

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

ЛЕКЦИЯ №3

Функции в языке программирования Python (структурное программирование).

СОСТАВИТЕЛЬ: КАНД. ТЕХН. НАУК БЫКАДОР В.С.

Для каких целей нужны функции

- 1) Минимизация кода за счёт исключения его избыточности, то есть если имеется повторяющийся программный код, то можно его выделить в отдельную функцию и далее вызывать данную функцию там, где это необходимо и столько раз сколько необходимо.
- 2) Процедурная декомпозиция, которая позволяет разделить сложную программу на составные части, отдельно написать код для каждой части и затем скомпоновать из функций целую программу (одни и те же функции можно использовать в разных программах).
- 3) Функции лежат в основе методов классов, которые являются основной частью парадигмы объектно-ориентированного программирования.

Общий синтаксис функции на языке Python

def

обязательное ключевое слово, которое обозначает, что код дальше будет относиться к функции.

имя_функции

идентификатор функции, по которому в дальнейшем будет выполняться обращение к этой функции.

[входные_параметры]

входные параметры позволяют передать в функцию данные из вне. В общем случае входные параметры не являются обязательными и могут быть функции без входных параметров.

тело_функции

рабочий код функции, который выполняет определённое преобразование данных.

!СМЕЩЕНИЕ КОДА!

```
def имя_функции([входные_параметры]):
```

```
    тело_функции
```

```
    [return результат_работы_функции]
```

:

Обязательное двоеточие в конце сигнатуры функции, указывающее на начало блока кода функции.

return

ключевое слово **return** используется для того, чтобы вернуть какие-либо данные из функции.

результат_работы_функции

это какая-либо переменная, возвращаемая функцией. Так же, как и входные параметры возвращаемый результат не является обязательным для функции, то есть функция может и не возвращать какие-либо данные.

Базовые примеры работы с функциями

```
def easy_func():  
    pass
```

Это пример простейшей функции, которая ничего не принимает, ничего не возвращает и ничего вообще не делает, но тем не менее эта функция синтаксически правильная и если её вызвать, то программный код будет работать.

```
2 def summa(a, b):  
3     c = a + b  
4     return c  
5  
6 # вызов функции  
7 res = summa(2, 5)  
8  
9 print(f'2 + 5 = {res}')
```

Вызов функции в коде

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  
  
bykvi/AppData/Local/Microsoft/WindowsApps/py  
YandexDisk/B работе/Лекции по Python/tests/m  
2 + 5 = 7  
PS C:\Users\bykvi\YandexDisk\B работе\Лекции
```

Эта функция принимает два параметра – числа *a* и *b*, которые затем суммируются в функции *summa*, которая затем возвращает результат своей работы.

Динамическая типизация и функции

Одна и та же функция возвращает различные результаты в зависимости от типа переданных в функцию параметров.

```
4
5 def summ(a, b):
6     return a + b
7
8
9 a, b = 1, 5
10 c = summ(a, b)
11 print(f'c = {a} + {b} = {c}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
• (venv) vityalybykador@MacBook-Air-Vitaly test % /Us
bykador/PROJECTS/test/for_lectures.py
```

c = 1 + 5 = 6

```
4
5 def summ(a, b):
6     return a + b
7
8
9 a, b = 2+3j, 1-2j
10 c = summ(a, b)
11 print(f'c = {a} + {b} = {c}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
(venv) vityalybykador@MacBook-Air-Vitaly test % /Us
bykador/PROJECTS/test/for_lectures.py
```

c = (2+3j) + (1-2j) = (3+1j)

```
5 ✓ def summ(a, b):
6     return a + b
7
8
9 a, b = 'A', 'Я'
10 c = summ(a, b)
11 print(f'c = {a} + {b} = {c}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

```
(venv) vityalybykador@MacBook-Air-Vitaly test % /Us
bykador/PROJECTS/test/for_lectures.py
```

c = A + Я = АЯ

Динамическая типизация и функции

Одна и та же функция возвращает различные результаты в зависимости от типа переданных в функцию параметров.

```
4
5 def summ(a, b):
6     return a + b
7
8
9 a, b = [1, 2, 'c'], [3, 'b', 3.91]
10 c = summ(a, b)
11 print(f'c = {a} + {b} = {c}')
12
13
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ

```
• (venv) vityalybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/for_lectures.py
```

```
c = [1, 2, 'c'] + [3, 'b', 3.91] = [1, 2, 'c', 3, 'b', 3.91]
```

```
5
4
5 def summ(a, b):
6     return a + b
7
8
9 a, b = 3, 'A'
10 c = summ(a, b)
11 print(f'c = {a} + {b} = {c}')
12
13
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

```
⊗ (venv) vityalybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv
bykador/PROJECTS/test/for_lectures.py
```

```
Traceback (most recent call last):
```

```
File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 10, in <module>
```

```
    c = summ(a, b)
```

```
File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 6, in summ
```

```
    return a + b
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Динамическая типизация и функции

Возвращать из функции можно тоже различные объекты языка программирования Python.

```
8  def my_filter(lst):
9      lst_digits, lst_strs, lst_lists, lst_others = [], [], [], []
10     for item in lst:
11         match item:
12             case int() | float() | complex():
13                 lst_digits.append(item)
14             case str():
15                 lst_strs.append(item)
16             case list():
17                 lst_lists.append(item)
18             case _:
19                 lst_others.append(item)
20     return [lst_digits, lst_strs, lst_lists, lst_others]
```

```
[1, 2.3, (2+3j)]
```

```
['b', 'привет']
```

```
[[1, 'a'], ['list', 2, 4]]
```

```
[None, datetime.date(2022, 9, 17)]
```

```
23  l = [1, [1, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 2, 4], date(2022, 9, 17)]
24  results = my_filter(l)
```

Ссылочные переменные

Функция, которая не возвращает данные.

```
30 def my_print(a, b):
31     print(a)
32     print(b)
33     print('')
34
35     a = 'привет'
36     b = 'пока'
37
38     my_print(a, b)
39
40     b = a
41
42     my_print(a, b)
43
44     a = 'что?'
45
46     my_print(a, b)
```

КОПИРОВАНИЕ

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КО

привет
пока

привет
привет

что?
привет

```
30 def my_print(a, b):
31     print(a)
32     print(b)
33     print('')
34
35     a = [1, 2]
36     b = ['a', 'b']
37
38     my_print(a, b)
39
40     b = a
41
42     my_print(a, b)
43
44     a = [2+1j, 3.14]
45
46     my_print(a, b)
```

КОПИРОВАНИЕ

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КО

[1, 2]
['a', 'b']

[1, 2]
[1, 2]

[(2+1j), 3.14]
[1, 2]

```
30 def my_print(a, b):
31     print(a)
32     print(b)
33     print('')
34
35     a = [1, 2]
36     b = ['a', 'b']
37
38     my_print(a, b)
39
40     b = a
41
42     my_print(a, b)
43
44     a[0] = 2+2j
45
46     my_print(a, b)
```

ССЫЛКА

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КО

[1, 2]
['a', 'b']

[1, 2]
[1, 2]

[(2+2j), 2]
[(2+2j), 2]

Ссылочные переменные как параметр функции

```
29
30 def test_list(a, b):
31     a += 1
32     b -= 10
33     res = a + b
34     return res
35
36 a, b = 2, 100
37
38 c = test_list(a, b)
39
40 print(f'a = {a}, b = {b}')
41
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАД

- (venv) vitalitybykador@MacBook-Air-Vitaly test %
bykador/PROJECTS/test/for_lectures.py

a = 2, b = 100

- (venv) vitalitybykador@MacBook-Air-Vitaly test

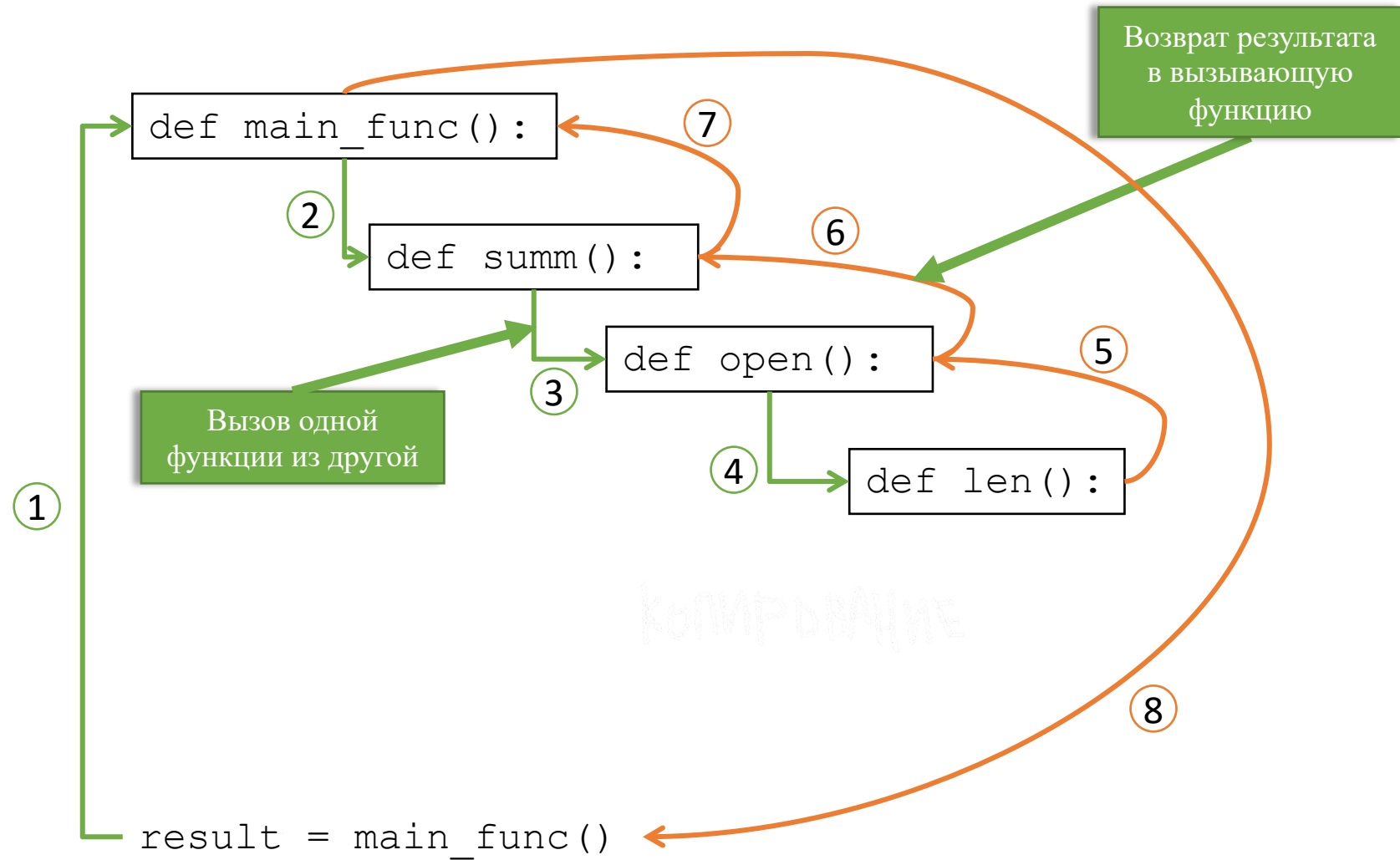
```
29
30 def test_list(lst):
31     local_lst = lst
32     local_lst[0] = 'поменял'
33     return local_lst
34
35 l = [1, 2, 10]
36
37 c = test_list(l)
38
39 print(f'l = {l}')
40
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАД

- (venv) vitalitybykador@MacBook-Air-Vitaly test %
bykador/PROJECTS/test/for_lectures.py

l = ['поменял', 2, 10]

Вложенный вызов функций



Вложенный вызов функций

```
25
26 def summ(lst):
27     s = 0
28     for item in lst:
29         if isinstance(item, int) or isinstance(item, float):
30             s += item
31     return s
32
33 def transformation(lst):
34     s = summ(lst)
35     return (s - s / 2)**2
36
37 l = [1, 2, 'привет', 3.14, None, 10]
38
39 result = transformation(l)
40
41 print(f'result = {result}')
42
```

Вызов встроенной функции

The diagram illustrates the execution flow of the code. It features two functions: `summ` (lines 26-31) and `transformation` (lines 33-35). A list `l` is defined on line 37, and `transformation(l)` is called on line 39. The result is printed on line 41. Numbered circles (1-4) and arrows indicate the sequence of calls: (1) points to the `transformation` function call; (2) points to the `summ` function definition; (3) points to the `summ` function call within `transformation`; (4) points to the `transformation` function call on line 39. A yellow arrow points to the `isinstance` function call on line 29, labeled 'Вызов встроенной функции'.

ПРОБЛЕМЫ

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

JUPYTER


```
• (venv) vitalybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test/venv/
bykador/PROJECTS/test/for_lectures.py
```

```
result = 65.124900000000001
```


```
○ (venv) vitalybykador@MacBook-Air-Vitaly test %
```

Рекурсивный вызов функции – функция вызывает сама себя

```
23 l = [1, [1, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 2, 4], date(2022, 9, 17)]  
24 results = my_filter(l)
```



```
8 def my_filter(lst):  
9     lst_digits, lst_strs, lst_lists, lst_others = [], [], [], []  
10    for item in lst:  
11        match item:  
12            case int() | float() | complex():  
13                lst_digits.append(item)  
14            case str():  
15                lst_strs.append(item)  
16            case list():  
17                lst_lists.append(item)  
18            case _:  
19                lst_others.append(item)  
20    return [lst_digits, lst_strs, lst_lists, lst_others]
```



[1, 2.3, (2+3j)]

['b', 'привет']


[[1, 'a'], ['list', 2, 4]]

[None, datetime.date(2022, 9, 17)]

Рекурсивный вызов функции – функция вызывает сама себя

```
l = [1, [7, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 21, 41], date(2022, 9, 17)]
```

```
def my_filter(lst):  
    lst_digits, lst_strs, lst_others = [], [], []  
    for item in lst:  
        match item:  
            case int() | float() | complex():  
                lst_digits.append(item)  
            case str():  
                lst_strs.append(item)  
            case list():  
                res = my_filter(item)  
                lst_digits.append(res[0])  
                lst_strs.append(res[1])  
                lst_others.append(res[2])  
            case _:  
                lst_others.append(item)  
    return [lst_digits, lst_strs, lst_others]
```



```
[1, [7], 2.3, (2+3j), [21, 41]]  
[['a'], 'b', 'привет', ['list']]  
[[], None, [], datetime.date(2022, 9, 17)]
```

Рекурсивный вызов функции – функция вызывает сама себя

```
l = [1, [7, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 21, 41], date(2022, 9, 17)]
```

```
5 |
6 | def get_element(lst_src, lst_dest):
7 |     for item in lst_src:
8 |         lst_dest.append(item)
9 |
10 |
11 | def my_filter(lst):
12 |     lst_digits, lst_strs, lst_others = [], [], []
13 |     for item in lst:
14 |         match item:
15 |             case int() | float() | complex():
16 |                 lst_digits.append(item)
17 |             case str():
18 |                 lst_strs.append(item)
19 |             case list():
20 |                 res = my_filter(item)
21 |                 get_element(res[0], lst_digits)
22 |                 get_element(res[1], lst_strs)
23 |                 get_element(res[2], lst_others)
24 |             case _:
25 |                 lst_others.append(item)
26 |     return [lst_digits, lst_strs, lst_others]
27 |
```



```
[1, 7, 2.3, (2+3j), 21, 41]
```

```
['a', 'b', 'привет', 'list']
```

```
[None, datetime.date(2022, 9, 17)]
```

Рекурсивный вызов функции – функция вызывает сама себя

```
l = [1, [7, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 21, 41, ['qqq', 100, 200, [None, 1002, 'i']]], date(2022, 9, 17)]
```

```
5 |
6 | def get_element(lst_src, lst_dest):
7 |     for item in lst_src:
8 |         lst_dest.append(item)
9 |
10 |
11 | def my_filter(lst):
12 |     lst_digits, lst_strs, lst_others = [], [], []
13 |     for item in lst:
14 |         match item:
15 |             case int() | float() | complex():
16 |                 lst_digits.append(item)
17 |             case str():
18 |                 lst_strs.append(item)
19 |             case list():
20 |                 res = my_filter(item)
21 |                 get_element(res[0], lst_digits)
22 |                 get_element(res[1], lst_strs)
23 |                 get_element(res[2], lst_others)
24 |             case _:
25 |                 lst_others.append(item)
26 |     return [lst_digits, lst_strs, lst_others]
27 |
```

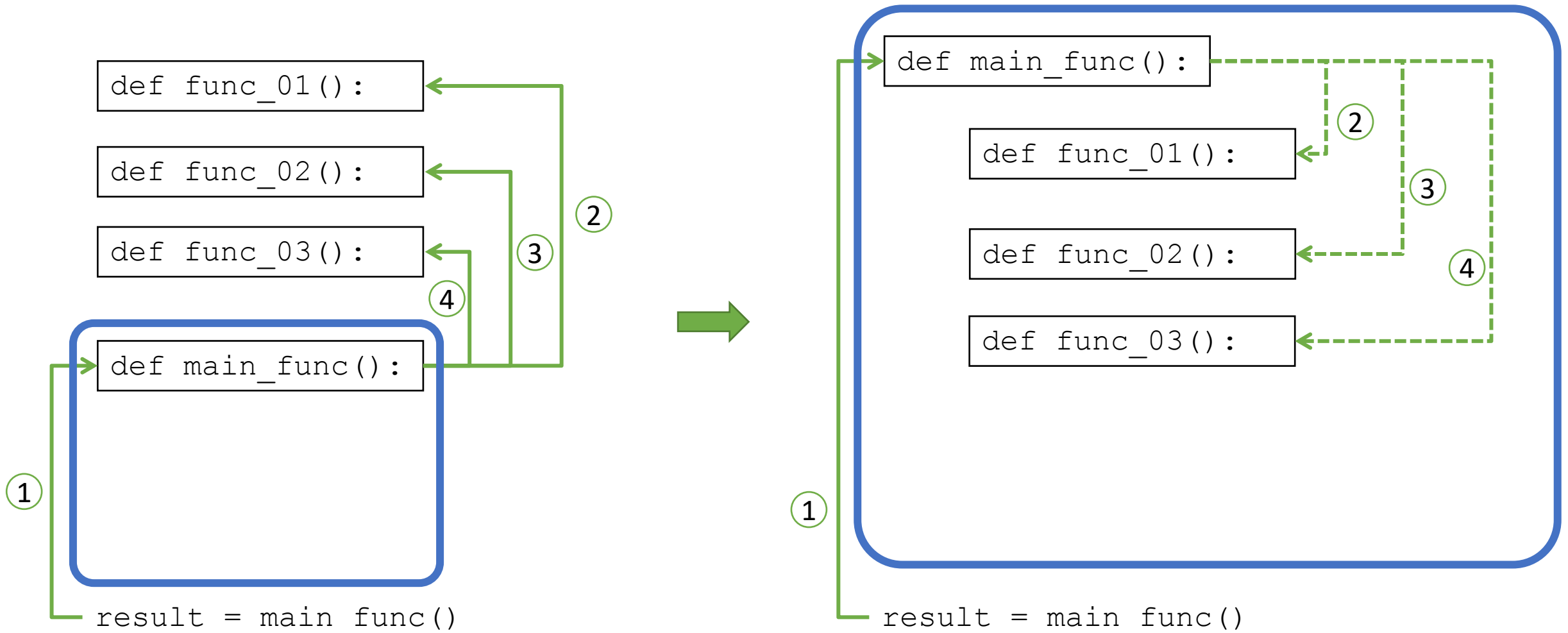


```
[1, 7, 2.3, (2+3j), 21, 41, 100, 200, 1002]
['a', 'b', 'привет', 'list', 'qqq', 'i']
[None, None, datetime.date(2022, 9, 17)]
```

Локальные функции

Локальная функция представляет собой объявление функции внутри другой функции.

Это удобно для организации сложных функций, программный код которой можно разделить на функции необходимые только для сложной функции и более нигде не вызываемых.



То есть локальные функции можно объявлять внутри других функций и вызывать ТОЛЬКО внутри функции объявления!

Локальные функции

```
l = [1, [7, 'a'], 2.3, 'b', 'привет', None, 2+3j, ['list', 21, 41, ['qqq', 100, 200, [None, 1002, 'i']]], date(2022, 9, 17)]
```

```
6
7 def my_filter(lst):
8     def get_element(lst_src, lst_dest):
9         for item in lst_src:
10             lst_dest.append(item)
11
12     lst_digits, lst_strs, lst_others = [], [], []
13     for item in lst:
14         match item:
15             case int() | float() | complex():
16                 lst_digits.append(item)
17             case str():
18                 lst_strs.append(item)
19             case list():
20                 res = my_filter(item)
21                 get_element(res[0], lst_digits)
22                 get_element(res[1], lst_strs)
23                 get_element(res[2], lst_others)
24             case _:
25                 lst_others.append(item)
26     return [lst_digits, lst_strs, lst_others]
```



```
[1, 7, 2.3, (2+3j), 21, 41, 100, 200, 1002]
['a', 'b', 'привет', 'list', 'qqq', 'i']
[None, None, datetime.date(2022, 9, 17)]
```

Функция как переменная

```
5 def summ(a, b):
6     return a + b
7
8 def sub(a, b):
9     return a - b
10
11 def mul(a, b):
12     return a * b
13
14 def div(a, b):
15     if b != 0:
16         return a / b
17     else:
18         return 'Ошибка! Деление на ноль.'
```

```
19
20 operation = {
21     '+': summ,
22     '-': sub,
23     '*': mul,
24     '/': div
25 }
26
27
28 a, b = 2, 3
29 op = '+'
30
31 if op in operation.keys():
32     func = operation[op]
33     result = func(a, b)
34 else:
35     result = 'Математическая операция не реализована.'
36
37 print(f'result = {result}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

```
• (venv) vitalitybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJE

result = 5
○ (venv) vitalitybykador@MacBook-Air-Vitaly test %
```

Функция как переменная

```
27
28 a, b = 2, 7
29 op = '/'
30
31 if op in operation.keys():
32     func = operation[op]
33     result = func(a, b)
34 else:
35     result = 'Математическая операция не реализована.'
36
37 print(f'result = {result}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

• (venv) vitalitybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test

result = 0.2857142857142857

• (venv) vitalitybykador@MacBook-Air-Vitaly test %

```
27
28 a, b = 2, 0
29 op = '/'
30
31 if op in operation.keys():
32     func = operation[op]
33     result = func(a, b)
34 else:
35     result = 'Математическая операция не реализована.'
36
37 print(f'result = {result}')
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

(venv) vitalitybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test

result = Ошибка! Деление на ноль.

(venv) vitalitybykador@MacBook-Air-Vitaly test %

Функция как переменная

```
28 a, b = 2, 0
29 op = '^'
30
31 if op in operation.keys():
32     func = operation[op]
33     result = func(a, b)
34 else:
35     result = 'Математическая операция не реализована.'
36
37 print(f'result = {result}')
38
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

(venv) vitalitybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJECTS/test

result = Математическая операция не реализована.

(venv) vitalitybykador@MacBook-Air-Vitaly test %

Функция как переменная

```
18         return 'Ошибка! Деление н
19
20     def pow(a, b):
21         return a**b
22
23
24     operation = {
25         '+': summ,
26         '-': sub,
27         '*': mul,
28         '/': div,
29         '^': pow
30     }
31
32
33     a, b = 2, 10
34     op = '^'
35
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ

(venv) vityalybykador@MacBook-Air-Vitaly test % ./Us

result = 1024

(venv) vityalybykador@MacBook-Air-Vitaly test %

Блочная область видимости переменных

Везде используется имя переменной **a**, но так как это имя расположено в разных блоках, то Python ассоциирует это имя с разными адресами памяти, то есть это разные переменные.

Хоть программа и работает, но это только для демонстрации, в рабочих программах так делать не нужно!

```
5 def func(a):
6     for a in a:
7         print(a)
8
9 a = [[1, 3, 4], ['a', 'b', 'q']]
10
11 for a in a:
12     func(a)
13     print('')
14
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ T

● (venv) vitalybykador@MacBook-Air-Vitaly test % /Users

1
3
4


a
b
q

○ (venv) vitalybykador@MacBook-Air-Vitaly test % █

Телепатов тут нет.

Сначала функцию объяви, а потом вызывай!

```
5 l = [[1, 3, 4], ['a', 'b', 'q']]
6
7 for item in l:
8     func(item) # вызов функции раньше объявления
9     print('')
10
11 def func(lst):
12     for item in lst:
13         print(item)
14
```



ПРОБЛЕМЫ 1 ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

(venv) vityalybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJ


Traceback (most recent call last):

File "/Users/vitalybykador/PROJECTS/test/for_lectures.py", line 8, in
func(item) # вызов функции раньше объявления

NameError: name 'func' is not defined

(venv) vityalybykador@MacBook-Air-Vitaly test %

```
5 def func(lst):
6     for item in lst:
7         print(item)
8
9 l = [[1, 3, 4], ['a', 'b', 'q']]
10
11 for item in l:
12     func(item) # вызов функции после объявления
13     print('')
14
15
```



ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ JUPYTER

(venv) vityalybykador@MacBook-Air-Vitaly test % /Users/vitalybykador/PROJ

1

3

4

a

b

q

(venv) vityalybykador@MacBook-Air-Vitaly test %