

## 7 РАБОТА С КОРТЕЖАМИ И СПИСКАМИ

### 7.1 Объявление кортежей

**Кортежи** в языке Python аналогичны структурам, которые называются **массивами**. Приемы работы с массивами рассматриваются при обучении таким популярным языкам программирования, как C++, Microsoft Visual Basic, Delphi и др. Стоит отметить, что массивы в них определяются как формальное объединение нескольких однотипных объектов (чисел, символов, строк и т. п.), рассматриваемое как единое целое.

В Python **кортеж** определяется как один из видов последовательностей, с которыми можно работать в этом языке программирования. Принципиальное отличие от привычных массивов данных в других языках программирования заключается в том, что последовательность данных, объединенная в кортеж, **не позволяет изменять свои значения**. Но в этом есть и определенные плюсы. Во-первых, увеличивается скорость обработки элементов кортежа, поскольку системе заранее известно, что значения не будут изменяться. Во-вторых, такая структура, как кортеж, может применяться для образования других структур: например, словари. В-третьих, кортеж занимает меньше памяти, чем, например, списки, и, кроме того, элементы кортежа защищены от случайных изменений.

Если массив представляет собой совокупность однотипных объектов, то кортеж в Python может содержать совершенно разнородные объекты, например, строковые и числовые значения или звуковые файлы, файлы изображений.

Синтаксис объявления кортежей следующий:

**Имя кортежа** = (элемент 1, элемент 2, ...элемент N)

Например

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

Возможна и другая запись, например, для значений строкового типа,

```
Имя кортежа=(«Элемент1»,  
                «Элемент 2»,  
                «Элемент 3»,  
                «Элемент N»)
```

причем в одной строке может располагаться несколько элементов.

**Доступ** к каждому элементу кортежа в программе осуществляется с помощью **индекса** - целого числа, служащего своеобразным именем элемента в кортеже. При упоминании в программе элемента кортежа сразу за его именем должен следовать индекс элемента в квадратных скобках,

например, `korteg[i]`

Важно отметить, что нумерация элементов кортежа начинается с нуля, а не с единицы. Операции по обработке кортежа осуществляются поэлементно в цикле с оператором **for**.

**Задача 7.1.1.** В кортеже, состоящего из заранее заданных целых чисел найдите сумму элементов.

**Решение.** Блок-схема алгоритма решения задачи представлена на рисунке 72.

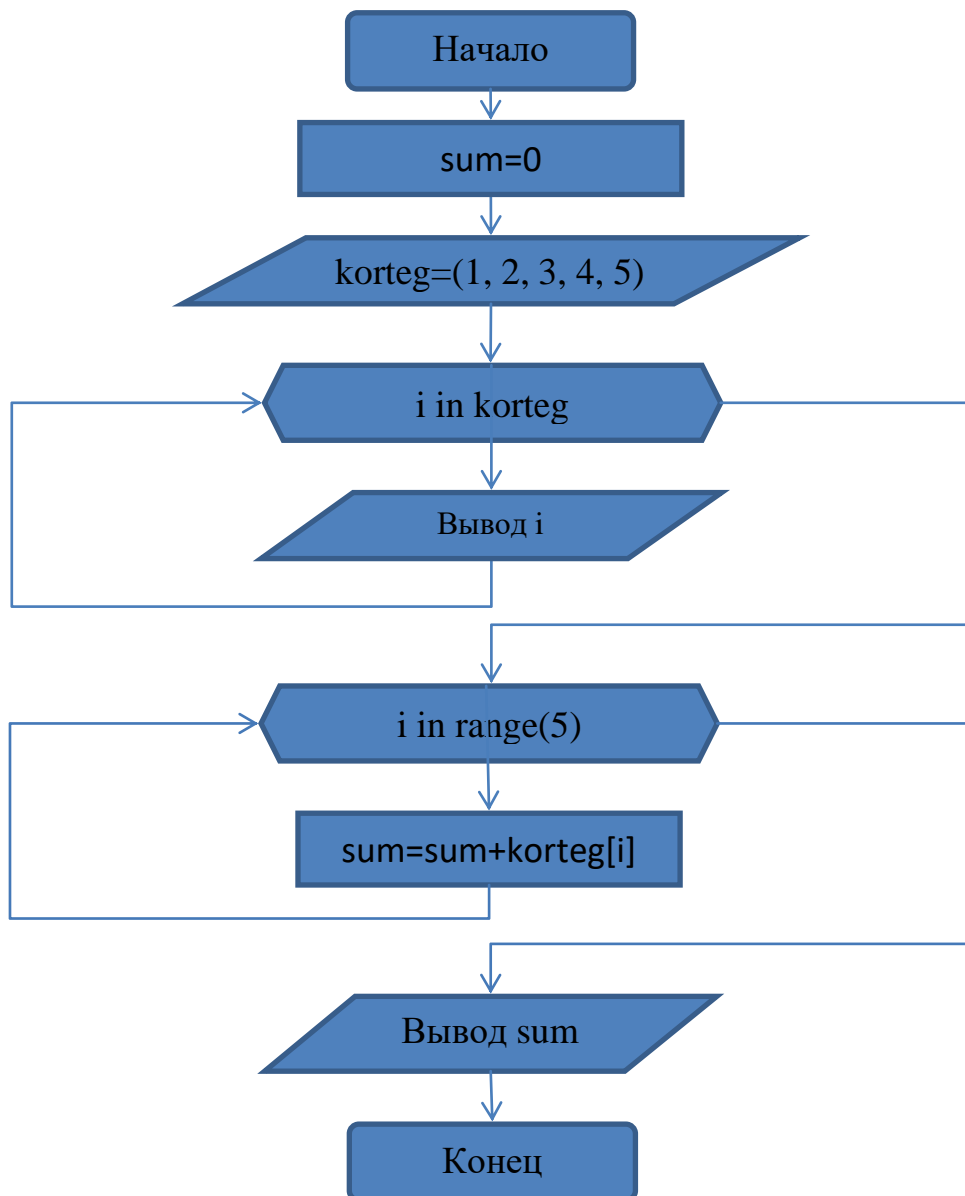


Рисунок 72 – Блок-схема алгоритма решения задачи 7.1.1

В листинге приведен код программы, отвечающий за решение задачи:

```
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)

```

Результат работы программы представлен на рисунке 73.

The screenshot shows a Python 3.7.2 Shell window with the following content:

```

Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000711.py
Кортеж
1 2 3 4 5
Сумма элементов кортежа = 15
>>> |

```

The window title is "Python 3.7.2 Shell". The status bar at the bottom right shows "Ln: 8 Col: 4".

Рисунок73 – Результат работы программы для нахождения суммы элементов кортежа в задаче 7.1.1

## 7.2 Классические способы обработки кортежей

Решение задач, связанных с обработкой последовательностей данных, базируется на их элементарных приемах обработки. Разбив задачу на логические части, можно значительно ускорить ее решение. Типичными задачами работы с кортежами являются определение наличия в нем заданного элемента, отбор элементов, удовлетворяющих определенным условиям и т. д.

Из всего многообразия существующих приемов, позволяющих манипулировать с данными, представленными в кортежах, можно выделить следующие:

1. Нахождение количества элементов при заданном условии.
2. Нахождение суммы значений элементов при заданном условии.
3. Нахождение произведения значений элементов при заданном условии.
4. Поиск экстремальных значений элементов кортежа (поиск максимального и/или минимального значения).
5. Объединение (сцепление) кортежей.
6. Обмен значений элементов в кортеже.
7. Срезы кортежей.

Ниже приведем реализацию перечисленных алгоритмов обработки

кортежей. Так как первые четыре способа просты в исполнении, а также учитывая, что данные алгоритмы неоднократно рассматривались при решении задач в предыдущих темах, то приведем их реализацию без комментариев. Стоит отметить, что в первых четырех листингах очередной элемент кортежа сравнивается с числом 10, однако на практике лучше организовать запрос условия, например, с помощью функции `input`.

*Нахождение количества элементов при заданном условии.*

Приведем пример нахождения количества элементов кортежа `korteg=(10, 8, 29, 35, 7)` больших или равных десяти. В листинге приведен код программы, отвечающий за решение данного примера:

```
kol=0
korteg=(10, 8, 29, 35, 7)
print(" Кортёж ")
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)
```

*Нахождение произведения значений элементов при заданном условии.*

Приведем пример нахождения суммы элементов кортежа `korteg=(10, 8, 29, 35, 7)` значения, которых больше или равно десяти. В листинге приведен код программы, отвечающий за решение данного примера:

```
sum=0
korteg=(10, 8, 29, 35, 7)
print(" Кортёж ")
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)
```

*Нахождение произведения значений элементов при заданном условии.*

Приведем пример нахождения произведения элементов кортежа `korteg=(10, 8, 29, 35, 7)` значения, которых больше или равно десяти. В листинге приведен код программы, отвечающий за решение данного примера:

```
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)

```

*Нахождение экстремальных значений в кортеже.*

Приведем пример нахождения максимального и минимального элементов кортежа `korteg=(10, 8, 29, 35, 7)`. В листинге приведен код программы, отвечающий за решение данного примера:

```

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)

```

*Объединение (сцепление) кортежей.*

**Задача 7.2.1.** Даны два кортежа. Требуется объединить их между собой.

**Решение.** Задача заключается в первоначальном формировании двух исходных кортежей и применении к первому кортежу операции сложения (+), после чего объединенный кортеж выводится на экран. В листинге приведен код программы, отвечающий за решение задачи:

```

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:

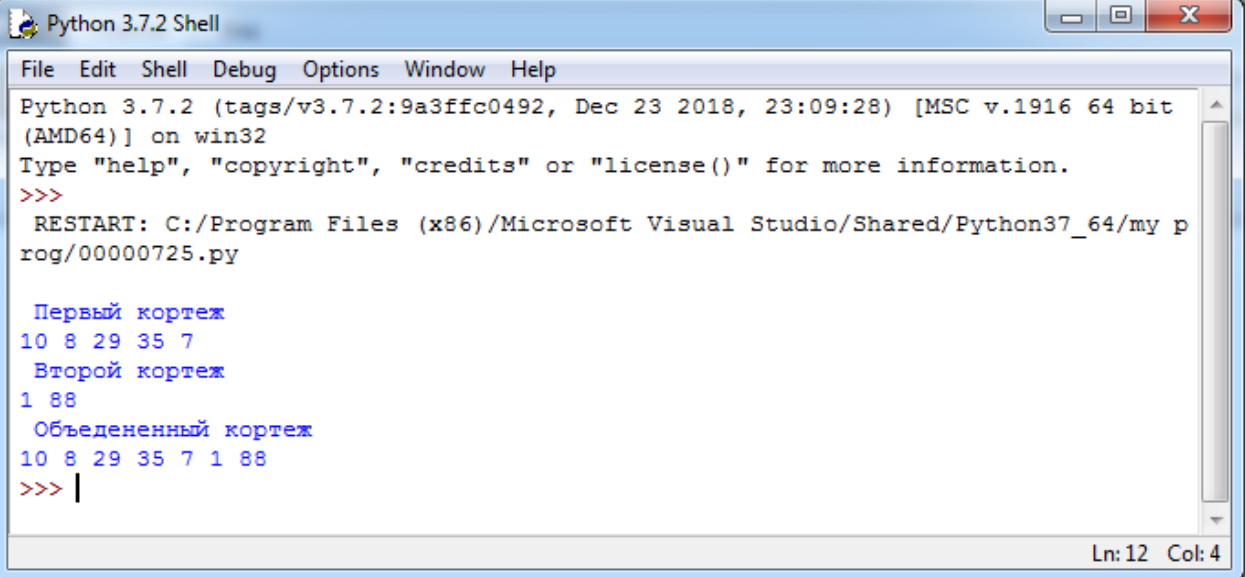
```

```

    print(i, end=" ")
print("\n Объединенный кортеж ")
korteg1 +=korteg2
for i in korteg1:
    print(i, end=" ")

```

Результат работы программы представлен на рисунке 74.



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000725.py

Первый кортеж
10 8 29 35 7
Второй кортеж
1 88
Объединенный кортеж
10 8 29 35 7 1 88
>>> |
Ln: 12 Col: 4

```

Рисунок 74 – Результат работы программы для объединения элементов двух кортежей в задаче 7.2.1

*Обмен значений элементов в кортеже.*

**Задача 7.2.2.** В кортеже целых чисел найдите максимальный и минимальный элементы, а также осуществите их обмен..

**Решение.** Приемы нахождения максимального и минимального элементов рассматривались ранее, поэтому стоит отметить, что сам обмен значений заключается в выполнении оператора **maxim,minim=minim,maxim**. В листинге приведен код программы, отвечающий за решение задачи:

```

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]

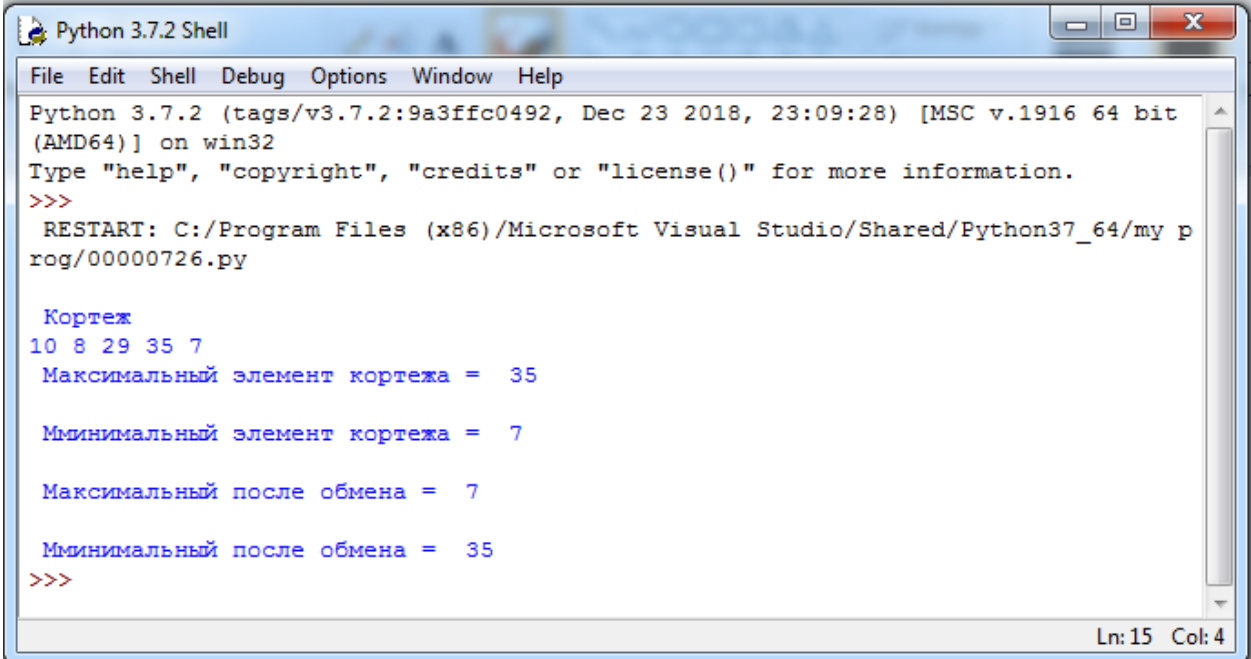
```

```

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)

```

Результат работы программы представлен на рисунке 75.



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000726.py

Кортеж
10 8 29 35 7
Максимальный элемент кортежа = 35

Мминимальный элемент кортежа = 7

Максимальный после обмена = 7

Мминимальный после обмена = 35
>>>
Ln:15 Col:4

```

Рисунок 75 – Результат работы программы задачи 7.2.2

### *Срезы кортежей.*

Срез кортежа получается в результате вывода элементов кортежа, находящихся между заранее заданными начальной **a** и конечной **b** позициями элементов. Реализация среза в программе осуществляется с помощью оператора **print(korteg[a:b])**.

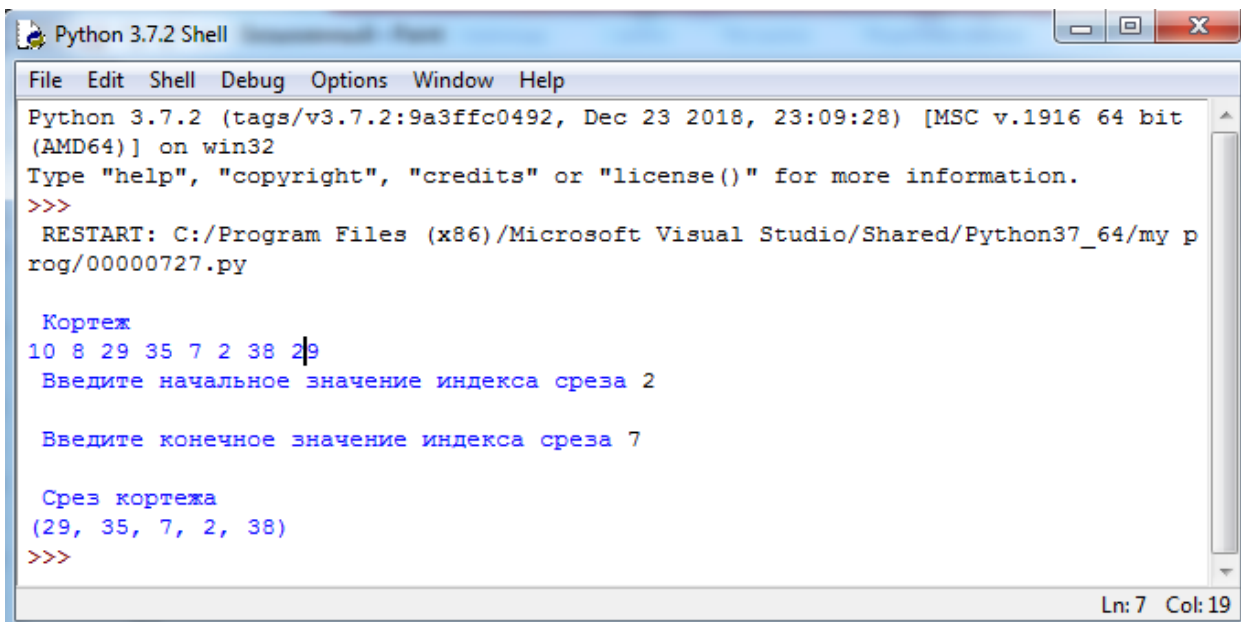
Приведем пример среза кортежа `korteg=(10, 8, 29, 35, 7, 2, 38, 29)`. В листинге приведен код программы, отвечающий за решение данного примера:

```

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])

```

Результат работы программы представлен на рисунке 76.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000727.py

Кортеж
10 8 29 35 7 2 38 2|
Введите начальное значение индекса среза 2
Введите конечное значение индекса среза 7

Срез кортежа
(29, 35, 7, 2, 38)
>>>
```

Рисунок 76 – Результат работы программы примера на срез кортежа

### 7.3 Работа со списками

Списки в Python функционируют подобно кортежам, с той разницей, что являются изменяемой последовательностью. Таким образом, если такие действия, как добавление нового элемента, удаление элемента, сортировка и др., в кортеже невозможны, то методы работы со списком позволяют это сделать.

Точно так же, как и элементы кортежа, элементы списка содержат лишь ссылку на объект, поэтому в списки могут входить разные типы данных. Данный подход отличается от обработки традиционных массивов в других языках программирования, хотя сходство есть в методах обработки списков в Python с методами обработки массивов.

Рассмотрим синтаксис объявления списков.

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

Например,

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

Типичными задачами при работе со списками являются:

- определение факта наличия в них заданного элемента;
- отбор элементов, удовлетворяющих определенным условиям.

В обоих случаях используется циклическое сравнение элементов списка с заданным образцом. Для определения факта наличия заданного образца в списке достаточно единственного совпадения, после чего дальнейший просмотр прекращается. Если условие отбора может



выполняться для нескольких элементов списка, то необходим просмотр всего списка до конца.

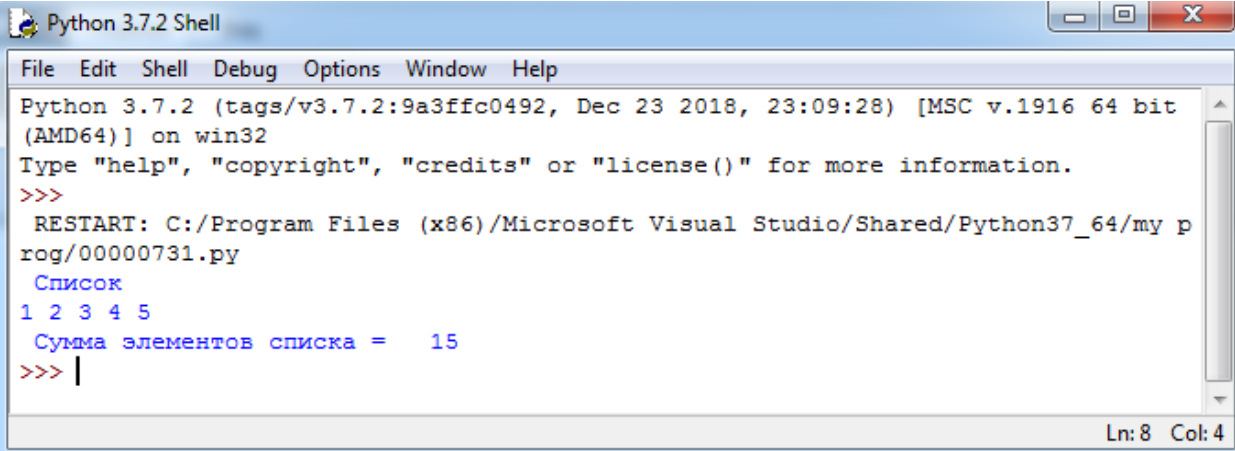
Все рассмотренные выше приемы, позволяющие манипулировать с данными, представленными в кортежах, действительны и для списков, однако их набор может быть существенно расширен в силу изменяемости списков. В частности, можно удалять и добавлять элементы списка, сортировать их.

Пользователь может создавать список, размещая его элементы, которые необходимо обработать, в квадратных скобках (согласно синтаксису, показанному выше), а может генерировать случайным образом. Рассмотрим эти способы подробнее на примере задачи «Найти сумму элементов списка». Ранее в этой теме мы рассмотрели решение точно такой же задачи, но для работы с кортежами. Здесь и далее будет полезно сравнить способы обработки последовательностей в Python.

```
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)
```

Как видно из листинга, текст и синтаксис операторов в программе, по сути, не изменились, за исключением того, что элементы списка, которые ввел пользователь для обработки, заключены в квадратные скобки.

Результат работы программы представлен на рисунке 77.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000731.py
Список
1 2 3 4 5
Сумма элементов списка = 15
>>> |
```

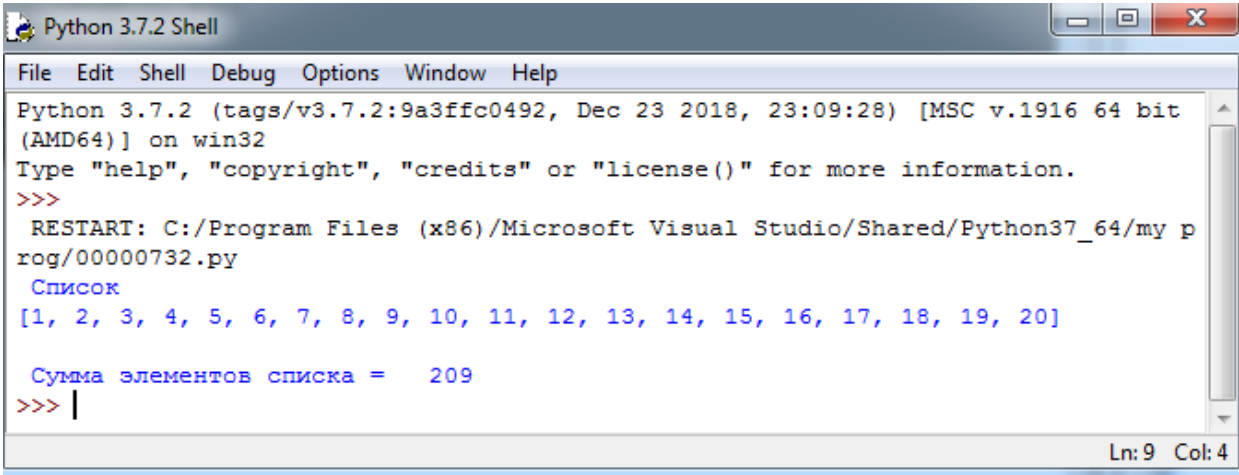
Рисунок 77 – Результат работы программы примера на нахождение суммы элементов списка

Рассмотрим возможность генерации списков случайным образом. Ее можно осуществить несколькими способами. Во-первых, в известной нам

функции **range** можно указать начальное и конечное значения списка, а функция **list** возвратит список. Тогда код предыдущей программы будет выглядеть так:

```
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)
```

Результат работы программы представлен на рисунке 78.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000732.py
Список
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

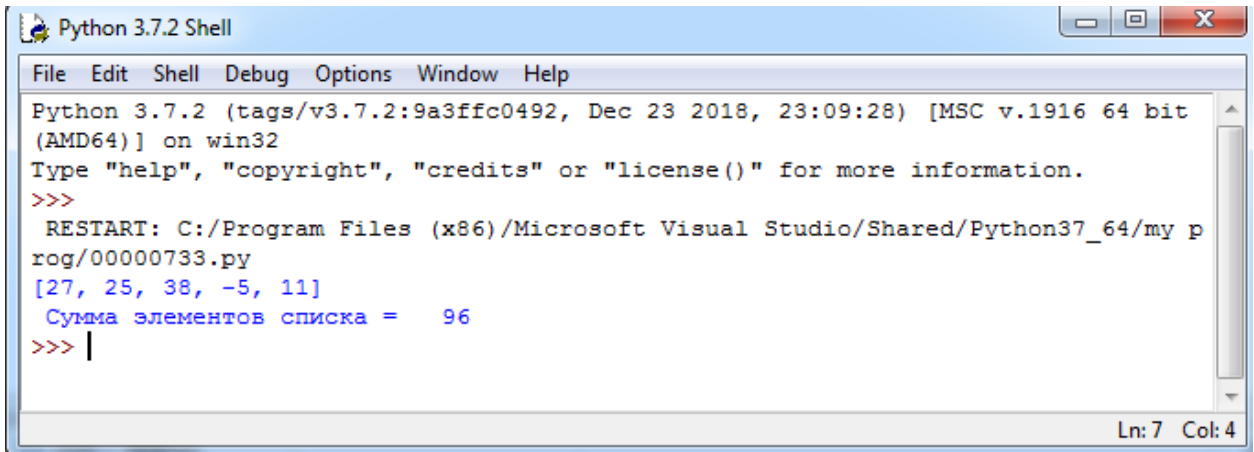
Сумма элементов списка = 209
>>> |
```

Рисунок 78 – Результат работы программы примера на нахождение суммы элементов списка сгенерированных при помощи **list(range(1, 21))**

Во-вторых, можно воспользоваться функцией **sample**, которая из исходной последовательности элементов списка будет возвращать указанное пользователем количество элементов. При этом необходимо воспользоваться модулем **random**, подключив его с помощью инструкции **import**. Как показано в нижеследующей программе, исходный список состоит из десяти элементов. Затем оператором **chislo=random.sample(spisok,5)** генерируются пять элементов из исходного списка.

```
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)
```

Результат работы программы представлен на рисунке 79.



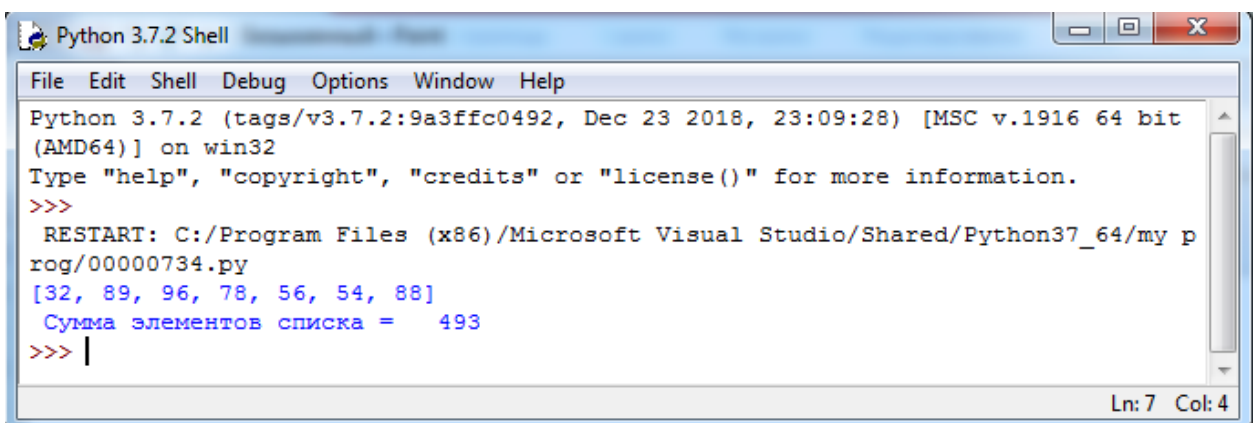
```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000733.py
[27, 25, 38, -5, 11]
Сумма элементов списка = 96
>>> |
```

Рисунок 79 – Результат работы программы примера нахождение суммы элементов списка сгенерированных при помощи **random.sample(spisok, 5)**

Скомбинируем предыдущие коды. Вместо заранее заданного списка сгенерируем его, воспользовавшись функцией **range**, а затем функцией **sample** (сделаем параметром число 7), тем самым ограничив количество элементов списка (до семи).

```
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)
```

Результат работы программы представлен на рисунке 80.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000734.py
[32, 89, 96, 78, 56, 54, 88]
Сумма элементов списка = 493
>>> |
```

Рисунок 80 – Результат работы программы примера нахождение суммы 7 элементов сгенерированных в диапазоне от 0 до 99

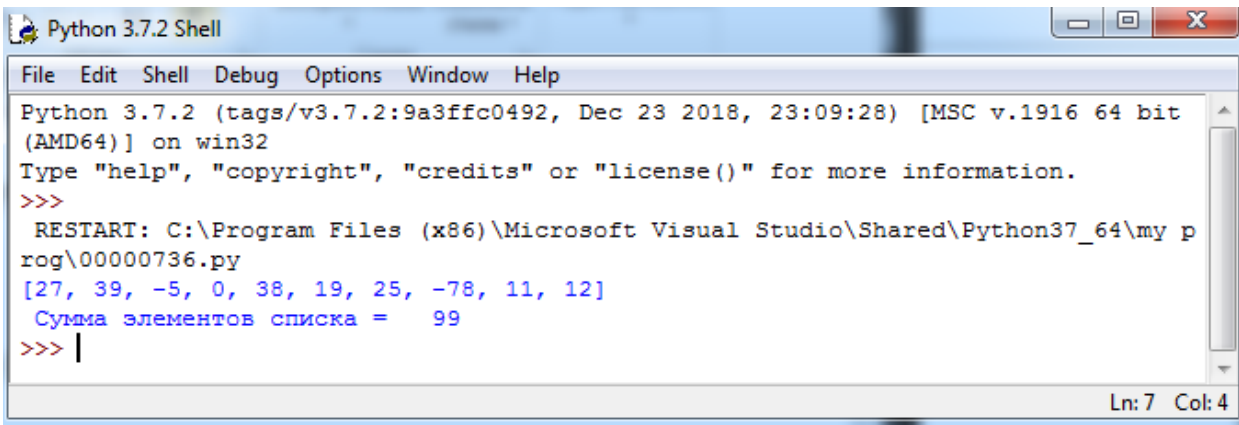
В-третьих, в модуле **random** есть функция **shuffle** с помощью, которой можно перемешать список случайным образом.

```

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)

```

Результат работы программы представлен на рисунке 81.



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\my p
rog\00000736.py
[27, 39, -5, 0, 38, 19, 25, -78, 11, 12]
Сумма элементов списка = 99
>>> |
Ln: 7 Col: 4

```

Рисунок 81 – Результат работы программы примера на нахождение суммы 5 элементов сгенерированных из исходного списка

Кроме рассмотренных способов создания списков в программе, можно применить прием динамического создания списка. Как показано в нижеприведенном коде, сначала объявляется пустой список оператором **sp=[]**. Затем у пользователя запрашивается количество элементов списка. В цикле пользователь начинает вводить элементы списка, а метод **append()** позволяет добавить их в список.

```

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=" ")

```

Результат работы программы представлен на рисунке 82.

```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000738.py

Введите количество элементов списка 5

Введите число 8

Введите число 9

Введите число 1

Введите число 0

Введите число 6
8 9 1 0 6
>>>
Ln: 18 Col: 4

```

Рисунок 82 – Результат работы программы создания списка «вручную»

Таким образом, мы рассмотрели основные способы создания списков в Python. Можно задавать списки, комбинируя рассмотренные выше основные способы создания списков. В дальнейшем некоторые из этих способов будут применяться при решении задач.

Для работы со списками в Python предусмотрены методы, некоторые из которых приведены в табл. 6. Рассмотрим их применение на примерах.

Таблица 6 - Методы работы со списками

Метод	Описание метода
<code>spisok.append(x)</code>	Добавляет значение <b>x</b> в конец списка
<code>spisok.insert(i, x)</code>	Вставляет значение <b>x</b> в позицию <b>i</b>
<code>spisok.extend(spisok1)</code>	Расширяет список, добавляя в конец все элементы списка <code>spisok1</code>
<code>spisok.remove(x)</code>	Удаляет первый элемент в списке, имеющий значение <b>x</b>
<code>spisok.pop(i)</code>	Возвращает значение в позиции номер <b>i</b> и одновременно удаляет его из списка. Если аргумент не передан, возвращается и удаляется последний элемент списка
<code>spisok.count(x)</code>	Возвращает количество элементов со значением <b>x</b>
<code>spisok.sort([reverse = True])</code>	Сортирует элементы по возрастанию. Параметр <b>reverse</b> является необязательным и принимает логические значения. Если передать значение <b>True</b> , то список сортируется по убыванию
<code>spisok.reverse()</code>	Возвращает обратный список

**Задача 7.3.1.** В список, сгенерированный случайным образом, добавить введенный пользователем элемент.

**Решение.** Как видно из нижеприведенной программы, сначала происходит генерация списка, затем запрашивается число, и с помощью метода **append** происходит добавление числа в конец списка.

```
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=" ")
```

Результат работы программы представлен на рисунке 83.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007311.py
[25, 32, 28, 23, 89, 59, 0]
Введите число 39
[25, 32, 28, 23, 89, 59, 0, 39]
>>>
```

Рисунок 83 – Результат работы программы добавления числа в конец списка

**Задача 7.3.2.** В список, сгенерированный случайным образом, добавить введенный пользователем элемент на указанную позицию.

**Решение.** В нижеприведенной программе, сначала происходит генерация списка, затем запрашивается число и номер позиции, на которую следует добавить число. Затем с помощью метода **insert()** с соответствующими параметрами **poz**, **chislo** выполняется вставка числа на указанную позицию.

```
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=" ")
```

Результат работы программы представлен на рисунке 84.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007312.py
[98, 61, 2, 16, 33, 55, 90]
Введите число -6

Введите позицию 0
[-6, 98, 61, 2, 16, 33, 55, 90]
>>>
Ln: 10 Col: 4
```

Рисунок 84 – Результат работы программы добавления введенного числа на введенную позицию в списке

**Задача 7.3.3.** Имеются два списка, сгенерированные случайным образом. Добавьте в конец первого списка все элементы второго списка.

**Решение.** Для начала реализуем сгенерированные случайным образом два списка **spisok1** и **spisok2**. За счет использования метода **extend()**, примененного к первому списку, мы расширяем его элементами второго списка.

```
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=" ")
```

Результат работы программы представлен на рисунке 85.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007313.py

Первый список
[32, 90, 89, 11, 77, 38, 2]
Второй список
[12, 40, 42, 81, 85]
Объединенный список
[32, 90, 89, 11, 77, 38, 2, 12, 40, 42, 81, 85]
>>>
Ln: 12 Col: 4
```

Рисунок 85 – Результат работы программы добавления к списку другого списка

**Задача 7.3.4.** Из заранее сформированного списка следует удалить элемент, введенный пользователем.

**Решение.** Исходный список будет содержать пять элементов. Пользователь вводит значение элемента, подлежащего удалению. Метод `remove()`, примененный к списку, осуществляет указанные действия.

```
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=" ")
```

Результат работы программы представлен на рисунке 86.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007315.py
8 9 -7 7 0
Введите число 0
[8, 9, -7, 7]
>>> |
Ln: 8 Col: 4
```

Рисунок 86 – Результат работы программы по удалению из списка введенного числа

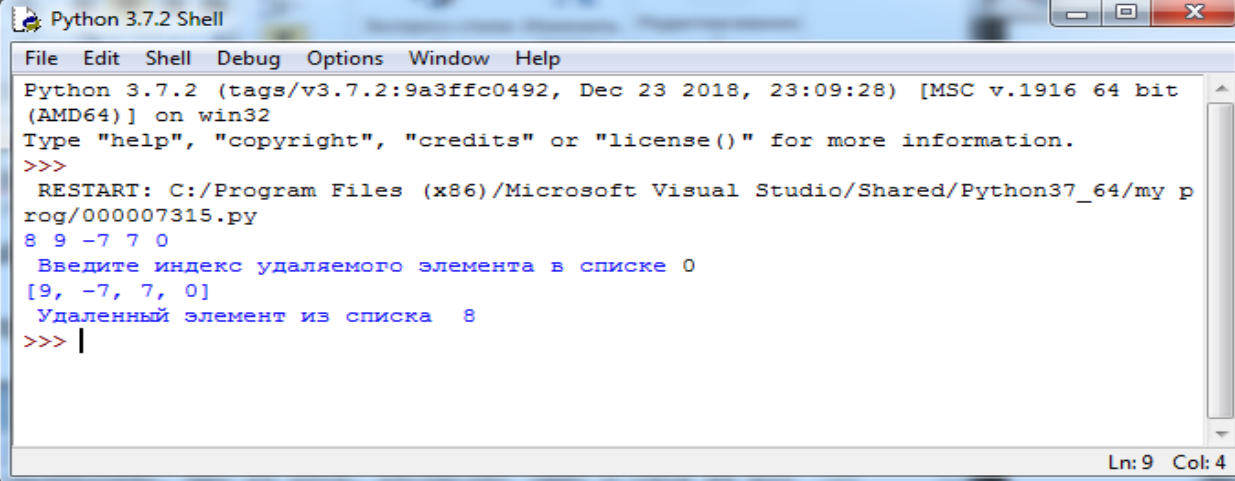
**Задача 7.3.5.** Из исходного списка следует удалить элемент, позицию которого указал пользователь.



**Решение.** Исходный список состоит из пяти элементов. Пользователь вводит индекс элемента, подлежащего удалению. Если аргумент не передан в метод **pop()**, то возвращается и удаляется последний элемент списка. Однако в качестве параметра метода **pop()** мы указываем индекс элемента. Кроме результирующего списка, выводим на экран значение удаленного элемента.

```
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)
```

Результат работы программы представлен на рисунке 87.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007315.py
8 9 -7 7 0
Введите индекс удаляемого элемента в списке 0
[9, -7, 7, 0]
Удаленный элемент из списка 8
>>> |
```

Рисунок 87 – Результат работы программы по удалению из списка числа по позиции

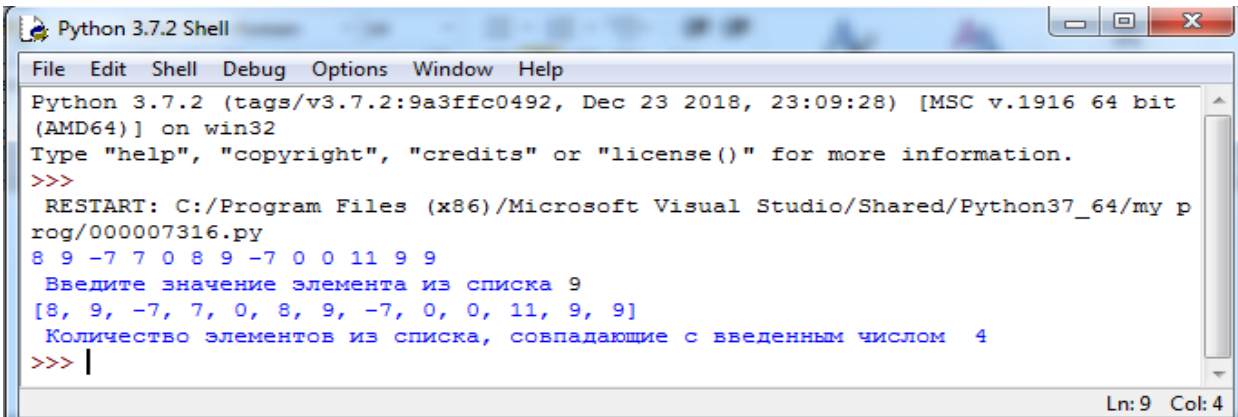
**Задача 7.3.6.** В исходном списке подсчитайте количество элементов с определенным значением.

**Решение.** Мы сформировали список, включив в него несколько повторяющихся элементов. Введенное пользователем значение **znach** является параметром метода **count()**, который вернет количество повторов элемента **znach**.

```
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)
```

Результат работы программы представлен на рисунке 88.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007316.py
8 9 -7 7 0 8 9 -7 0 0 11 9 9
Введите значение элемента из списка 9
[8, 9, -7, 7, 0, 8, 9, -7, 0, 0, 11, 9, 9]
Количество элементов из списка, совпадающие с введенным числом 4
>>> |
```

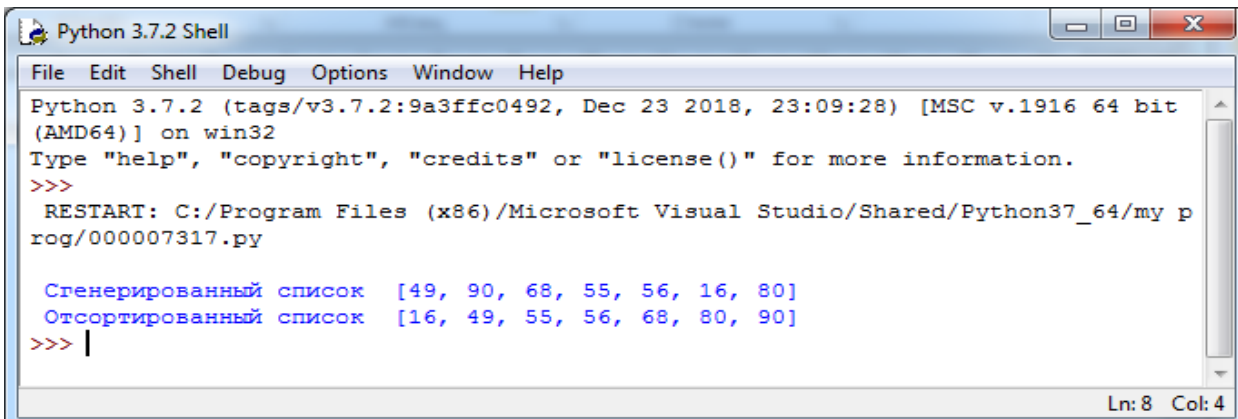
Рисунок 88 – Результат работы программы нахождения повторений в списке введенного числа

**Задача 7.3.7.** Список, сгенерированный случайным образом, отсортируйте по возрастанию.

**Решение.** Метод `sort()`, применяемый в Python для сортировки списка, имеет необязательный параметр `reverse`. В нижеприведенном коде он выставлен в положение `False`, что дает возможность отсортировать список по возрастанию.

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

Результат работы программы представлен на рисунке 89.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007317.py

Сгенерированный список [49, 90, 68, 55, 56, 16, 80]
Отсортированный список [16, 49, 55, 56, 68, 80, 90]
>>> |
```

Рисунок 89 – Результат работы программы сортировки списка по возрастанию

**Задача 7.3.8.** Вывести в обратном порядке список, сгенерированный случайным образом.

**Решение.** Метод **reverse** без параметров, примененный к исходному списку, сгенерированному случайным образом списку, возвращает обратный список.

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

Результат работы программы представлен на рисунке 90.

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/000007318.py

Сгенерированный список [90, 87, 0, 81, 35, 22, 57]
Список в обратном порядке [57, 22, 35, 81, 0, 87, 90]
>>> |
```

Рисунок 90 – Результат работы программы вывода списка в обратном порядке

## 7.4 Работа со словарями

В Python есть возможность работать со структурами данных, которые имеют не только номера (индексы) элементов, но и строковые значения. Так как часто программисту приходится работать с данными, которые имеют буквенно-цифровые обозначения, то это очень удобно. Обработка таких данных может быть использована, например, в информационных системах, применяемых в компаниях по перевозке грузов и пассажиров.

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному объекту (числу, строке, кортежу), в языке Python называется **словарем**. Некий произвольный объект является **ключом**, который и открывает доступ к набору объектов, содержащихся в словаре.

Синтаксис создания словаря может быть разным, приведем один из них:

**Имя словаря = {ключ:значение,  
ключ:значение,  
ключ:значение,  
ключ:значение}**

Как видно, каждый элемент словаря состоит из пары «ключ:значение». Ключ определяет элемент словаря, а значение соответствует данным, которые соответствуют данному ключу. Каждая пара может записываться в программах на разных строках, а может располагаться в одной.

Смысл такой конструкции напоминает использование оператора выбора (Case, Select Case, switch), работа с которым ведется в таких популярных языках программирования, как C#, Microsoft Visual Basic, Pascal и др., но в Python возможности использования словарей значительно шире. Словари можно использовать, когда необходимо установить соответствие между объектами, подсчитывать количество объектов, хранить данные, связанные с объектом.

При использовании словарей следует учитывать следующие правила:

1. Двух одинаковых ключей в парах «ключ:значение» быть не может.
2. Словари не относятся к последовательностям. Таким образом, те функции, которые использовались для работы со списками и кортежами, для словарей неприемлемы.
3. Ключ должен быть неизменяемым, им может быть целое или действительное число, строка, кортеж. Сделано это для того, чтобы добиться выполнения требования в пункте первом.
4. Значения могут принимать любой тип данных, быть изменяемыми и неизменяемыми.

Рассмотрим несколько примеров работы программ с использованием словаря.

**Задача 7.4.1.** Разработайте программу, которая осуществляет перевод нескольких слов русского языка на английский.

**Решение.** Создание словаря в данной программе происходит иначе, чем рассказывалось выше. Сначала оператором `slovar=dict()` мы создаем пустой словарь с именем **slovar**. Далее мы создаем в нашем словаре пары, состоящие из ключей (русские слова) и значений (слова английского языка). Когда пользователь вводит слово, которое он хочет перевести на английский язык, организуется перебор элементов словаря, и если какой-то из них совпал с введенным словом, выводится ответ.

```
slovar=dict()
slovar['Мама']='Mother'
slovar['Папа']='Dad'
slovar['Бабушка']='Grandmother'
slovar['Дедушка']='Grandfather'
```

```
slovo=input("\n Введите слово для перевода ")
if slovo in slovar:
    perevod=slovar[slovo]
    print("\n", slovo, "переводится как", perevod)
```

Результат работы программы представлен на рисунке 91.

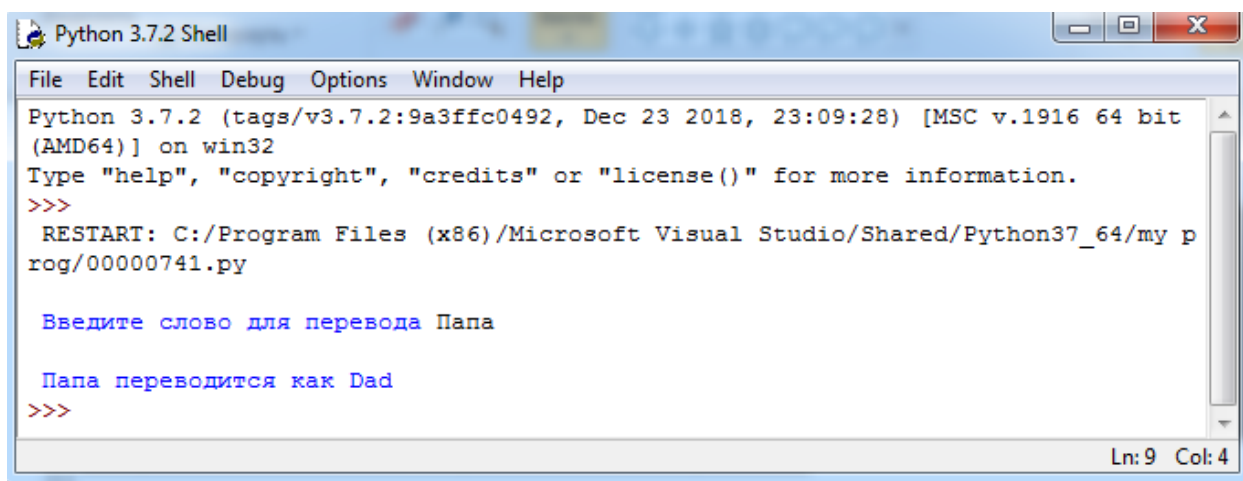
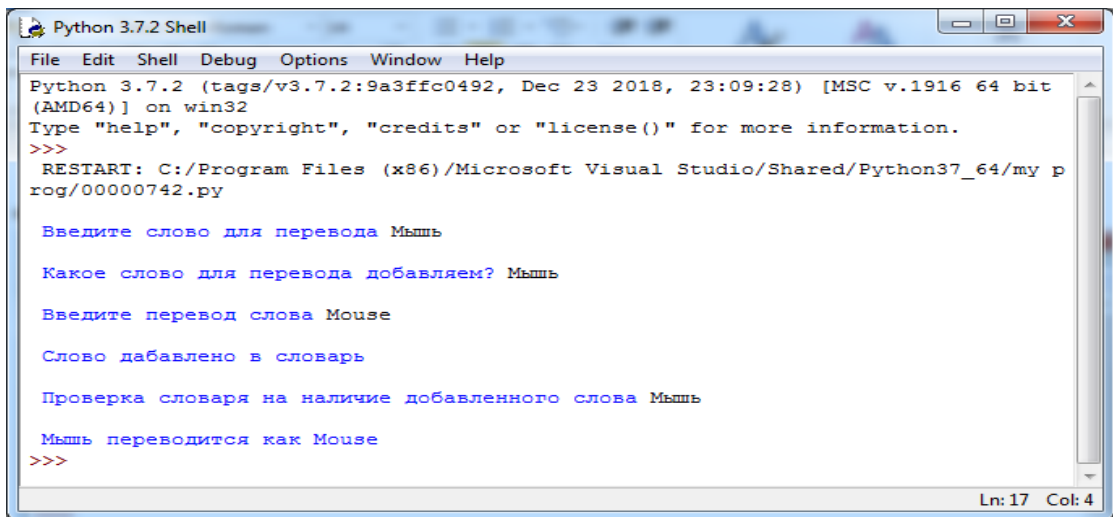


Рисунок 91 – Результат выбора слова из словаря

Дополним программу возможностью включения в словарь нового слова. Для этого, в случае ввода слова не из словаря, организуем ввод слова и перевода этого слова в словарь. Для того чтобы убедиться в наличии введенного слова в словаре, организуем проверку словаря.

```
slovar=dict()
slovar['Мама']='Mother'
slovar['Папа']='Dad'
slovar['Бабушка']='Grandmother'
slovar['Дедушка']='Grandfather'
slovo=input("\n Введите слово для перевода ")
if slovo in slovar:
    perevod=slovar[slovo]
    print("\n", slovo, "переводится как", perevod)
slovo=input("\n Какое слово для перевода добавляем? ")
if slovo not in slovar:
    perevod=input("\n Введите перевод слова ")
    slovar[slovo]=perevod
    print("\n Слово дабавлено в словарь")
slovo=input("\n Проверка словаря на наличие добавленного слова ")
if slovo in slovar:
    perevod=slovar[slovo]
    print("\n", slovo, "переводится как", perevod)
```

Результат работы программы представлен на рисунке 92.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000742.py

Введите слово для перевода Мышь

Какое слово для перевода добавляем? Мышь

Введите перевод слова Mouse

Слово дабавлено в словарь

Проверка словаря на наличие добавленного слова Мышь

Мышь переводится как Mouse
>>>
```

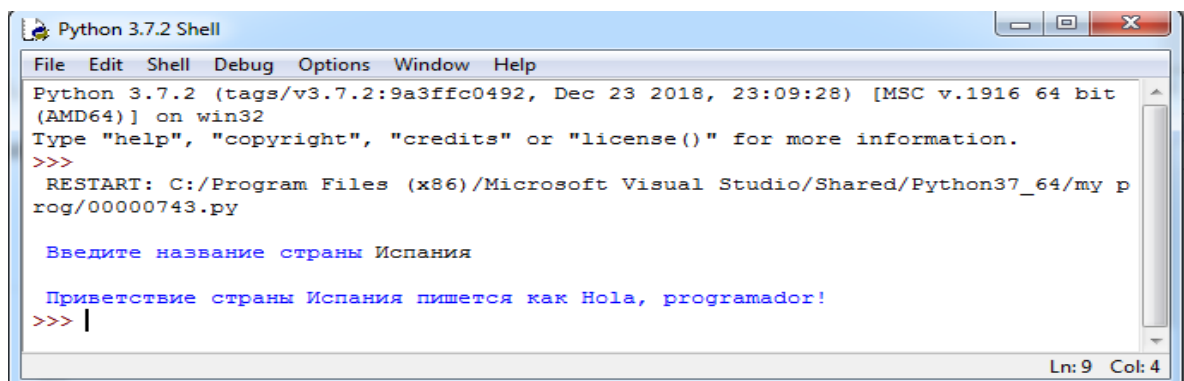
Рисунок 92 – Результат добавления слова в словарь

**Задача 7.4.2.** Разработайте программу, которая выводит фразу «Привет, программист» на одном из иностранных языков.

**Решение.** При решении данной задачи мы создадим словарь из нескольких пар, в которых есть ключ - страна и значение, которым является приветствие. Все имеющиеся пары мы заключаем в фигурные скобки. Далее с помощью условного оператора **if** проверяем, соответствует ли введенное название страны ключу в словаре и, если условие оказывается истинным, выводим соответствующее значение.

```
slovar={"Россия":"Привет, программист!",
        "Англия":"Hello, programmer!",
        "Германия":"Hallo, Programmierer!",
        "Испания":"Hola, programador!",
        "Италия":"Ciao, programmatore!"}
slovo=input("\n Введите название страны ")
if slovo in slovar:
    perevod=slovar[slovo]
    print("\n", "Приветствие страны", slovo, "пишется как", perevod)
```

Результат работы программы представлен на рисунке 93.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000743.py

Введите название страны Испания

Приветствие страны Испания пишется как Hola, programador!
>>> |
```

Рисунок 93 – Результат выбора страны из словаря приветствий

**Задача 7.4.3.** Разработайте программу, которая имитирует заказ билета в кинотеатр. Ее результатом должен стать вывод информации о выбранном кинофильме, времени сеанса, зале, номере ряда и места..

**Решение.** Создадим три словаря. В первом из них, имеющем имя **film**, элементы словаря будут соответствовать названию фильма и времени сеанса. Предполагается, что в зависимости от времени сеанс будет проходить в определенном зале, поэтому во втором словаре (**time**) элементами будут номер зала и время начала сеанса. Третий словарь (**place**) необходим для хранения номера ряда и места.

После ввода названия фильма условием **if name in film** проверяем наличие фильма в словаре, и если он в нем содержится, то выводим доступное время сеанса для запрошенного фильма. Далее запрашиваем время сеанса, и если оно заявлено в словаре (**if vrem in time**), то относительно заданного времени будет выбран соответствующий зал, после чего пользователя попросят ввести номер ряда и место. Код программы представлен в листинге ниже.

```
film={"Мавританец":"16-00 18-00 20-00 22-00",
      "Конь Юлий и большие скачки":"9-00 11-00 13-00",
      "Лига справедливости Зака Снайдера":"12-00 14-15 16-30 19-00"}
time={"9-00":"Зал 1",
      "11-00":"Зал 1",
      "12-00":"Зал 2",
      "13-00":"Зал 1",
      "14-15":"Зал 3",
      "16-00":"Зал 1",
      "16-30":"Зал 2",
      "18-00":"Зал 3",
      "19-00":"Зал 1",
      "20-00":"Зал 2",
      "22-00":"Зал 3"}
place={"1":"1-20",
        "2":"1-20",
        "3":"1-20",
        "4":"1-20",
        "5":"1-20",
        "6":"1-20",
        "7":"1-20",
        "8":"1-20",
        "9":"1-20",
        "10":"1-20",
        "11":"1-20",
        "12":"1-20"}
print("Доступные фильмы: Конь Юлий и большие скачки, Лига
```

```

справедливости Зака Снайдера, Мавританец")
name=input("\n Введите название фильма: ")
if name in film:
    print("Доступное время сеансов: ", film[name])
    vrem=input("Введите время сеанса: ")
    if vrem in time:
        zal=time[vrem]
        if zal=="Зал 1":
            ryad=input("Введите ряд от 1 до 12: ")
            if ryad in place:
                print("Доступные места - ", place[ryad])
                mesto=int(input("Введите место: "))
                if (mesto>0) and (mesto<21):
                    print("\n Выбранный фильм:", name, " Выбранное время: ",
vrem, " Зал: ", zal," Ряд: ", ryad, " Место: ", mesto)
                else:
                    print("Введено некорректное место")
            else:
                print("Введен некорректный номер ряда")
        elif zal=="Зал 2":
            ryad=input("Введите ряд от 1 до 12: ")
            if ryad in place:
                print("Доступные места - ", place[ryad])
                mesto=int(input("Введите место: "))
                if (mesto>0) and (mesto<21):
                    print("\n Выбранный фильм:", name, " Выбранное время: ",
vrem, " Зал: ", zal," Ряд: ", ryad, " Место: ", mesto)
                else:
                    print("Введено некорректное место")
            else:
                print("Введен некорректный номер ряда")
        elif zal=="Зал 3":
            ryad=input("Введите ряд от 1 до 12: ")
            if ryad in place:
                print("Доступные места - ", place[ryad])
                mesto=int(input("Введите место: "))
                if (mesto>0) and (mesto<21):
                    print("\n Выбранный фильм:", name, " Выбранное время: ",
vrem, " Зал: ", zal," Ряд: ", ryad, " Место: ", mesto)
                else:
                    print("Введено некорректное место")
            else:
                print("Введен некорректный номер ряда")
        else:
            print("Неверно введено время сеанса")

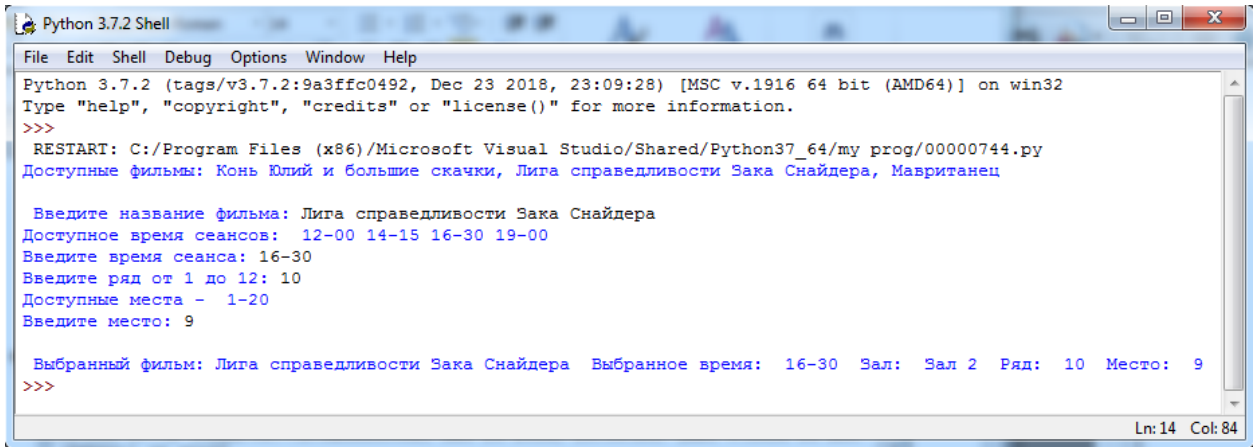
```



else:

```
print("Неверное введено название фильма")
```

Результат работы программы представлен на рисунке 94.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my prog/00000744.py
Доступные фильмы: Конь Юлий и большие скачки, Лига справедливости Зака Снайдера, Мавританец

Введите название фильма: Лига справедливости Зака Снайдера
Доступное время сеансов: 12-00 14-15 16-30 19-00
Введите время сеанса: 16-30
Введите ряд от 1 до 12: 10
Доступные места - 1-20
Введите место: 9

Выбранный фильм: Лига справедливости Зака Снайдера Выбранное время: 16-30 Зал: Зал 2 Ряд: 10 Место: 9
>>>
```

Рисунок 94 – Результат программы имитирующей работу кассы кинотеатра

## 7.5 Примеры решения задач

**Задача 7.5.1.** В кортеже целых чисел вычислите произведение отрицательных элементов, имеющих нечетные индексы.

**Решение.** В листинге приведен код программы, отвечающий за решение задачи:

```
korteg=(-11, -12, 35, -8, -25, 39, 0, -12)
print(" Кортёж ")
for i in korteg:
    print(i, end=" ")
kol=0
proizv=1
for i in range(8):
    if i%2==1: # Поиск элементов кортежа с нечетными индексами
        if korteg[i]<0:
            proizv=proizv*korteg[i] # Произведение отрицательных из них
            kol=kol+1 # Количество отрицательных из них
print("\n Произведение отрицательных элементов кортежа, имеющих
нечетные индексы = ", proizv)
print("\n Количество отрицательных элементов кортежа, имеющих
нечетные индексы = ", kol)
```

Блок-схема алгоритма решения задачи представлена на рисунке 95.

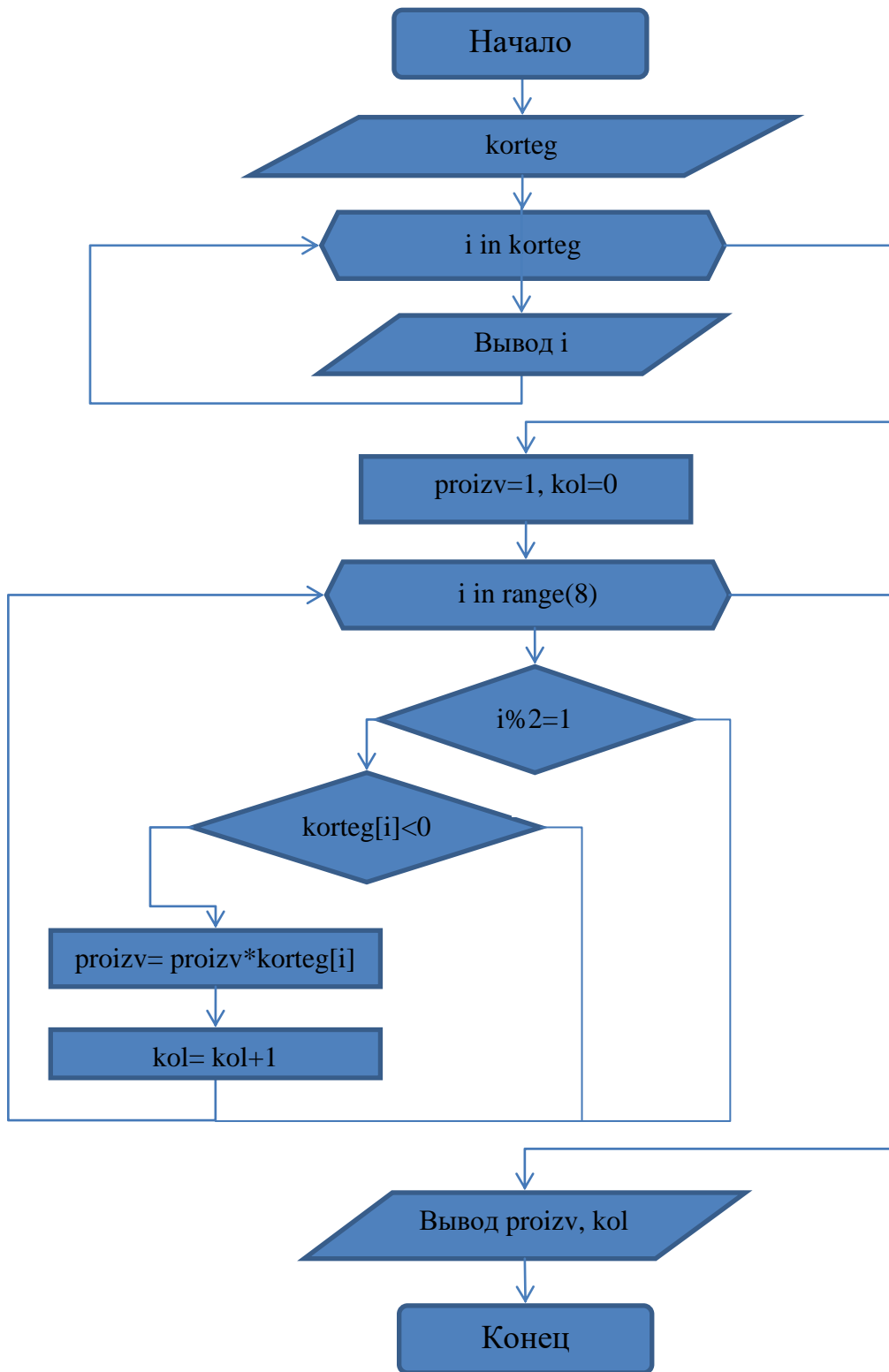
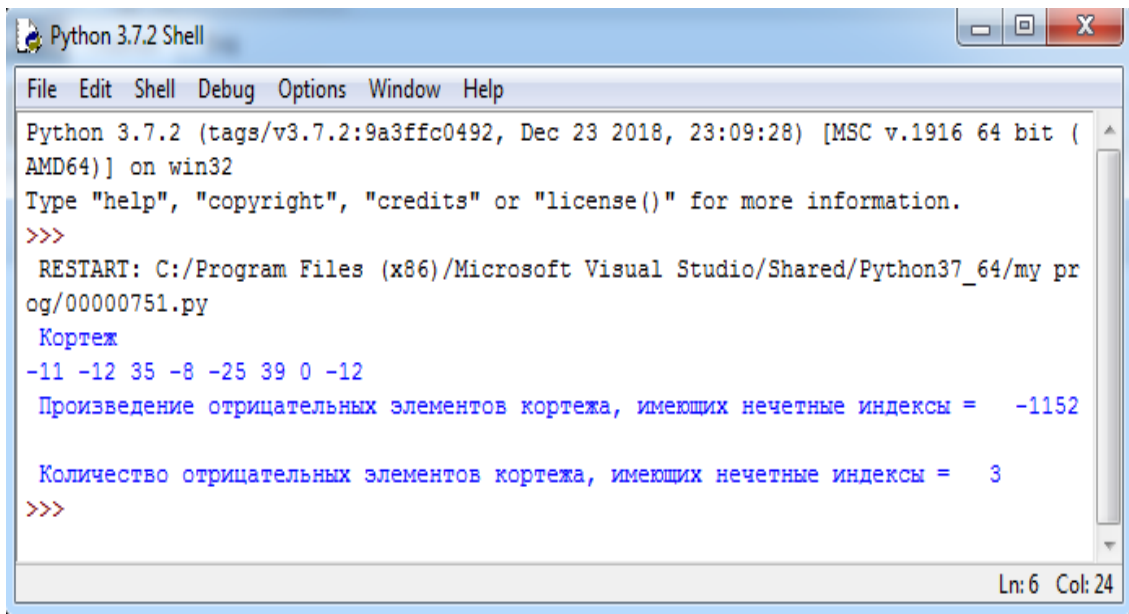


Рисунок 95 – Блок-схема алгоритма решения задачи 7.5.1

Результат работы программы представлен на рисунке 96.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my prog/00000751.py
Кортеж
-11 -12 35 -8 -25 39 0 -12
Произведение отрицательных элементов кортежа, имеющих нечетные индексы = -1152
Количество отрицательных элементов кортежа, имеющих нечетные индексы = 3
>>>
Ln: 6 Col: 24
```

Рисунок 96 – Результат работы программы задачи 7.5.1

**Задача 7.5.2.** В кортеже целых чисел вычислите среднее арифметическое значение квадратов положительных элементов.

**Решение.** В листинге приведен код программы, отвечающий за решение задачи:

```
korteg=(-11, -12, 35, -8, -25, 39, 0, -12)
print(" Кортеж ")
for i in korteg:
    print(i, end=" ")
kol=0
sum=0
for i in range(8):
    if korteg[i]>0:
        sum=sum+korteg[i]*korteg[i] # Сумма квадратов положительных
элеменов кортежа
        kol=kol+1 # Количество положительных элеменов кортежа
sr=sum/kol
print("\n Среднее арифметическое квадратов положительных элеменов
кортежа = ", sr)
print("\n Количество положительных элеменов кортежа = ", kol)
```

Блок-схема алгоритма решения задачи представлена на рисунке 97.

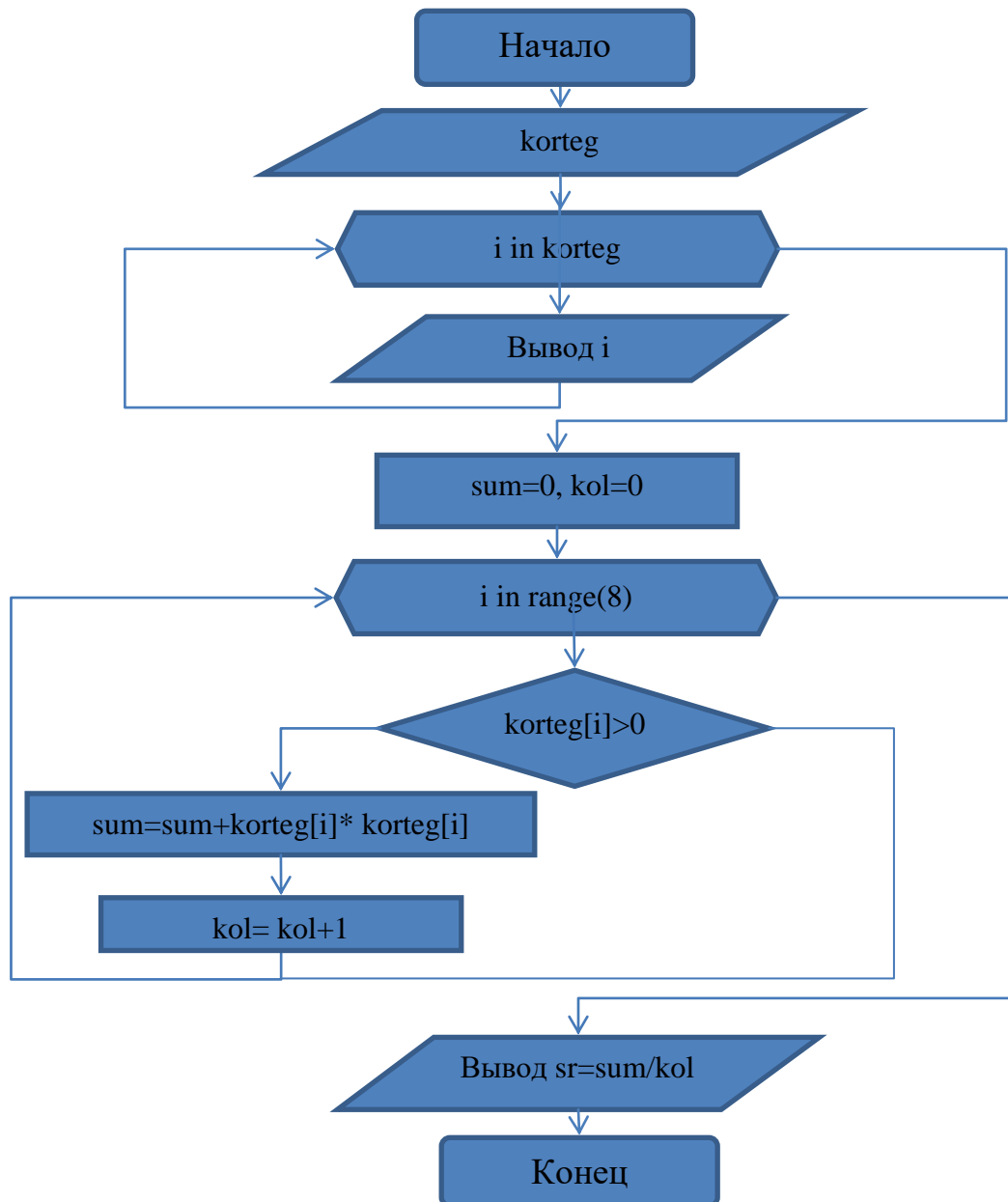


Рисунок 97 – Блок-схема алгоритма решения задачи 7.5.2

Результат работы программы представлен на рисунке 98.

```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my prog/00000752.py
Кортеж
-11 -12 35 -8 -25 39 0 -12
Среднее арифметическое квадратов положительных элементов кортежа = 1373.0

Количество положительных элементов кортежа = 2
>>> |
Ln: 10 Col: 4
  
```

Рисунок 96 – Результат работы программы задачи 7.5.2

**Задача 7.5.3.** В кортеже целых чисел определите число инверсий. Инверсией называется пара элементов, в которой большее число расположено слева от меньшего. Например, дан кортеж:



Стрелками показано количество инверсий.

**Решение.** Организовав цикл от 1 до 6, мы сравниваем каждый элемент последовательности с предыдущим и определяем, не больше ли он него. Нет смысла организовывать цикл от 1 до 7, потому что, выполнив проверку  $kor[i] > kor[i+1]$ , мы седьмой элемент кортежа будем сравнивать с восьмым, а такого не существует. В случае истинности проверяемого условия увеличиваем счетчик на единицу оператором  $inv = inv + 1$  и выводим ответ.

Блок-схема алгоритма решения задачи представлена на рисунке 97.

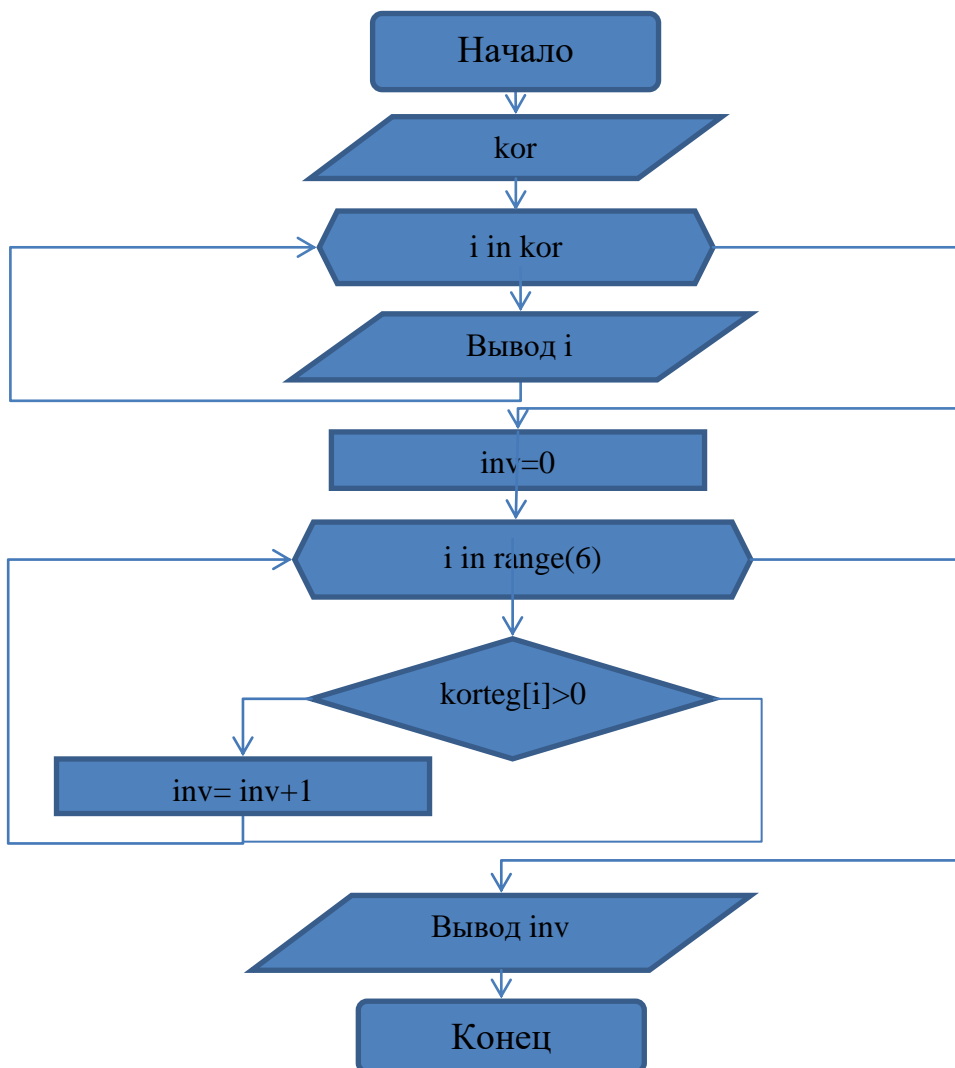


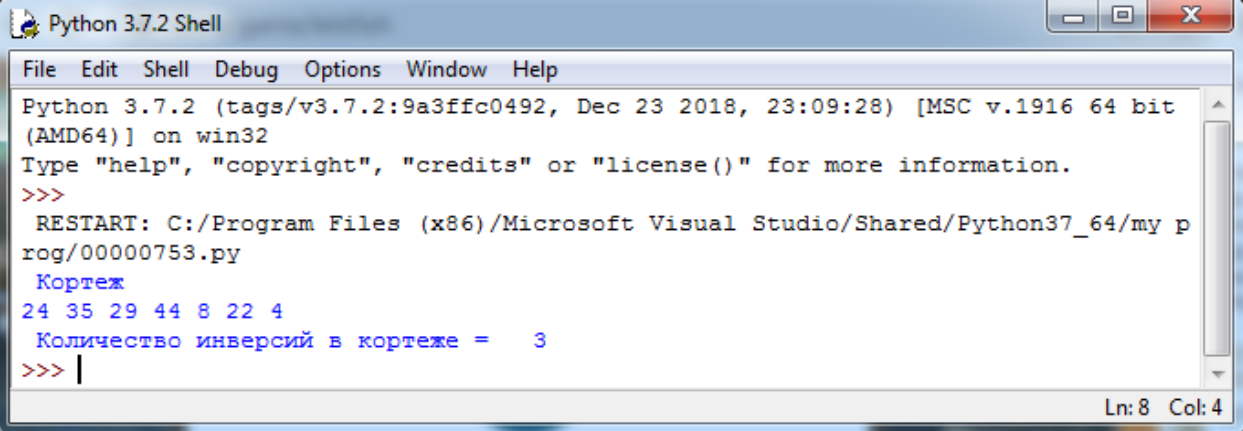
Рисунок 97 – Блок-схема алгоритма решения задачи 7.5.3

```

В листинге приведен код программы, отвечающий за решение задачи:
kor=(24, 35, 29, 44, 8, 22, 4)
print(" Кортеж ")
for i in kor:
    print(i, end=" ")
inv=0
for i in range(6):
    if kor[i]>kor[i+1]:
        inv=inv+1
print("\n Количество инверсий в кортеже = ", inv)

```

Результат работы программы представлен на рисунке 98.



```

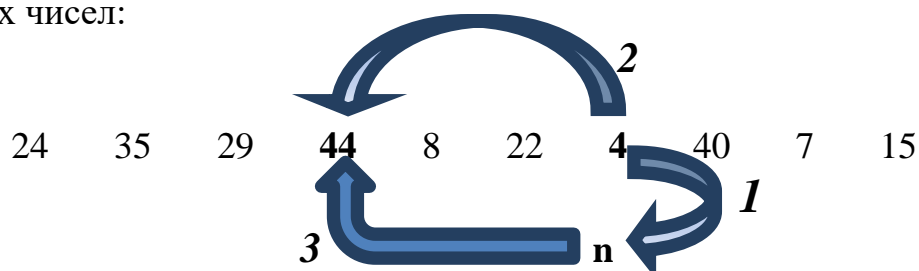
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000753.py
Кортеж
24 35 29 44 8 22 4
Количество инверсий в кортеже = 3
>>> |
Ln: 8 Col: 4

```

Рисунок 98 – Результат работы программы задачи 7.5.3

**Задача 7.5.4.** В списке из десяти целых чисел, сформированном случайным образом, найдите максимальный и минимальный элементы, а также осуществите их обмен.

**Решение.** Для того чтобы обменять элементы списка местами, нам потребуются две ячейки **nommax** и **nommin**. Изначально считаем, что номер максимального элемента в списке равен нулю и номер минимального элемента также равен нулю. Отсюда и операторы в программе: **nommax=0** и **nommin=0**. Приемы нахождения максимального и минимального чисел рассматривались при программировании циклических алгоритмов, поэтому останавливаться на них не будем. Для того чтобы осуществить обмен ячеек, необходима третья ячейка - например, **n**. Предположим, сгенерирован список целых чисел:



Минимальный элемент в списке равен **4**, с порядковым номером **6**, а максимальный элемент в списке равен **44**, с порядковым номером **3**. Возьмем ячейку - **n**. **Первым действием (1)** будет **перемещение** в нее найденного **минимального числа**. **Вторым действием (2)** на освободившееся место **перемещаем максимальное число**. И, наконец, **третьим действием (3)** **минимальное число из ячейки n помещается на то место** в списке, где только что находилось **максимальное число**. В программе эти действия можно было бы описать операторами:

```
n=chislo[6]
chislo[6]=a[3]
a[3]=n
```

Поскольку заранее не известно, какой элемент в списке будет минимальный, а какой максимальный, и каковы будут их номера, такую последовательность операторов заменяем на следующую:

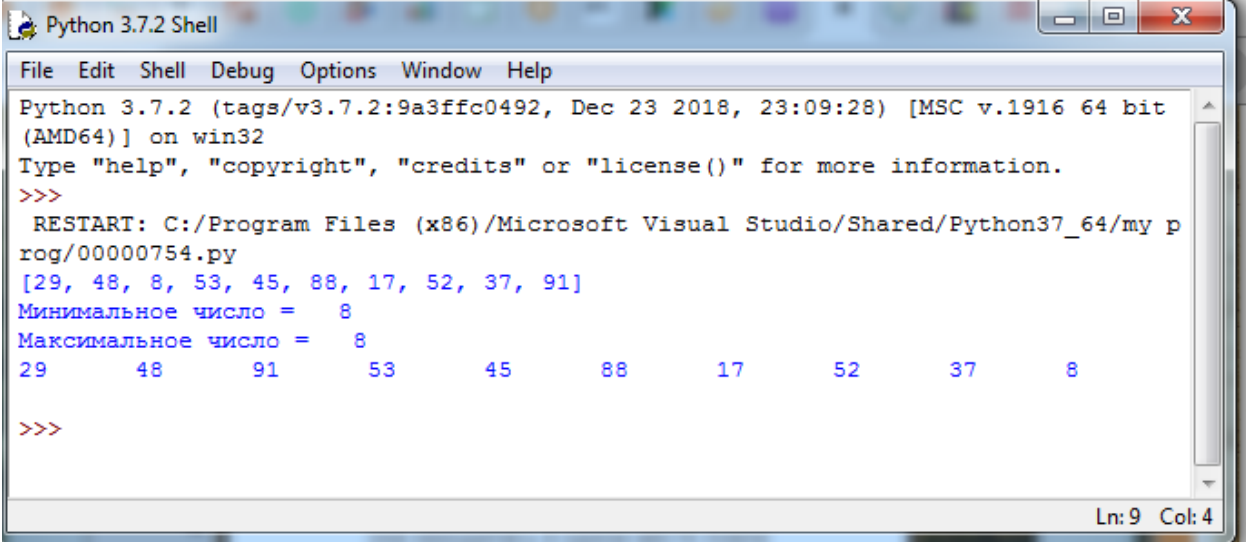
```
n=chislo[nommin]
chislo[nommin]=chislo[nommax]
chislo[nommax]=n
```

В листинге приведен код программы, отвечающий за решение задачи.

```
import random
chislo=random.sample(range(100), 10)
print(chislo)
nommin=0 # Считаем, что первоначально номер минимального
элемента в списке 0
nommax=0 # Считаем, что первоначально номер максимального
элемента в списке 0
min=32767
max=-32768
for i in range(1, 10):
    if chislo[i]<min: # Поиск минимального элемента в списке
        min=chislo[i]
        nommin=i # Поиск индекса минимального элемента в списке
    if chislo[i]>max: # Поиск максимального элемента в списке
        max=chislo[i]
        nommax=i # Поиск индекса максимального элемента в списке
print("Минимальное число = ", min)
print("Максимальное число = ", max)
n=chislo[nommin] #
chislo[nommin]=chislo[nommax]
chislo[nommax]=n
for i in range(0, 10):
```

```
print(chislo[i], end='\t')
```

Результат работы программы представлен на рисунке 99.



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000754.py
[29, 48, 8, 53, 45, 88, 17, 52, 37, 91]
Минимальное число = 8
Максимальное число = 8
29      48      91      53      45      88      17      52      37      8
>>>
Ln: 9 Col: 4
```

Рисунок 99 – Результат работы программы задачи 7.5.4

**Задача 7.5.5.** Дан кортеж, состоящий из вещественных чисел. Запишите положительные элементы кортежа в список. Определите количество положительных элементов. Найдите произведение элементов списка.

**Решение.** Первоначально сформированный кортеж содержит семь вещественных чисел, среди которых четыре положительных. Пустой список объявляется с помощью оператора `spisok=[]`. Оператором `if kort[i]>0` проверим условие на положительность числа, входящего в кортеж, и, если оно истинно, то с помощью метода `append()` добавляем его в список. Счетчик количества положительных чисел `n` увеличивается на единицу оператором `n=n+1`.

Далее организуется цикл с уже известным количеством положительных чисел, хранящихся в ячейке `n`, и в нем находим произведение элементов списка, которое, в качестве ответа, выводим на экран.

В листинге приведен код программы, отвечающий за решение задачи.

```
kort=(5.8, -1.7, 0, -7.5, 1.2, 0.8, 0.2)
spisok=[]
n=0
proizv=1
print("")
for i in kort:
    print(i, end=" ")
for i in range(7):
    if kort[i]>0:
        spisok.append(kort[i])
```

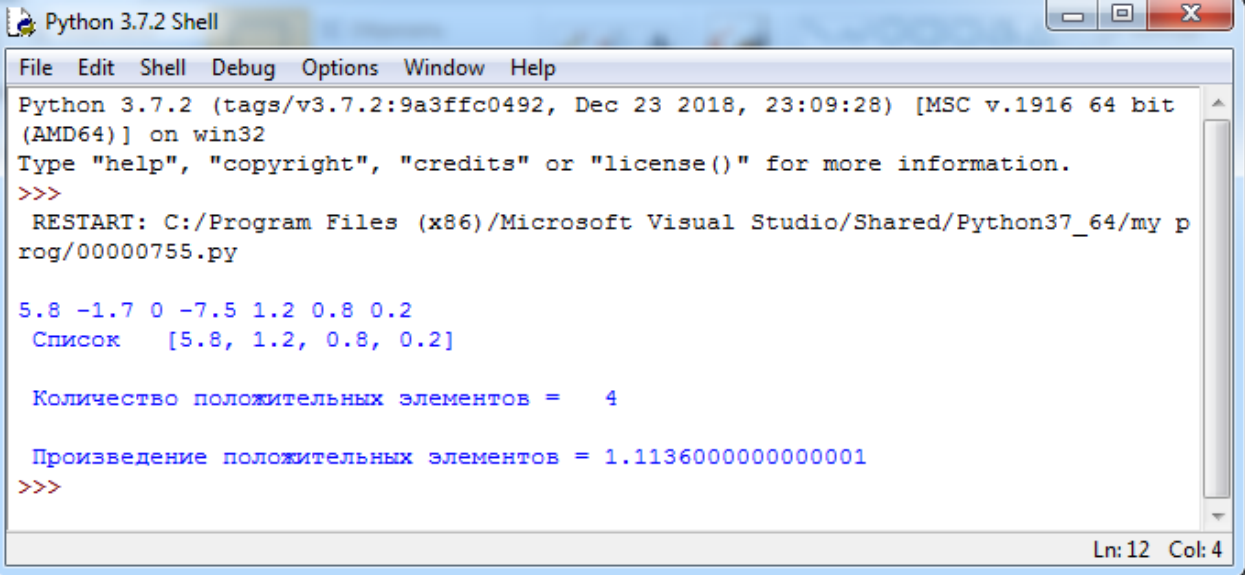


```

    n=n+1
for i in range(n):
    proizv=proizv*spisok[i]
print("\n Список ", spisok)
print("\n Количество положительных элементов = ", n)
print("\n Произведение положительных элементов =", proizv)

```

Результат работы программы представлен на рисунке 100.



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000755.py

5.8 -1.7 0 -7.5 1.2 0.8 0.2
Список [5.8, 1.2, 0.8, 0.2]

Количество положительных элементов = 4

Произведение положительных элементов = 1.1136000000000001
>>>
Ln:12 Col:4

```

Рисунок 100 – Результат работы программы задачи 7.5.5

**Задача 7.5.6.** Разработайте программу, заполняющую список из  $N$  элементов случайными целыми числами разного знака. Выведите на экран компьютера созданный список и сформируйте другой список, элементы которого по модулю больше некоторого введенного значения  $C$ .

**Решение.** Первоначально сформируем последовательность чисел с использованием функции **sample**, которая возвращает указанное пользователем количество элементов. Затем, после ввода значения  $c$ , инициализируем список  $y$ , открываем цикл, в котором проверяем логическое выражение **abs(x[i])>c** и в случае его истинности, используя метод **append()**, добавляем элемент исходной последовательности в список  $y$ .

В листинге приведен код программы, отвечающий за решение задачи.

```

import random
from math import *
n=int(input("Введите количество элементов в списке: "))
x=random.sample(range(-5, 15), n)
for i in x:
    print(i, end=" ")
c=int(input("\n Введите значение переменной C: "))

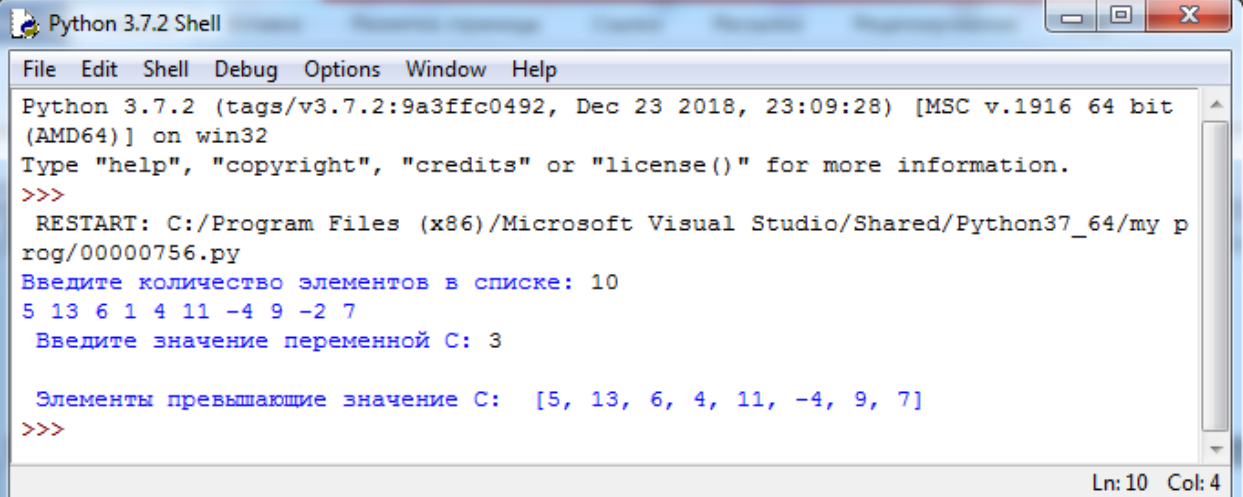
```

```

y=[]
for i in range(n):
    if abs(x[i])>c:
        y.append(x[i])
print("\n Элементы превышающие значение C: ", y)

```

Результат работы программы представлен на рисунке 101.



```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my p
rog/00000756.py
Введите количество элементов в списке: 10
5 13 6 1 4 11 -4 9 -2 7
Введите значение переменной C: 3
Элементы превышающие значение C: [5, 13, 6, 4, 11, -4, 9, 7]
>>>
Ln:10 Col:4

```

Рисунок 101 – Результат работы программы задачи 7.5.6

**Задача 7.5.7.** Сформируйте исходный список имен  $N$  студентов. Организуйте формирование двух результирующих списков по четным и нечетным номерам исходного списка. Список с нечетными номерами упорядочите по убыванию, с четными номерами - по возрастанию.

**Решение.** Создаем три пустых списка (**spisok**, **chetn**, **nechetn**) и организуем ввод имен студентов, а также добавление их методом **append()** в исходный список. В следующем цикле осуществляем проверку номера элемента на четность с использованием операции **%** (деление по модулю). Если остаток равен нулю, то элемент добавляется в список четных номеров, в противном случае формируется список нечетных номеров. Используя методы упорядочивания списков (согласно таблице 6), выводим их на экран.

В листинге приведен код программы, отвечающий за решение задачи.

```

n=int(input("Введите количество студентов: "))
spisok=[]
chetn=[]
nchetn=[]
for i in range(n):
    sp=input("Введите имя студента: ")
    spisok.append(sp)
print("\n Список студентов ", spisok)
for i in range(n):

```

```

    if i%2==0:
        ch=spisok[i]
        chetn.append(ch)
    else:
        nech=spisok[i]
        nechtn.append(nech)
print("\n Четные номера списка: ", chetn)
print("\n Нечетные номера списка: ", nechtn)
for i in chetn:
    chetn.sort()
print("\n Четные номера списка упорядоченные по убыванию: ", chetn)
for i in nechtn:
    nechtn.sort(reverse=True)
print("\n Нечетные номера списка упорядоченные по возрастанию: ",
nechtn)

```

Результат работы программы представлен на рисунке 102.

```

Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Program Files (x86)/Microsoft Visual Studio/Shared/Python37_64/my prog/00000757.py
Введите количество студентов: 10
Введите имя студента: Абенов
Введите имя студента: Нургазин
Введите имя студента: Мурзин
Введите имя студента: Капасов
Введите имя студента: Нурсадьков
Введите имя студента: Абденов
Введите имя студента: Жакупов
Введите имя студента: Сергазин
Введите имя студента: Абаев
Введите имя студента: Закинов

Список студентов ['Абенов', 'Нургазин', 'Мурзин', 'Капасов', 'Нурсадьков', 'Абденов', 'Жакупов', 'Сергазин', 'Абаев', 'Закинов']
Четные номера списка: ['Абенов', 'Мурзин', 'Нурсадьков', 'Жакупов', 'Абаев']
Нечетные номера списка: ['Нургазин', 'Капасов', 'Абденов', 'Сергазин', 'Закинов']
Четные номера списка упорядоченные по убыванию: ['Абаев', 'Абенов', 'Жакупов', 'Мурзин', 'Нурсадьков']
Нечетные номера списка упорядоченные по возрастанию: ['Сергазин', 'Нургазин', 'Капасов', 'Закинов', 'Абденов']
>>> |
Ln: 26 Col: 4

```

Рисунок 102 – Результат работы программы задачи 7.5.7

## 7.6 Контрольные вопросы

1. В чем состоит главная особенность кортежей?
2. Каковы преимущества кортежей с точки зрения их использования в программах?
3. Напишите синтаксис объявления кортежей.
4. Каким образом осуществляется доступ к каждому элементу кортежа при его обработке?
5. Перечислите классические способы обработки кортежей.
6. Каким образом можно реализовать в программе срез кортежа?

7. Поясните, каким образом осуществляется обмен значений элементов кортежа.

8. Поясните, в чем состоит отличие списков, созданных на языке Python, от кортежей.

9. Напишите синтаксис объявления списков.

10. Какие возможности языка Python используются для генерации списков?

11. Перечислите и поясните основные методы работы со списками.

12. Дайте определение такой структуры данных языка Python, как словарь.

13. Напишите синтаксис создания словаря.

14. Какие правила следует использовать при создании словаря?

### 7.7 Задачи для самостоятельного решения

1. В кортеже целых чисел вычислите произведение отрицательных элементов, имеющих нечетные индексы.

2. Из исходного списка целых чисел сформируйте два списка: список четных чисел **B** и список нечетных чисел **C**.

3. Определите среднее арифметическое элементов кортежа, удовлетворяющих условию **abs(korteg[i])>C**. Значение **C** вводится с клавиатуры.

4. Разработайте программу, в которой определяются максимальный и минимальный элементы среди положительных нечетных элементов целочисленного кортежа **A(10)**.

5. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **-20** до **40**. Выведите на экран компьютера созданный список. В списке положительные элементы уменьшите вдвое, а отрицательные замените на значения их индексов.

6. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **40**. Определите, сколько процентов всего количества элементов списка составляют нечетные элементы.

7. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **50**. Выведите на экран компьютера созданный список и упорядочите элементы данного списка по возрастанию их значений.

8. Из списка произвольных чисел **A[10]** сформируйте другой список таким образом, чтобы вначале были отрицательные элементы исходного списка, затем положительные и, наконец, нулевые.

9. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **30**. Выведите на экран компьютера созданный список и найдите максимальный

элемент, его номер и поменяйте местами максимальный и первый элемент списка.

10. Разработайте программу, которая включает заданное число **D** в список **A[10]**, упорядоченный по возрастанию, с сохранением упорядоченности.

11. Разработайте программу, в которой удалите из списка **A**, состоящего из **n** элементов, первые четыре нулевых элемента.

12. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **-30** до **50**. Выведите на экран компьютера созданный список и определите, есть ли в нем серии элементов, состоящих из знакопередающихся чисел. Если есть, то выведите на экран количество таких серий.

13. Разработайте программу, которая выводит на экран два кортежа **A(10)**, содержащих диаметры и веса шин. Следует отобрать две шины, диаметры которых отличаются не более чем на **D** см, а вес - не более чем на **W** килограмм.

14. Из списка произвольных чисел **A[10]** сформируйте два списка, содержащих номера положительных и отрицательных элементов.

15. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **-20** до **40**. Выведите на экран компьютера созданный список и вычислите среднее арифметическое значение квадратов положительных элементов.

16. В кортеже целых чисел произведите обмен соседних элементов, стоящих на четных местах, с элементами, стоящими на нечетных местах.

17. Разработайте программу, заполняющую список из **N** элементов случайными вещественными числами, находящимися в интервале от **1** до **30**. Все элементы списка с четными номерами, предшествующие первому по порядку элементу с наибольшим значением, домножьте на него.

18. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **-15** до **20**. Выведите на экран компьютера созданный список и найдите наибольший элемент из отрицательных.

19. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **50**. Выведите на экран компьютера созданный список и найдите количество тех элементов, значения которых находятся в диапазоне от **A** до **B**.

20. Пользователь вводит с клавиатуры элементы списка **A[n]**. Определите, является ли заданная последовательность чисел  $a_1, a_2, \dots, a_N$  монотонно убывающей.

21. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **30**. Замените нулями элементы между максимальным и минимальным значениями, кроме них самих.

22. Разработайте программу, в которой в кортеже целых чисел требуется найти индекс последнего по счету отрицательного элемента.

23. Разработайте программу, которая определяет, имеется ли в заданном целочисленном кортеже **A(10)** хотя бы одна пара совпадающих по значению чисел.

24. Разработайте программу, которая выводит на экран два кортежа, содержащих кортежи ростов игроков двух команд (в см), и определяет, имеется ли в данных командах игроки одинакового роста.

25. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами, находящимися в интервале от **1** до **50**. Выведите на экран компьютера созданный список и найдите максимальный и минимальный элементы, вычислите их разность.