

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с кортежами в языке Python»

ОТЧЕТ
по лабораторной работе №8
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.

Ссылка на репозиторий: https://github.com/KuvshinChick/Py_L8.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone "https://github.com/KuvshinChick/Py_L8.git"
Cloning into 'Py_L8'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

```
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py_L8/

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L8 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 8.1 – Клонирование репозитория и создание ветки develop

```

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py_L8 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py_L8 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py_L8 (develop)
$ git commit -m "modified .gitignore & readme"
[develop bb08d28] modified .gitignore & readme
2 files changed, 5 insertions(+), 3 deletions(-)

```

Рисунок 8.2 – Обновление .gitignore и readme

8. Создайте проект PyCharm в папке репозитория.

7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

8. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

Пример 1.

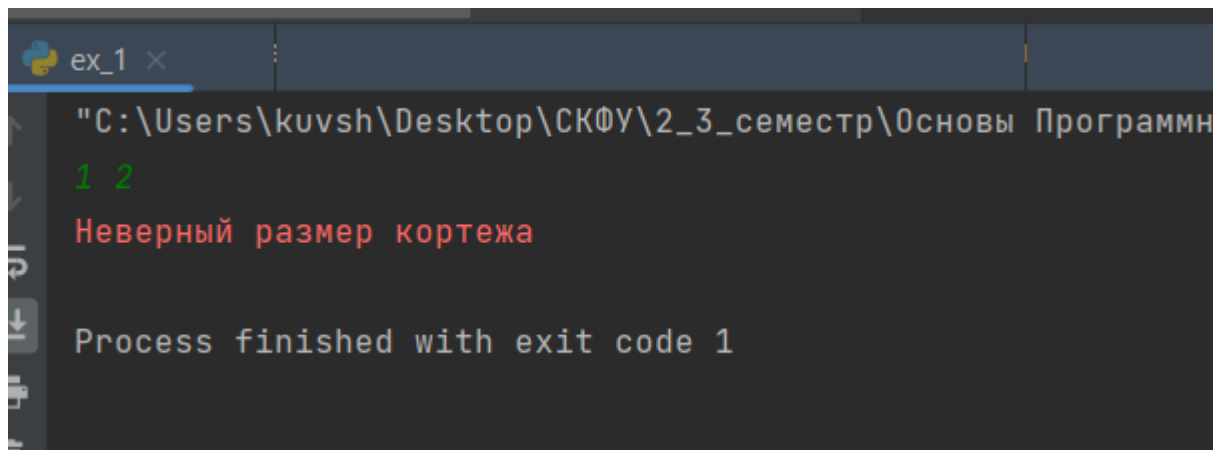
Ввести кортеж A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран. Использовать в программе вместо списков кортежи.

```
ind_1.py × ex_1.py × ex_1_2.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 ▶ if __name__ == '__main__':
8     # Ввести кортеж одной строкой.
9     A = tuple(map(int, input().split()))
10    # Проверить количество элементов кортежа.
11    if len(A) != 10:
12        print("Неверный размер кортежа", file=sys.stderr)
13        exit(1)
14
15    # Найти искомую сумму.
16    s = 0
17    for item in A:
18        if abs(item) < 5:
19            s += item
20
21    print(s)
22
```

Рисунок 8.3 – Код программы-примера

```
ex_1 ×
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инже
1 3 5 -4 8 9 5 1 2 3
6
Process finished with exit code 0
```

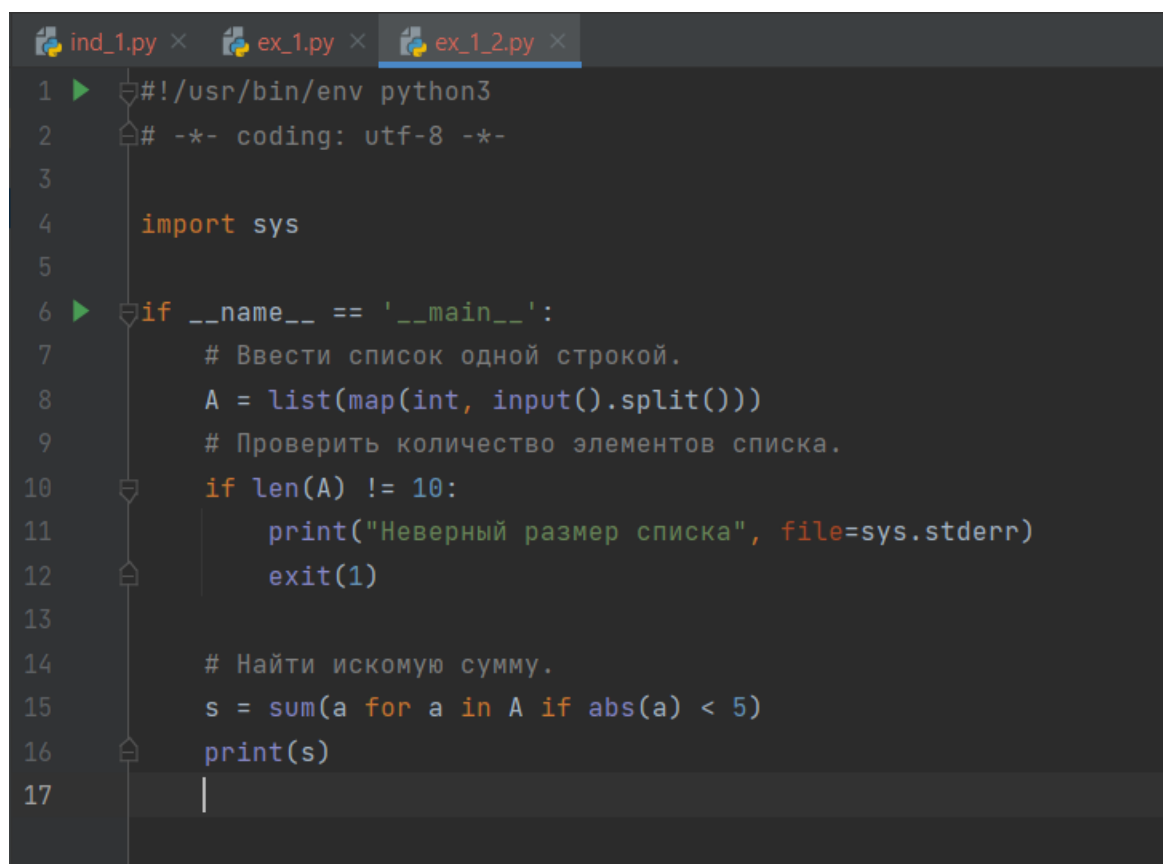
Рисунок 8.4 – Результат программы



A screenshot of a terminal window with a dark background. The title bar shows a file named 'ex_1'. The terminal output shows the file path 'C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программн', the input '1 2', a red error message 'Неверный размер кортежа' (Incorrect tuple size), and the message 'Process finished with exit code 1'.

```
ex_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программн
1 2
Неверный размер кортежа
Process finished with exit code 1
```

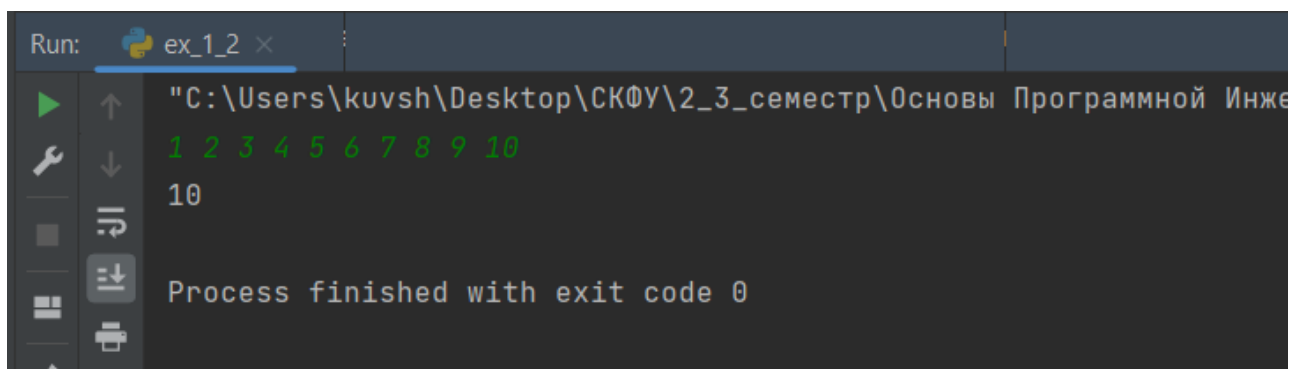
Рисунок 8.5 – Результат программы (ошибка)



A screenshot of a code editor with three tabs: 'ind_1.py', 'ex_1.py', and 'ex_1_2.py'. The 'ex_1_2.py' tab is active, showing Python code. The code includes a shebang, encoding declaration, sys import, and logic to check if the input list has 10 elements. If not, it prints an error and exits with code 1. Otherwise, it calculates the sum of absolute values less than 5 and prints it.

```
ind_1.py x ex_1.py x ex_1_2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     # Ввести список одной строкой.
8     A = list(map(int, input().split()))
9     # Проверить количество элементов списка.
10    if len(A) != 10:
11        print("Неверный размер списка", file=sys.stderr)
12        exit(1)
13
14    # Найти искомую сумму.
15    s = sum(a for a in A if abs(a) < 5)
16    print(s)
17
```

Рисунок 8.6 – Код программы-примера (с помощью списковых включений)



A screenshot of a terminal window with a dark background. The title bar shows 'Run: ex_1_2 x'. The terminal output shows the file path 'C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инже', the input '1 2 3 4 5 6 7 8 9 10', the output '10', and the message 'Process finished with exit code 0'.

```
Run: ex_1_2 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инже
1 2 3 4 5 6 7 8 9 10
10
Process finished with exit code 0
```

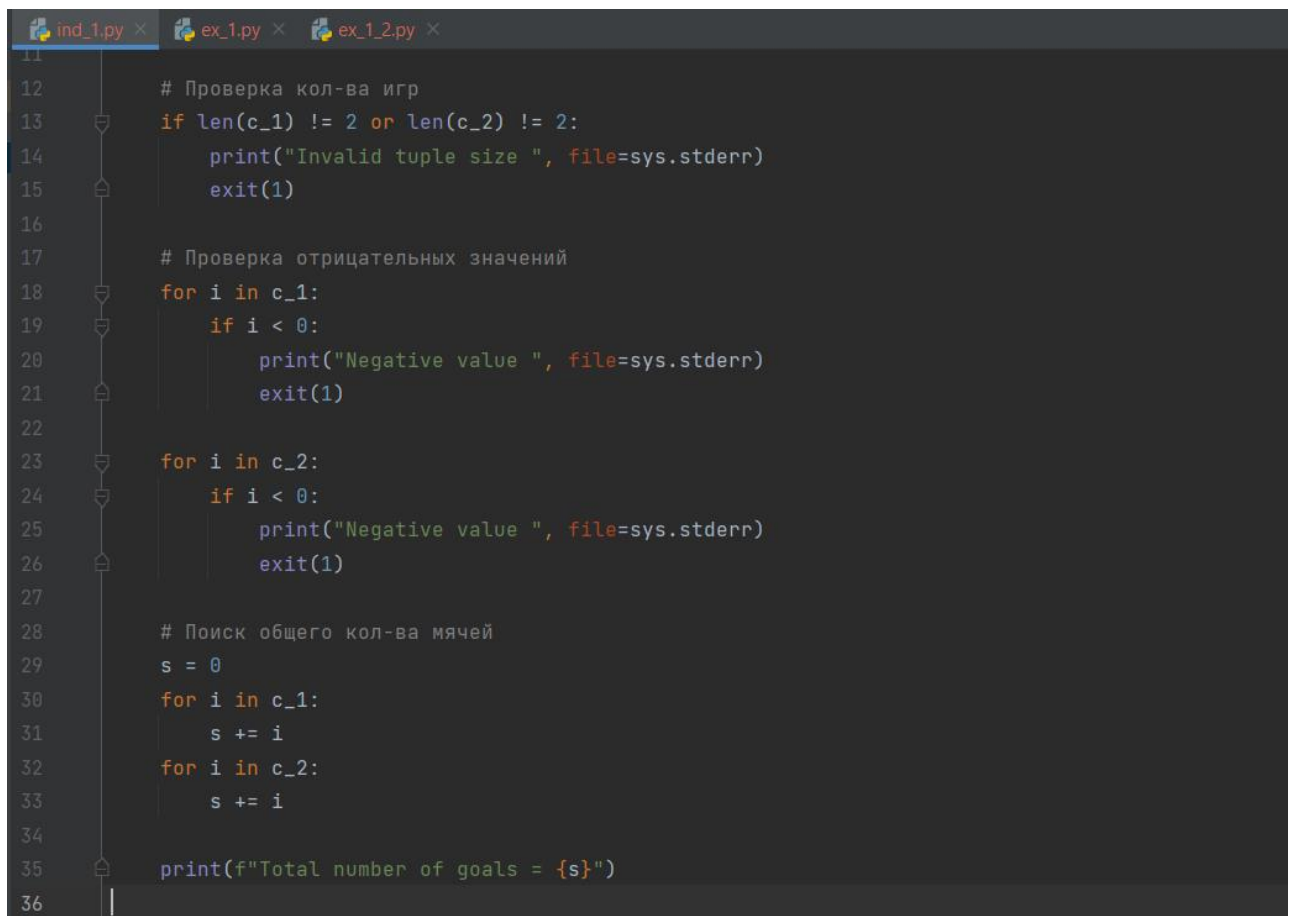
Рисунок 8.7 – Результат программы

9. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.
10. Зафиксируйте сделанные изменения в репозитории.
11. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
12. Выполните слияние ветки для разработки с веткой main / master.
13. Отправьте сделанные изменения на сервер GitHub.
14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальные задания: (Вариант 15)

Задание 1.

15. Известно количество мячей, забитых футбольной командой за каждую игру в двух чемпионатах, которое хранится в двух кортежах. В каждом из чемпионатов команда сыграла 26 игр. Найти общее количество мячей, забитых командой в двух чемпионатах.



```
11
12     # Проверка кол-ва игр
13     if len(c_1) != 2 or len(c_2) != 2:
14         print("Invalid tuple size ", file=sys.stderr)
15         exit(1)
16
17     # Проверка отрицательных значений
18     for i in c_1:
19         if i < 0:
20             print("Negative value ", file=sys.stderr)
21             exit(1)
22
23     for i in c_2:
24         if i < 0:
25             print("Negative value ", file=sys.stderr)
26             exit(1)
27
28     # Поиск общего кол-ва мячей
29     s = 0
30     for i in c_1:
31         s += i
32     for i in c_2:
33         s += i
34
35     print(f"Total number of goals = {s}")
36
```

Рисунок 8.8 – Код программы

Рисунок 8.9 – Код программы через List Comprehensions

```
ind_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L8\PyCharm\venv\Script
Goals in first championship : 0 1 0 3 6 4 7 2 0 8 1 2 5 3 2 1 4 0 7 1 3 5 6 7 8 5
Goals in first championship : 1 3 4 2 3 5 7 1 0 7 0 3 1 4 5 6 7 3 1 2 0 2 1 4 6 7
Total number of goals = 176
Process finished with exit code 0
```

Рисунок 8.10 – Результат программы

```
ind_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной
Goals in first championship : 1 2
Goals in first championship : 3 4
Invalid tuple size
Process finished with exit code 1
```

Рисунок 8.11 – Результат программы (ошибка)

```
Run: ind_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py_L8\PyCharm\venv
Goals in first championship : -1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8
Goals in first championship : 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8
Negative value
Process finished with exit code 1
```

Рисунок 8.12 – Результат программы (ошибка)

Вопросы для защиты работы

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов.

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, вторая - кортежи в памяти занимают меньший объем по сравнению со списками. В – третьих - прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

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)
```

При желании можно воспользоваться функцией `tuple()`.

```
>>> a = tuple([1, 2, 3, 4])
>>> print(a)
(1, 2, 3, 4)
```

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 "<pyshell#24>", line 1, in <module>

```
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. Но есть способ лучше! Как мы кортеж собираем, так его можно и разобрать:

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

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

```
name # 'Bob'
```

```
age # 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. Смотрите:

```
(a, b, c) = (1, 2, 3)
```

```
a # 1
```

```
b # 2
```

```
c # 3
```

Используя множественное присваивание, можно провернуть интересный трюк: обмен значениями между двумя переменными. Вот код:

```
a = 100
```

```
b = 'foo'
```

$(a, b) = (b, a)$

$a \# 'foo'$

$b \# 100$

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

$T2 = T1[i:j]$

здесь $T2$ – 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

$$T2 = T1 * n$$

здесь

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.

11. Какие методы работы с кортежами Вам известны?

Метод index(). Поиск позиции элемента в кортеже

Метод count(). Количество вхождений элемента в 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.

В этом примере показано использование списковых включений для расчета суммы, однако в отличие от выражения [a for a in A ...] , которое на выходе дает нам список, выражение (a for a in A ...) дает на выходе специальный объект генератора, а не кортеж. Для преобразования генератора в кортеж необходимо воспользоваться вызовом tuple() .