

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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники института
перспективной инженерии

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4
дисциплины «Программирование на Python»
«Работа со списками и кортежами в языке Python»
Вариант 12

Выполнила:
Коробка В.А.,
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Проверил:
Воронкин Р.А.,
доцент департамента цифровых,
робототехнических систем и
электроники института перспективной
инженерии

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

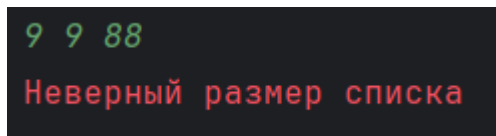
Цель: приобретение навыков по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Адрес репозитория: <https://github.com/13r4lera/Python4.git>

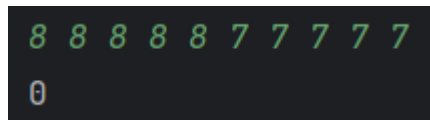
Был создан общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, выполнено клонирование данного репозитория. Файл .gitignore был дополнен правилами, необходимыми для работы с PyCharm (раскомментирована строка с .idea/).

Был проработан пример 1 из лабораторной работы. Была написана программа, в которой пользователь должен ввести 10 элементов, а внутри программы находится сумма элементов, меньших по модулю 5. Затем эта сумма выводится на экран. Вместо списков в программе используются кортежи.



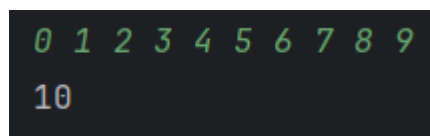
```
9 9 88
Неверный размер списка
```

Рисунок 1. Ошибочный ввод данных



```
8 8 8 8 8 7 7 7 7 7
0
```

Рисунок 2. Ввод кортежа со слишком большими числами



```
0 1 2 3 4 5 6 7 8 9
10
```

Рисунок 3. Результат работы программы

Полный код примера:

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import sys
```

```

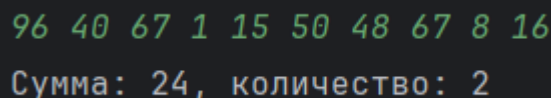
if __name__ == '__main__':
    A = tuple(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    s = sum(a for a in A if abs(a) < 5)
    print(s)

```

Было выполнено индивидуальное задание 1 по варианту 12. Условие задания: ввести список A из 10 элементов, найти сумму элементов, больших 2 и меньших 20 и кратных 8, их количество и вывести результаты на экран.

Программа считывает с клавиатуры 10 целых чисел и сохраняет их в кортеж A. Сначала выполняется проверка, что введено ровно 10 элементов, иначе программа выводит сообщение об ошибке и завершает работу. Затем с помощью цикла for программа проходит по всем элементам кортежа и находит сумму и количество чисел, которые больше 2, меньше 20 и кратны 8. Результаты выводятся на экран в одной строке с помощью f-строки, показывая сумму и количество подходящих элементов.

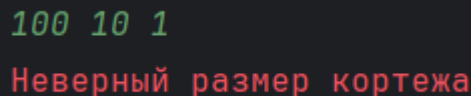


```

96 40 67 1 15 50 48 67 8 16
Сумма: 24, количество: 2

```

Рисунок 4. Результат работы программы



```

100 10 1
Неверный размер кортежа

```

Рисунок 5. Вывод ошибки

Полный код программы:

```

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

```

```

import sys

if __name__ == '__main__':
    A = tuple(map(int, input().split()))

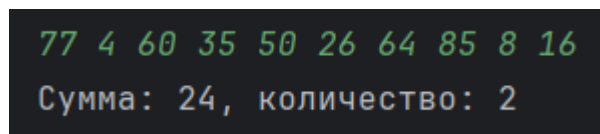
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        sys.exit(1)

    s = 0
    count = 0
    for item in A:
        if 2 < item < 20 and item % 8 == 0:
            s += item
            count += 1

    print(f"Сумма: {s}, количество: {count}")

```

Затем был написан второй вариант программы, но с использованием List Comprehensions. С помощью list comprehension создаётся новый список В, содержащий числа, которые больше 2, меньше 20 и кратны 8. На экран выводится сумма элементов списка В и их количество в одной строке с помощью f-строки.



```

77 4 60 35 50 26 64 85 8 16
Сумма: 24, количество: 2

```

Рисунок 6. Результат работы программы

Полный код программы:

```

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

```

```
import sys
```

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

```
    A = tuple(map(int, input().split()))
```

```
    if len(A) != 10:
```

```
        print("Неверный размер кортежа", file=sys.stderr)
```

```
        sys.exit(1)
```

```
my_list = [x for x in A if 2 < x < 20 and x % 8 == 0]
```

```
s = sum(my_list)
```

```
l = len(my_list)
```

```
print(f"Сумма: {s}, количество: {l}")
```

Было выполнено индивидуальное задание 2 по варианту 12. Условие задания: В списке, состоящем из вещественных элементов, вычислить:

1. количество элементов списка, лежащих в диапазоне от A до B;
2. сумму элементов списка, расположенных после максимального элемента.

Программа считывает с клавиатуры список вещественных чисел и сохраняет его в список `my_list`. Далее выполняется проверка на пустой список, при которой в случае ошибки выводится сообщение и работа программы завершается. После этого вводятся границы диапазона `a` и `b`, и с помощью генераторного выражения определяется количество элементов списка, лежащих в заданном диапазоне. Затем находится максимальный элемент списка и вычисляется сумма элементов, расположенных после него. Результаты вычислений выводятся на экран в текстовом виде.

```
Введите элементы списка: 66 70 78 63 14
Введите нижнюю границу диапазона: 44
Введите верхнюю границу диапазона: 64
В заданном диапазоне находится 1 элементов.
Сумма элементов списка, расположенных после максимального: 77.0
```

Рисунок 7. Результат работы программы

Полный код программы:

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import sys
```

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

```
    my_tuple = tuple(map(float, input("Введите элементы списка: ").split()))
```

```
    if len(my_tuple) == 0:
```

```
        print("Список пуст", file=sys.stderr)
```

```
        sys.exit(1)
```

```
    a = float(input("Введите нижнюю границу диапазона: "))
```

```
    b = float(input("Введите верхнюю границу диапазона: "))
```

```
    s = sum(1 for x in my_tuple if a <= x <= b)
```

```
    print(f"В заданном диапазоне находится {s} элементов.")
```

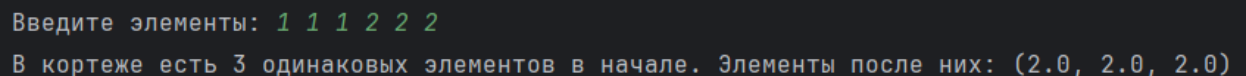
```
    max_index = my_tuple.index(max(my_tuple))
```

```
    s_after_max = sum(my_tuple[max_index + 1:])
```

```
    print(f"Сумма элементов списка, расположенных после  
максимального: {s_after_max}")
```

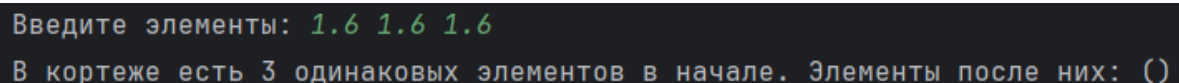
Было выполнено индивидуальное задание 3 по варианту 12. Условие задания: В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующими за последним из них. Рассмотреть возможность того, что весь массив заполнен одинаковыми элементами. Условный оператор не использовать.

Была написана программа, которая считывает с клавиатуры элементы кортежа вещественных чисел. Выполняется проверка на пустоту кортежа, при которой в случае отсутствия элементов работа программы завершается с сообщением об ошибке. Далее определяется количество одинаковых элементов, расположенных в начале кортежа, путём поиска первого элемента, отличного от начального. После этого выводится количество таких элементов и все элементы, следующие за последним из них.



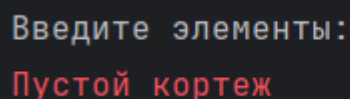
```
Введите элементы: 1 1 1 2 2 2
В кортеже есть 3 одинаковых элементов в начале. Элементы после них: (2.0, 2.0, 2.0)
```

Рисунок 8. Результат работы программы



```
Введите элементы: 1.6 1.6 1.6
В кортеже есть 3 одинаковых элементов в начале. Элементы после них: ()
```

Рисунок 9. Ввод одинаковых чисел



```
Введите элементы:
Пустой кортеж
```

Рисунок 10. Вывод ошибки

Полный код программы:

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

```
import sys
```

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

```
my_tuple = tuple(map(float, input("Введите элементы: ").split()))
```

```
if len(my_tuple) == 0:
```

```
    print("Пустой кортеж", file=sys.stderr)
```

```
    sys.exit(1)
```

```
first = my_tuple[0]
```

```
count = 0
```

```
while count < len(my_tuple) and my_tuple[count] == first:
```

```
    count += 1
```

```
print(f"В кортеже есть {count} одинаковых элементов в начале. Элементы  
после них: {my_tuple[count:]}")
```

Ответы на контрольные вопросы

1. Что такое списки в языке Python?

Список (list) - это структура данных для хранения объектов различных типов. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить его элементы в квадратные скобки. Например, `my_list = ['один', 10, 2.25, [5, 15], 'пять']`.

3. Как организовано хранение списков в оперативной памяти?

При создании списка в памяти резервируется область, которую можно условно назвать некоторым "контейнером", в котором хранятся ссылки на другие элементы данных в памяти.

4. Каким образом можно перебрать все элементы списка?

Читать элементы списка можно с помощью цикла `for elem in my_list`. Если нужны индексы элементов, то можно использовать `for i in range(len(my_list))`. Для получения в цикле одновременно элемента списка и

его индекса необходимо использовать функцию `enumerate`, которая генерирует кортежи, состоящие из двух элементов - индекса элемента и самого элемента (`for i, item in enumerate(my_list)`).

5. Какие существуют арифметические операции со списками?

Для объединения списков можно использовать оператор сложения `+`. Список можно повторять с помощью оператора умножения `*`.

6. Как проверить, есть ли элемент в списке?

Чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор `in` (`if 3 in lst`). Если требуется, чтобы элемент отсутствовал в списке, необходимо использовать оператор `not in`.

7. Как определить число вхождений заданного элемента в списке?

Метод `count` можно использовать для определения числа сколько раз данный элемент встречается в списке.

8. Как осуществляется добавление (вставка) элемента в список?

Метод `insert` можно использовать, чтобы вставить элемент в список. Индексы для вставляемых элементов также начинаются с нуля.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод `sort`. Для сортировки списка в порядке убывания необходимо вызвать метод `sort` с аргументом `reverse=True`.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе `pop`. Если не указывать индекс, то функция удалит последний элемент. Элемент можно удалить с помощью метода `remove`. Оператор `del` можно использовать для тех же целей. Можно удалить несколько элементов с помощью оператора среза. Можно удалить все элементы с помощью метода `clear`.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса

языка, которая предоставляет простой способ построения списков. Например, `a = [i for i in range(int(input()))]`. Функции `map` и `filter` позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов. Например, `b = [i for i in a if i % 2 == 0]`.

12. Как осуществляется доступ к элементам списков с помощью срезов?

Слайс задается тройкой чисел, разделенных запятой `start:stop:step`. `Start` - позиция, с которой нужно начать выборку, `stop` - конечная позиция, `step` - шаг. Выборка не включает элемент определяемый `stop`.

13. Какие существуют функции агрегации для работы со списками?

`len(L)` - получить число элементов в списке `L`. `min(L)` - получить минимальный элемент списка `L`. `max(L)` - получить максимальный элемент списка `L`. `sum(L)` - получить сумму элементов списка `L`, если список `L` содержит только числовые значения. Для функций `min` и `max` элементы списка должны быть сравнимы между собой.

14. Как создать копию списка?

Для создания копии списка необходимо использовать либо метод `copy`, либо использовать оператор среза, не использовать "псевдонимы" при работе со списками не рекомендуется.

15. Самостоятельно изучите функцию `sorted` языка Python. В чем ее отличие от метода `sort` списков?

Функция `sorted()` возвращает новый отсортированный список и может принимать любой итерируемый объект, не изменяя исходный. Метод `sort()` изменяет сам список и работает только со списками, возвращая `None`.

16. Что такое кортежи в языке Python?

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

17. Каково назначение кортежей в Python?

С их помощью можно обезопасить данные от случайного изменения. Если есть желание поработать с массивом данных, но при этом данные

непосредственно менять не нужно, тогда стоит использовать кортежи. Кортежи занимают в памяти меньший объем по сравнению со списками. Кортежи работают быстрее, чем списки. Кортежи можно использовать в качестве ключа у словаря.

18. Как осуществляется создание кортежей?

Кортеж с заданным содержанием создается также как список, только вместо квадратных скобок используются круглые. При желании можно воспользоваться функцией `tuple()`. Чтобы создать кортеж из одного элемента, нужно написать в круглых скобках элемент и поставить после него запятую.

19. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка - через указание индекса.

20. Зачем нужна распаковка (деструктуризация) кортежа?

Мы можем разобрать кортеж с помощью `(name, age) = name_and_age`. Для кортежа из одного элемента: `(a,) = (42,)`. Если после имени переменной не поставить запятую, то синтаксической ошибки не будет, но в переменную `a` кортеж запишется целиком, т.е. ничего не распакуется.

21. Какую роль играют кортежи в множественном присваивании?

Из-за того, что кортежи легко собирать и разбирать, в Python удобно делать множественное присваивание: `(a, b, c) = (1, 2, 3)`. Можно совершить обмен значениями между двумя переменными.

22. Как выбрать элементы кортежа с помощью среза?

`T2 = T1[i:j]`, где `i` и `j` - верхняя и нижняя границы среза. Операция взятия среза для кортежа может иметь модификации такие же как и для списков.

23. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом `+`: `T3 = T1 + T2`. Кортеж может быть образован путем операции повторения, обозначаемой символом `*`: `T2 = T1 * n`.

24. Как выполняется обход элементов кортежа?

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

25. Как проверить принадлежность элемента кортежу?

С помощью операции `in` (`if (item in A)`).

26. Какие методы работы с кортежами Вам известны?

Чтобы получить индекс (позицию) элемента в кортеже, нужно использовать метод `index()`. Чтобы определить количество вхождений заданного элемента в кортеж используется метод `count()`.

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

Да, допустимо.

28. Как создать кортеж с помощью спискового включения?

Выражение `(a for a in A ...)` дает на выходе специальный объект генератора, а не кортеж. Для преобразования генератора в кортеж необходимо воспользоваться вызовом `tuple()`.

Вывод: В ходе лабораторной работы были изучены и применены операции со списками и кортежами в Python, включая вычисление сумм и количества элементов по условиям, работу с диапазонами, генераторные выражения, срезы и методы агрегации.

