер сізге тұрақты жиым қажет болса, бірақ элемент индексінің ең үлкен мәні өте үлкен, бірақ мүмкін индекстерді («сирек массив» деп аталатын) емес, жадты сақтау үшін ассоциативті жиынды қолдануға болады.

1. Сөздік жасау Бос сөздік dict () функциясымен немесе бос жақшалармен {} (осы себепті бұйра жақшалар бос жиынтығын жасау үшін пайдаланыла алмайды) көмегімен жасалуы мүмкін. Бастапқы мәндердің кейбір жиынтығымен сөздікті жасау үшін келесі конструкцияларды пайдалануға болады: Мысал 2. Capitals = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington'} Capitals = dict(Russia = 'Moscow', Ukraine = 'Kiev', USA = 'Washington') Capitals = dict([("Russia", "Moscow"), ("Ukraine", "Kiev"), ("USA", "Washington")]) Capitals = dict(zip(["Russia", "Ukraine", "USA"], ["Moscow", "Kiev", "Washington"])) print(Capitals) Алғашқы екі әдіс тек шағын сөздіктерді жасау үшін пайдаланылуы мүмкін, олардың барлық элементтері. параметрлері атындағы функцияларды араласу сияқты Сонымен қатар, екінші әдісі, кілттер беріледі, сондықтан бұл жағдайда, кілттер ғана идентификаторлары дұрыс, онда жолдары, болуы мүмкін. Үшінші және төртінші оқиғасы дәлелдер алуға болады дайын тізімдер өтуге, егер сіз, үлкен сөздіктер жасауға болады міндетті барлық элементтерін тасымалдауға және бағдарламасын орындау барысында салынған кез келген басқа құралы болып табылады. негізгі және мәні: берілетін үшінші әдісі DICT функциясы тізімінде әр элементі екі элементтерінің кортеж болып табылады. пернелер тізімі мен құндылықтардың тізімі: төртінші әдісі екі тең ұзындығы тізімін алады funktsiyazip пайдаланады.

2. Сөздік элементтерімен жұмыс істеу Негізгі операция: кілттің көмегімен элементтің мәнін алу тізімдерге ұқсас: A [key]. Егер көрсетілген кілтдегі элемент сөздікте болмаса, KeyError ерекше жағдай орын алады. Кілт бойынша мәнді анықтаудың тағы бір жолы - алу әдісі: A.get (кілт) Егер алу кілтімен элемент сөздікте болмаса, онда Жоқ мәні қайтарылады. емес сөздікте негізгі элементі негізгі, егер екі дәлелдер A.get (негізгі, Val) әдісімен белгілер мән Val қайтарады. Сіз операцияларда емес, сондай-ақ жиынтықтардағы сөздіктің мүшелігін тексере аласыз. Сөздікке жаңа элемент қосу үшін, оны кейбір мәнді тағайындаңыз: A [перне] = мән.
   Элементті сөздіктен жою үшін, әрекетті пайдалануыңызға болады
   Мұндай негізгі сөздікте болмаса, егер дель (A [негізгі] операция, ерекше KeyError көтереді. сөздіктен элементін жою үшін Мұнда екі қауіпсіз әдісін. 3-мысал.A = {'ab' : 'ba', 'aa' : 'aa', 'bb' : 'bb', 'ba' : 'ab'} key = 'ac' if key in A: del A[key] try: del A[key] except KeyError: print('There is no element with key "' + key + '" in dict') print(A) Бірінші жағдайда, біз алдымен элементтің болуын тексереміз, ал екінші жағдайда біз қоспаны өңдейміз және өңдейміз. Сөздіктен элементті жоюдың тағы бір жолы - pop әдісі: A.pop (keyy). Бұл әдіс, егер осы пернемен элемент сөздікте болмаса, алып тасталатын элементтің мәнін қайтарады, ерекшелік тасталады. Егер екінші әдіс поп әдісіне берілсе, онда элемент сөздікте болмаса, pop әдісі осы параметрдің мәнін қайтарады. Бұл сөздіктен элементті қауіпсіз шығарып алудың ең оңай жолы: A.pop (кілт, жоқ).
3. Сөздік элементтерін іздеу Сөздіктегі барлық элементтердің кілттерін іздеуді ұйымдастыру оңай: 4-мысал. A = dict(zip('abcdef', list(range(6)))) for key in A: print(key, A[key]) Келесі әдістер сөздік элементтерінің көріністерін қайтарады. Өкілдер көптеген жолдармен ұқсас, бірақ олар сөздік элементтерінің мәндерін өзгерткен кезде өзгереді. Кілттер әдісі барлық элементтердің кілттерінің көрінісін қайтарады, мәндер әдісі барлық мәндердің көрсетілуін қайтарады және элементтер әдісі кілттер мен мәндердің барлық жұбының (түймелерінің) көрінісін қайтарады. Тиісінше, сөздік A элементінің барлық мәндерінің арасында VAL мәндерінің төмендегіні тексеруге болады: valval in A.values () және айнымалы мәнді айнымалы айнымалы элементтің кілті бар және val айнымалы мәнінде оның мәні болуы мүмкін:

Мысал 5.

```
A = dict(zip('abcdef', list(range(6))))
for key, val in A.items():
    print(key, val)
```

Жұмыс жоспары Тапсырма 1. 1-5 мысалдар келтіріңіз Тапсырма 2. Сөздің көрінісінің саны. Тек жол мәтінді қамтиды. Осы мәтіннің әрбір сөзі үшін, осы мәтінде бұрын қанша рет табылғанын есептеңіз. Сөз - дәйекті емес таңбалардың дәйектілігі, сөздер бір немесе бірнеше бос орындармен немесе соңғы сызық таңбаларымен бөлінеді. Тапсырма 3. Синонимдердің сөздігі. Сізге жұп сөздерден тұратын сөздік беріледі. Әр сөз - оған жұп сөздер синонимі. Сөздікдегі барлық сөздер басқаша. Соңғы жолда жазылған сөздіктен алынған сөз үшін синонимді анықтаңыз.

№8-9 ПРАКТИКАЛЫҚ САБАҚ

ЖОЛДЫҚ АЙНЫМАЛЫЛАРДЫ ҚОЛДАНЫП ЕСЕПТЕРДІ ШЫҒАРУ

Тақырып 5.1: Python бағдарламалау тілі. Жол деректерімен жұмыс істеу

> Жұмыстың мақсаты: жолды белгілеу, жолдың бөлігін алу, берілген ішкі жолды іздестіру, берілген ішкі жолдың жолында индексті анықтау, берілген ішкі жолды жою.

Қысқаша теориялық мәлімет

```
Жол inputt () функциясынан стандартты енгізуден оқылады.
```

Екінші жолда, қосу (конкатенация) операциясы анықталып, жолды санға көбейту операциясы да анықталған. Жол символдар тізбегінен тұрады. Len функциясын таңбалардың санын (жол ұзындығы) табу үшін пайдалануға болады.

1. Python-да кез келген басқа объект оған сәйкес келетін жолға аударылуы мүмкін. Мұны істеу үшін () функциясын шақырыңыз, оны жолға аударылған нысанды дәлел ретінде жіберу. Шын мәнінде, Python тұрғысынан әр жол, str классы. Басқа кластың басқа нысанының нысанын алу үшін оған қандай да бір жолмен жауап беру функциясын пайдалануға болады. Бұл функцияның атауы объектіге сілтеме жасайтын класстың аты сияқты. (Білгіштер үшін: бұл функция осы класстың нысандарының конструкторы.) Мысал: int - бүтін сандар үшін класс. Мысал 1. Жолды санға аудару s = input() print(len(s)) t = input() number = int(t) u = str(number) print(s * 3) print(s + ' ' + u)
2. Қабыршақтар Бөлік (кесінді) - фрагментке немесе подпоследовательности осы жолдың алу бір таңбаны немесе подстрока. Үш пішінді кесектер бар. Кесу қарапайым түрі: бір-желісі сипатын ескере отырып, атап айтқанда, S [I] - бірқатар Мен бар бір сипаттағы, тұратын б лік болып табылады. S = 'Hello', содан кейін S [0] == 'H', S [1] == «e», S [2] == 'L', егер ол, нөмірлеу Яғни санының 0. басталады деп көзделіп отыр , S [3] == 'L', S [4] == 'O'.
   Python бір түрі жол емес екенін ескеріңіз. жол түрі, сондай-ақ көш табылады - [I] кесінді S нәтижесінде алынған Әрбір объект. жолда бір таңбалар Numbers (сондай-ақ басқа да деректер құрылымын: тізімдері, луын) индексі деп аталады. Сіз индексі теріс мән көрсетпесеңіз, саны саны 1 бастап, соңына түскен есептеледі болады. Яғни, S [-1] == 'O', S [-2] == 'L', S [-3] == 'L', S [-4] == «e», S [-5] == 'H'.

Немесе мынадай кесте түрінде: Строка S H e l l o Индекс

```
S[0]
S[1]
S[2]
S[3]
S[4]
```

Индекс

```
S[-5]
S[-4]
S[-3]
S[-2]
S[-1]
```

Егер S-line бөліміндегі таңба саны LEN (S) немесе одан жоғары мәнге тең болса, немесе -len (S) мәнінен аз болса, ИндексError: Индекс индексі қателіктерден тыс болады, егер бұл жол қолжетімді болса. Екі параметрлерімен бөлік: S [a: б] б үзіндісін қайтарады - сипаты C indeksoma басталатын таңбаларды, яғни оған қоса алғанда емес символы индексі б, үшін. Мысалы, S [1: 4] == 'ell', сол кезде S [-4: -1] жазады. Ол, мысалы, бір бөлімінде оң және теріс екі кодтарын пайдалану S болады [1: -1] - бірінші және соңғы символы жоқ жол (б лік символы индексі 1 басталады және 1 индексінің, бірақ, оның ішінде емес аяқталады).
Егер қате түрін пайдалансаңыз, IndexError ешқашан орын алмайды. (Желілік 100-ден астам таңба болмаса): [100 1]: Мысалы, б лік S [1 5], екінші индексі өте үлкен жасауға Егер 'Ello «жол оралады, нәтиже мысалы, S, бірдей болады. Екінші параметрді өткізіп тастасаңыз (бірақ қос нүктені қойыңыз), онда тілек сызықтың соңына алынады. Мысалы, жолдан бірінші таңбаны (оның индексі 0) алып тастау үшін S [1:] тілімшесін қабылдауға болады. Сол сияқты, егер сіз бірінші параметрді алып тастасаңыз, жолдың басынан б лік алай аласыз. Яғни, S [: - 1] тілімшесін пайдаланып, соңғы таңбаны сызықтан алып тастай аласыз. S [:] тілінің қисық сызығы S.
Жолдың қимасы бар кез-келген операциялар жаңа жолдар жасайды және бастапқы жолды ешқашан өзгертпейді. Python-да желілер әдетте өзгермейді, оларды өзгерту мүмкін емес. Сіз ескі айнымалыға жаңа жолды ғана тағайындай аласыз.
Шындығында, питонда ешқандай айнымалылар жоқ. Кез келген нысандармен байланысты тек атаулар бар. Алдымен атыңызды бір нысанмен байланыстыра аласыз, содан кейін басқа. Бірнеше атауды сол нысанмен байланыстыруға болады. Үш параметрлер S бар болса орнатыңыз кесінді [A: B: D], үшінші параметр қадам белгілейді, іс funktsieyrange сияқты, яғни индекстерінің a + D, A + 2 * D, т.б. таңбаларды қабылданады ... Сіз үшінші параметр мәні б лігіне, 2 тең орнатылған кезде қажы екінші сипаты алады, және сіз -1 тең жарығы сөнген мәні алуға болса, онда таңбалар кері енеді. Мысалы, сіз S [:: - 1] тілімшесі бар сызықты аудара аласыз.

2-мысал. Нәтижесінде бағдарлама не шығарады?

```
s = 'abcdefg'
print(s[1])
print(s[-1])
print(s[1:3])
print(s[1:-1])
print(s[:3])
print(s[2:])
print(s[:-1])
print(s[::2])
print(s[1::2])
print(s[::-1])
```

Кесудің үшінші параметрі ауқымның range() функциясының үшінші параметріне қалай ұқсас екенін ескеріңіз:

```
Мысал 3. Диапазонды () функциясын пайдалану. Бағдарлама қандай болады?
s = 'abcdefghijklm'
print(s[0:10:2])
for i in range(0, 10, 2):
    print(i, s[i])
```

1. әдістері Әдіс - бұл объектке қолданылатын функция, бұл жағдайда жол. Бұл әдіс деп аталады ObjectName. MethodName (параметрлер) Мысалы, S.find («е») «е» параметрінің бір параметрімен іздеу әдісінің S жолына қолданба. 3.1. Әдістерді табыңыз және жасаңыз Табылған әдіс берілген жолдағы осы әдісті (ол әдіс қолданылатын) табады. Функция қажетті ішкі жолдың бірінші пайда болу индексін қайтарады. Егер подстрок табылмаса, әдіс -1 қайтарады. Мысал 4. Бағдарлама нені шығарады және неге? S = 'Hello' print(S.find('e')) print(S.find('ll')) print(S.find('L')) Сол сияқты, rfind әдісі берілген жолдың соңғы пайда болу индексін қайтарады («іздеу құқығы»). Мысал 5. Бағдарлама нені шығарады? S = 'Hello' print(S.find('l')) print(S.rfind('l'))

Егер сіз іздеу әдісін S.find (T, a, b) үш параметрімен шақырсаңыз, іздеу [a: b] тілінде орындалады. Егер сіз тек қана S.find (T, a) параметрлерін анықтасаңыз, онда іздеу S [a:] тілінде орындалады, яғни индекстің a және соңындағы символдан басталады.

```
S.find әдісі (T, a, b) индексті тілге емес, S жолына қайтарады.
```

3.2. Ауыстыру әдісі Ауыстыру әдісі бір жолдың барлық қайталануларын басқасымен ауыстырады. Пішім: S.replace (ескі, жаңа) - S сызығымен ауыстырыңыз. Мысал: Мысал 6. Ішкі жолды ауыстыру

```
print('Hello'.replace('l', 'L'))
```

Басқа параметрді орнату үшін әдісін ауыстыру, егер: S.replace (ескі, жаңа, санау), ол, оның ішінде бірінші санау қарағанда ғана көп емес барлық даналарын ауыстырылады емес болады. Мысал 7. Бірнеше ішкі жолдарды ауыстыру.

```
Басып шығару ( «Abrakadabra'.replace ( 'A', 'A', 2))
```

Есептеу әдісі Басқа жолдағы бір жолдың қайталану санын есептейді. қарапайым түрі S.count қоңырау (T) Бұл жағдайда жолдың C. ішінде жол T қайталанулар санын қайтарады, тек емес қабаттасатын жағдай есептеледі, мысалы: Мысал 8. Жолда астын сызуды санау.

```
Басып шығару ( «Abracadabra'.count ( 'A'))
Басып шығару (( 'A' * 10) .count ( «AA»))
```

S.count үш параметрлерін көрсете кезде (T, A, B), [A: B] желісі T srezeS жылы қайталанулар санын санау үшін теңшеледі. Жұмыс жоспары: Тапсырма 1. 1-8 мысалдарын орындаңыз. Бағдарламалардың нәтижелерін жазып алыңыз. Тапсырма 2. Бағдарламада тілімдер жасаймыз

- Жол беріледі.
- Алдымен осы жолдың үшінші сипатын шығарыңыз.
- Екінші жолда осы жолдың алдын-ала сипатын басып шығарыңыз.
- Үшінші жолда осы жолдың алғашқы бес таңбасын басып шығарыңыз.
- Төртінші жолда соңғы екі таңбаны қоспағанда, бүкіл жолды шығарыңыз.
- Бесінші жолда индекстеудің индекстері бар барлық таңбалар шығарылады (индекстеу 0 басталады, сондықтан таңбалар біріншіден шығады).
- Алтыншы жолда тақ индекстермен барлық таңбаларды шығарыңыз, яғни жолдың екінші таңбасынан бастап.
- Жетінші жолда кері таңбаның барлық таңбаларын шығарыңыз.
- Сегізінші жолда жолдың барлық таңбаларын бірінен соң бірін кері тәртіпте басып шығарыңыз.
- Тоғызыншы жолда осы жолдың ұзындығын шығарыңыз. Тапсырма 3. Сөздердің саны. Бос орындармен бөлінген сөздерден тұратын жолды ескеріңіз. Онда қанша сөз бар екенін анықтаңыз. Мәселені шешу үшін есептеу әдісін қолданыңыз. Тапсырма 4. Екі жарты. Жолды ескере отырып. Оны екі тең бөлікке бөліңіз (сызықтың ұзындығы тең болса және жолдың ұзындығы тақ болса, онда бірінші бөліктің ұзындығы тағы бір таңба болуы керек). Осы екі бөлікті орындарға ауыстырыңыз, нәтижені жаңа жолға және дисплейге жазыңыз. Бұл мәселені шешкен кезде, if if операторын қолдануға болмайды. Тапсырма 5. Екі сөзді қайталаңыз. Бұл кеңістік бөлінген дәл екі сөзден тұратын жол. Бұл сөздерді орындарға салыңыз. Нәтижені жолға жазыңыз және алынған жолды басып шығарыңыз.

№10-11 ПРАКТИКАЛЫҚ САБАҚ

ПРОЦЕДУРАЛАР МЕН ФУНКЦИЯЛАР

Тақырып 6.1: Python бағдарламалау тілі. Таңдамалы функциялар

> Жұмыстың мақсаты: Бағдарламалық кодтың бөлімдерін функциялар түрінде жобалау және оларды негізгі бағдарламада қалай пайдалану керектігін үйрену.

Қысқаша теориялық мәлімет Функциялар - бағдарламаның қалған бөлігінен оқшауланған және олар шақырылған кезде ғана орындалатын бағдарлама коды. Біз sqrt (), len () және print () функцияларына кездестік. Олардың барлығы ортақ меншікке ие: олар параметрлерді (нөл, бір немесе бірнеше) қабылдай алады және олар қайтара алады (бірақ олар қайтарылмауы мүмкін). Мысалы, sqrt () функциясы бір параметрді қабылдап, мәнді (санның түбірін) қайтарады. Print () функциясы параметрлердің айнымалы санын алады және ештеңені қайтармайды. 1-мысал. Санның бір параметрін қабылдайтын new function factorial () функциясын жасауды қарастырыңыз және мәнді қайтарады - бұл санның факторы. Естеріңізге сала кетейік, математикада n санының факториалы n ретінде анықталады! = 1 · 2 · ... · n. Мысалы, 5! = 1 · 2 · 3 · 4 · 5 = 120. Әрине, факторикалы цикл үшін оңай есептелуі мүмкін. Біздің бағдарламамызда (немесе кодының әртүрлі бөліктерінде) әртүрлі сандар факториясын бірнеше рет есептеу керек екенін елестетіп көрейік. Әрине, фореклиалды есептеулерді бір рет жаза аласыз, содан кейін Copy-Paste функциясын оны қажет жерде қою үшін пайдаланыңыз. # вычислим 3!

```
res = 1
for i in range(1, 4):
```

res *= i

```
print(res)
```

вычислим 5!

```
res = 1
for i in range(1, 6):
```

res *= i

```
print(res)
```

Алайда, бастапқы кодта бір рет қате жасасақ, онда бұл қате фореклиалды есептеуді көшірген барлық жерлерде кодқа түседі. Және, жалпы алғанда, коды мүмкін болатын қарағанда көбірек орын алады. Бірдей логиканы қайталамау үшін бағдарламалау тілдерінде функциялар бар.

```
def factorial(n):
    res = 1
    for i in range(1, n + 1):
```

res *= i

```
    return res
print(factorial(3))
print(factorial(5))
```

Функцияларды жасау кезінде сіз бірнеше ережелерді сақтауыңыз керек:

1. Функционалды код бағдарламаның басында, немесе дәлірек, factorial () функциясын пайдаланғымыз келетін нүктеге орналастырылуы керек.
2. Бірінші жол біздің функциямыздың сипаттамасы. factorial - бұл идентификатор, яғни біздің функцияның атауы. Идентификатордан кейін, функция қабылдайтын параметрлердің тізімі жақшада болады. Тізім үтірмен бөлінген параметр идентификаторларынан тұрады. Біздің жағдайда, тізім n бір саннан тұрады. Жолдың соңында қос нүкте орналастырылған.
3. Бұдан былай блок түрінде жасалған, яғни шегініспен, функцияның денесі пайда болады. Функцияның ішіндегі n мәнінің факторы есептеледі және ол айнымалы res сақталады. Функция функцияны тоқтатып, айнымалы res мәнін қайтаратын қайтару нұсқауы арқылы тоқтатылады.
4. Функция кез келген жерде пайда болуы мүмкін, оны орындау функцияны тоқтатады және көрсетілген мәнді қоңырау орнына қайтарады. Қайтар- мағыналар Атқаратын қызметім мағынаны қайтармаса, қайтар- мағынаның қайтар-мағына болмайды. Мәндерді қайтарудың қажеті жоқ функцияларда қайтару мәлімдемесі болмауы мүмкін.

МЫСАЛ 2. Максимум () функциясын жазыңыз, ол екі нөмірді алып, олардың барлығын қайтарады (іс жүзінде мұндай функция Python-ға кіріктірілген, бірақ біз өзіміз жазатын боламыз).

```
def max(a, b):
```

if a > b:

```
        return a
    else:
        return b
print(max(3, 5))
print(max(5, 3))
print(max(int(input()), int(input())))
```

```
Мысал 3. Макс3 () функциясын жазыңыз, ол үш нөмірді алып, олардың барлығын қайтарады.
def max(a, b):
```

if a > b:

```
        return a
    else:
        return b
def max3(a, b, c):
    return max(max(a, b), c)
print(max3(3, 5, 4))
```

Мысал 4. Python ішіндегі кірістірілген max () функциясы аргументтердің айнымалы санын қабылдап, олардың барлығын қайтара алады. Міне, осындай функцияның қалай жазылуы мүмкін екендігі туралы мысал.

```
def max(*a):
    res = a[0]
    for val in a[1:]:
```

if val > res:

```
            res = val
    return res
print(max(3, 5, 4))
```

Бұл функцияға жіберілген барлық параметрлер атаумен бір функцияны сипаттайтын жолда жұлдызшамен көрсетілгендей бір апарып жиналады. Жұмыс жоспары: Тапсырма 1. 1-4 мысалдарын іске қосыңыз. Тапсырма 2. Сегменттің ұзындығы. Төрт нақты нөмірлерді ескере отырып: x1, y1, x2, y2. (X1, y1) және (x2, y2) арасындағы қашықтықты есептейтін қашықтықты (x1, y1, x2, y2) жазыңыз. Төрт нақты сандарды оқып шығыңыз және осы функцияның нәтижесін шығарыңыз. Тапсырма 3. Теріс дәрежесі. Нақты нақты сан және n бүтін санты ескере отырып. Есептеу. Ерітінді қуат функциясы ретінде құрастырылған (a, n).
Стандартты экспонциялау функциясын пайдалану мүмкін емес. Тапсырма 4. Фибоначчи сандары. N-е Fibonacci нөмірін теріс емес бүтін сан үшін қайтаратын fib (n) функциясын жазыңыз.

№12-13 ПРАКТИКАЛЫҚ САБАҚ

ЖАЗБАЛАР ЖӘНЕ ФАЙЛДАР ТАҚЫРЫП 7.1: ТОПТАР АРАСЫНДАҒЫ ҚАТЫНАСТАР.

> Мақсаты: қолданбалы міндеттерді шешуге арналған жиынтықтардың өзара байланысын қалай пайдалану керектігін үйрену.

Қысқаша теориялық мәлімет Топтамалармен көрсетілген операцияларға қосымша, жиі реляциялық операциялармен айналысуға тура келеді (жиындармен жұмыс істеу контекстінде орындалған). Теңдік тақырыбына арналған жиындарды қалай салыстыру керек және элементті жиынтықтың бөлігі екендігін қалай білуге болады, біз білеміз. Бірінші жағдайда == оператор пайдалы, ал екінші жағдайда оператор. A және B екі жиынтығы A (B) деп аталады, егер A элементінің әрбір элементі B жиынына енсе, онда B жиынтығының әрбір элементі A
A жиынтығы A жиынтығының әрбір элементі B жиынтығының элементі болса, B арқылы белгіленсе, A және B⊆ B немесе A ⊂көптеген B жиынтығы (A белгілері сәйкес келуі мүмкін) Реляциялық операторлар ретінде стандартты салыстыру операторлары пайдаланылады, ал кейбіреулері үшін қосымша әдістер бар. Негізгі операторлар және олардың тиісті операциялары 1-кестеде келтірілген. A және B арқылы біз қандай операциялар орындалатындығын анықтаймыз. 1-кесте.

Топтамаларға қатысты қатынастардың негізгі операциялары Оператор немесе әдіс Сипаттама < A <B өрнекінің нәтижесі True болып табылады, егер A жиынтығының барлық элементтері B жиынтығына (көптеген A - жиынтықтың 6 жиынтығы) енгізілсе және екі жиын тең емес. Әйтпесе, өрнек мәні Fa l s e болып табылады

```
<=  немесе
issubset ( )
```

A <= B өрнегінің нәтижесі (немесе өрнектің A. issubset (B)) шын мәнінде A жиынтығының барлық элементтері B жиынтығына (жиынтық A - жиынтығы B жиынтығы) енгізілсе және жиынтықтардың теңдігі рұқсат етіледі. Әйтпесе, өрнектің мәні F a l s e болып табылады > A> B өрнегінің нәтижесі T e e болып табылады, егер B жиынтығының барлық элементтері A жиынына (жиынтықта A жиынтығының жиынтығы) енгізілсе және екі жиын тең емес. Әйтпесе, өрнектің мәні F a 1 s e болып табылады

```
>=
немесе issuperset ( )
```

A = B өрнекінің нәтижесі (немесе өрнек A.issupeset (B)) шын мәнінде B жиынтығының барлық элементтері A жиынына енеді (жиынтықта A жиынтығының жиынтығы) және жиындардың теңдігі рұқсат етіледі. Әйтпесе, өрнек мәні False

```
==
```

A == B өрнегіндегі көлемнің нәтижесі A жиынтығының барлық элементтері B жиынтығына кірсе, онда B жиынтығының барлық элементтері A жиынтығына (жиынтық теңдігі) кіреді. Әйтпесе, өрнек мәні Fa l s e болып табылады

```
isdisjoint ( )
```

A 'өрнекінің нәтижесі. isdisjoint '(B) - бұл мән. A және B жиындарының қиылысуы бос жиын болып табылса (яғни A және B жиындарының ортақ элементтері болмаса). Әйтпесе, өрнек мәні False. in A i n A өрнекінің нәтижесі A элементі A жиынтығына кірсе True, әйтпесе, өрнек мәні Fa 1 s e тең øБос жиын - ешбір элементтері жоқ (әдетте ø арқылы белгіленетін) жиын.
Айта кету керек, циклдар циклында айнымалы айнымалының мәндерін аудару үшін қолдануға болады. бұл тізімдермен болды. Есте сақтаудың жалғыз нәрсе - тізімнен өзгеше, жиынтықта элементтердің реті белгіленбеген. Айнымалы айналымның мәні қандай немесе сол иерарацияны қабылдайды, айту өте қиын. Жиындарды жасау үшін, жиынтық генераторларды пайдалануға болады. Бұл қағида генераторларға тек тіктөртбұрыш емес, ал бұйра жақшалардан тұратын жиынтықтар үшін ғана беріледі. Иллюстрациялар ретінде мынадай мәселенің бағдарламалық әдістері шешу жинақтарын пайдалану қарастыру: төмендегі шарттарды қанағаттандыратын 1-ден 100 диапазонында қандай сан анықтау қажет: 5-бөліктегі сандар 2 немесе 4-де қалдықтар береді;

- 7-бөлімде сандар 3 қалдықта беріледі;
- қалдықтардан сандарды 3-ке бөлгенде, сіз 1-ден басқа сандарды аласыз. Бұл қарапайым тапсырма және әр түрлі жолдармен шешілуі мүмкін. жағдай жоғарыда аталған екенін тексеру үшін әрбір нөмірі үшін 1-ден 100 - барлық табиғи сандар арқылы дәйекті цикл - ақыл келеді, ең қарапайым және айқын идея. Бірақ, біз, әрине, мәселенің әдіснамалық жағы сияқты соңғы нәтижеге көп көңіл бөлмейміз. Сондықтан, Python-дағы бұл мәселені шешу үшін, біз Python тілінің мүмкіндіктері мен көптеген теориясын құрастырамыз. Мәселені шешу алгоритмін түсіну үшін келесі жағдайларды ескеру қажет.
  1-ден 100-ге дейінгі табиғи сандар жиынтығы E арқылы белгіленеді. 5-ке бөлінген кезде, қалған 2-де A1-ді білдіреді. 5-ке бөлінген кезде, қалдық 4 деп көрсетілген сандардың жиынтығы A2-мен белгіленеді. 7 бөлінген саны, жиынтығы B. жәйттерді 3 бөлінген, сандар жиынтығы 100 артық емес сандар туралы айтып отырмыз осы барлық жағдайларда C таңбалау арқылы, 1 қалған береді, 3 қалған береді. Сондықтан осы көптеген көптеген жиынтығы E. қалдық 2 немесе 4 беруге 5 бөлінген сандар жиыны, Әлбетте A. ретінде белгіленеді, A = A1∪ A2. қалдық 2 немесе 4 беруге 5 бөлінген сандар жиыны, және осындай санының екі жиынтығы A (қалдық 2 немесе 4 5 бөлінген) екі тиесілі тиіс, өйткені, қалдық білдіру анықталады 3 ∩V беруге 7 бөлінеді , ал B жиынтығына (7 бөлімі бойынша 3-ін қалдырады). біз шартты қолдануға болса G бөлінген, тіпті осындай саны 1 қалған берген жоқ, бұл, осы сандардың жиынтығы C (3 бөлінген 1 қалдығына) тиесілі емес екенін білдіреді. Демек, A ∩B жиынтығы көптеген C жиналуын білдіреді. Осылайша, көптеген

D = (A ∩B) \ C Тиісті бағдарлама коды # Верхняя граница n= 100 # Множество натуральных чисел E= { s for s in range (1 , n+ 1) } # Множество чисел , которые при делении на 5 дают в остатке 2 A 1 = {s for s in E if s % 5==2 } # Множество чисел , которые при делении на 5 дают в остатке 4 A 2 = {s for s in E if s % 5 == 4 } # Множество чисел , которые при делении на 5 дают в остатке 2 или 4 A=A1 | A2 # Множество чисел , которые при делении на 7 дают в остатке 3 B= {s for s in E if s % 7==3 } # Множество чисел , которые при делении # на 3 дают в остатке 1 C= {s for s in E if s% 3 = = 1 } # Множество чисел , которые при делении на 5 дают в остатке 2 или 4 , при делении на 7 дают в остатке 3 , а при делении на 3 не дают в остатке 1 D= (A & B) - C # Отображаем результат print (" Приведенные ниже числа о т 1 до " , n) p r i n t (" 1) при деле нии на 5 дают в остатке 2 или 4 ; ") p r i n t (" 2) при деле нии на 7 дают в остатке 3 ; ") p r i n t (" 3) при делении на 3 не дают в остатке 1 : ") p r i n t (D)

Бұл жағдайда келесі нәтиже аламыз: Төмендегі сандар 1-ден 1 0 0-ге дейін 5-ке бөлінгенде, 2 немесе 4 қалдықты беріңіз; 7-ге бөлінген кезде қалдықтар 3; 3-ке бөлінгенде, олар 1 қалдықты бермейді:

```
{ 2 4 , 1 7 , 5 9 , 8 7 }
```

Сандық ауқымның жоғарғы шегінің мәні n айнымалы мәніне жазылады. Командада 1-ден n-ге дейінгі ауқымда мәндері бар оң бүтін сандар жиынтығы жасалады E = (1, n + 1) диапазонында s s үшін. Мұнда жиынтықтың генераторын пайдаланғанымызда: жақшаларда, s үшін s (I, n + I) аралығындағы s s операторлары көптеген элементтердің 1-ден n-ге дейінгі табиғи сериялардың мәндері арқылы өтетін айнымалы s анықтайды. Set aI = I = {s болса, S кезінде 5% == 2} командасымен жасалады. Енді жиынтықтың генераторында айнымалы s көптеген элементтердің мәндерінен шығып кетеді. Элемент S алгебрада 5% == 2 жағдайында сақталады, яғни, егер 5-ке тең бөлісу саны 2-ге тең болса. Ұқсас бұйрықтар да бар

```
A2 = {s  for  s in E  if  s % 5 = = 4 }
B= {s for  s  in E if s % 7 == 3 }
C= {s  for  s  in  E if  s % 3 == 1 }
```

басқа жиынтықтар жасалады. 5-ке бөлінген кезде, қалған 2 немесе 4-дегі сандар жиынтығы Al және A2 жиындарын біріктіру арқылы есептеледі - A = Al ∪ A2 пәрменін қолданып есептеледі. Ақырында, D = (A & B) - C пәрменін қолданып, қажетті нәтижені табамыз. Сіз «классикалық» тәсіл қолдансаңыз болады. Мысал 2. Біз бірдей мәселені шешеміз. # Верхняя граница

```
n= 100
```

Пустое множество

```
D= set()
```

Бөлуге 5, 2 немесе 4 кірістілік бөлінген кезде, олардың саны 3-ке 3-ке бөлінеді, ал 3-ке бөлінген кезде олар қашықтықта берілмейді

```
For  s  in range ( 1 , n + 1 ) :
    i f s % 5==2  or   s % 5 = = 4 :
         if  s % 7 == 3 :
```

if s % 3 ! = 1 :

```
                    D .add ( s )
```

Отображаем резул ьтат

```
p r i n t ( " Приведенные ниже числа о т 1 до " , n )
p r i n t ( " 1 ) при делении на 5 дают в о с т а т ке 2 или 4 ; " )
p r i n t ( " 2 ) при делении на 7 дают в о с т а тке 3 ; " )
p r i n t ( " 3 ) при делении на 3 не дают в ост атке 1 : " )
p r i n t ( D )
```

Осы тұрғыда, біз бірінші бос жиынын әмір, содан кейін цикл мәлімдеме мен кірістірілген шартты есептілігін тобын пайдалану) D = жиынтығы (жасау жаңа элементтер тізіміне қосу - әрине, бұл элементтері барлық критерийлерге (шартты есептілігінде шарттары) қанағаттандыратын болса. Элементті жиынға қосу үшін add () әдісі қолданылады. Python түріндегі s e t (set) түріне қосымша frozenset (өзгермейтін жиын) түрі бар. Өзгеріссіз жиынтығы мен кәдімгі жиынтықтың арасындағы негізгі айырмашылық - бұл өзгермейтін жиынтығы, оңай деп болжауға болатындықтан, өзгерту мүмкін емес. Сіз өзгермейтін жиынтығын frozenset () функциясын пайдаланып жасай аласыз, мысалы, тізім немесе мәтін жолы. Кәдімгі жиынтықтарға қолданылатын көптеген әдістер өзгермейтін жиынтықтарға қолданылуы мүмкін - сөйлеу, әрине, бастапқы жиынның құрылымын өзгертпейтін әдістер туралы. Жұмыс жоспары: Тапсырма 1. Мысалдың бағдарлама кодын енгізіңіз 1. Жоғарыда көрсетілген шешіммен алынған нәтижені салыстырыңыз. Тапсырма 2. Кодты енгізіңіз, мысалы 2. Нәтижені тексеріңіз. 3-міндет. 1-ден 20-ға дейінгі диапазондағы барлық жұп сандардан бөліп шығатын бағдарламаны жазыңыз. Мысалы, бұл 2, 6, 10 және т.б. сандар. Жұмыстың нәтижесі есеп түрінде.

№14-15 ПРАКТИКАЛЫҚ САБАҚ

СТАНДАРТТЫ МОДУЛЬДЕР ТАҚЫРЫП 8.1: ФАЙЛДАРМЕН ЖҰМЫС ІСТЕУ

> Мақсаты: файлдағы ақпаратты қалай сақтау керектігін, файлдардан ақпаратты оқып үйрену, бағдарламалардағы файлдық жүйені пайдалану.

Қысқаша теориялық мәлімет Файл - аты бар дискідегі деректер жиынтығы. Бағдарламашы тұрғысынан екі түрдегі файлдар бар:

1. мәтіні бар мәтіндер, сызықтарға бөлінеді; Мәселен, мәтіндік файлдардағы барлық арнайы таңбалардан тек жаңа жол белгілері болуы мүмкін;
2. кез келген деректерді және шектеулердісіз кез келген кодты қамтитын екілік; екілік файлдарда сақталған суреттер, дыбыстар, бейнелер және т.б. Біз тек мәтіндік файлдармен жұмыс істейміз.
   Бағдарламадан файлмен жұмыс үш фазаны қамтиды.
   Алдымен файлды ашу керек, яғни оны бағдарламаға белсенді етеді. Егер файл ашық болмаса, онда бағдарлама оған қол жеткізе алмайды.
   Файлды ашқанда, жұмыс режимін көрсетіңіз: оқу, жазу немесе файлдың соңына деректерді қосу. Көбінесе, басқа бағдарламалар оны пайдалана алмайтын етіп, ашық файл құлыпталады. Файл ашылғанда (белсенді) бағдарлама онымен барлық қажетті әрекеттерді орындайды. Осыдан кейін, файлды жабу керек, яғни босатыңыз, бағдарламамен байланыс үзіңіз. Бағдарламада жасалған барлық соңғы өзгерістердің дискке жазылуын аяқтаған кезде. Python бағдарламасында файлды ашады және файлдың сілтегішін қайтарады - бұл файлмен бірге жұмыс істейтін айнымалы. Ашық функция екі параметрді қабылдайды: Файл атауы (немесе бағдарлама жазылған каталогта болмаса, файлдың жолы) және файлдың ашық режимі «r» - оқу үшін ашық,
3. «w» - жазу үшін ашық,
4. «a» - қосу үшін ашық. Функцияны ашу функциясы: file_name = open («толық файл атауы», режим) Мысал: f = open ('text.txt', 'r') Бағдарламаның өзі сол қалтада орналасқан 'text.txt' файлы ашылады. Файл оқу үшін ашылды. Қалтада мұндай файл жоқ болса, қате туралы хабар пайда болады. Егер файлдың ашық режимі көрсетілмесе, ол оқу үшін ашылады.
   Егер бар файл жазу үшін ашылса, оның мазмұны жойылады.
   Бағдарлама аяқталғаннан кейін барлық ашық файлдар автоматты түрде жабылады.
   Файлды жабу үшін жабу әдісі де қолданылады.

```
Python файлынан оқу. Read () әдісі
Read () әдісі ашық файлдан жолды (бір тізім элементі) оқиды.
Read () әдісінің синтаксисі.
Filename_read.read ([count])
```

Қосымша санау параметрі ашық файлдан оқылуы қажет байттардың саны. Бұл әдіс файлдың басынан ақпаратты және егер параметр параметрі көрсетілмесе, файлдың соңына дейін оқиды. Мысал 1. D: text1.txt файлын операциялық жүйенің түбірлік каталогында жасаңыз және атыңызды және тегіңізді жазыңыз. Бұл файлды ашатын және оның мазмұнын экранға шығаратын бағдарламаны жазыңыз.

```
file1 = open("D:text1.txt", "r")
s=file1.read()
print('В файле записана информация: ', s)
file1.close()
```

Python файлына жазу. Жазу әдісі. Write () әдісі ашық файлға кез келген жолды жазады. Python жолдары мәтінді ғана емес, екілік деректерді де қамтуы мүмкін екендігін есте ұстау маңызды.

```
Жазу () әдісі файлдың соңына жол үзілімін ('\ n') қосады.
Жазу () әдісінің синтаксисі.
Name_of_file_name.write (string_name)
```

Мысал 2. Жаңа файл жасаңыз және кез келген жолды жазыңыз.

```
file1 = open("D:text2.txt", "w")
file1.write("Мне нравится Python!\nЭто классный язык!")
file1.close()
```

Бұл файлды Explorer бағдарламасында ашыңыз және бағдарламаның қалай жұмыс істейтінін тексеріңіз. Мысал 3. D: text2.txt файлын ашып, оған мәтін қосыңыз: «Бұл файл менің бағдарламамен ашылды!»

```
file1 = open("D:text2.txt", "a")
file1.write("Этот файл был открыт моей программой!»)
file1.close()
```

Мысал 4. Пайдаланушы пернетақтаның сандарын енгізеді. Оларды мәтінге жазыңыз2. Содан кейін оның мазмұнын тексеріп, тексеріңіз.

```
file1 = open("D:text2.txt", "w")
a=[0]*10            #Создадим массив на 10 элементов, заполненный нулями
for i in range(10):
    print('Ведите' ,i,' число')
    a[i]=input()            #Вводим число  с клавиатуры и записываем в массив
    file1.write(a[i])             #Записывает эту строку-число в файл

file1.close()                #Закрываем файл

file1 = open("D:text2.txt", "r")        #Открываем файл для чтения
s=file1.read()
print('В файл было записано:',s)
```

Нәтижесінде біз барлық сызықтарды бір-біріне жабыстыратын сызықты алды. Неге бұлай болды? Файлға сызық жазу кезінде бос орын немесе жаңа жол таңбасы қосылмайды. Сандар арасында бос орын болу үшін бағдарламаны түзетіңіз. Ол үшін файлға жаңа жолға жазылатын сызықты түзетеміз:

```
file1.write (a [i] + '')
```

Енді өңдеуге болатын сандары бар файл бар.

Мысал 5. text2.txt файлын ашып, сол жерден сандарды қамтитын жолды оқып, жолды жеке таңбалар-сандарға бөліңіз. Рәмметтерді бүтін санға түрлендіріп, олардың сомасын табыңыз. Біз мәселені екі кезеңде шешеміз: алдымен жолды оқып, бөлек рәміздерге бөліңіз:

```
file1 = open("D:text2.txt", "r")
s=file1.read()
print('В файл было записано:',s)
print('Преобразуем строку из чисел в список из цифр-символов')
a =s.split()
print (a)
```

Осы бағдарламаның нәтижесі: Файлға келесі файл жазылған: 2 3 4 5 6 5 4 3 2 2 Сандарды сандар тізіміне түрлендіру

```
[2, 3, 4, 5, 6, 5, 4, 3, 2, 2)
```

Әрі қарай, таңбалар тізімінің әрбір элементін бүтін санға аударатын бағдарлама кодын қосыңыз және барлық сандардың қосындысын есептейді.sum=0

```
for i in range(10):
    a[i]=int(a[i])
    sum=sum+a[i]
print ('Сумма всех чисел равна ', sum)
```

Осы бағдарламаның нәтижесі: Файлға келесі файл жазылған: 2 3 4 5 6 5 4 3 2 2

```
Сандарды сандар тізіміне түрлендіру
[2, 3, 4, 5, 6, 5, 4, 3, 2, 2)
```

Әрі қарай, таңбалар тізімінің әрбір элементін бүтін санға аударатын бағдарлама кодын қосыңыз және барлық сандардың қосындысын есептейді. Мысал 6. Біз жасаған text2.txt және text2.txt файлдарын оқыңыз:

```
file1 = open("D:text2.txt", "r")
file2 = open("D:text1.txt", "r")
s1=file1.read()
s2=file2.read()
print(s1,'\n',s2)
file1.close()
```

Сұрақ: операторға басып шығару (s1, '\ n', s2) '\ n' параметрі көрсетіледі. Егер сіз бұл сұраққа жауап бере алмасаңыз, осы параметрді жойыңыз. Бағдарламаның нәтижесі қалай жұмыс істеді? Мысал 7. Бағдарлама кодын пайдалану арқылы осы компьютермен жұмыс істейтін оқушылардың атауларынан тұратын кіру тізімін жасаңыз. Осы тізімді text3.txt файлында жазыңыз.

```
file1= open("D:text3.txt", "w")
s=['Иванов','Петров','Сидоров']
for i in range(len(s)):
file1.writelines(s[i]+'\n')
print('end')
file1.close()
```

Мысал 8. Пайдаланушы тегінің атын сұрайтын бағдарламаны жазыңыз. Егер бұл тізімге қолжетімді болса (6-мысал), онда біз: «Сіздің тегіңіз пайдаланушылардың тізімінде!» Деген хабарды басып шығарамыз. Егер жоқ болса - «Сіздің атыңыз кіру тізімінде жоқ».

```
file1= open("D:text3.txt", "r")
```

#s1 – Пайдаланушылардың аттарын файлдан оқылатын тізім сақталады # fam- файлдың бір атын оқу үшін жол айнымалысы

```
s1=[fam for fam in file1]
print(s1);
```

Бұл бағдарламаны іске қосыңыз. Не болды? Әрбір соңғы атаудың соңында '\ n' жолдың соңғы таңбасы. Оны жою үшін әрбір айнымалы айнымалының соңғы таңбасын өшіру керек. Жолды өңдеу s1 = [fam [: - 1] файлындағы fam файл үшін] арналған [fam for fam1 file1]. Нәтижені тексеріңіз. Бағдарламаны жазуды жалғастырамыз:

```
imy=input('Введите свою фамилию')
fl=0
for i in range(len(s)):
    if imy==s1[i]:
        print ('Ваша фамилия есть в списке пользователей')
        fl=1
if fl==0:
    print('Вашей фамилии нет в списке доступа')
file1.close()
```

Жұмыс жоспары: Тапсырма 1. 1-8 мысалдар бағдарламаларын енгізу, откладтау Тапсырма 2. Пайдаланушы тізімінде пайдаланушы атын жаза алатындай етіп, 8-мысалдағы бағдарламаны өзгертіңіз. Тапсырма 3. Файлды ашатын бағдарламаны жазыңыз, одан ақпаратты ақпаратты жолдың айнымалы мәніне енгізіңіз, содан кейін дан жолындағы таңбалардың санын есептеп, оны сол файлға жазады. Мысалы: text1.txt файлында жазылған: Иванов Иван Иванович Иванов Иван Иванович 21 болады

Ұсынылатын әдебиеттер тізімі Негізгі әдебиеттер

---

1 МАҚАШЕВ, Е.П. АЛГОРИТМДЕР ЖӘНЕ ПРОГРАММАЛАУ [МӘТІН] = ОҚУ ҚҰРАЛЫ / Е.П. МАҚАШЕВ - АЛМАТЫ, 2015.- 2 ЭКЗ

Смайлова, Ұ.М. Программалау: Алгоритм құру технологиялары [Мәтін] = Оқу құралы / Ұ.М.Смайлова - Алматы, 2011.- 2 экз Васильев А.Н. Python на примерах. Практический курс ·по программированию. - СПб.: Наука и Техника, 2016. - 432 с. Қосымша әдебиеттер

1. Поляков К.Ю., Еремин Е.А.. Информатика, 10-11 класс, учеб. пособие, изд.БИНОМ, 2015 г.
2. Прохоренок Н. А. Python 3 и PyQt. Разработка приложений.--:- СПб.: БХВ-Петербург, 2016.-704
3. Шапошникова С. Лаборатория юного линуксоида http://younglinux.info, 2011
4. 050704- Есептеу техникасы және бағдарламамен қамтамасыз ету [Мәтін] = Алгоритмдер теориясы., Сараптау жіне интелектуальды жүйелер. Интернет технологиялары. Комьпютерлік желілер.: Оқу-әдістемелік құралы / Құрас. аға оқытушы Шалбаева Ш., Алтынбеков Ш.Е., Абдрахманова М.Б, п.ғ.к. Жайдақбаева Л.Қ., ж.т.б.- Шымкент: ХГТУ, 2012.- 228 б.-40экз
5. Манакова, И.П. М23. Информационные системы и технологии. Языки программирования высокого уровня. Программирование на языке Python [Электронный ресурс] : учеб.-метод. пособие / И.П. Манакова; М-во образования и науки РФ; ФГАОУ ВО «УрФУ им. первого Президента России Б.Н. Ельцина», Нижнетагил. технол. ин-т (филиал). – Нижний Тагил : НТИ (филиал) УрФУ, 2016. – 37 с. Мультимедиялық қамтамасыз ету және электрондық ресурстар HYPERLINK "https://melimde.com/algoritmder-jene-derekter-rilimi.html"https://melimde.com/algoritmder-jene-derekter-rilimi.html HYPERLINK "https://open.kaznu.kz/courses/course-v1:kaznu+PrK+2021C2/about"https://open.kaznu.kz/courses/

course-v1:kaznu+PrK+2021C2/about HYPERLINK "http://rmebrk.kz/bilim/association/boribaev-algoritmdeu.pdf"http://rmebrk.kz/bilim/association/boribaev-algoritmdeu.pdf