

Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»
Институт цифрового развития

ОТЧЕТ ПО ДИСЦИПЛИНЕ
ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Лабораторная работа №7
Работа со списками в языке Python

Выполнила:

Мирзаева Камилла Мирзаевна
2 курс, группа ПИЖ-б-о-20-1

Принял:

Воронкин Роман Александрович

Ставрополь, 2021 г.

Ход работы:

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

modul1 ×

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

[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]

Рисунок 1 - Проход (итерация) по списку

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

modul1 ×

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

5

Рисунок 2 - Вложенный список

```
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)

Рисунок 3 - Использование функции enumerate

```
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]

Рисунок 4 - Арифметические операции со списками

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

modul1 ×

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

Рисунок 5 - Повторение списка с помощью оператора умножения (*)

```
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/ОП/7
Список содержит число 3

Рисунок 6 - Поиск элемента в списке Python

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

if 0 not in lst

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1
Список не содержит нулей

Рисунок 7 - Поиск элемента в списке Python

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

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/7/modul1
1

Рисунок 8 - Индекс элемента в списке

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

modul1 ×

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

2

Рисунок 9 - Число вхождений элемента в список

```
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

Рисунок 10 - Изменение списка Python

```
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]

Рисунок 11 - Вставить 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']

Рисунок 12 - Добавить элемент в список

```
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 x

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

Рисунок 13 - Добавить элемент в список

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

modul1 x

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

Рисунок 14 - Перевернуть список


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

modul1 x

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

Рисунок 15 - Удалить элемент из списка

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

modul1 x

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

Рисунок 16 - Удалить элемент из списка

```
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']

Рисунок 17 - Применение оператора del

```
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']

Рисунок 18 - Применение оператора del

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

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБ

[1, 2, 3, 4, 5]

[]

Рисунок 19 - Удалить все элементы из списка

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

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБ

a = [1, 2, 3, 4, 5, 6, 7]

b = [1, 4, 9, 16, 25, 36, 49]

Рисунок 20 - List Comprehensions как обработчик списков

```
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]

Рисунок 21 - List Comprehensions как обработчик списков

```
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]

Рисунок 22 - Построение нового списка, состоящего только из четных чисел

```
a = [1, 2, 3, 4, 5, 6, 7]
b = list(filter(lambda x: x % 2 == 0, 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 = [2, 4, 6]

Рисунок 23 - Построение нового списка, состоящего только из четных чисел

```
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 x

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

a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]

Рисунок 24 - Построение нового списка, состоящего только из четных чисел

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

modul1 x

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

Рисунок 25 - Функции агрегации

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

modul1 x

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

Рисунок 26 - Функции агрегации

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

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБНИК/Python/27.py

5

Рисунок 27 - Функции агрегации

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

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБНИК/Python/28.py

15

Рисунок 28 - Функции агрегации

```
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

Рисунок 29 - Сравнение списков

```
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']

Рисунок 30 - Списки и строки


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

modul1 x

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

Рисунок 31 - Списки и строки

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

modul1 x

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

Рисунок 32 - Списки и строки

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

modul1 x

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

Рисунок 33 - Метод join

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

modul1 x

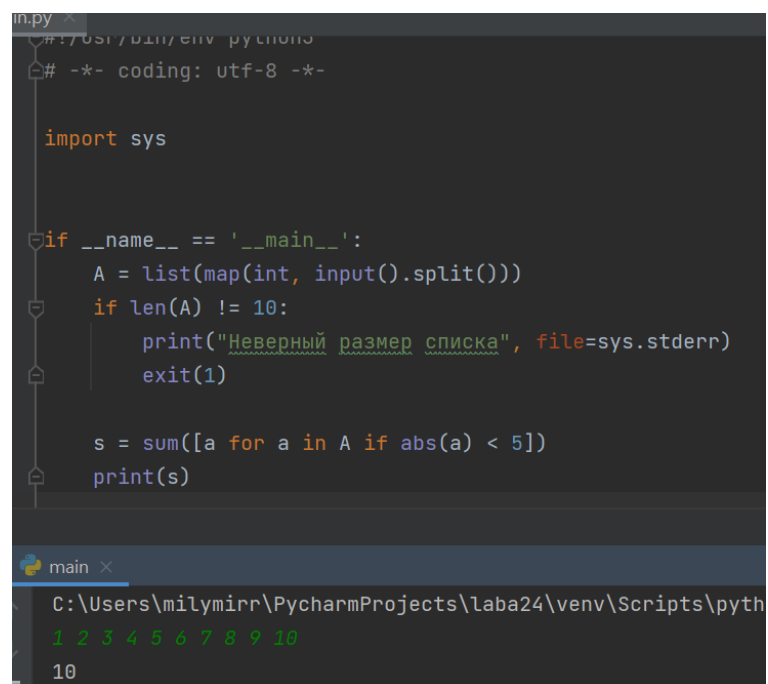
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОП/7/т
['Monty', 'Java:)']

Рисунок 34 - Псевдонимы

```
>>> 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
```

Рисунок 35 - Применение метода copy

Пример №1



```
in.py x
C:\Users\milymirr\PycharmProjects\laba24\venv\Scripts\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 = sum([a for a in A if abs(a) < 5])
    print(s)
```

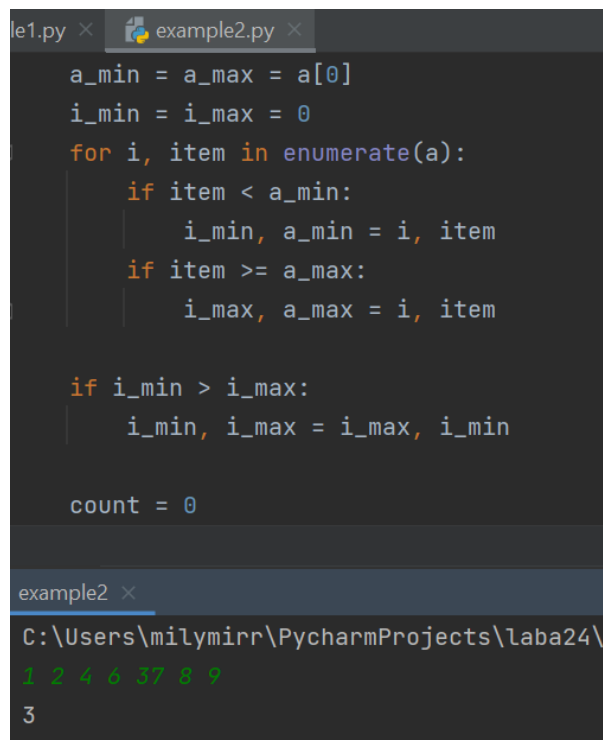
main x

C:\Users\milymirr\PycharmProjects\laba24\venv\Scripts\python3

1 2 3 4 5 6 7 8 9 10

10

Пример №2



```
le1.py x example2.py x
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

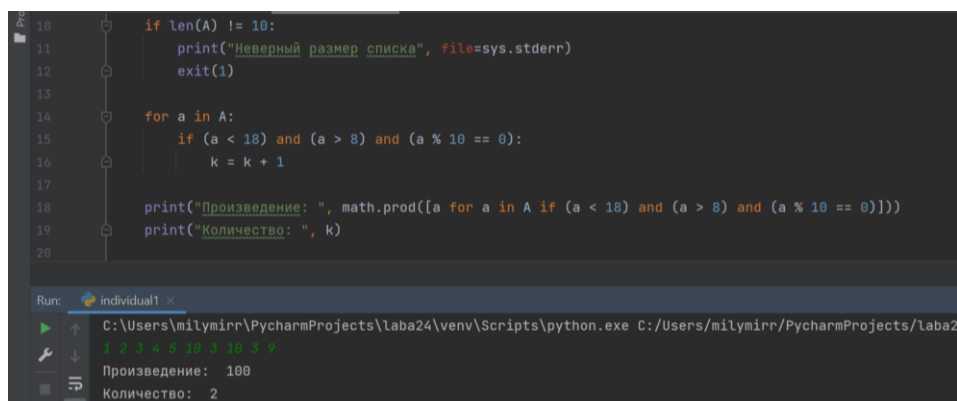
count = 0

example2 x
C:\Users\milymirr\PycharmProjects\laba24\
1 2 4 6 37 8 9
3
```

Индивидуальное задание №1.

Вариант 15

Ввести список А из 10 элементов, найти произведение элементов, больших 8 и меньших 18 и кратных 10, их количество и вывести результаты на экран.



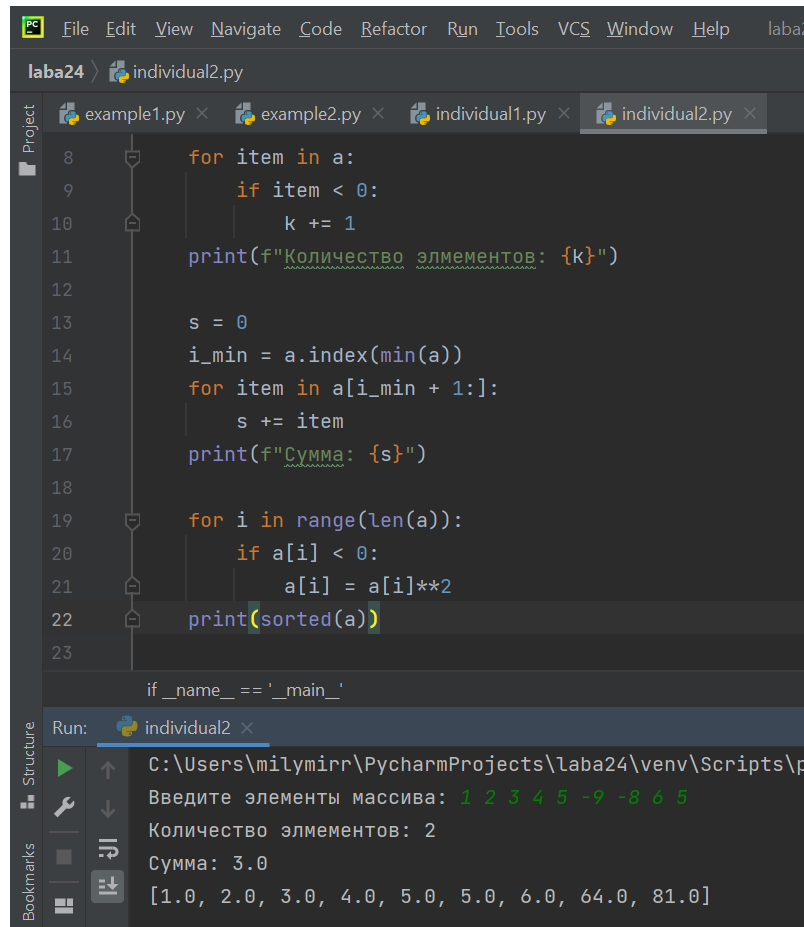
```
10 if len(A) != 10:
11     print("Неверный размер списка", file=sys.stderr)
12     exit(1)
13
14 for a in A:
15     if (a < 18) and (a > 8) and (a % 10 == 0):
16         k = k + 1
17
18 print("Произведение: ", math.prod([a for a in A if (a < 18) and (a > 8) and (a % 10 == 0)]))
19 print("Количество: ", k)
20
Run: individual1 x
C:\Users\milymirr\PycharmProjects\laba24\env\Scripts\python.exe C:/Users/milymirr/PycharmProjects/laba2
1 2 3 4 5 10 1 10 1 9
Произведение: 100
Количество: 2
```

Индивидуальное задание №2.

В списке, состоящем из вещественных элементов, вычислить:

1. количество отрицательных элементов списка;

2. сумму модулей элементов списка, расположенных после минимального по модулю элемента. Заменить все отрицательные элементы списка их квадратами и упорядочить элементы списка по возрастанию.



The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `individual2.py` with the following code:

```
8     for item in a:
9         if item < 0:
10             k += 1
11     print(f"Количество элементов: {k}")
12
13     s = 0
14     i_min = a.index(min(a))
15     for item in a[i_min + 1:]:
16         s += item
17     print(f"Сумма: {s}")
18
19     for i in range(len(a)):
20         if a[i] < 0:
21             a[i] = a[i]**2
22     print(sorted(a))
23
24     if __name__ == '__main__':
```

The Run console at the bottom shows the execution of the script. The output is as follows:

```
Run: individual2
C:\Users\milymirr\PycharmProjects\laba24\venv\Scripts\python.exe
Введите элементы массива: 1 2 3 4 5 -9 -8 6 5
Количество элементов: 2
Сумма: 3.0
[1.0, 2.0, 3.0, 4.0, 5.0, 5.0, 6.0, 64.0, 81.0]
```

Вопросы для защиты работы:

Что такое списки в языке Python? Список (list) - структура данных для хранения объектов различных типов.

Как осуществляется создание списка в Python? Для создания списка нужно заключить элементы в квадратные скобки.

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.

Каким образом можно перебрать все элементы списка? Элементы можно перебрать с помощью цикла for.

Какие существуют арифметические операции со списками? Списки можно складывать и умножать.

Как проверить есть ли элемент в списке? Проверить есть ли элемент в списке можно с помощью цикла if.

Как определить число вхождений заданного элемента в списке? Чтобы определить число вхождений заданного элемента в списке, нужно воспользоваться методом count().

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.

Как выполнить сортировку списка? Для сортировки списка нужно использовать метод sort.

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.

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. 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?

Создать псевдоним. Создать новый список и присвоить значения имеющегося.

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']
```