Функции

Функции

- Часть переиспользуемого, или требующего оформления в отдельный функционал, кода.
- Базовый синтаксис

```
def fun():
    pass
```

```
1  def func_welcome():
2      print("Quick start!")
3
```

- Есть входные и возвращаемые параметры

Возвращаемое значение

- Return – ключевое слово

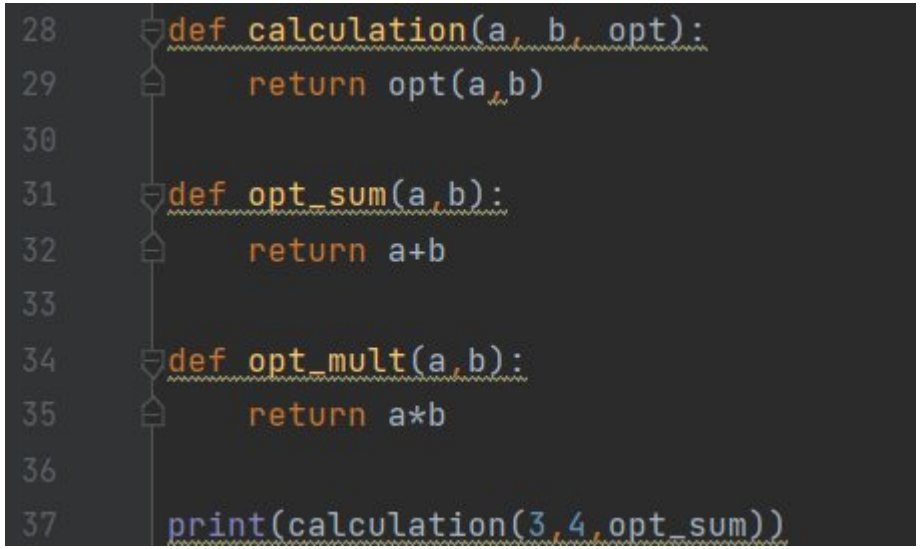
```
5   import random
6
7   def func_rand_value():
8       val = random.uniform(0,1)
9       return val
```

```
13  def f1():
14      print("I'm function 1!")
15
16  def f2():
17      print("I'm function 2!")
18
19  def func_choise():
20      val = random.uniform(0, 1)
21      if val< 0.5:
22          return f1()
23      else:
24          return f2()
25
26
```

```
print(func_choise())
```

Передача параметров

- По умолчанию: тип передаваемого параметра подразумевается при написании кода функции



```
28  def calculation(a, b, opt):
29      return opt(a,b)
30
31  def opt_sum(a,b):
32      return a+b
33
34  def opt_mult(a,b):
35      return a*b
36
37  print(calculation(3,4,opt_sum))
```

Передача множества параметров

```
44  def func_many_args(*argv):
45      for arg in argv:
46          print(arg, type(arg))
47
48
49  print(func_many_args(3,5,'study',(1,2), {1:2, 4:5}))
```

```
def func_named_args(**kwargs):
    for key, value in kwargs.items():
        print(key, value)

func_named_args(one = 1, two = 2.0, three  ='three')
```

```
57  def func_one_plus_argv(a,*argv):
58      if a > 0:
59          for arg in argv:
60              print(arg, type(arg))
61      else:
62          print('Change negative on positive')
63
64  func_one_plus_argv(-1,2)
```

```
51      def func_named_args(**kwargs):
52          for key, value in kwargs.items():
53              print(key, value)
54
55      func_named_args(one = 1, two = 2.0, three  ='three')
56
57      params = {'one':1, 'two':2.0, 'three': 'three'}
58      func_named_args(**params)
```

Передача изменяемых параметров

```
151  def foo(a, l=None):
152      if l is None:
153          l = []
154      l.append(a)
155      return l
156
157  l = []
158  foo(3, l)
159  print(l)
```

Вложенные функции

```
72  def f1():
73      print('F1 is called')
74      def f2():
75          print('F2 is called')
76
77
78  f1()
79  f2()
```

ошибка видимости

переменные, объявленные внутри функции, видны только внутри функции (область видимости), с соответствующим «временем жизни»

Функция как параметр

```
90  def func_choise():
91      val = random.uniform(0, 1)
92
93      def f1():
94          print("I'm function 1!")
95          return 1
96
97      def f2():
98          print("I'm function 2!")
99          return 2
100
101     if val< 0.5:
102         return f1
103     else:
104         return f2
```

```
106 f = func_choise()
107 print(f())
```

Декоратор

```
110  def sum(a,b):
111      return a+b
112
113  def mult(a,b):
114      return a*b
115
116  def decorator_math_scalar(func):
117      def wrapper(*argv):
118          result = func(argv[0],argv[1])* argv[2]
119          return result
120      return wrapper
121
122  print(decorator_math_scalar(sum)(1,2,4.5))
```

```
124  @decorator_math_scalar
125  def diff(a,b):
126      return a-b
127
128  print(diff(3,4,0.5))
```

Декораторы

```python
def decorator_math_scalar(c):

    def decorator(func):

        def wrapper(*argv):
            result = func(argv[0],argv[1])*c
            return result
        return wrapper

    return decorator

@decorator_math_scalar(0.5)
def diff(a,b):
    return a-b

print(diff(2,3))
```

```python
def decorator_math_scalar(func):
    def wrapper(*argv):
        result = func(argv[0],argv[1])* argv[2]
        return result
    return wrapper

print(decorator_math_scalar(sum)(1,2,4.5))

@decorator_math_scalar
def diff(a,b):
    return a-b

print(diff(3,4,0.5))
```