

Северо-Кавказский федеральный университет  
Кафедра инфокоммуникаций СКФУ

Отчет  
По лабораторной работе №6  
По предмету: «Основы  
кроссплатформенного программирования»

**Исполнитель:**

Студента группы ИТС-б-з-22-1

Направление подготовки 11.03.02

Инфокоммуникационные

технологии и системы связи

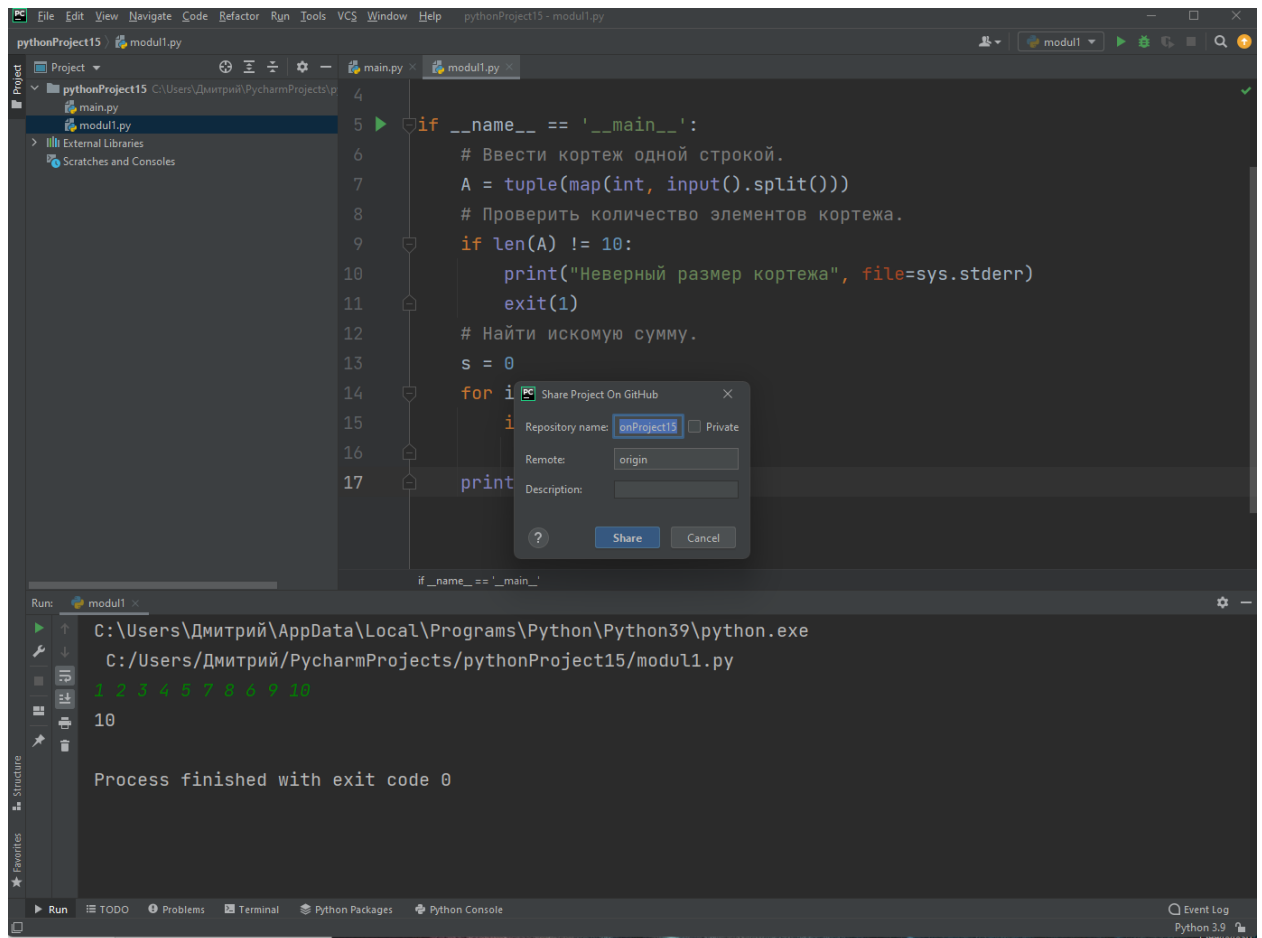
Пальников Станислав Петрович  
(Ф.И.О.)

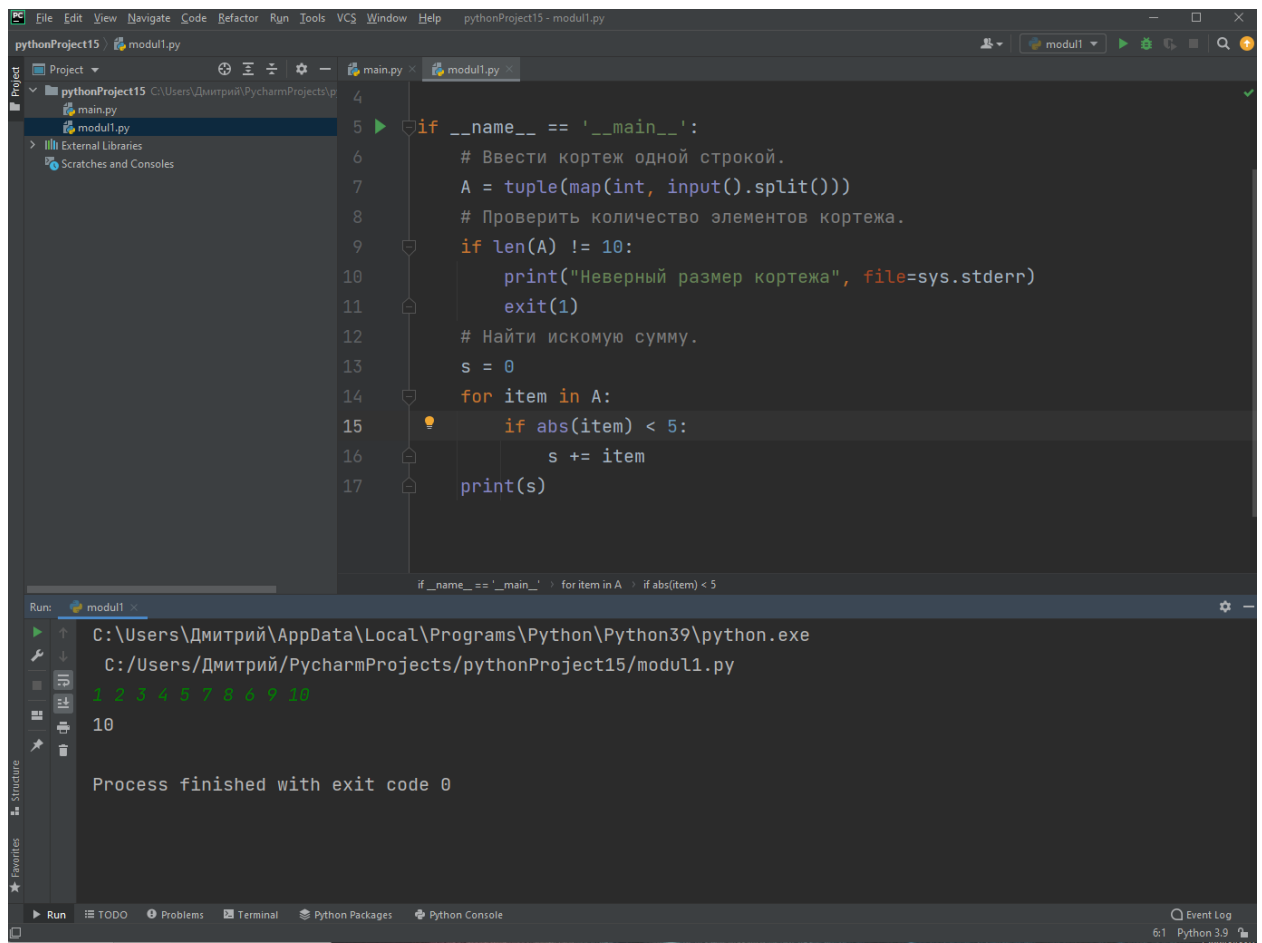
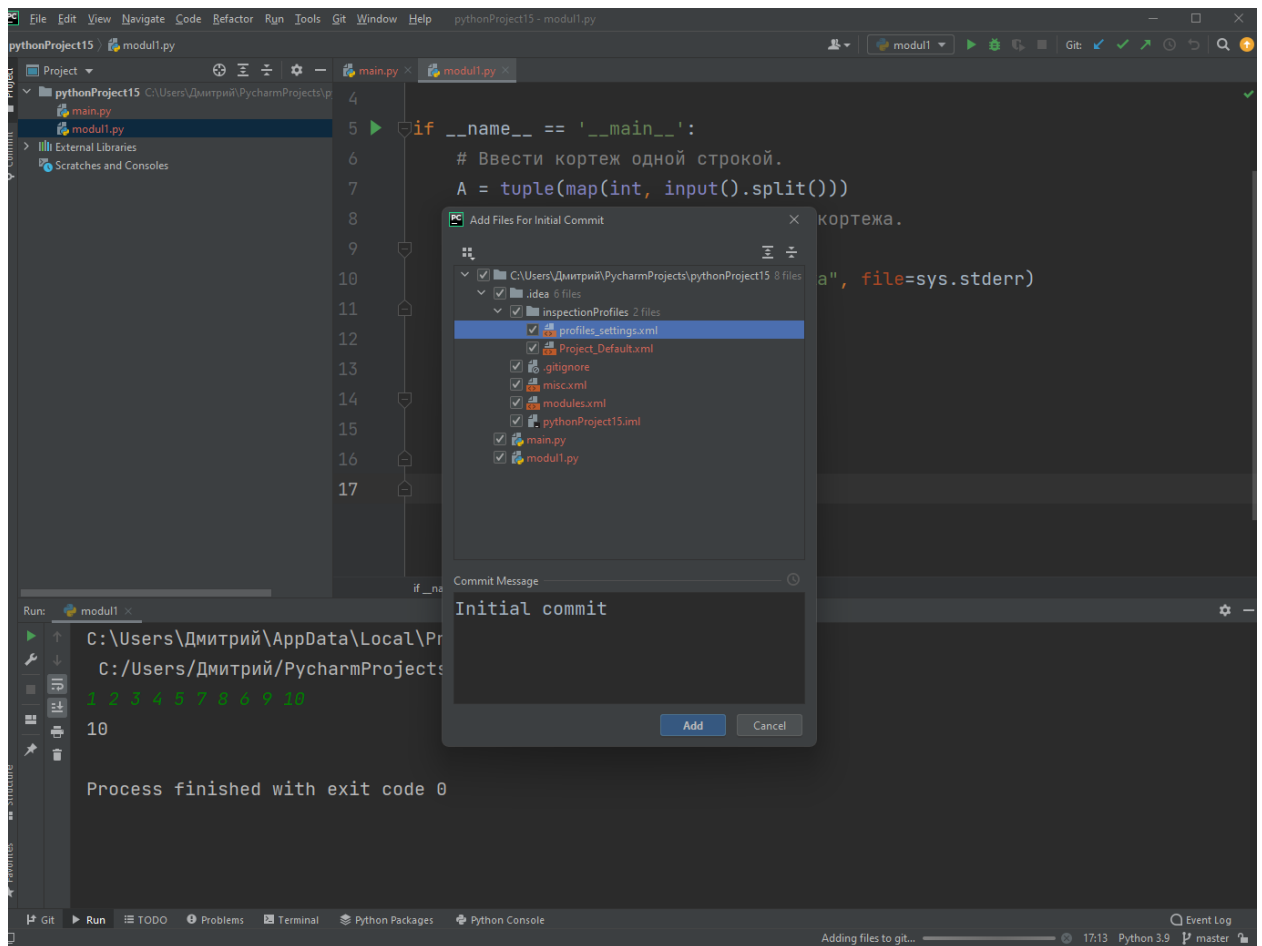
**Руководитель дисциплины:**

Воронкин Роман Александрович

Ставрополь, 2023

## Ход выполнения работы:





```
pythonProject15 - modu1.py
4
5 if __name__ == '__main__':
6     # Ввести кортеж одной строкой.
7     A = tuple(map(int, input().split()))
8     # Проверить количество элементов кортежа.
9     if len(A) != 10:
10        print("Неверный размер кортежа", file=sys.stderr)
11        exit(1)
12    # Найти искомую сумму.
13    s = 0
14    for item in A:
15        if abs(item) < 5:
16            s += item
17    print(s)

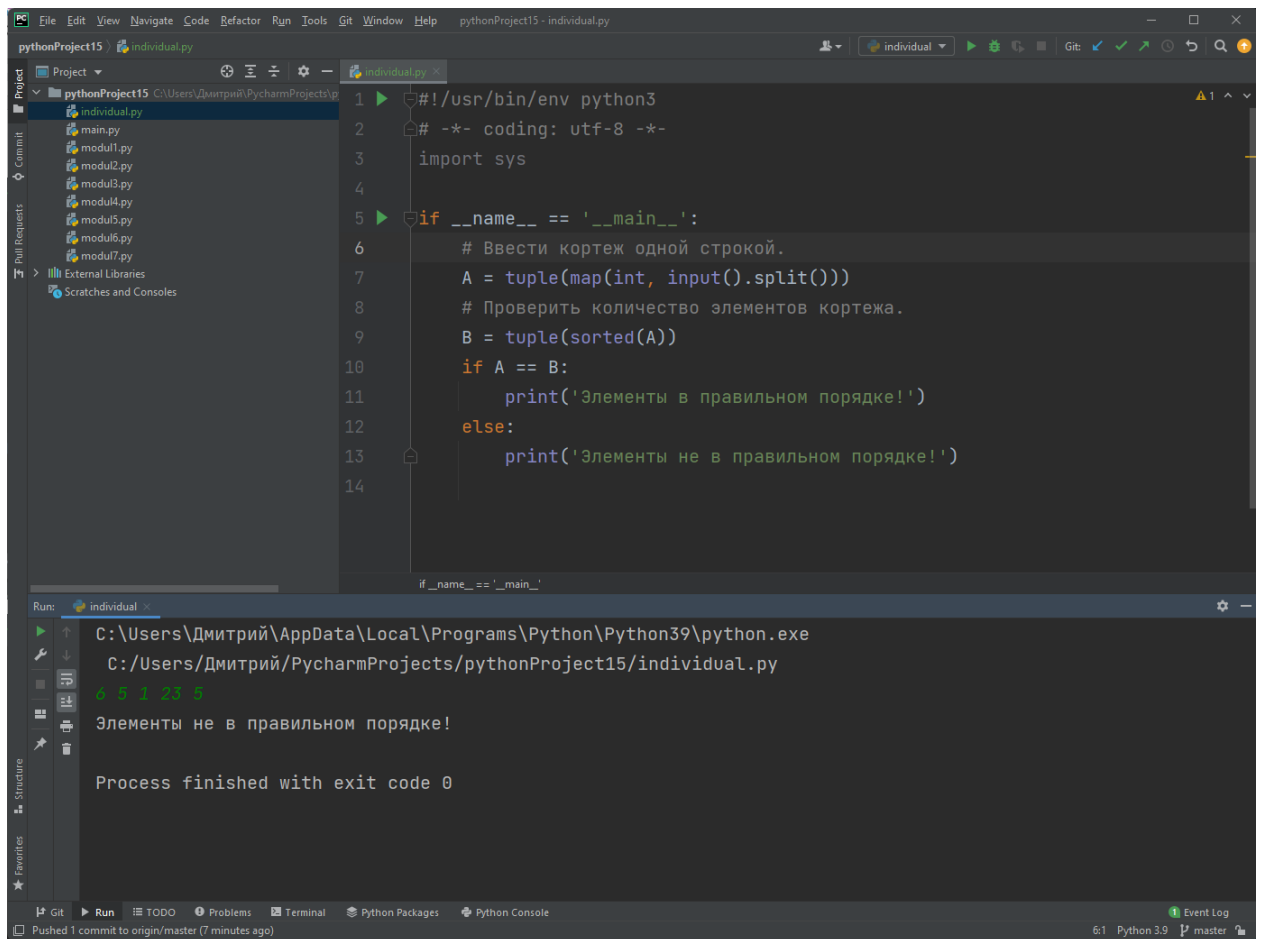
Run: modu1
C:\Users\Дмитрий\AppData\Local\Programs\Python\Python39\python.exe
C:/Users/Дмитрий/PycharmProjects/pythonProject15/modu1.py
2 5 10 5 20 4 5 6 10 89
6
Process finished with exit code 0
```

Successfully shared project on GitHub  
pythonProject015

```
pythonProject15 - individual.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4
5 if __name__ == '__main__':
6     # Ввести кортеж одной строкой.
7     A = tuple(map(int, input().split()))
8     # Проверить количество элементов кортежа.
9     B = tuple(sorted(A))
10    if A == B:
11        print('Элементы в правильном порядке!')
12    else:
13        print('Элементы не в правильном порядке!')
14

Run: individual
C:\Users\Дмитрий\AppData\Local\Programs\Python\Python39\python.exe
C:/Users/Дмитрий/PycharmProjects/pythonProject15/individual.py
1 2 3 4
Элементы в правильном порядке!
Process finished with exit code 0
```

Pushed 1 commit to origin/master (7 minutes ago)



The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `individual.py`. The script's content is as follows:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys

if __name__ == '__main__':
    # Ввести кортеж одной строкой.
    A = tuple(map(int, input().split()))
    # Проверить количество элементов кортежа.
    B = tuple(sorted(A))
    if A == B:
        print('Элементы в правильном порядке!')
    else:
        print('Элементы не в правильном порядке!')
```

The left sidebar shows the project structure for `pythonProject15`, including files like `main.py`, `modul1.py`, `modul2.py`, `modul3.py`, `modul4.py`, `modul5.py`, `modul6.py`, and `modul7.py`. The bottom panel shows the Run console output:

```
C:\Users\Дмитрий\AppData\Local\Programs\Python\Python39\python.exe
C:/Users/Дмитрий/PycharmProjects/pythonProject15/individual.py
6 9 1 23 9
Элементы не в правильном порядке!
Process finished with exit code 0
```

Ответы на контрольные вопросы:

## 1. Что такое списки в языке Python?

Список – это изменяемый тип данных. т. е. если у нас есть список `a = [1, 2, 3]` и мы хотим заменить второй элемент с 2 на 15, то мы можем это сделать, напрямую обратившись к элементу списка.

## 2. Каково назначение кортежей в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список.

Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придется как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками.

### 3. Как осуществляется создание кортежей?

Для создания пустого кортежа можно воспользоваться одной из следующих команд

```
>>> a = ()
```

```
>>> print(type(a))
```

```
<class tuple>
```

```
>>> b = tuple()
```

```
>>> print(type(b))
```

```
<class tuple>
```

Кортеж с заданным содержанием создается также как список, только вместо квадратных скобок используются круглые.

```
>>> a = (1, 2, 3, 4, 5)
```

```
>>> print(type(a))
```

```
<class tuple>
```

```
>>> print(a)
```

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

Определять кортежи очень просто, сложности могут возникнуть только с кортежами, содержащими ровно один элемент. Если мы просто укажем значение в скобках, то Python подумает, что мы хотим посчитать арифметическое выражение со скобками:

```
not_a_tuple = (42) # 42
```

Чтобы сказать Python, что мы хотим создать именно кортеж, нужно поставить после элемента кортежа запятую:

```
tuple = (42,) # (42,).
```

### 4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка — через указание индекса. Но, как уже было сказано — изменять элементы кортежа нельзя!

```
>>> a = (1, 2, 3, 4, 5)
```

```
>>> print(a[0])
```

```
1
```

```
>>> print(a[1:3])
```

```
(2, 3)
```

```
>>> a[1] = 3
```

Traceback (most recent call last):

File "", line 1, in

```
a[1] = 3
```

TypeError: 'tuple' object does not support item assignment.

#### 5. Зачем нужна распаковка (деструктуризация) кортежа?

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто.

```
name_and_age = ('Bob', 42)
```

```
(name, age) = name_and_age
```

```
name # 'Bob'
```

```
age # 42
```

Именно таким способом принято получать и сразу разбирать значения, которые возвращает функция (если таковая возвращает несколько значений, конечно)

```
(quotient, modulo) = div_mod(13, 4)
```

Соответственно кортеж из одного элемента нужно разбирать так:

```
(a,) = (42,)
```

```
a # 42
```

#### 6. Какую роль играют кортежи в множественном присваивании?

Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание. Используя множественное присваивание, можно проверить интересный трюк: обмен значениями между двумя переменными. Строку  $(a, b) = (b, a)$  нужно понимать как "присвоить в  $a$  и  $b$  значения из кортежа, состоящего из значений переменных  $b$  и  $a$ ".

#### 7. Как выбрать элементы кортежа с помощью среза?

С помощью операции взятия среза можно получить другой кортеж. Общая форма операции взятия среза для кортежа следующая:

$T2 = T1[i:j]$ , где  $T2$  – новый кортеж, который получается из кортежа  $T1$ ;  $T1$  – исходный кортеж, для которого происходит срез;  $i, j$  – соответственно нижняя и верхняя границы среза. Фактически берутся ко вниманию элементы, лежащие на позициях  $i, i+1, \dots, j-1$ . Значение  $j$  определяет позицию за последним элементом среза. Операция взятия среза для кортежа может иметь модификации такие же как и для списков.

#### 8. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом  $+$ . В простейшем случае для конкатенации двух кортежей общая форма операции следующая:

$T3 = T1 + T2$ , где  $T1, T2$  – кортежи, для которых нужно выполнить операцию конкатенации. Операнды  $T1, T2$  обязательно должны быть кортежами. При выполнении операции конкатенации для кортежей, использовать в качестве операндов любые другие типы (строки, списки) запрещено;  $T3$  – кортеж, который есть результатом.

Кортеж может быть образован путем операции повторения, обозначаемой символом  $*$ . При использовании в выражении общая форма операции следующая:

$T2 = T1 * n$ , где  $T2$  – результирующий кортеж;  $T1$  – исходный кортеж, который нужно повторить  $n$  раз;  $n$  – количество повторений кортежа  $T1$ .

#### 9. Как выполняется обход элементов кортежа?

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла `while` или `for`.

#### 10. Как проверить принадлежность элемента кортежу?



Операция in:

```
# Проверка вхождения элемента в кортеж
# Оператор in
# Заданный кортеж, который содержит строки
A = ("abc", "abcd", "bcd", "cde")
# Ввести элемент
item = str(input("s = "))
if (item in A):
    print(item, " in ", A, " = True")
else:
    print(item, " in ", A, " = False")
```

Результат выполнения программы:

```
s = abc
abc in ('abc', 'abcd', 'bcd', 'cde') = True
```

11. Какие методы работы с кортежами Вам известны?

Метод `index()`. Поиск позиции элемента в кортеже. Чтобы получить индекс (позицию) элемента в кортеже, нужно использовать метод `index()`. Общая форма вызова метода следующая:

`pos = T.index(item)`, где `T` – кортеж, в котором осуществляется поиск; `pos` – позиция (индекс) элемента `item` в кортеже. Первому элементу соответствует позиция 0. Если элемента нет в кортеже, генерируется исключительная ситуация. Поэтому, перед использованием метода `index()` рекомендуется делать проверку на наличие элемента (с помощью операции `in`).

Метод `count()`. Количество вхождений элемента в кортеж. Чтобы определить количество вхождений заданного элемента в кортеж используется метод `count`, общая форма которого следующая: `k = T.count(item)`, где `T` – исходный кортеж; `k` – результат (количество элементов); `item` – элемент, количество вхождений которого нужно определить. Элемент может быть составным (строка, список, кортеж).

12. Допустимо ли использование функций агрегации таких как `len()` , `sum()` и т. д. при работе с кортежами?

Да, допустимо.

13. Как создать кортеж с помощью спискового включения.

```
>>> a = [1, 2, 3]
```

```
>>> print(a)
```

```
[1, 2, 3]
```

```
>>> a[1] = 15
```

```
>>> print(a)
```

```
[1, 15, 3]
```