

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины «Программирование на Python»
Вариант 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокоммуникаций, доцент
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

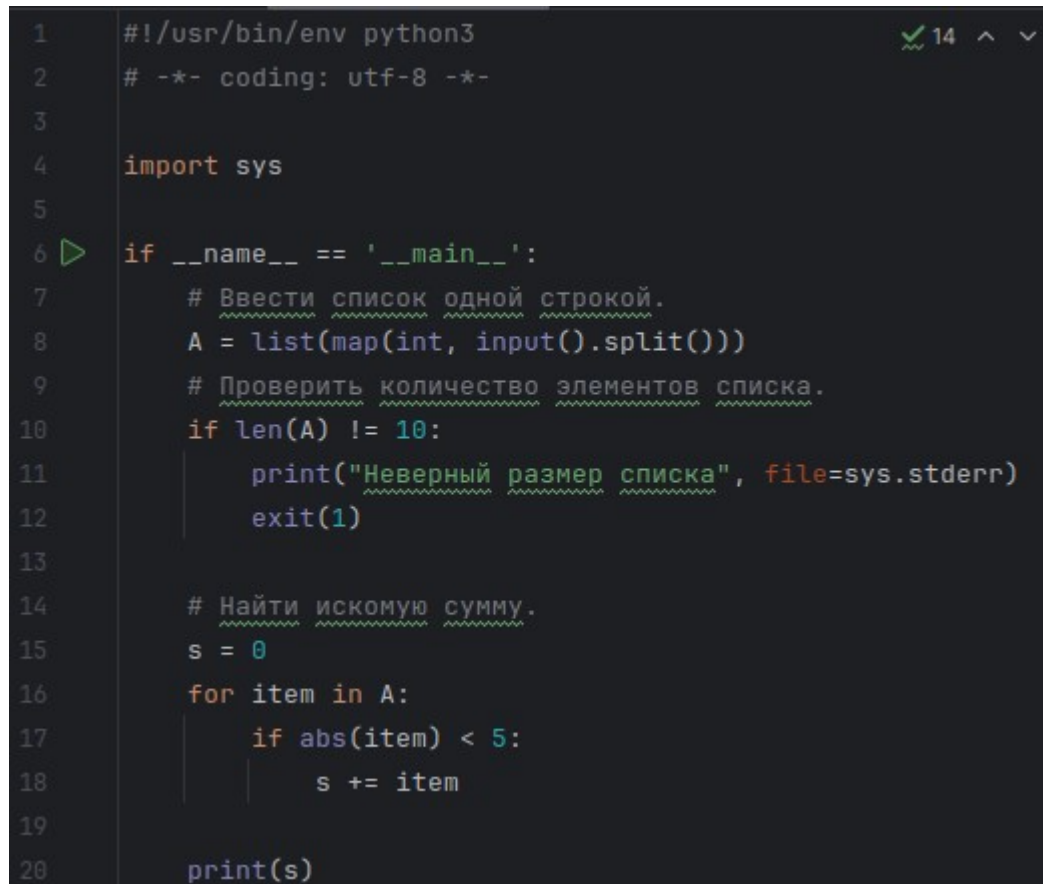
Ставрополь, 2023 г.

Тема: Работа со списками в языке Python

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

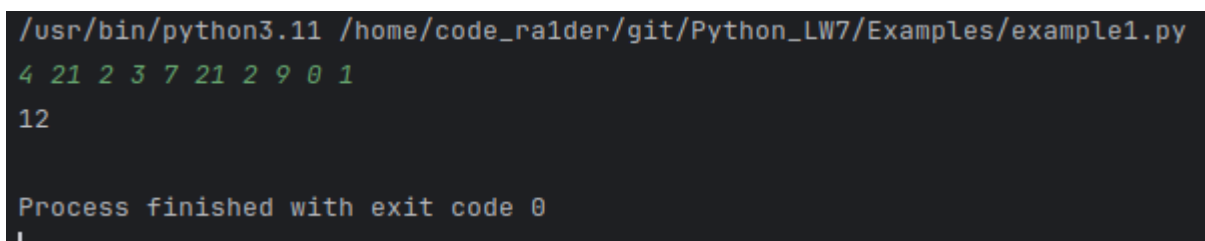
Порядок выполнения работы

1. Проработал пример №1: Ввести список A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран



```
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 = 0
16     for item in A:
17         if abs(item) < 5:
18             s += item
19
20     print(s)
```

Рисунок 1. Программа примера 1



```
/usr/bin/python3.11 /home/code_raider/git/Python_LW7/Examples/example1.py
4 21 2 3 7 21 2 9 0 1
12
Process finished with exit code 0
```

Рисунок 2. Результат программы

2. Проработал пример №2: Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4
5  if __name__ == '__main__':
6      # Ввести список одной строкой.
7      a = list(map(int, input().split()))
8      # Если список пуст, завершить программу.
9      if not a:
10         print("Заданный список пуст", file=sys.stderr)
11         exit(1)
12
13     # Определить индексы минимального и максимального элементов.
14     a_min = a_max = a[0]
15     i_min = i_max = 0
16     for i, item in enumerate(a):
17         if item < a_min:
18             i_min, a_min = i, item
19         if item >= a_max:
20             i_max, a_max = i, item
21
22     # Проверить индексы и обменять их местами.
23     if i_min > i_max:
24         i_min, i_max = i_max, i_min
25
26     # Посчитать количество положительных элементов.
27     count = 0
28     for item in a[i_min + 1:i_max]:
29         if item > 0:
30             count += 1
31
32     print(count)

```

Рисунок 3. Программа примера 2

```

/usr/bin/python3.11 /home/code_raider/git/Python_LW7/Examples/example2.py
56 6432 764 7 -34 52 5 -75 214 -54 878 90
0
0
0
2
0
0
0
2
2
2
2
2
2
Process finished with exit code 0

```

Рисунок 4. Результат программы

3. Выполнил индивидуальное задание №1: Составить программу, выдающую индексы заданного элемента или сообщаящую, что такого элемента в списке нет

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  elements = list(map(int, input('Введите ряд из чисел, через пробел: ').split()))
5  indexer = int(input('Введите число, ищущее индексы элементы списка: '))
6
7  if indexer in elements:
8      index = elements.index(indexer)+1
9      print('Индекс заданного элемента:', index)
10 else:
11     print('Заданного элемента нет в списке')

```

Рисунок 5. Программа индивидуального задания 1

```

/usr/bin/python3.11 /home/code_raider/git/Python_LW7/Individual/individ1.py
Введите ряд из чисел, через пробел: 1 2 3 4 5
Введите число, ищущее индексы элементы списка: 3
Индекс заданного элемента: 3
Process finished with exit code 0

```

Рисунок 6. Результат программы

4. Выполнил индивидуальное задание №2: В списке, состоящем из целых элементов, вычислить:

- 1) максимальный по модулю элемент списка
- 2) сумму модулей этого списка, расположенных после первого элемента, равного нулю

Преобразовать список таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине — элементы, стоявшие в нечетных позициях

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  list = list(map(int, input('Введите целые элементы через пробел: ').split()))
5  even = []
6  odd = []
7  count = 0
8  sum = 0
9  min = 0
10 minnum = 0
11
12 for i in range(len(list)):
13     k = list[i]
14     # Запись четных и нечетных элементов
15     if (list.index(k) + 1) % 2 == 0:
16         even.append(list[i])
17     else:
18         odd.append(list[i])
19     # Задача №1
20     if list.index(k) == 0:
21         min = abs(list[i])
22         minnum = list[i]
23     if abs(list[i]) < min:
24         min = abs(list[i])
25         minnum = list[i]
26     # Задача №2
27     if list[i] == 0:
28         count += 1
29     if count != 0:
30         sum += list[i]
31
32 print('')
33 print('Минимальный по модулю элемент списка:', minnum)
34 if sum != 0:
35     print('Сумма элементов после первого нуля:', sum)
36 else:
37     print('В списке нет элементов, равных нулю')
38 print('Реорганизованный список:')
39 print(even + odd)

```

Рисунок 7. Программа индивидуального задания 2

```
/usr/bin/python3.11 /home/code_raider/git/Python_LW7/Individual/individ2.py
Введите целые элементы через пробел: 1 -5 3 6 -2 5

Минимальный по модулю элемент списка: 1
В списке нет элементов, равных нулю
Реорганизованный список:
[-5, 6, 5, 1, 3, -2]

Process finished with exit code 0
```

Рисунок 8. Результат программы

Ответы на вопросы

1. Список — структура данных для хранения объектов различных типов. Списки можно изменять, дополнять данными и даже менять тип данных. Переменные, определяемые как списки, содержат ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры
2. Для создания списка в Python применяется несколько способов: либо через квадратные скобки (`list = [1, 2, 3, 4]`), либо через метод `list()` (`list = list(1, 2, 3, 4)`)
3. При создании списка в памяти резервируется область, в которой хранятся ссылки на другие элементы. Содержимое этой области (или так называемого контейнера) можно изменять в отличие от чисел или строк
4. Для этого можно использовать цикл `for` с атрибутом `range(len(list))` или с помощью `enumerate()`
5. Для списков существуют операции объединения («+») и повторения при помощи оператора умножения («*»)
6. Для определения элемента в списке можно для заданного элемента подставить метод `in` к данному списку
7. Чтобы определить число вхождений элемента в список можно сделать цикл с счетчиком `count`, считающий эти вхождения

8. Для добавления элемента в список можно применить метод `append()`, а сразу для нескольких — `extend()`

9. Чтобы выполнить сортировку списка нужно использовать метод `sort()`. Существует несколько способов сортировки. Как пример, `sort(reverse=True)` сортирует все переменные в порядке убывания

10. Для удаления элементов списка используют несколько методов — `pop()` и `remove()`. Если требуется удалить несколько переменных, можно воспользоваться методом `del`. Пример — `del list[1:3]`

11. Списковое включение — один из способов построения списков. Он имеет формат `a = [i for i in range(7)]`, где `range(7)` — размер создаваемого списка (от 0 до 6), а элемент `i` — элемент, который включается в данный список

12. С помощью срезов доступ к элементам списков осуществляется так:

- `list[:]` - копия списка
- `list[0:n]` — первые `n` элементы
- `list[n:m]` — получение элементов от `n` до `m`
- `list[:n]` — берутся элементы списка с шагом `n`
- `list[n:m:s]` — то же, что и 3 вариант, но также присутствует шаг `s`

13. Существуют такие функции агрегации как получение числа элементов `len()`, получение минимального элемента `min()`, получение максимального элемента `max()` и получение суммы элементов списка `sum()`

14. Чтобы копировать список можно либо использовать метод `copy()`, либо использовать срез `a = b[:]`

15. Функция `sorted()` возвращает новый отсортированный список, оставляя исходный список неизменным. Она принимает список в качестве аргумента и возвращает новый список, содержащий отсортированные элементы. Основное отличие между `sort()` и `sorted()` заключается в том, что `sorted()` возвращает новый отсортированный список, оставляя исходный список неизменным, в то время как `sort()` изменяет текущий список, сортируя элементы на месте

Вывод: В ходе выполнения работы были изучены списки, методы и функции работы с ними и получены навыки управления ими с помощью языка программирования Python версии 3.x