

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа со списками в языке Python»

ОТЧЕТ
по лабораторной работе №7
дисциплины
«Основы программной инженерии»

Выполнил:

Мизин Глеб Егорович

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.

Проработка примеров из лабораторной работы:

```
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     # Проверить количество элементов списка.
9     if len(A) != 10:
10         print("Неверный размер списка", 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
18     print(s)
```

if __name__ == '__main__'

Run: Ind × ex1 ×

F:\GitLab\PyLab-2.5\venv\Scripts\python.exe F:\GitLab\PyLab-2.5\ex1.py

1 5 6 78 9 4 -2 -3 3 3

6

Рисунок 1 – Пример №1

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4
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)
```

if __name__ == '__main__' > if len(A) != 10

Run: Ind × ex2 ×

F:\GitLab\PyLab-2.5\venv\Scripts\python.exe F:\GitLab\PyLab-2.5\ex2.py

1 5 6 78 9 4 -2 -3 3 3

6

Рисунок 2 – Пример №1 с использованием List Comprehensions

```
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    # Если список пуст, завершить программу.
11    if not a:
12        print("Заданный список пуст", file=sys.stderr)
13        exit(1)
14
15    # Определить индексы минимального и максимального элементов.
16    a_min = a_max = a[0]
17    i_min = i_max = 0
18    for i, item in enumerate(a):
19        if item < a_min:
20            i_min, a_min = i, item
21        if item >= a_max:
22            i_max, a_max = i, item
23
24    # Проверить индексы и обменять их местами.
25    if i_min > i_max:
26        i_min, i_max = i_max, i_min
27    # Посчитать количество положительных элементов.
28    count = 0
29    for item in a[i_min + 1:i_max]:
30        if item > 0:
31            count += 1
32    print(count)
33
34 if __name__ == '__main__' > for i, item in enumerate(a) > if item >= a_max
```

Run: Ind × ex2 ×

▶ F:\GitLab\PyLab-2.5\venv\Scripts\python.exe F:\GitLab\PyLab-2.5\ex2.py

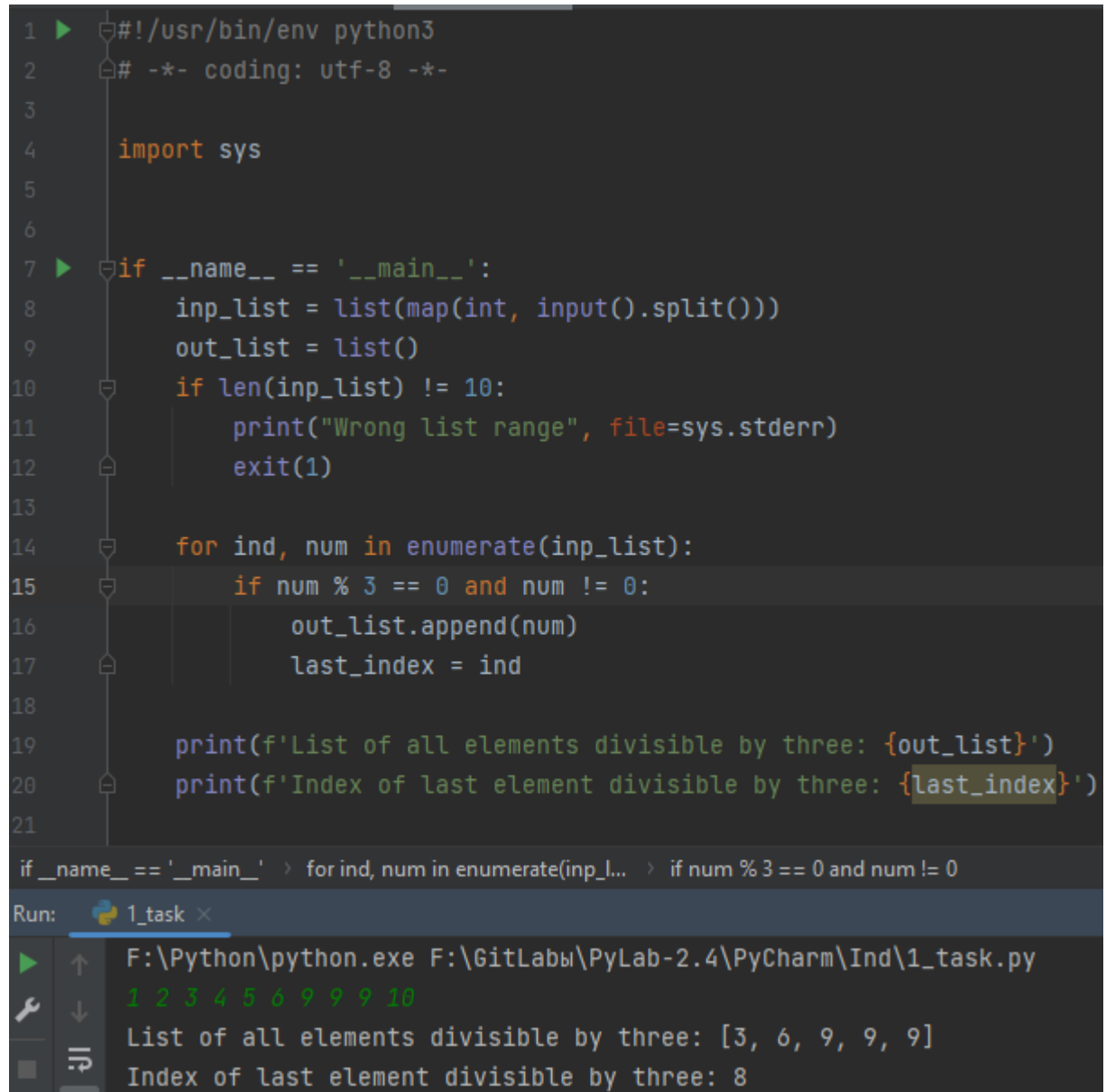
1 5 4 8 7 -4 -5 1 2 3 4 5 789 12 3 4 5

5

Рисунок 3 – Пример №2

Индивидуальное задание:

Задание №1: Ввести список А из 10 элементов. Определить количество элементов, кратных 3 и индекс последнего такого элемента.



```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 ▶ if __name__ == '__main__':
8     inp_list = list(map(int, input().split()))
9     out_list = list()
10    if len(inp_list) != 10:
11        print("Wrong list range", file=sys.stderr)
12        exit(1)
13
14    for ind, num in enumerate(inp_list):
15        if num % 3 == 0 and num != 0:
16            out_list.append(num)
17            last_index = ind
18
19    print(f'List of all elements divisible by three: {out_list}')
20    print(f'Index of last element divisible by three: {last_index}')
21
```

Run: 1_task ×

F:\Python\python.exe F:\GitLab\PyLab-2.4\PyCharm\Ind\1_task.py

1 2 3 4 5 6 9 9 9 10

List of all elements divisible by three: [3, 6, 9, 9, 9]

Index of last element divisible by three: 8

Рисунок 4 – Индивидуальное задание №1

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 ▶ if __name__ == '__main__':
8     il = list(map(int, input().split()))
9     if len(il) != 10:
10         print("Wrong list range", file=sys.stderr)
11         exit(1)
12
13     ol = [num for ind, num in enumerate(il) if num % 3 == 0 and num != 0]
14     il = [ind for ind, num in enumerate(il) if num % 3 == 0 and num != 0]
15
16     print(f'List of all elements divisible by three: {ol}')
17     print(f'Index of last element divisible by three: {il[-1]}')
18
```

if __name__ == '__main__' > if len(il) != 10

Run: 1_task(List Comprehensions) ×

F:\Python\python.exe "F:\GitLab\PyLab-2.4\PyCharm\Ind\1_task(List Comprehensions).py"

1 2 3 4 5 6 9 9 9 10

List of all elements divisible by three: [3, 6, 9, 9, 9]

Index of last element divisible by three: 8

Рисунок 5 – Индивидуальное задание №1 с использованием List Comprehensions

Задание №2: в списке, состоящем из вещественных элементов, вычислить:

1. произведение положительных элементов списка;
2. сумму элементов списка, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     inp_list = list(map(float, input().split()))
6     final_out = list()
7     prod = 1
8     Sum = 0
9
10    # Multiply all non-negative elements
11    for item in inp_list:
12        if item > 0:
13            prod *= item
14
15    # Sum all the elements until we meet the minimum
16    for item in inp_list:
17        if item > min(inp_list):
18            Sum += item
19        else:
20            break
21
22    # Split the initial list into two, with even and odd elements
23    f_out = inp_list[::2]
24    s_out = inp_list[1::2]
25
```

Рисунок 6 – Индивидуальное задание №2

```
26     # Sort new lists
27     f_out.sort()
28     s_out.sort()
29
30     # Create a new list by inserting elements one by one from the sorted lists
31     for i, v in enumerate(f_out):
32         if i == len(s_out):
33             final_out.append(f_out[i])
34             break
35         else:
36             final_out.append(f_out[i])
37             final_out.append(s_out[i])
38
39     # Output
40     print(f'Product of non-negative numbers: {prod}')
41     print(f'The sum of the elements before meeting the minimum: {Sum}')
42     print(f'List sorted according to task condition: {final_out}')
43
44 if __name__ == '__main__':
45     for item in inp_list:
46         else
```

Run: 2_taks ×

F:\Python\python.exe F:\GitLab\PyLab-2.4\PyCharm\Ind\2_taks.py

-10.4 78.1 77.6 -11 -0.45 22 64 780

Product of non-negative numbers: 6655949414.399999

The sum of the elements before meeting the minimum: 145.29999999999998

List sorted according to task condition: [-10.4, -11.0, -0.45, 22.0, 64.0, 78.1, 77.6, 780.0]

Process finished with exit code 0

Рисунок 7 – Индивидуальное задание №2

Контрольные вопросы

1. Что такое списки в языке Python?

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

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки:

```
my_list = [1, 2, 3, 4, 5]
```

3. Как организовано хранение списков в оперативной памяти?

Как уже было сказано выше, список является изменяемым типом данных. При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличие от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Читать элементы списка можно с помощью следующего цикла:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']  
for elem in my_list:  
    print(elem)
```

5. Какие существуют арифметические операции со списками?

Для объединения списков можно использовать оператор сложения (+):

Список можно повторить с помощью оператора умножения (*):

6. Как проверить есть ли элемент в списке?

Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор 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

```
my_list = [1, 2, 3, 4, 5]
my_list.insert(1, 'привет')
print(my_list)
```

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе pop

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
removed = my_list.pop(2)
print(my_list)
print(removed)
```

Результат:

```
['один', 'два', 'четыре', 'пять']
три
```

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. Проще всего работу list comprehensions показать на примере. Допустим вам

необходимо создать список целых чисел от 0 до n, где n предварительно задается. Классический способ решения данной задачи выглядел бы так:

```
>>> n = int(input())
7
>>> a=[]
>>> for i in range(n):
        a.append(i)

>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

Использование *list comprehensions* позволяет сделать это значительно проще:

```
>>> n = int(input())
7
>>> a = [i for i in range(n)]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

или вообще вот так, в случае если вам не нужно больше использовать *n*:

```
>>> a = [i for i in range(int(input()))]
7
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

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. При этом необходимо помнить, что выборка не включает элемент определяемый stop.

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции:

- `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?

Поэтому для создания копии списка необходимо использовать либо метод `copy`, либо использовать оператор среза.

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

Функция `sorted()` в Python, выполняет сортировку. Выполняет сортировку последовательности по возрастанию/убыванию.

Метод `sort()` работает только со списками и сортирует уже имеющийся список. Данный метод ничего не возвращает. А метод `sorted()` работает с любыми итерируемыми объектами и возвращает новый отсортированный список. В качестве итерируемых объектов могут выступать списки, строки, кортежи и другие.