

**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 №7 по дисциплине:
основы программной инженерии**

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2021 г.

Ход работы:

1. Проход (итерация) по списку

```
my_list = [1, 2, 3, 4, 5]
for i in range(len(my_list)):
    my_list[i] += 5
    print(my_list)
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py

[6, 2, 3, 4, 5]
[6, 7, 3, 4, 5]
[6, 7, 8, 4, 5]
[6, 7, 8, 9, 5]
[6, 7, 8, 9, 10]

2. Вложенный список

```
my_list = ['один', 10, 2.25, [5, 15], 'пять']
print(len(my_list))
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py

5

3. Использование функции enumerate

```
a = [10, 20, 30, 40]
for i, item in enumerate(a):
    print(f"({i}, {item})")
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/

(0, 10)
(1, 20)
(2, 30)
(3, 40)

4. Арифметические операции со списками

```
list_1 = [1, 2, 3]
list_2 = [4, 5, 6]
print(list_1 + list_2)
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОП

[1, 2, 3, 4, 5, 6]

5. Повторение списка с помощью оператора умножения (*)

```
list_1 = [1, 2, 3]
print(list_1 * 2)
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7
[1, 2, 3, 1, 2, 3]

6. Поиск элемента в списке Python

```
lst = [3, 5, 2, 4, 1]
if 3 in lst:
    print("Список содержит число 3")
else:
    print("Список не содержит число 3")
|
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7
Список содержит число 3

```
lst = [3, 5, 2, 4, 1]
if 0 not in lst:
    print("Список не содержит нулей")
```

if 0 not in lst

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ
Список не содержит нулей

7. Индекс элемента в списке

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

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul
1

8. Число вхождений элемента в список

```
lst = [1, 2, 2, 3, 3]
print(lst.count(2))
|
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/0

2

9. Изменение списка Python

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
elem = my_list[-1]
print(elem)
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.

пять

10. Вставить элемент в список

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

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
[1, 'Привет', 2, 3, 4, 5]

11. Добавить элемент в список

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list.append('ещё один')
print(my_list)
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
['один', 'два', 'три', 'четыре', 'пять', 'ещё один']

```
my_list = ['cde', 'fgh', 'abc', 'klm', 'opq']
list_2 = [3, 5, 2, 4, 1]
my_list.sort()
list_2.sort()
print(my_list)
print(list_2)
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
['abc', 'cde', 'fgh', 'klm', 'opq']
[1, 2, 3, 4, 5]
```

12. Перевернуть список

```
my_list = [1, 2, 3, 4, 5]
my_list.reverse()
print(my_list)
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
[5, 4, 3, 2, 1]
```


13. Удалить элемент из списка

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

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py  
['один', 'два', 'четыре', 'пять']  
три
```

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

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py  
['один', 'два', 'три', 'четыре']  
пять
```

14. Применение оператора del

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

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py  
['один', 'два', 'четыре', 'пять']
```

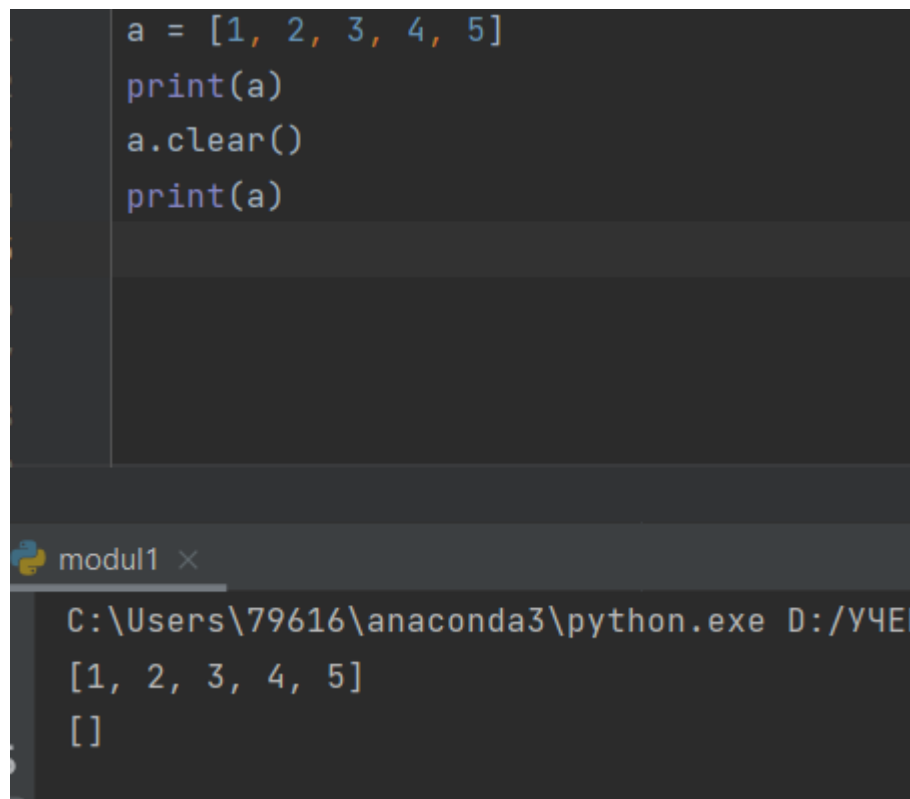
```
my_list = ['один', 'два', 'три', 'четыре', 'пять']  
del my_list[1:3]  
print(my_list)
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py  
['один', 'четыре', 'пять']
```

15. Удалить все элементы из списка

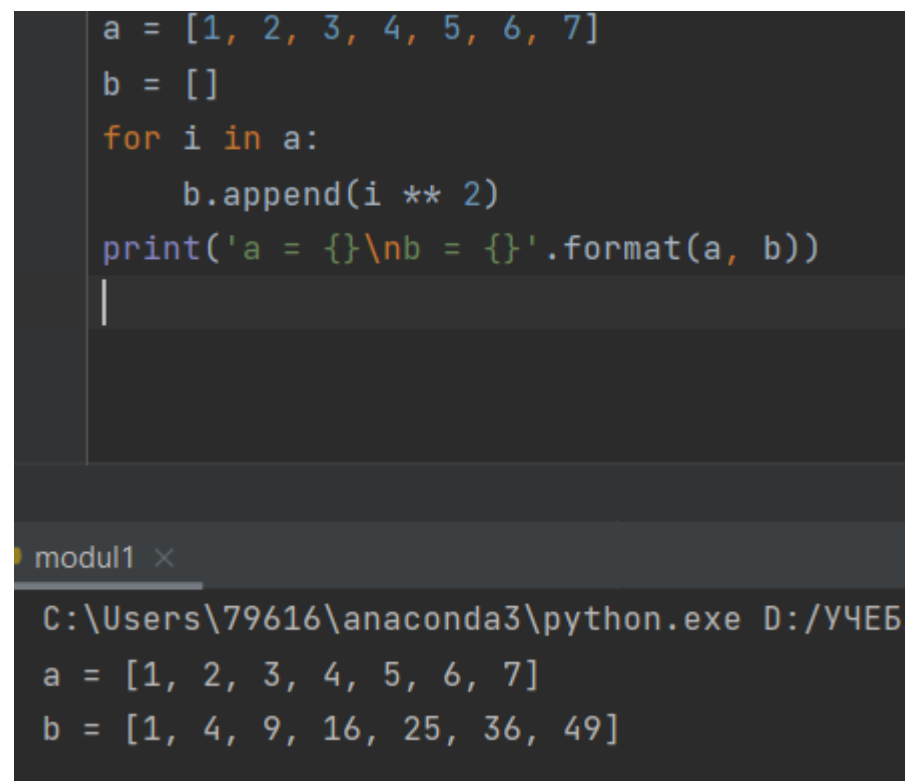
```
a = [1, 2, 3, 4, 5]
print(a)
a.clear()
print(a)
```



The screenshot shows a Python IDE with a file named 'modul1'. The code in the editor defines a list 'a' with elements [1, 2, 3, 4, 5], prints it, calls the 'clear()' method, and prints it again. The output window shows the list [1, 2, 3, 4, 5] followed by an empty list [], confirming that all elements have been removed.

16. List Comprehensions как обработчик списков

```
a = [1, 2, 3, 4, 5, 6, 7]
b = []
for i in a:
    b.append(i ** 2)
print('a = {}\nb = {}'.format(a, b))
```



The screenshot shows a Python IDE with a file named 'modul1'. The code defines a list 'a' with elements [1, 2, 3, 4, 5, 6, 7], initializes an empty list 'b', and uses a for loop to append the square of each element in 'a' to 'b'. The output window shows the original list 'a' and the new list 'b' containing the squares [1, 4, 9, 16, 25, 36, 49].

```
a = [1, 2, 3, 4, 5, 6, 7]
b = list(map(lambda x: x**2, a))
print('a = {}\nb = {}'.format(a, b))
```

modul1 x

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

17. Построение нового списка, состоящего только из четных чисел

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 b = []
3 for i in a:
4     if i%2 == 0:
5         b.append(i)
6 print('a = {}\nb = {}'.format(a, b))
7
8
```

modul1 x

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

```
a = [1, 2, 3, 4, 5, 6, 7]
b = list(filter(lambda x: x % 2 == 0, a))
print('a = {}\nb = {}'.format(a, b))
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

```
a = [1, 2, 3, 4, 5, 6, 7]
b = [i for i in a if i % 2 == 0]
print('a = {}\nb = {}'.format(a, b))
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

18. Функции агрегации

```
my_list = [5, 3, 2, 4, 1]
print(len(my_list))
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/YЧЕ
5

```
my_list = [5, 3, 2, 4, 1]
print(min(my_list))
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/YЧЕ
1

```
my_list = [5, 3, 2, 4, 1]
print(max(my_list))
```

|

modul1 x

C:\Users\79616\anaconda3\python.exe D:/Y4B
5

```
my_list = [5, 3, 2, 4, 1]
print(sum(my_list))
```

|

modul1 x

C:\Users\79616\anaconda3\python.exe D:/Y4B
15

19. Сравнение списков

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
list_2 = ['три', 'один', 'пять', 'два', 'четыре']
if (my_list == list_2):
    print('совпадают')
else:
    print('не совпадают')
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
не совпадают

20. Списки и строки

```
my_str = 'Monty Python'
my_list = list(my_str)
print(my_list)
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py
['M', 'o', 'n', 't', 'y', ' ', 'P', 'y', 't', 'h', 'o', 'n']


```
my_str = 'Monty Python'  
my_list = my_str.split()  
print(my_list)
```

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/Y4E5/  
['Monty', 'Python']
```

```
my_str = 'Monty Python'  
my_list = my_str.split('-')  
print(my_list)
```

|

modul1 ×

```
C:\Users\79616\anaconda3\python.exe D:/Y4E5A/0  
['Monty Python']
```

21.Метод join

```
my_list = ['Monty', 'Python']
delimiter = ' '
output = delimiter.join(my_list)
print(output)
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОП/7/1
Monty Python

22.Псевдонимы

```
my_list = ['Monty', 'Python']
list_2 = my_list
list_2[1] = 'Java:)'
print(my_list)
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/1
['Monty', 'Java:)']

23. Применение метода copy

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a.copy()
>>> b
[1, 2, 3, 4, 5]
>>> a == b
True
>>> a is b
False
>>> a is not b
True
```

Выполнение 1 примера

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

import sys

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

    s = 0
    for item in A:
        if abs(item) < 5:
            s += item
    print(s)
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1.py

2 5 1 1 7 6 7 8 9 3

7

Выполнение примера 2

```
import sys

if __name__ == '__main__':
    a = list(map(int, input().split()))
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    a_min = a_max = a[0]
    i_min = i_max = 0
    for i, item in enumerate(a):
        if item < a_min:
            i_min, a_min = i, item
        if item >= a_max:
            i_max, a_max = i, item

    if i_min > i_max:
        i_min, i_max = i_max, i_min

    count = 0
    for item in a[i_min + 1:i_max]:
        if item > 0:
            count += 1

if __name__ == '__main__': > for i, item in enumerate(a) > if item >= a_max
```

modul2 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul2.py

2 5 1 1 7 6 7 8 9 3

5

Выполнение индивидуального задания. Вариант 13

1. Ввести список A из 10 элементов, найти сумму элементов, меньших по модулю 3 и кратных 9, их количество и вывести результаты на экран.

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

import sys

k = 0
if __name__ == '__main__':
    A = list(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    s = 0
    for item in A:
        if abs(item) < 3:
            k += 1
            s += item

    print(k)
    print(s)
```

```
C:\Users\79616\anaconda3\python.exe
3 4 5 6 7 8 7 2 2 1
3
5
```

2. В списке, состоящем из вещественных элементов, вычислить:

1) количество элементов списка, равных 0;

2) сумму элементов списка, расположенных после минимального элемента.

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

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

```
import sys
```

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

```
    a = list(map(float, input('Введите элементы массива: ').split()))
```

```
    s = 0
```

```
    sum = 0
```

```
    e = 0
```

```
    for item in a:
```

```
        if item == 0:
```

```
            s += 1
```

```
    print(f"Количество элементов равных нулю: {s}")
```

```
    a_min = 1000.0
```

```
    for i, item in enumerate(a):
```

```
        if item < a_min:
```

```
            a_min = item
```

```
            e = i
```

```
    for i, item in enumerate(a):
```

```
        if i > e:
```

```
            sum += item
```

```
    print(f"Сумма равна {sum}")
```

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/
```

```
Введите элементы массива: 1 2 3 0 4 5 0 2
```

```
Количество элементов равных нулю: 2
```

```
Сумма равна 11.0
```

Вопросы для защиты работы:

1. Что такое списки в языке Python? Список (list) - структура данных для хранения объектов различных типов.
2. Как осуществляется создание списка в Python? Для создания списка нужно заключить элементы в квадратные скобки.
3. Как организовано хранение списков в оперативной памяти? 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? Элементы можно перебрать с помощью цикла for.
5. Какие существуют арифметические операции со списками? Списки можно складывать и умножать.
6. Как проверить есть ли элемент в списке? Проверить есть ли элемент в списке можно с помощью цикла if.
7. Как определить число вхождений заданного элемента в списке? Чтобы определить число вхождений заданного элемента в списке, нужно воспользоваться методом count().
8. Как осуществляется добавление (вставка) элемента из списка? Метод insert можно использовать, чтобы вставить элемент в список. Метод append можно использовать для добавления элемента в список.
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. Элемент можно удалить с помощью метода remove. Оператор del можно также использовать для удаления элемента. Можно удалить все элементы из списка 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. В языке Python есть две очень мощные функции для работы с коллекциями: `map` и `filter`. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как `list`, `tuple`, `set`, `dict` и т.п. Списковое включение позволяет обойтись без этих функций.

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

Созданный список:

```
>>> a = [i for i in range(10)]
```

Доступ к его элементам:

```
>>> # Получить копию списка
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> # Получить первые пять элементов списка
>>> a[0:5]
[0, 1, 2, 3, 4]

>>> # Получить элементы с 3-го по 7-ой
>>> a[2:7]
[2, 3, 4, 5, 6]

>>> # Взять из списка элементы с шагом 2
>>> a[::2]
[0, 2, 4, 6, 8]

>>> # Взять из списка элементы со 2-го по 8-ой с шагом 2
>>> a[1:8:2]
[1, 3, 5, 7]
```

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.

14. Как создать копию списка?

1) Создать псевдоним. Создать новый список и присвоить значения имеющегося.

2) Создать копию, используя метод `copy`.

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

Функция `sorted` возвращает новый отсортированный список, который получен из итерируемого объекта, который был передан как аргумент. Функция также поддерживает дополнительные параметры, которые позволяют управлять сортировкой.

```
In [1]: list_of_words = ['one', 'two', 'list', '', 'dict']  
  
In [2]: sorted(list_of_words)  
Out[2]: ['', 'dict', 'list', 'one', 'two']
```

