7 КОРТЕЖДЕРМЕН ЖӘНЕ ТІЗІМДЕРМЕН ЖҰМЫС ІСТЕУ

7.1 Кортеждерді жариялау(хабарлау)

Python тіліндегі **кортеждер** массивтер деп аталатын құрылымдарға ұқсас. Массивтермен жұмыс істеу әдістері C++, Microsoft Visual Basic, Delphi және т.б. сияқты танымал бағдарламалау тілдерін оқытуда қарастырылады. Айта кету керек, оларда массивтер біртұтас болып саналатын бірнеше ұқсас объектілердің (сандар, таңбалар, жолдар және т. б.) ресми бірлестігі ретінде анықталады.

Python-да кортеж осы бағдарламалау тілінде жұмыс істеуге болатын тізбектің бір түрі ретінде анықталады. Басқа бағдарламалау тілдеріндегі массив мәліметтерінен түбегейлі айырмашылығы-бұл біріктірілген мәліметтер тізбегі **олардың мәндерін өзгертуге мүмкіндік бермейді**. Бірақ бұл жерде белгілі бір артықшылықтары да бар. Біріншіден, кортеж элементтерін өңдеу жылдамдығы артады, өйткені жүйе мәндер өзгермейтінін алдын-ала біледі. Екіншіден, кортеж сияқты құрылымды басқа құрылымдарды қалыптастыру үшін қолдануға болады: мысалы, сөздіктерде. Үшіншіден, мысалы, тізімдерге қарағанда аз жад алады, сонымен қатар, кортеж элементтері кездейсоқ өзгерістерден қорғалған.

Егер массив бір типтегі объектілердің жиынтығы болса, онда кортеж Python-да толық әртекті объектілер болуы мүмкін мысалы, жолдық және сандық мәндер немесе дыбыстық файлдар, бейнелеу файлдары сияқты.

Кортеж жариялау синтаксисі келесі түрде:

Кортеж аты = (элемент 1, элемент 2, ...элемент N)

Мысалы

```
korteg=( 1, 2, 3,4, 5)
```

Басқа жазба болуы мүмкін, мысалы, жол түріндегі мәндер үшін

```
Кортеж аты  =(«Элемент1»,
              «Элемент 2»,
              «Элемент 3»,
              «Элемент N»)
```

сонымен қатар, бір жолда бірнеше элементтер орналасуы мүмкін .

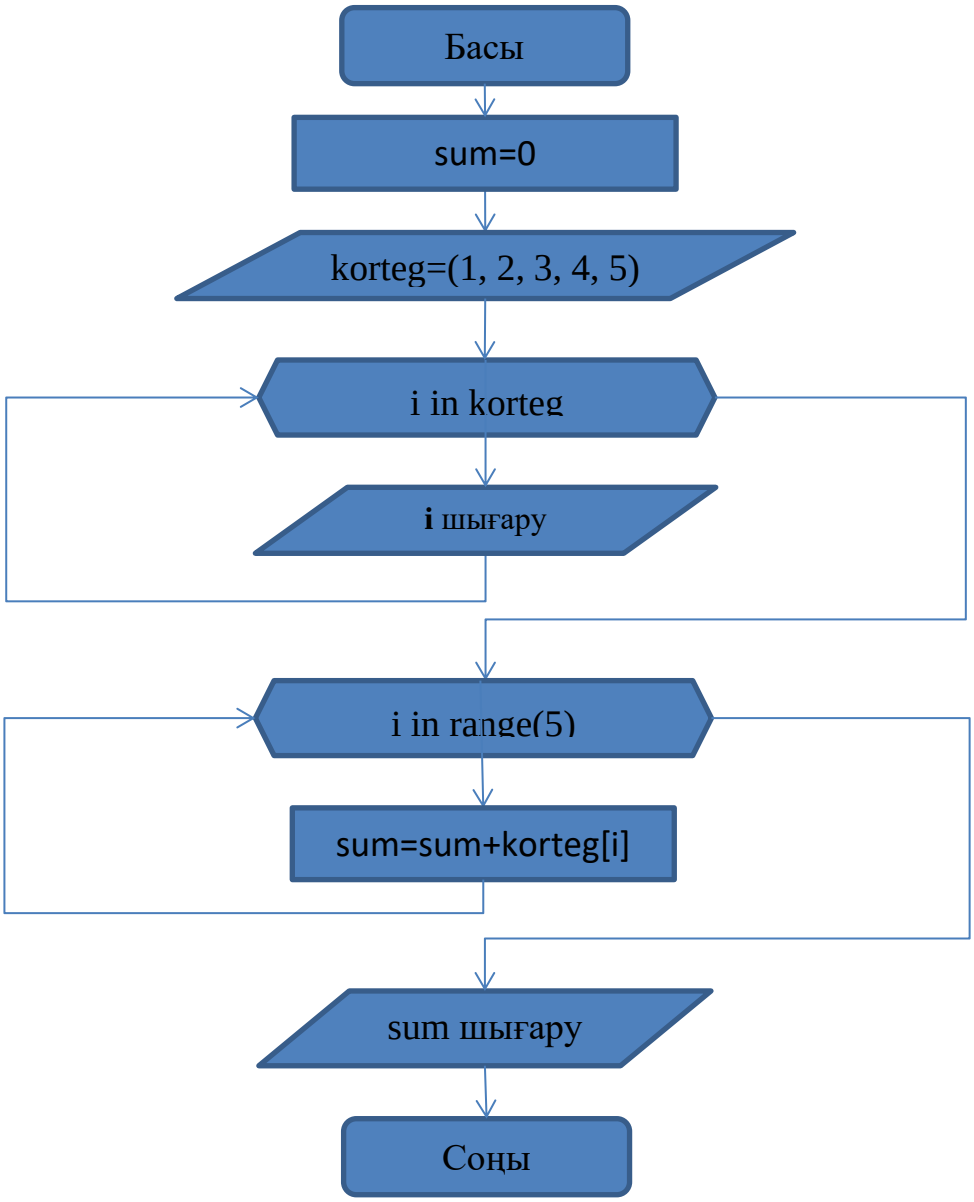
Бағдарламадағы кортеждің әрбір элементіне **қол жеткізу индекстің** көмегімен жүзеге асырылады - кортеж элементінің өзіндік атауы болып табылатын бүтін сан. Бағдарламада кортеж элементі туралы айтқан кезде, оның атауының артында тік жақшадағы элемент индексі болуы керек, мысалы, **korteg[i]**

Айта кету керек, кортеж элементтерін нөмірлеу бірліктен емес, нөлден

басталады. Кортежді өңдеу бойынша операциялар **for** операторымен циклде элемент бойынша жүзеге асырылады.

Есеп 7.1.1. Алдын ала анықталған бүтін сандардан тұратын кортежде элементтердің қосындысын табыңыз.

Шешімі. Есепті шешу алгоритмінің блок-схемасы 72-ші суретте көрсетілген.



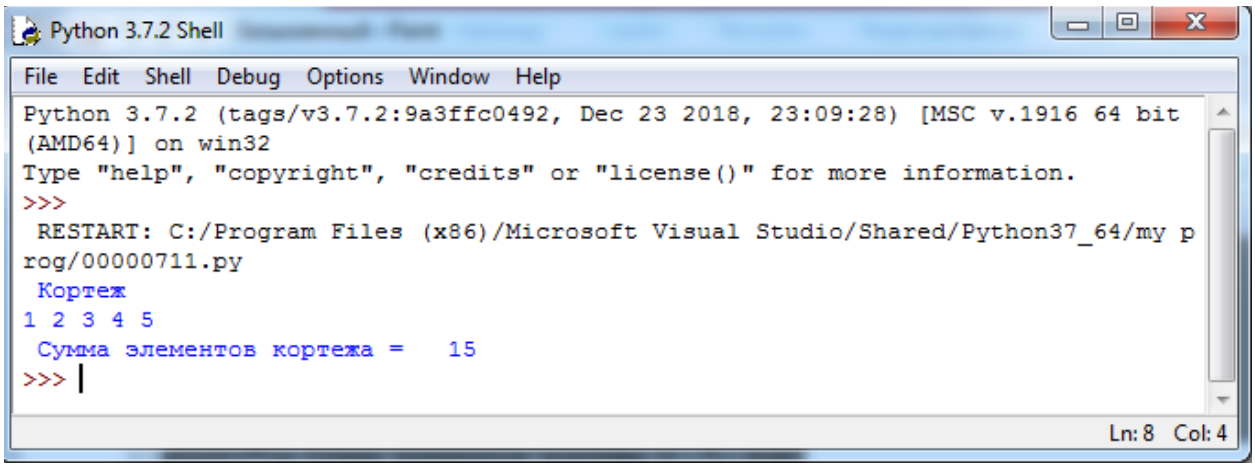
Сурет 72 – 7.1.1 есептің шешу алгоритмның блок-схемасы

Листингте есепті шешуге арналған программа коды көрсетілген:

```
sum=0
korteg=(1, 2, 3, 4, 5)
print(" Кортеж ")
for i in korteg:
```

```
    print(i, end=" ")
for i in range(5):
    sum=sum+korteg[i]
print("\n Сумма элементов кортежа =  ", sum)
```

Бағдарлама жұмысының нәтижесі 73-ші суретте көрсетілген.



Сурет 73 – Кортеж элементерінің қосындысының нәтижесі

7.2 Кортеждерді классикалық өңдеу әдістері

Деректер тізбегін өңдеумен байланысты мәселелерді шешу олардың қарапайым өңдеу әдістеріне негізделген. Тапсырманы логикалық бөліктерге бөліп, оны шешуді жеңілдетуге болады. Кортеждермен жұмыс істеудің әдеттегі міндеттері-онда берілген элементтің болуын анықтау, белгілі бір шарттарды қанағаттандыратын элементтерді таңдау және т. б.
Кортежде ұсынылған деректерді басқаруға мүмкіндік беретін барлық қолданыстағы әдістердің ішінен мыналарды бөліп көрсетуге болады:
1. Берілген жағдайда элементтер санын табу.
2. Берілген жағдайда элементтер мәндерінің қосындысын табу.
3. Берілген жағдайда элементтер мәндерінің көбейтіндісін табу.
4. Кортеж элементтерінің экстремалды мәндерін іздеу (максималды және/немесе минималды мәнді іздеу).
5. Кортеждерді біріктіру (тіркесу).
6. Кортеж ішіндегі элементтердің мәндерінің алмастыру.
7. Кортеждің бөліктері(кескінділері).

Төменде кортежді өңдеудің тізбектелген алгоритмдері келтірілген. Алғашқы төрт әдісті орындау оңай болғандықтан, сонымен қатар бұл алгоритмдер алдыңғы тақырыптардағы есепті шешу кезінде бірнеше рет қарастырылғанын ескере отырып, біз оларды Түсіндірмесіз жүзеге асырамыз. Айта кету керек, алғашқы төрт тізімде тағы бір кортеж элементі 10 санымен салыстырылады, бірақ іс жүзінде, мысалы, **input** функциясын қолдана отырып, шарт сұрауын ұйымдастырған дұрыс.

Берілген жағдайда элементтер санын табу.
Кортеж элементтерінің **korteg =(10, 8, 29, 35, 7)** 10-нан үлкен немесе тең санын табуға мысал келтірейік. Листингте осы мысалды шешуге жауап беретін бағдарлама коды көрсетілген:

```
kol=0
korteg=(10, 8, 29, 35, 7)
print(" Кортеж ")
for i in korteg:
   print(i, end=" ")
for i in range(5):
   if korteg[i]>=10:
      kol=kol+1
print("\n Количество элементов кортежа больших или равных десяти =  ", kol)
```

Берілген жағдайда элементтер мәндерінің көбейтіндісін табу.
Кортеж элементтерінің **korteg =(10, 8, 29, 35, 7)** 10-нан үлкен немесе тең қосындысын табуға мысал келтірейік. Листингте осы мысалды шешуге жауап беретін бағдарлама коды көрсетілген:

```
sum=0
korteg=(10, 8, 29, 35, 7)
print(" Кортеж ")
for i in korteg:
   print(i, end=" ")
for i in range(5):
   if korteg[i]>=10:
      sum=sum+korteg[i]
print("\n Сумма элементов кортежа больших или равных десяти =  ", sum)
```

Берілген жағдайда элементтер мәндерінің көбейтіндісін табу.
Кортеж элементтерінің **korteg =(10, 8, 29, 35, 7)** 10-нан үлкен немесе тең көбейтіндісін табуға мысал келтірейік. Листингте осы мысалды шешуге жауап беретін бағдарлама коды көрсетілген:

```
pr=1
korteg=(10, 8, 29, 35, 7)
print(" Кортеж ")
for i in korteg:
   print(i, end=" ")
for i in range(5):
   if korteg[i]>=10:
      pr=pr*korteg[i]
print("\n Произведение элементов кортежа больших или равных десяти =  ", pr)
```

Кортежде экстремалды мәндерді табу.
Біз кортеждің korteg =(10, 8, 29, 35, 7). максималды және минималды элементтерін табуға мысал келтіреміз Листингте осы мысалды шешуге жауап беретін бағдарлама коды көрсетілген:

```
maxim=-32768
minim=32767
korteg=(10, 8, 29, 35, 7)
print(" Кортеж ")
for i in korteg:
   print(i, end=" ")
for i in range(5):
   if korteg[i]>maxim:
      maxim=korteg[i]
   if korteg[i]<minim:
      minim=korteg[i]
print("\n Максимальный элемент кортежа = ", maxim)
print("\n Минимальный элемент кортежа = ", minim)
```

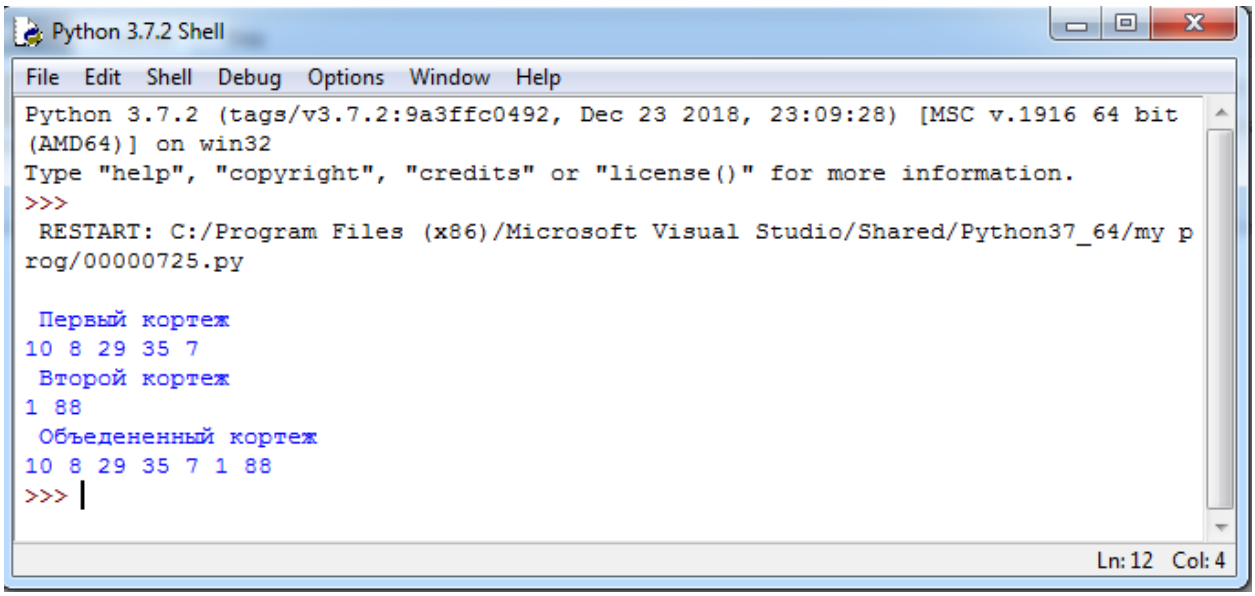
Кордеждерді біріктіру (тіркестіру).

Есеп 7.2.1. Екі кортеж берілген**.** Оларды өзара біріктіру керек.

Шешімі . Тапсырма алдымен екі түпнұсқа кортежді қалыптастыру және бірінші кортежге қосу (+) операциясын қолдану, содан кейін біріктірілген кортеж экранда көрсетіледі. Листингте есепті шешуге жауап беретін бағдарлама коды бар:

```
korteg1=(10, 8, 29, 35, 7)
korteg2=(1, 88)
print("\n Первый кортеж ")
for i in korteg1:
   print(i, end=" ")
print("\n Второй кортеж ")
for i in korteg2:
   print(i, end=" ")
print("\n Объедененный кортеж ")
korteg1 +=korteg2
for i in korteg1:
   print(i, end=" ")
```

Бағдарламаның нәтижесі 74-ші суретте көрсетілген.



Сурет 74 – 7.2.1 тапсырмасындағы екі кортеждің элементтерін біріктіру бағдарламасының нәтижесі

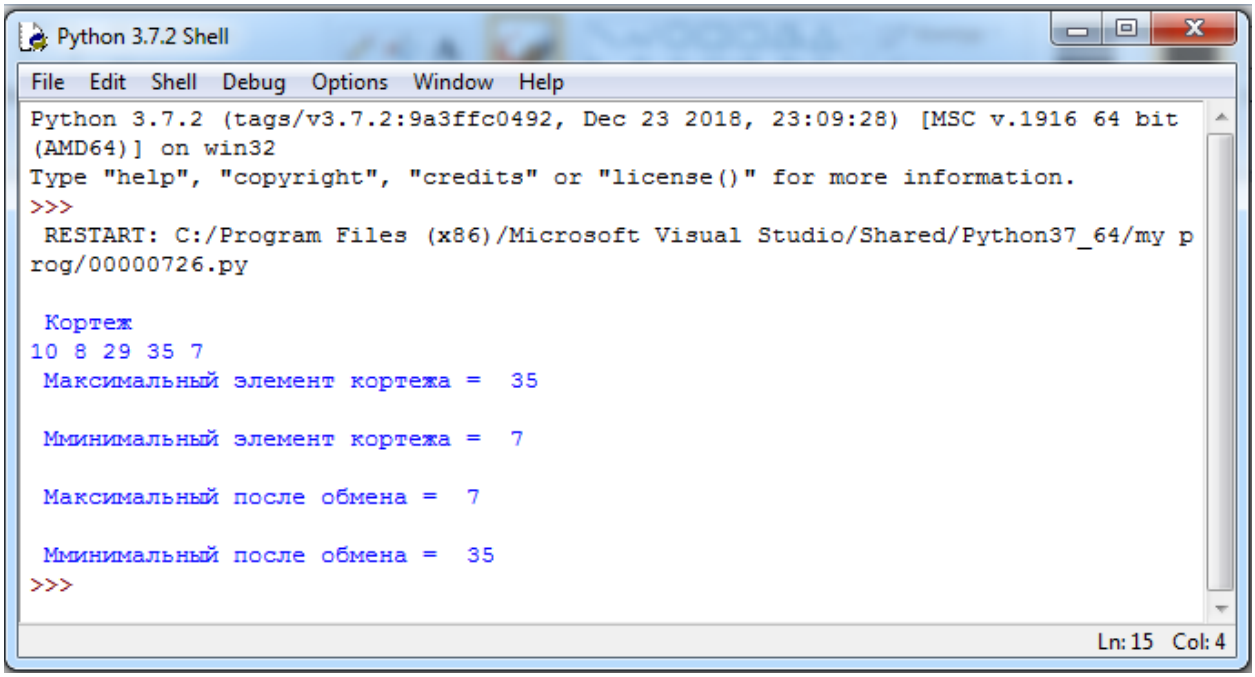
Кортеждегі элементтердің мәндерін ауыстыру.

Есеп 7.2.2. Бүтін сандар кортежіндегі максималды және минималды элементтерді тауып, оларды алмастырыңыз.

Шешім. Максималды және минималды элементтерді табу әдістері алдыңғы дәрістерде қарастырылған, сондықтан алмасудың өзі **maxim,minim=minim,maxim** операторларын орындалуынан тұратындығын ескерген жөн. Листингте есепті шешуге жауап беретін бағдарлама коды көрсетілген:

```
korteg=(10, 8, 29, 35, 7)
print("\n Кортеж ")
for i in korteg:
   print(i, end=" ")
minim=32767
maxim=-32768
for i in range(5):
   if korteg[i]>maxim:
      maxim=korteg[i]
   if korteg[i]<minim:
      minim=korteg[i]
print("\n Максимальный элемент кортежа = ", maxim)
print("\n Мминимальный элемент кортежа = ", minim)
maxim,minim=minim,maxim
print("\n Максимальный после обмена = ", maxim)
print("\n Мминимальный после обмена = ", minim)
```

Бағдарламаның нәтижесі 75-ші суретте көрсетілген.



Сурет 75 – 7.2.1 тапсырмасындағы бағдарламасының нәтижесі

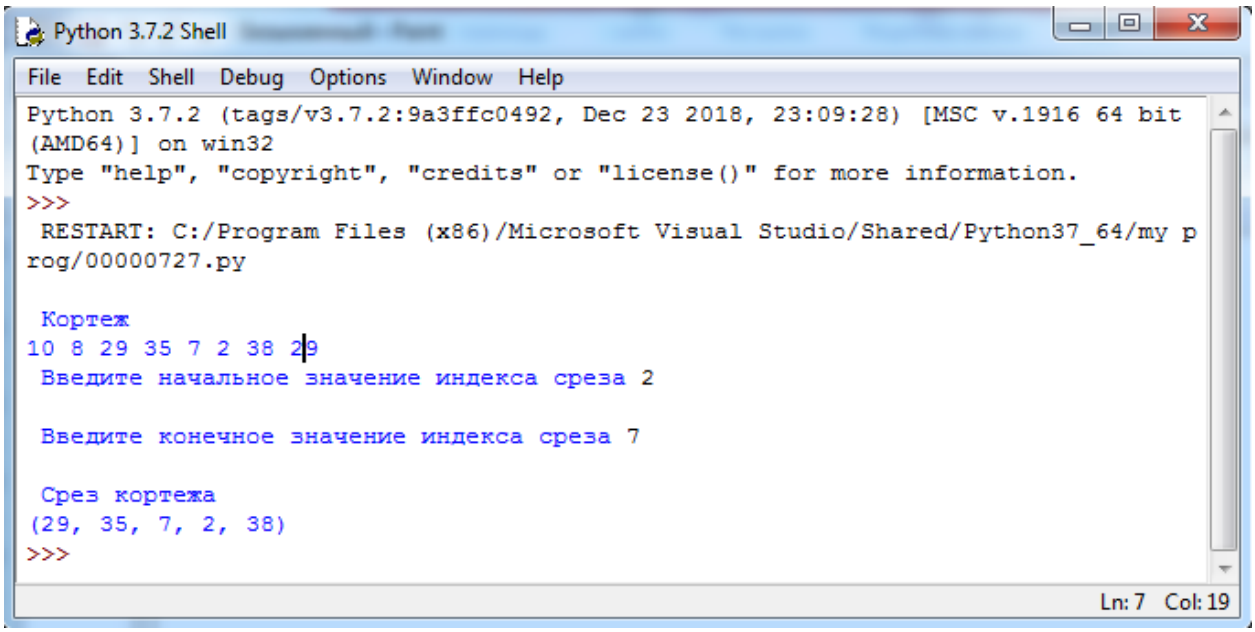
Кортеждердің кескінділері.

Кортеждің кесіндісі элементтердің алдын-ала белгіленген **a** және **b** позицияларының арасында орналасқан кортеж элементтерін шығару арқылы алынады. Кескіндер бағдарламада print (korteg [a: b]) операторының көмегімен жүзеге асырылады.

Мысалы берілген korteg=(10, 8, 29, 35, 7, 2, 38, 29) кортеж кескінділері . Листингте есепті шешуге жауап беретін бағдарлама коды көрсетілген:

```
korteg=(10, 8, 29, 35, 7, 2, 38, 29)
print("\n Кортеж ")
for i in korteg:
    print(i, end=" ")
a=int(input("\n Введите начальное значение индекса среза "))
b=int(input("\n Введите конечное значение индекса среза "))
print("\n Срез кортежа ")
print(korteg[a:b])
```

Бағдарламаның нәтижесі 76-шы суретте көрсетілген.



Сурет 76 – Кортеж кесінділеріне арналған бағдарлама нәтижесі

7.3 Тізімдермен жұмыс істеу

Python-дағы тізімдер кортежге ұқсас жұмыс істейді айырмашылығы ауыспалы реттілікте. Осылайша, егер жаңа элементті қосу, элементті жою, сұрыптау және т.б. әрекеттерді кортежде орындалуы мүмкін болмаса, онда тізіммен жұмыс істеу әдістері бұған мүмкіндік береді.
Кортеж элементтері сияқты, тізім элементтері тек объектіге сілтеме жасайды, сондықтан тізімдерге әртүрлі типті мәліметтерді енгізуге болады. Бұл тәсіл дәстүрлі массивтерді басқа бағдарламалау тілдерінде өңдеуден ерекшеленеді, дегенмен Python-да тізімдерді өңдеу әдістерінде массивтерді өңдеу әдістерімен ұқсастық бар.
Тізімді жариялаудың синтаксисын қарастырайық .

Имя списка = [элемент 1, элемент 2, ...элемент N]

Мысалы,

spisok=[l, 2, 3, 4, 5]

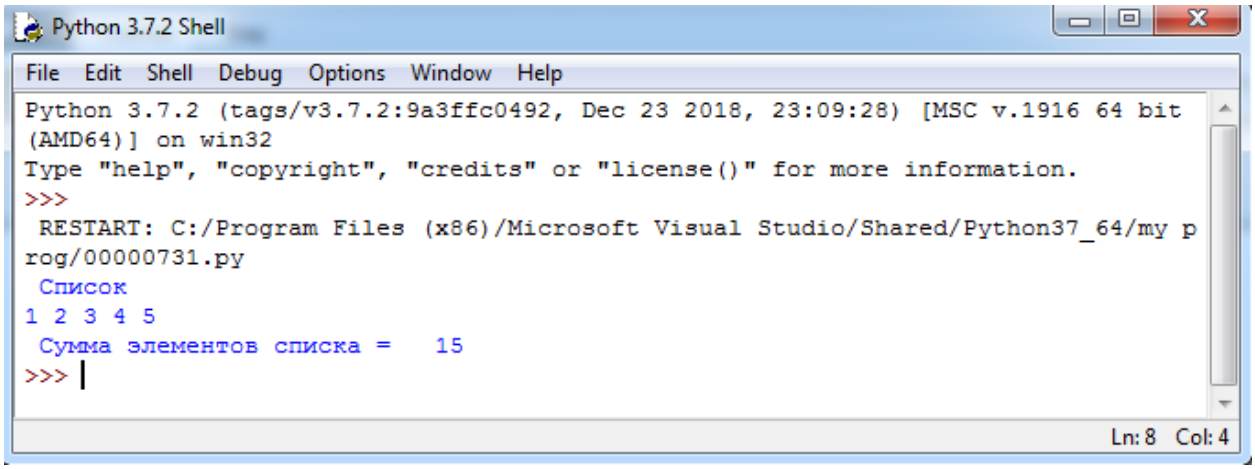
Тізімдермен жұмыс істеу кезіндегі типтік тапсырмалар:
- оларда берілген элементтің болу фактісін анықтау;
- белгілі бір шарттарды қанағаттандыратын элементтерді таңдау.

Екі жағдайда да тізім элементтерін берілген үлгімен циклдік салыстыру қолданылады. Берілген үлгінің тізімде болу фактісін анықтау үшін бір сәйкестік жеткілікті, содан кейін одан әрі қарастыру тоқтатылады. Егер таңдау шарты тізімнің бірнеше элементтері үшін орындалса, онда бүкіл тізімді соңына дейін қарастыру керек.
Жоғарыда қарастырылған барлық әдістер тізімдер үшін жарамды, бірақ

тізімдердің өзгеруіне байланысты олардың жиынтығы айтарлықтай кеңейтілуі мүмкін. Атап айтқанда, тізім элементтерін жоюға және қосуға, сұрыптауға болады.
Пайдаланушы оның элементтерін төртбұрышты жақшаға орналастыру арқылы тізім жасай алады (жоғарыда көрсетілген синтаксиске сәйкес) немесе оны кездейсоқ жасай алады. «Тізім элементтерінің қосындысын табу» тапсырмасының мысалын пайдаланып, осы әдістерді толығырақ қарастырайық. Осы тақырыптың басында біз дәл осындай есептің шешуді қарастырдық, бірақ кортеждермен жұмыс істеу үшін. Әрі қарай, Python-да тізбекті өңдеу әдістерін салыстыру пайдалы болады.

```
sum=0
spisok=[1, 2, 3, 4, 5]
print(" Список ")
for i in spisok:
    print(i, end=" ")
for i in range(5):
    sum=sum+spisok[i]
print("\n Сумма элементов списка =  ", sum)
```

Тізімнен көріп отырғаныңыздай, бағдарламадағы мәтіндер мен сөйлемдер синтаксисі, іс жүзінде, өзгерген жоқ, тек пайдаланушы өңдеуге енгізген тізім элементтері тік жақшаға алынады.
Бағдарламаның нәтижесі 77-ші суретте көрсетілген.



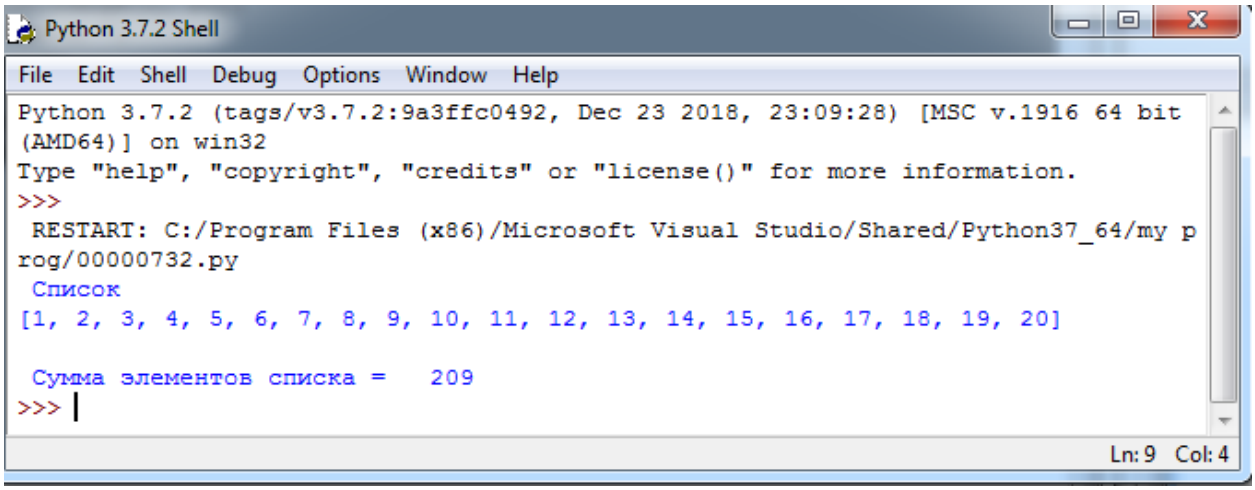
Сурет 77 – Тізім элементтерінің қосындысын табуға арналған бағдарламаның нәтижесі

Тізімдерді кездейсоқ құру мүмкіндігін қарастырайық. Оны бірнеше жолмен жасауға болады. Біріншіден, біз білетін **range** функциясында тізімнің бастапқы және соңғы мәндерін көрсетуге болады, ал **list** функциясы тізімді қайтарады. Содан кейін алдыңғы бағдарламаның коды келесі түрде болады:

```
sum=0
```

```
spisok=list(range(1, 21))
print(" Список ")
print(spisok)
for i in range(1, 20, 1):
    sum=sum+spisok[i]
print("\n Сумма элементов списка =  ", sum)
```

Бағдарлама жұмысының нәтижесі 78-ші суретте көрсетілген.

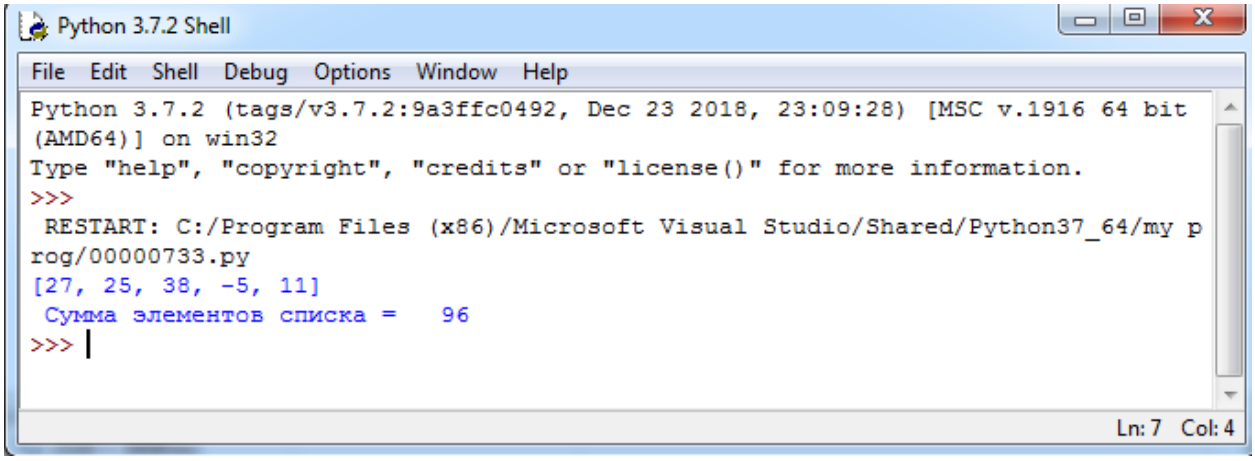


Сурет 78 – List(range(1, 21) көмегімен жасалған тізім элементтерінің қосындысын табуға арналған мысал бағдарламасының нәтижесі

Екіншіден, сіз **sample** функциясын қолдана аласыз, ол тізім элементтерінің бастапқы тізбегінен пайдаланушы көрсеткен элементтер санын қайтарады. Бұл жағдайда **random** модулін пайдалану керек, оны **import** нұсқаулығын қолдана отырып қосу керек. Төмендегі бағдарламада көрсетілгендей, бастапқы тізім он элементтен тұрады. Содан кейін **chislo=random.sample(spisok,5)** операторымен бастапқы тізімнен бес элемент туындайды.

```
import random
sum=0
spisok=[11, 25, 38, -5, 0, 12, -78, 27, 39, 19]
chislo=random.sample(spisok, 5)
print(chislo, end=" ")
for i in range(0, 5):
    sum=sum+chislo[i]
print("\n Сумма элементов списка =  ", sum)
```

Бағдарламма нәтижесі 79-ші суретте көрсетілген .

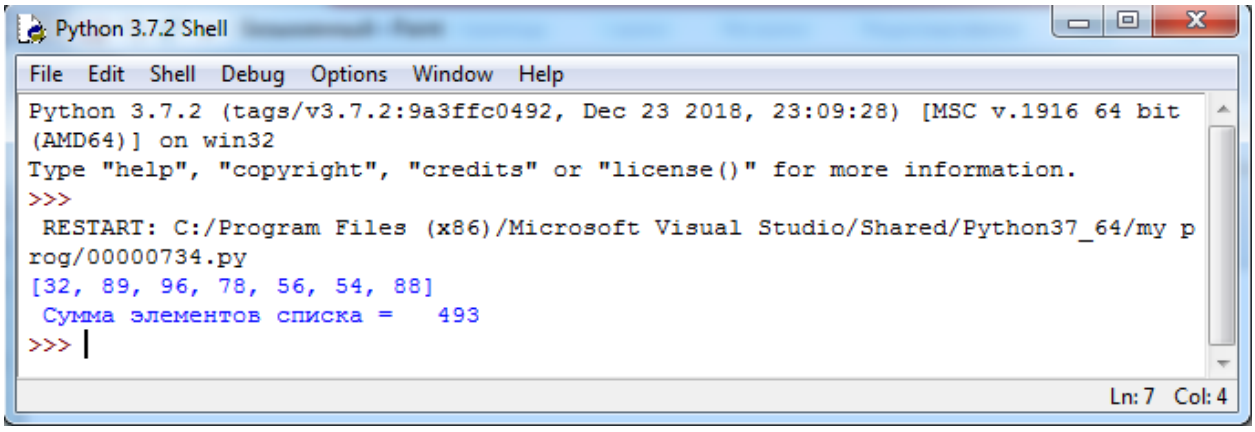


Сурет 79- **random.sample(spisok, 5)** көмегімен жасалған тізім элементтерінің қосындысын табуға арналған мысалдың нәтижесі.

Алдыңғы кодтарды комбинациялаңыз. Алдын ала анықталған тізімнің орнына біз оны range функциясын пайдаланып өндіреміз(генерациялаймыз) , содан кейін **sample** функциясын қолданып(7 санын параметрге айналдырамыз), осылайша тізім элементтерінің санын шектейміз (жетіге дейін).

```
import random
sum=0
chislo=random.sample(range(100), 7)
print(chislo, end=" ")
for i in range(0, 7):
    sum=sum+chislo[i]
print("\n Сумма элементов списка =  ", sum)
```

Бағдарлама жұмысының нәтижесі 80-ші суретте көрсетілген.



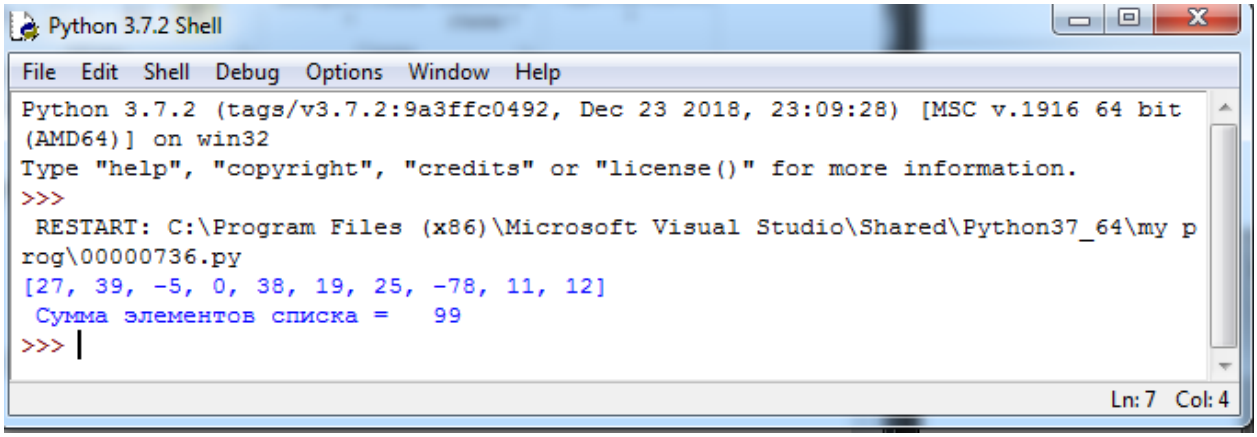
Сурет 80- **0-ден 99**-ға дейінгі диапазонда құрылған 7 элементтің қосындысын табуға арналған мысалдың нәтижесі

Үшіншіден, **random** модулінде тізімді кездейсоқ араластыруға болатын

shuffle функциясы бар.

```
import random
sum=0
spisok=[11, 25, 38, -5, 0, 12, -78, 27, 39, 19]
random.shuffle(spisok)
print(spisok, end=" ")
for i in range(0, 5):
   sum=sum+spisok[i]
print("\n Сумма элементов списка =  ", sum)
```

Бағдарлама жұмысының нәтижесі 81-ші суретте көрсетілген.

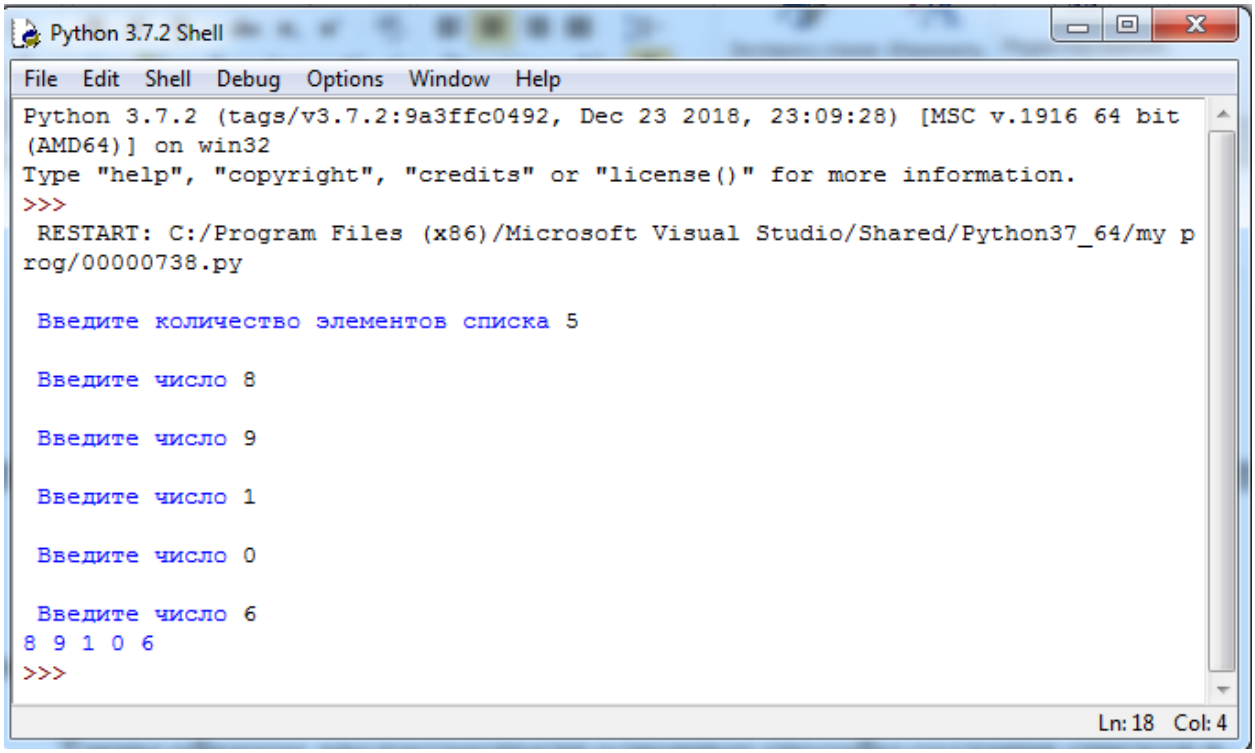


Сурет 81- Бастапқы тізімнен алынған 5 элементтің қосындысын табуға арналған мысал бағдарламасының нәтижесі

Бағдарламада тізімдерді құрудың қарастырылған тәсілдерінен басқа, сіз динамикалық тізімді құру әдісін қолдана аласыз. Төмендегі кодта көрсетілгендей, бос тізімді алдымен **sp = []** операторы жариялайды. Содан кейін пайдаланушыдан тізім элементтерінің саны сұралады. Циклде пайдаланушы тізім элементтерін енгізе бастайды, ал **append()** әдісі оларды тізімге қосуға мүмкіндік береді.

```
sp=[]
n=int(input("\n Введите количество элементов списка "))
for i in range(n):
   chislo=int(input("\n Введите число "))
   sp.append(chislo) # Добавляем элементы списка
for i in range(n): # Выводим элемента списка
   print(sp[i], end=" ")
```

Бағдарлама жұмысының нәтижесі 82-ші суретте көрсетілген.



Сурет 82 –Тізімді «қолмен» құру бағдарламасының нәтижесі»

Осылайша, біз Python-да тізімдер құрудың негізгі тәсілдерін қарастырдық. Тізімдерді құрудың жоғарыда қарастырылған негізгі әдістерін біріктіру арқылы тізімдерді орнатуға болады. Әрі қарай осы әдістердің кейбіреулері есептерді шешуде қолданылады.

Python-да тізімдерімен жұмыс істеу үшін әдістер қарастырылған, олардың кейбіреулері 6-шы кестеде келтірілген. Олардың қолданылуын мысалдармен қарастыраыйқ.

Кесте 6 – Тізімдер жұмыс істеу әдістері

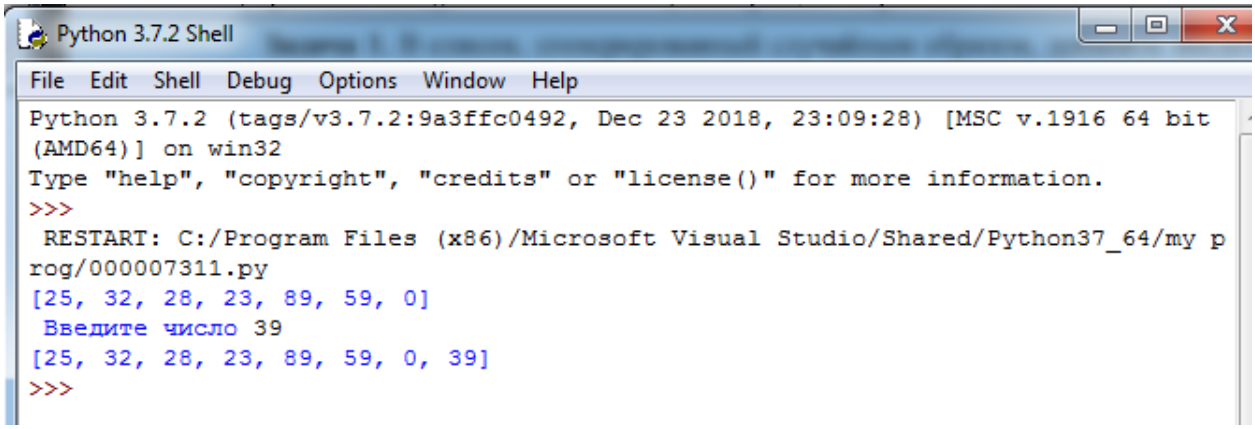
Әдіс	Әдістің сипаттамасы
spisok.append(x)	Тізімнің соңына **x** мәнін қосады
spisok.insert(i, x)	**i** позициясына **x** мәнін қосады
spisok.extend(spisokl)	Spisokl тізімінің барлық элементтерін соңына қосу арқылы тізімді кеңейтеді
spisok.remove(x)	**x** мәні бар тізімдегі бірінші элементті жояды
spisok.pop(i)	**i** нөмірінің мәнін позицияға қайтарады және сонымен бірге оны тізімнен жояды. Егер аргумент берілмесе, тізімнің соңғы элементі қайтарылады және жойылады
spisok.count(x)	**x** мәні бар элементтер санын қайтарады
spisok.sort([reverse = True])	Элементтерді өсу бойынша сұрыптайды. **Reverse** параметрі міндетті емес және логикалық мәндерді қабылдайды. Егер сіз **True** мәнін берсеңіз, тізім кему бойынша сұрыпталады
spisok.reverse()	Кері тізімді қайтарады

Есеп 7.3.1. Кездейсоқ құрылған тізімге пайдаланушы енгізген элементті қосыңыз

Шешімі. Төмендегі бағдарламадан көріп отырғаныңыздай, алдымен тізімнің генерациясы жасалады, содан кейін сан сұралады және **append** әдісін қолдана отырып, тізімнің соңына сан қосылады.

```
import random
spisok=random.sample(range(100), 7)
print(spisok, end=" ")
chislo=int(input("\n Введите число "))
spisok.append(chislo) # Добавляем введенное число в конец списка
print(spisok, end=" ")
```

Бағдарлама жұмысының нәтижесі 83-ші суретте көрсетілген



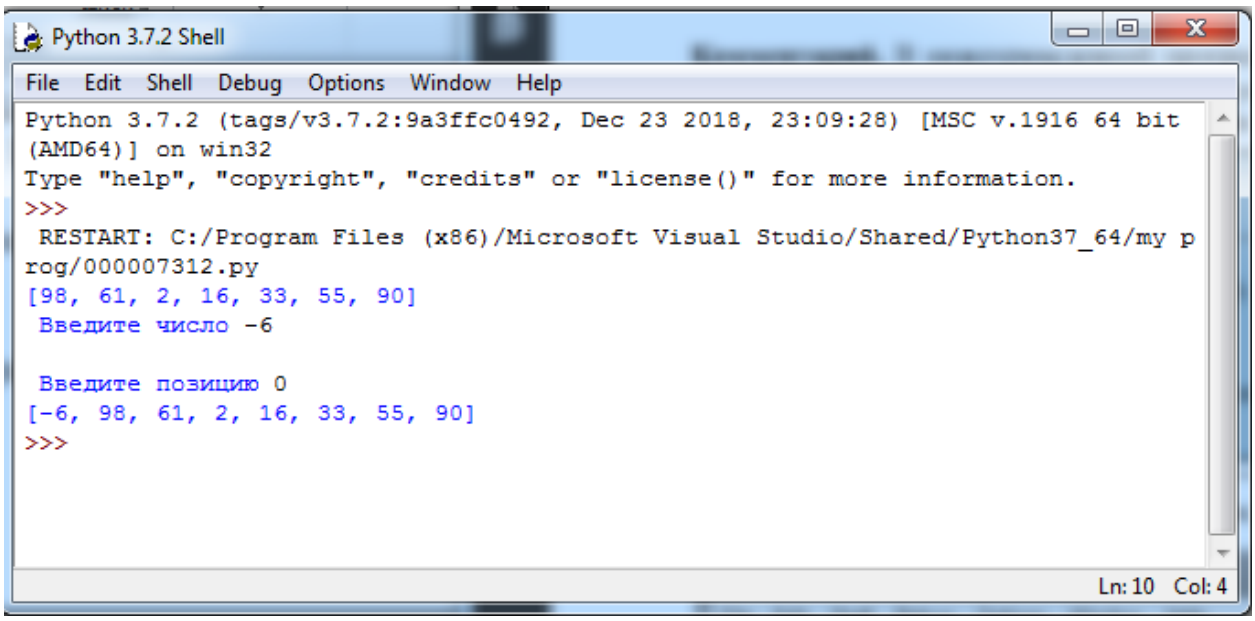
Сурет 83 – Тізімнің соңына санды қосу бағдарламасының нәтижесі

Есеп 7.3.2. Кездейсоқ құрылған тізімге пайдаланушы енгізген элементті көрсетілген позицияға қосыңыз.

Шешімі. Төмендегі бағдарламада алдымен тізім жасалады, содан кейін, санды қосуға арналған , сан мен позиция нөмірі сұралады. Содан кейін **insert()** әдісін қолдана отырып тиісті **poz ,chislo** параметрлері бар, санды көрсетілген позицияға енгізеді.

```
import random
spisok=random.sample(range(100), 7)
print(spisok, end=" ")
chislo=int(input("\n Введите число ")) # Вводим число
poz=int(input("\n Введите позицию ")) # Вводим позицию для добавления введенного выше числа в список
spisok.insert(poz, chislo) # Добавляем введенное число
print(spisok, end=" ")
```

Бағдарлама жұмысының нәтижесі 84-ші суретте көрсетілген



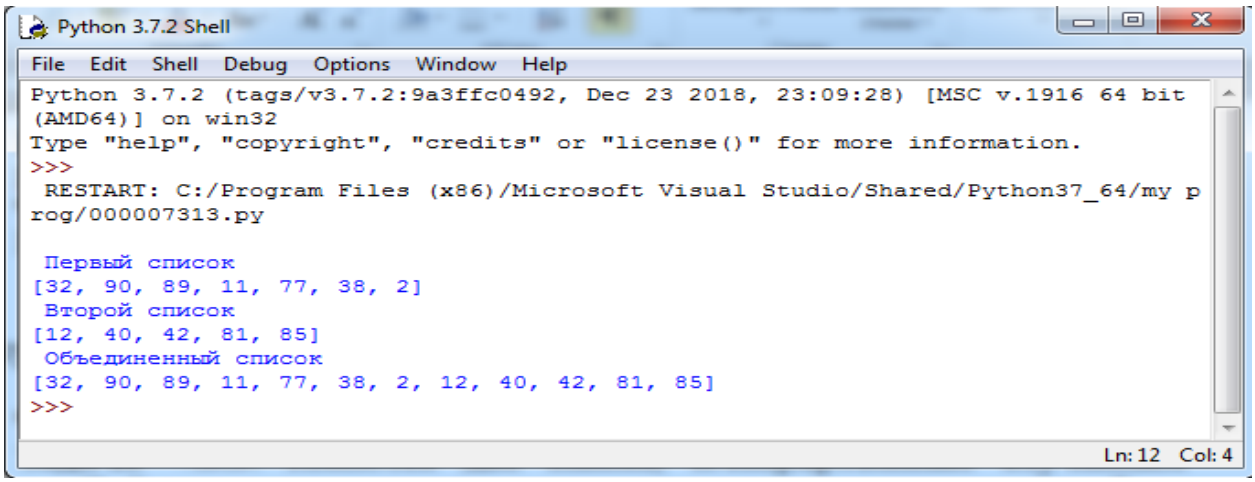
Сурет 84 – Енгізілген санды тізімдегі енгізілген позицияға қосу бағдарламасының нәтижесі

Есеп 7.3.3. Кездейсоқ құрылған екі тізім бар. Бірінші тізімнің соңына екінші тізімнің барлық элементтерін қосыңыз.

Шешімі . Алдымен **spisokl** және **spisok2** кездейсоқ құрылған екі тізімді іске асырамыз. Бірінші тізімге **extend ()** әдісін қолдану арқылы біз екінші тізімнің элементтерімен оны кеңейтеміз.

```
import random
spisok1=random.sample(range(100), 7)
print("\n Первый список ")
print(spisok1, end=" ")
spisok2=random.sample(range(100), 5)
print("\n Второй список ")
print(spisok2, end=" ")
spisok1.extend(spisok2) # Добавляем к первому списку элементы второго списка
print("\n Объединенный список ")
print(spisok1, end=" ")
```

Бағдарлама жұмысының нәтижесі 85-ші суретте көрсетілген



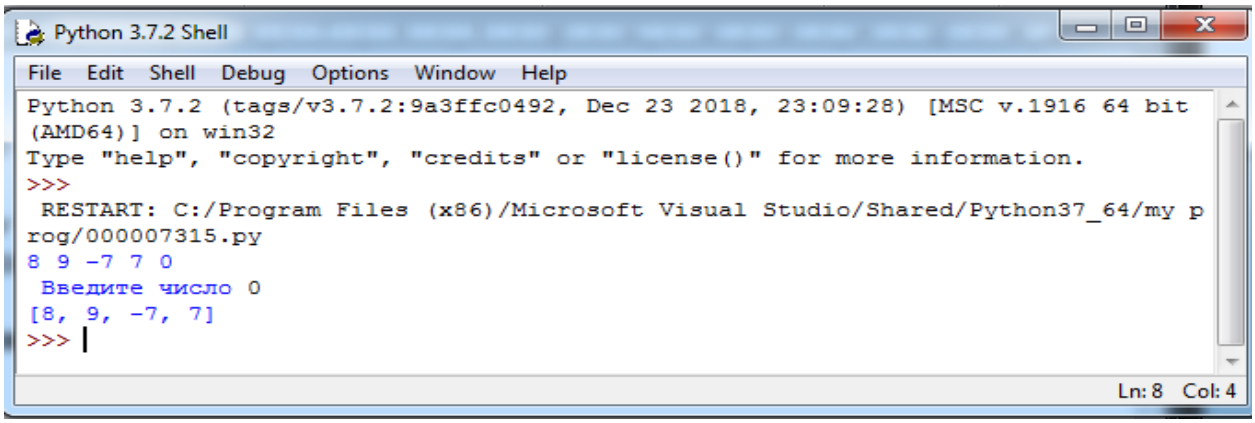
Сурет 85 – Басқа тізімнен тізімге қосу бағдарламасының нәтижесі

Есеп 7.3.4. Алдын ала құрылған тізімнен пайдаланушы енгізген элементті жою керек

Шешімі. Бастапқы тізімде бес элемент болады. Пайдаланушы жойылатын элементтің мәнін енгізеді. Тізімге қолданылған **remove ()** әдісі көрсетілген әрекеттерді жүзеге асырады.

```
spisok=[8, 9, -7, 7, 0]
for i in spisok:
    print(i, end=" ")
chislo=int(input("\n Введите число "))
spisok.remove(chislo)
print(spisok, end=" ")
```

Бағдарлама жұмысының нәтижесі 86-шы суретте көрсетілген



Сурет 86 – Енгізілген санды тізімінен жою бойынша бағдарламаның нәтижесі

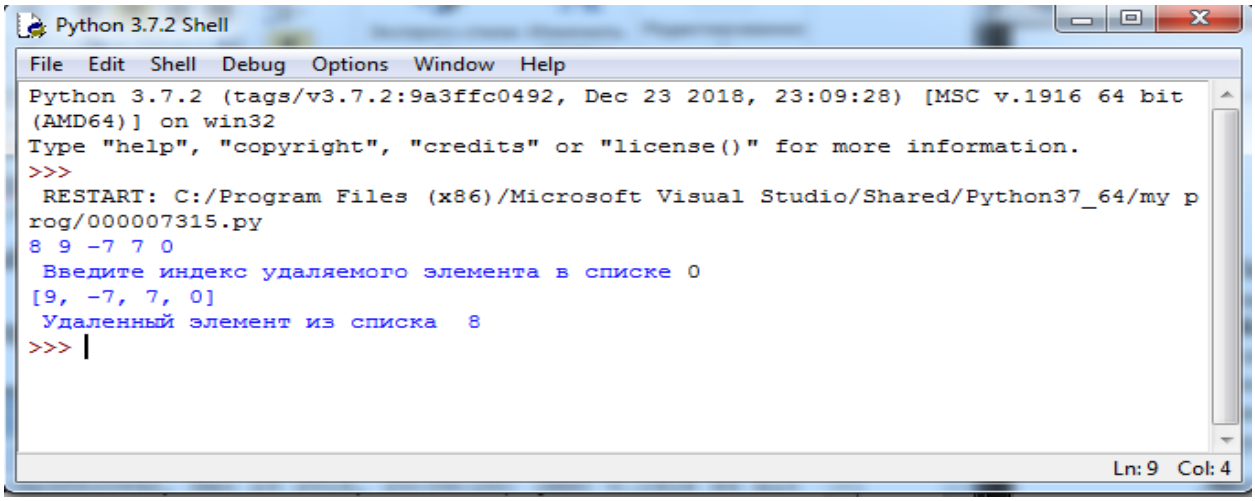
Есеп 7.3.5. Бастапқы тізімнен пайдаланушы көрсеткен позициядан элементті жою керек.

Шешімі . Бастапқы тізім бес элементтен тұрады. Пайдаланушы жойылатын элементтің индексін енгізеді. Егер аргумент **pop ()** әдісіне берілмесе, онда

тізімнің соңғы элементі қайтарылады және жойылады. Алайда, **pop ()** әдісінің параметрі ретінде біз элементтің индексін көрсетеміз. Экранға нәтижелік тізімнен басқа, жойылған элементтің мәнін шығарамыз .

```
spisok=[8, 9, -7, 7, 0]
for i in spisok:
    print(i, end=" ")
ind=int(input("\n Введите индекс удаляемого элемента в списке "))
chislo=spisok.pop(ind)
print(spisok, end=" ")
print("\n Удаленный элемент из списка ", chislo)
```

Бағдарлама жұмысының нәтижесі 87-ші суретте көрсетілген



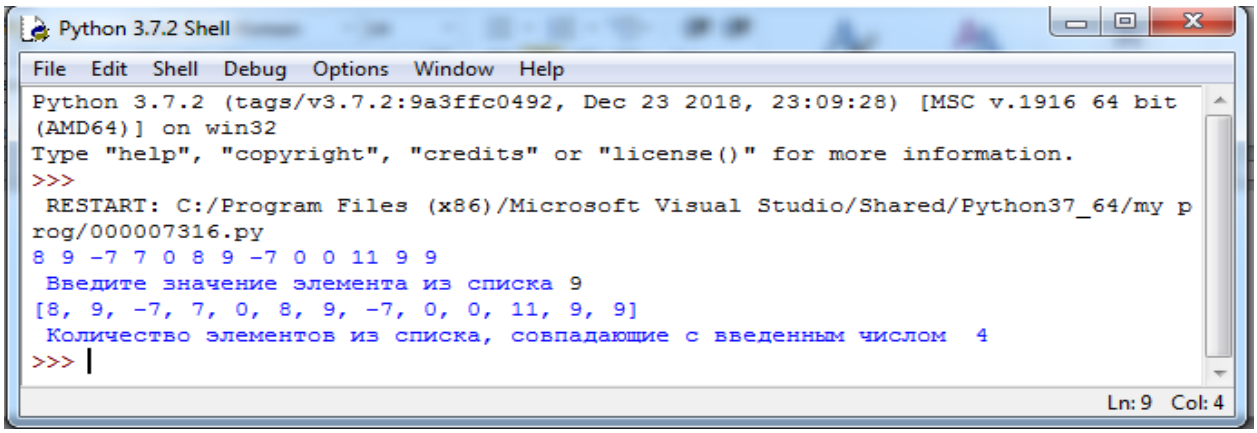
Сурет 87 – позиция бойынша тізімнен санды жою бағдарламаның нәтижесі

Есеп 7.3.6. Бастапқы тізімде белгілі бір мәні бар элементтер санын есептеңіз.

Шешімі . Біз тізімді бірнеше қайталанатын элементтерді кірістіру арқылы құрастырдық. Пайдаланушы енгізген **znach** мәні- **count ()** әдісінің параметрі , ол **znach** элементінің қайталану санын қайтарады.

```
spisok=[8, 9, -7, 7, 0, 8, 9, -7, 0, 0, 11, 9, 9]
for i in spisok:
    print(i, end=" ")
znach=int(input("\n Введите значение элемента из списка "))
kol=spisok.count(znach)
print(spisok, end=" ")
print("\n Количество элементов из списка, совпадающие с введенным числом ", kol)
```

Бағдарлама жұмысының нәтижесі 88-ші суретте көрсетілген



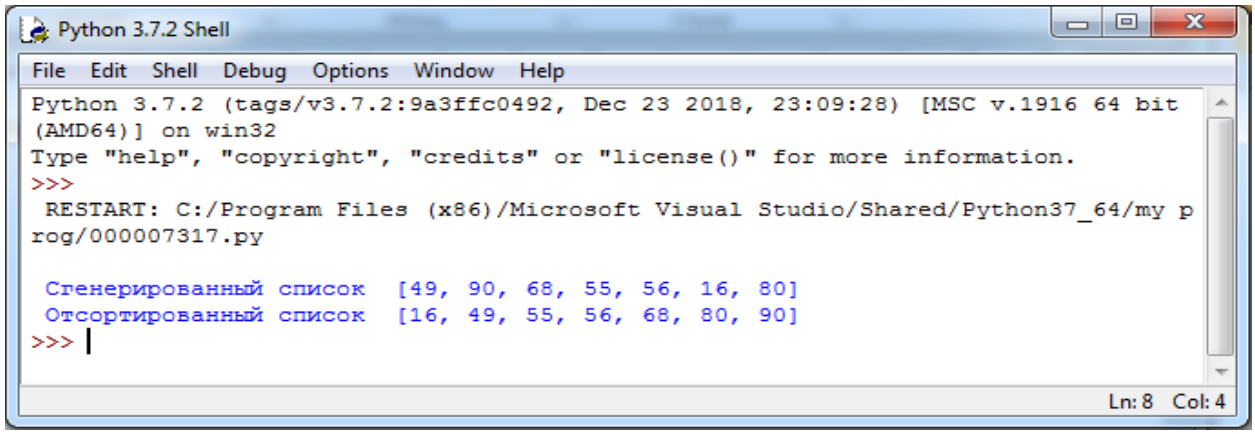
Сурет 88 – Енгізілген санның тізімінде қайталануларды табу бағдарламасының нәтижесі

Есеп 7.3.7.Кездейсоқ құрылған тізімді өсу бойынша сұрыптау керек.

Шешімі . Python-да тізімді сұрыптау үшін міндетті емес **reverse** параметрі бар **sort ()** әдісін қолданады. Төмендегі кодта ол **False** жағдайда орнатылған , бұл тізімді өсу бойынша сұрыптауға мүмкіндік береді.

```
import random
spisok=random.sample(range(100), 7)
print("\n Сгенерированный список ", spisok, end=" ")
spisok.sort(reverse=False)
print("\n Отсортированный список ", spisok, end=" ")
```

Бағдарлама жұмысының нәтижесі 89-ші суретте көрсетілген



Сурет 89 – Тізімді өсу бойынша сұрыптау бағдарламасының нәтижесі

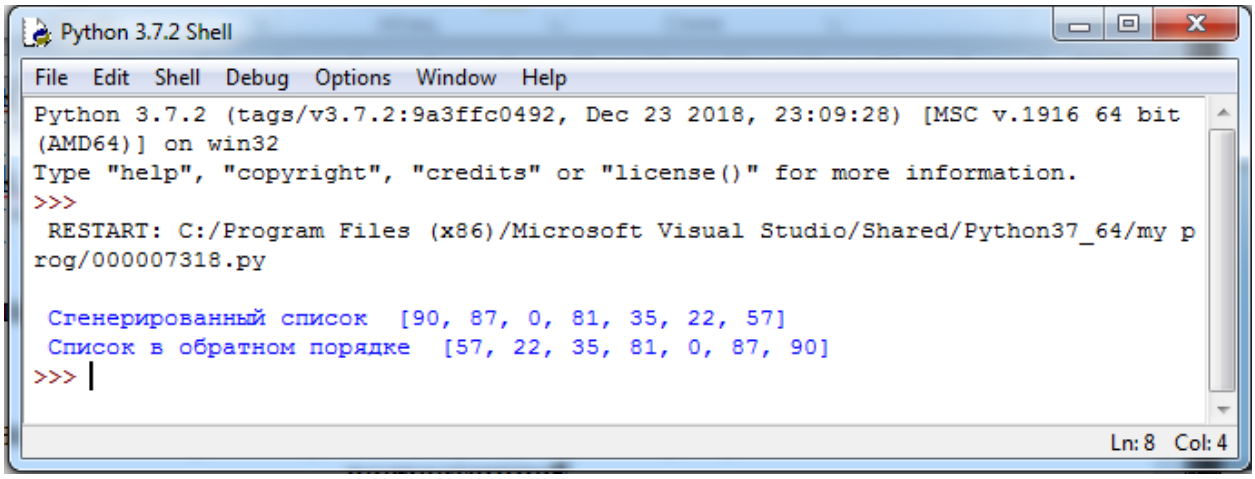
Есесп 7.3.8. Кездейсоқ түрде жасалған тізімді кері ретпен көрсетіңіз.

Шешімі. Кездейсоқ құрылған бастапқы тізімге қолданылған параметрлерсіз **reverse** әдісі кері тізімді қайтарады.

```
import random
```

```
spisok=random.sample(range(100), 7)
print("\n Сгенерированный список ", spisok, end=" ")
spisok.reverse()
print("\n Список в обратном порядке ", spisok, end=" ")
```

Бағдарлама жұмысының нәтижесі 90-шы суретте көрсетілген



Сурет 90 – Кері ретімен тізімді шығару бағдарламасының нәтижесі