

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе №8
«Работа с кортежами в языке Python»**

по дисциплине «Основы программной инженерии»

Выполнила:
Первых Дарья Александровна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
>>> a = [1, 2, 3]
>>> print(a)
[1, 2, 3]
>>> a[1] = 15
>>> print(a)
[1, 15, 3]
>>>
```

Рисунок 1 – Пример замены элементов списка

```
>>> b = (1, 2, 3)
>>> print(b)
(1, 2, 3)
>>> b[1] = 15
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Рисунок 2 – Пример невозможности замены элемента в кортеже

```
>>> lst = [10, 20, 30]
>>> tpl = (10, 20, 30)
>>> print(lst.__sizeof__())
104
>>> print(tpl.__sizeof__())
48
>>>
```

Рисунок 3 – Пример памяти, занимаемой кортежем и списком

```
>>> a = ()
>>> print(type(a))
<class 'tuple'>
>>> b = tuple()
>>> print(type(b))
<class 'tuple'>
```

Рисунок 4 – Пример создания кортежа

```
>>> a = (1, 2, 3, 4, 5)
>>> print(type(a))
<class 'tuple'>
>>> print(a)
(1, 2, 3, 4, 5)
>>>
```

Рисунок 5 – Пример представления кортежа

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

Рисунок 6 – Пример использования функции tuple() для создания кортежа

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

Рисунок 7 – Пример кортежа с одним элементом

```
>>> a = (1, 2, 3, 4, 5)
>>> print(a[0])
1
>>> print(a[1:3])
(2, 3)
>>> a[1]
2
>>> a[1] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Рисунок 8 – Пример доступа к элементам кортежа

```
>>> a = (1, 2, 3, 4, 5)
>>> del a[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
>>>
```

Рисунок 9 – Пример удаления одного элемента кортежа

```
>>> del a
>>> print (a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>>
```

Рисунок 10 – Пример удаления кортежа полностью

```
>>> lst = [1, 2, 3, 4, 5]
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[1, 2, 3, 4, 5]
>>> tpl = tuple(lst)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(1, 2, 3, 4, 5)
>>>
```

Рисунок 11 – Пример преобразования списка в кортеж

```
>>> tpl = (2, 4, 6, 8, 10)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(2, 4, 6, 8, 10)
>>> lst = list(tpl)
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[2, 4, 6, 8, 10]
>>>
```

Рисунок 12 – Пример преобразования кортежа в список

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

(name, age) = name_and_age
name # 'Bob'
age  # 42
```

Рисунок 13 – Пример деструктуризации кортежа

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

Рисунок 14 – Пример способа, который получает и сразу разбирает значение функции

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

```
a #42
```

Рисунок 15 – Пример разбора кортежа с одним элементом

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

```
a #1
```

```
b #2
```

```
c #3
```

Рисунок 16 – Пример множественно присваивания значения элементам кортежа

```
a = 100
```

```
b = 'foo'
```

```
(a, b) = (b, a)
```

```
a # 'foo'
```

```
b # 100
```

Рисунок 17 – Пример обмена значениями между двумя переменными

```
# Операция tuple()
# 1. Создание кортежа из слова 'Hello'
d = tuple('Hello') # d = ('H', 'e', 'l', 'l', 'o')

# 2. Создание кортежа из списка
# Заданный список
lst = [2, "abc", 3.88]

# Создать кортеж
e = tuple(lst) # e = (2, 'abc', 3.88)

# 3. Создание кортежа из другого кортежа
f = tuple((3, 2, 0, -5)) # f = (3, 2, 0, -5)
```

Рисунок 18 – Пример создания кортежа из итерированного объекта

```

# Операция [i:j] - взятие среза
# 1. Кортеж, содержащий целые числа
A = (0, 1, 2, 3)
item = A[0:2] # item = (0, 1)

# 2. Кортеж, содержащий список
A = (2.5, ['abcd', True, 3.1415], 8, False, 'z')
item = A[1:3] # item = (['abcd', True, 3.1415], 8)

# 3. Кортеж, содержащий вложенный кортеж
A = (3, 8, -11, "program")
B = ("Python", A, True)
item = B[:3] # item = ('Python', (3, 8, -11, 'program'), True)
item = B[1:] # item = ((3, 8, -11, 'program'), True)

```

Рисунок 19 – Пример работы операции T[i:j]. Взятие среза в кортеже

```

# Кортежи. Конкатенация +
# Конкатенация двух кортежей
A = (1, 2, 3)
B = (4, 5, 6)
C = A + B # C = (1, 2, 3, 4, 5, 6)

# Конкатенация кортежей со сложными объектам
D = (3, "abc") + (-7.22, ['a', 5]) # D = (3, 'abc', -7.22, ['a', 5])

# Конкатенация трёх кортежей
A = ('a', 'aa', 'aaa')
B = A + (1, 2) + (True, False) # B = ('a', 'aa', 'aaa', 1, 2, True, False)

```

Рисунок 20 – Пример конкатенации кортежей

```

# Кортежи. Повторение *
# Кортеж, который содержит простые числа
A = (1, 2, 3) * 3 # A = (1, 2, 3, 1, 2, 3, 1, 2, 3)

# Кортеж, который содержит вложенные объекты
B = ("ab", ["1", "12"])*2 # A=('ab', ['1', '12'], 'ab', ['1', '12'])

```

Рисунок 21 – Пример повторения кортежа

```

# Обход кортежа в цикле
# 1. Цикл for
# Заданный кортеж
A = ("abc", "abcd", "bcd", "cde")

# Вывести все элементы кортежа
for item in A:
    print(item)

# 2. Цикл while
# Исходный кортеж - целые числа
A = (-1, 3, -8, 12, -20)

# Вычислить количество положительных чисел
i = 0
k = 0 # количество положительных чисел

while i < len(A):
    if (A[i]<0):
        k = k + 1
    i = i + 1

# Вывести результат
print("k = ", k)

# 3. Обход в цикле for
# Заданный кортеж, содержащий строки
A = ("abc", "ad", "bcd")

# Сформировать новый список из элементов кортежа A
# в новом списке B, каждый элемент удваивается
B = [item * 2 for item in A]

```

Рисунок 22 – Примеры обхода кортежа в цикле

```
abc
abcd
bcd
cde
k = 3
A = ('abc', 'ad', 'bcd')
B = ['abcabc', 'adad', 'bcdbcd']
```

Рисунок 23 – Результат выполнения программы

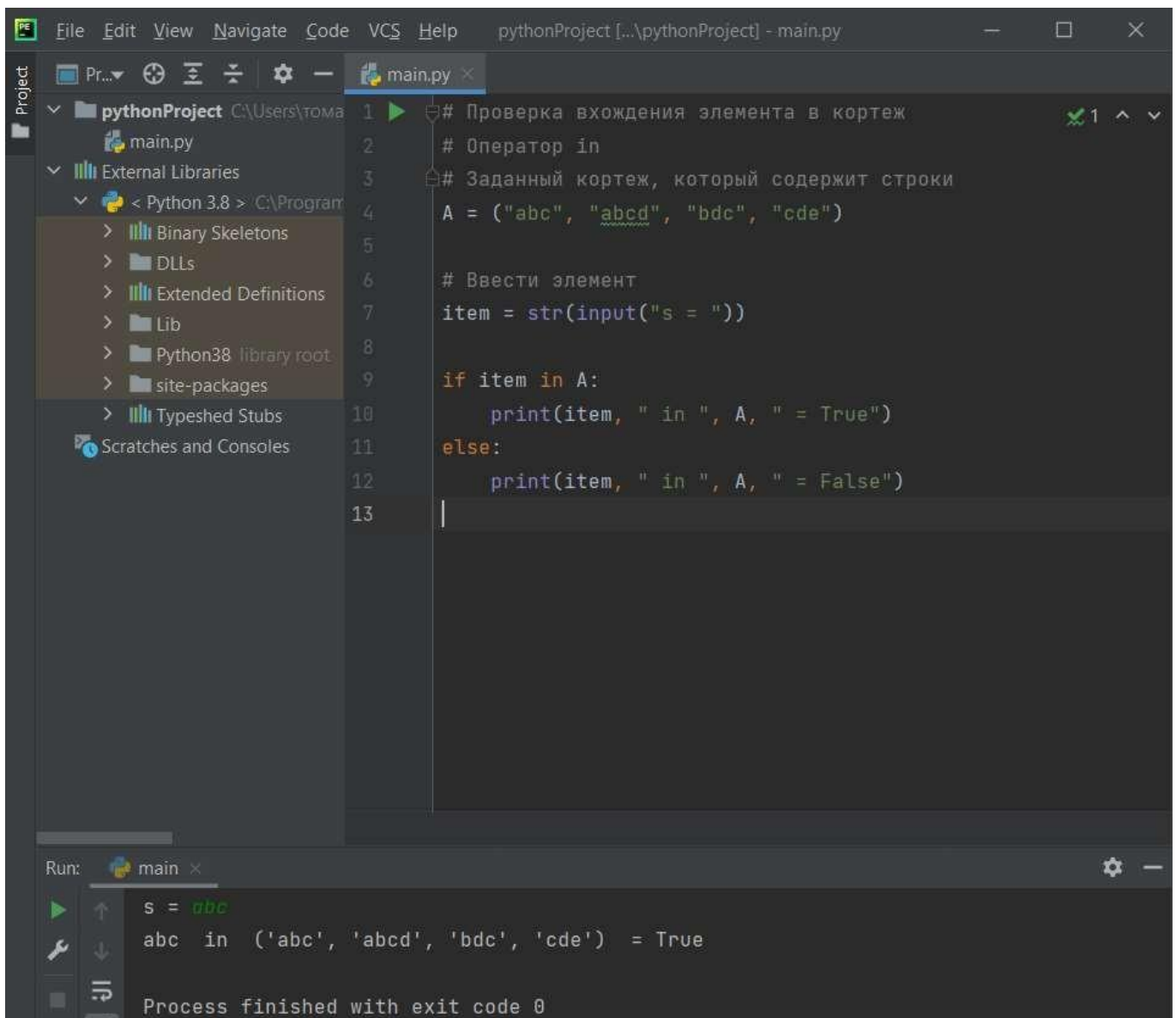


Рисунок 24 – Пример операции in. Проверка вхождения элемента в кортеж


```
# Метод index - определяет позиции (индекс) элемента в кортеже
# Заданный кортеж
A = ("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")

# Запрос к вводу названия дня недели
day = str(input("Enter day: "))

# Корректно вычислить индекс
if day in A: # проверка, есть ли строка day в кортеже A
    num = A.index(day)
    print("Number of day = ", num + 1)
else:
    num = -1
    print("Wrong day.")
```

Рисунок 25 – Пример работы метода index(). Поиск позиции элемента в кортеже

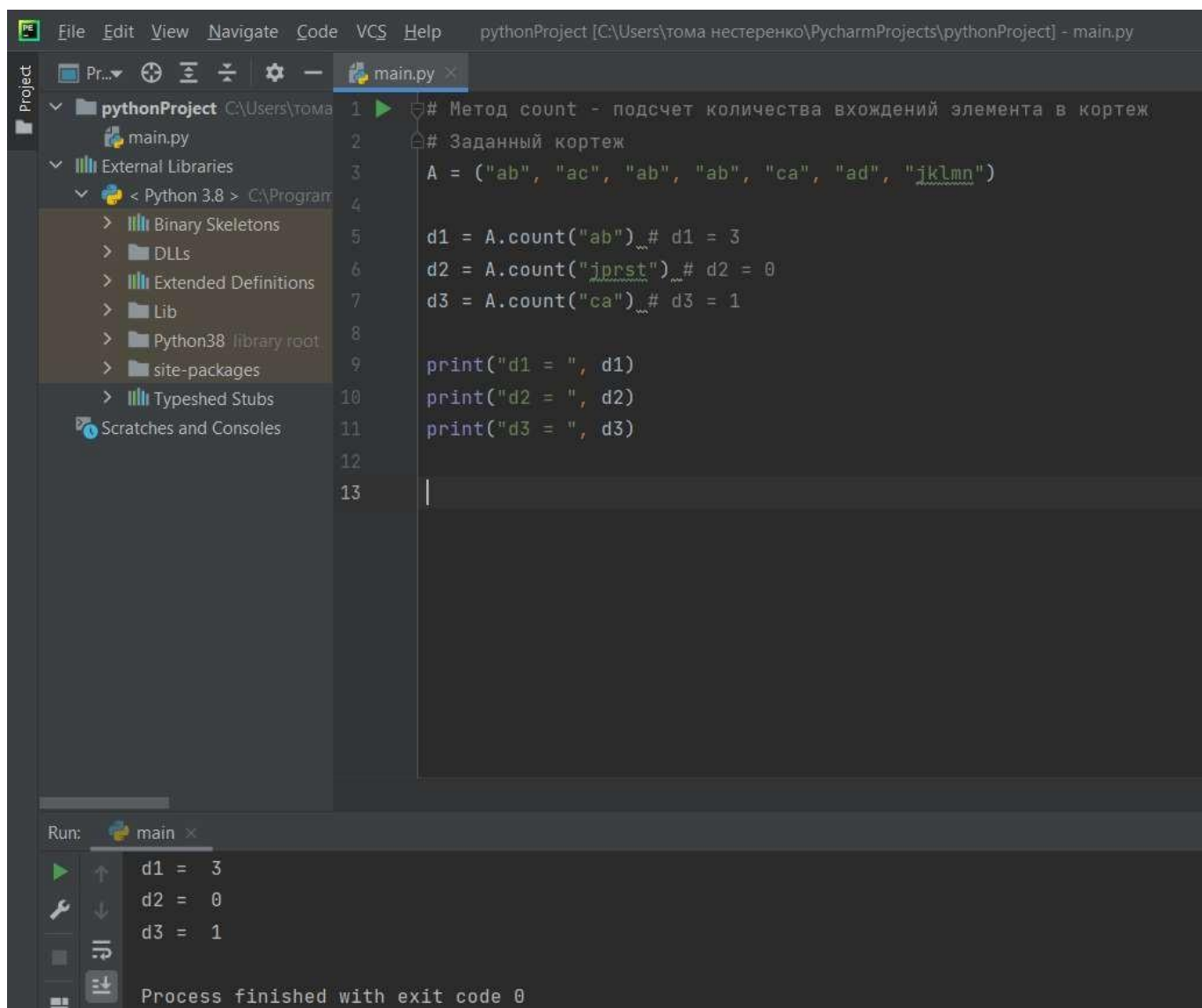


Рисунок 26 – Пример работы метода count(). Количество вхождений элемента в кортеж

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     # Ввести кортеж одной строкой.
8     A = tuple(map(int, input().split()))
9     # Проверить количество элементов кортежа.
10    if len(A) != 10:
11        print("Неверный размер кортежа", file=sys.stderr)
12        exit(1)
13
14    # Найти искомую сумму.
```

main ×

C:\Users\podar\study\anaconda\envs\8LR\python.exe "C:/Users/pod
5 6 7 2 1
Неверный размер кортежа

```
▶ #!/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.
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
```

main ×

C:\Users\podar\study\anaconda\envs\8LR\python.exe "C:/Users/po
4 5 6 7 7 12 2 45 12 4
10

Рисунок 27 – Пример решения задачи: ввести кортеж A из 10 элементов, найти сумму элементов, меньших модулю 5, и вывести её на экран. Использовать в программе вместо списков кортежи.

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

main x

C:\Users\podar\study\anaconda\envs\8LR\python.exe "C:/Users/pod

4 5 6 12 3 1

Неверный размер списка

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

import sys

▶ if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum(a for a in A if abs(a) < 5)
    print(s)
```

main x

C:\Users\podar\study\anaconda\envs\8LR\python.exe "C:/Users/pod

4 5 9 12 27 7 2 57 5 7

6

Рисунок 30 – Пример решения задачи с помощью списковых включений
Индивидуальное задание. Известны данные о вместимости (в гигабайтах) и

стоимости (в рублях) каждого из 13 типов жестких магнитных дисков (винчестеров).
Напечатать вместимость тех винчестеров, которые стоят больше s рублей.

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

```
import sys
```

```
if __name__ == '__main__':
```

```
    A = (2, 2.1, 4, 7.8, 3.2, 2.1, 7.9, 4.2, 8.4, 33.8, 2, 4.2, 2.1)
```

```
    B = (560, 790, 1500, 2300, 1200, 790, 2500, 1500, 3000, 8550, 610, 1450, 950)
```

```
    s = float(input("Введите сумму s: "))
```

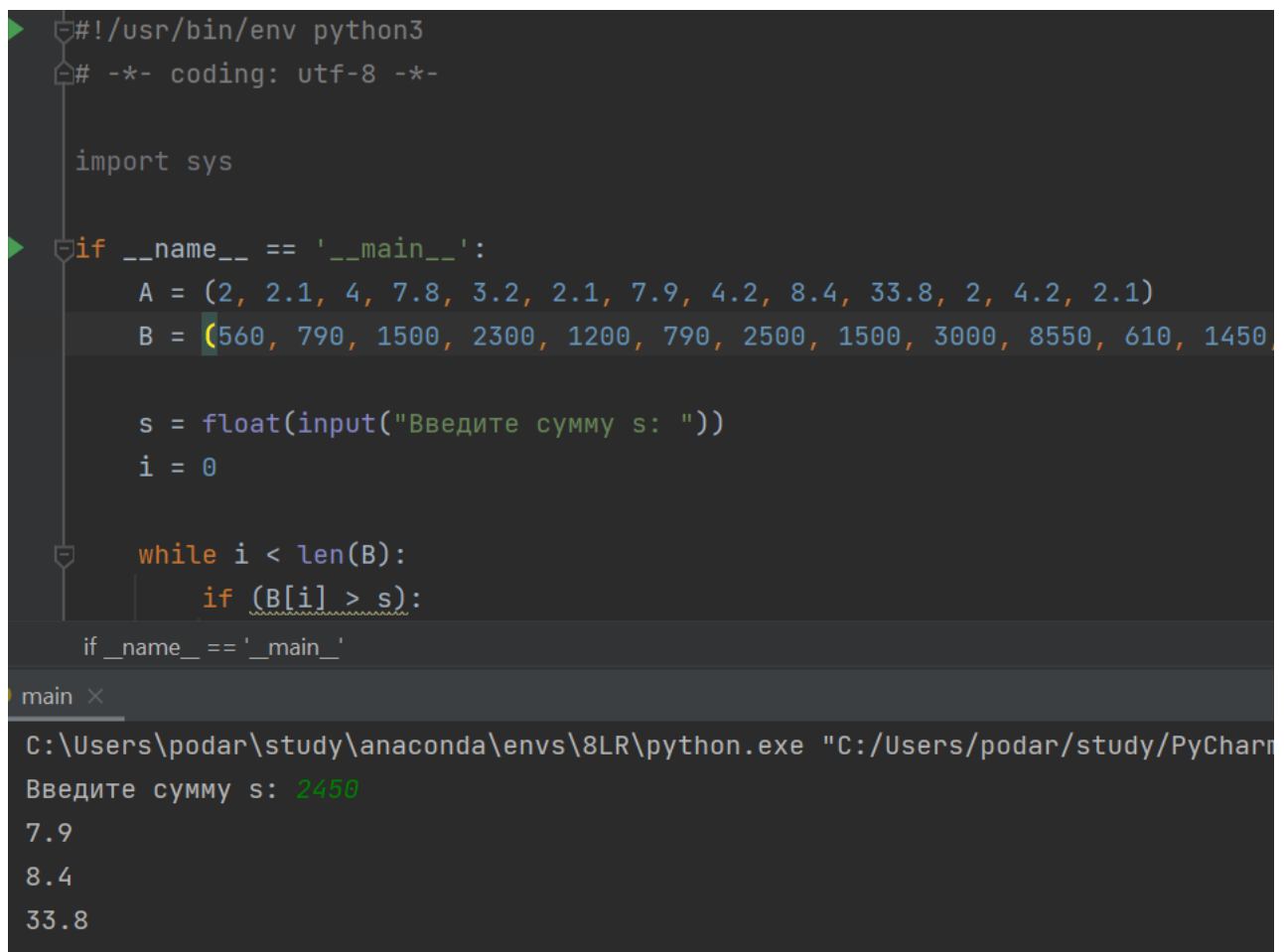
```
    i = 0
```

```
    while i < len(B):
```

```
        if (B[i] > s):
```

```
            print (A[i])
```

```
        i = i + 1
```



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

import sys

if __name__ == '__main__':
    A = (2, 2.1, 4, 7.8, 3.2, 2.1, 7.9, 4.2, 8.4, 33.8, 2, 4.2, 2.1)
    B = (560, 790, 1500, 2300, 1200, 790, 2500, 1500, 3000, 8550, 610, 1450, 950)

    s = float(input("Введите сумму s: "))
    i = 0

    while i < len(B):
        if (B[i] > s):
            print (A[i])
        i = i + 1

if __name__ == '__main__':
```

main x

C:\Users\podar\study\anaconda\envs\8LR\python.exe "C:/Users/podar/study/PyCharm" Введите сумму s: 2450

7.9

8.4

33.8

Рисунок 33 – Результат решения

Вопросы

1. Что такое списки в языке Python?

Список (list) - структура данных для хранения объектов различных типов. Так вот, список очень похож на массив, только, как было уже сказано выше, в нем можно хранить объекты различных типов. Размер списка не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

2. Каково назначение кортежей в языке Python?

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

Назначение кортежей:

- 1) Обезопасить данные от случайного изменения.
- 2) Экономия места.
- 3) Прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки.

3. Как осуществляется создание кортежей?

- 1) `A = ()`
- 2) `A = tuple([1, 2, 3])`

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется через указание индекса. `print(a[0])` или `print(a[1:3])`

5. Зачем нужна распаковка (деструктуризация) кортежа?

Деструктуризация кортежа нужна для быстрого разбиения кортежа на отдельные элементы, для более удобного доступа и работы.

6. Какую роль играют кортежи в множественном присваивании?

Элементам кортежа можно сразу последовательно присваивать значения. А также используя множественное присваивание, можно проверить интересный трюк: обмен значениями между двумя переменными.

7. Как выбрать элементы кортежа с помощью среза?

Необходимо ввести: `item = A[0:2]`, `item = B[:3]`, `item = B[1:]` Общая форма операции взятия среза для кортежа:

```
t2 = t1[i:j]
```

8. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом `+`. Общая форма операции:

```
T3 = T1 + T2
```

Кортеж может быть образован путём операции повторения, обозначаемой символом *.
Общая форма операции:

```
T2 = T1 * n
```

9. Как выполняется обход элементов кортежа?

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

```
# Обход кортежа в цикле
# 1. Цикл for
# Заданный кортеж
A = ("abc", "abcd", "bcd", "cde")

# Вывести все элементы кортежа
for item in A:
    print(item)

# 2. Цикл while
# Исходный кортеж - целые числа
A = (-1, 3, -8, 12, -20)

# Вычислить количество положительных чисел
i = 0
k = 0 # количество положительных чисел

while i < len(A):
    if (A[i] > 0):
        k = k + 1
    i = i + 1

# Вывести результат
print("k = ", k)

# 3. Обход в цикле for
# Заданный кортеж, содержащий строки
A = ("abc", "ad", "bcd")

# Сформировать новый список из элементов кортежа A
# в новом списке B, каждый элемент удваивается
B = [item * 2 for item in A]
```

10. Как проверить принадлежность элемента кортежу?

Проверить принадлежность элемента кортежу можно с помощью операции in.

```
if (item in A):  
    print(item, " in ", A, " = True")  
else:  
    print(item, " in ", A, " = False")
```

11. Какие методы работы с кортежами Вам известны?

Метод `index()`: `pos = A.index(item)` Метод `count()`: `k = A.count(item)` где `item` – элемент кортежа.

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.

`x = 10`

`a = tuple([i for i in range(x)])`

`a = tuple(int(i) for i in input().split())`