

Python: functions

budnyjj@pirates.by

- ▶ Функции
- ▶ Модули
- ▶ Стандартная библиотека

Определение и вызов функции

```
1 def cube(x):  
2     return x ** 3  
3  
4 print(cube(12))  
5  
6 print(cube)  
7 print(isinstance(cube, object))  
8 print(dir(cube))  
9
```

Передача аргументов в функцию

```
1 def modify(string, lst):
2     string = "new " + string
3     for idx, val in enumerate(lst):
4         lst[idx] = "new " + val
5
6 names = ["cat", "book", "cinema"]
7 s = "machine"
8
9 print "BEFORE MODIFY()"
10 print names
11 print s
12
13 modify(s, names)
14
15 print "AFTER MODIFY()"
16 print names
17 print s
18
```

- ▶ Обязательные
- ▶ Именованные
- ▶ По умолчанию
- ▶ Переменной длины
- ▶ Ключевые

Значение аргументов по умолчанию¹

```
1 def power(x, y=2):  
2     r = 1  
3     for i in range(y):  
4         r = r * x  
5     return r  
6  
7 print power(3)  
8 print power(3, 3)  
9 print power(5, 5)  
10
```

¹<http://zetcode.com/lang/python/functions/>

Задание значения аргумента по имени

```
1 def display(name, age, sex="M"):
2     print "Name: ", name
3     print "Age: ", age
4     print "Sex: ", sex
5
6 display("Lary", 43, "M")
7 display("Lary", age=43)
8 display("Lary", age=43, sex="M")
9 display(age=43, name="Lary", sex="M")
10
11 display(age=24, name="Joan", "F")           # error
```

Список аргументов переменной длины

```
1 print sum
2
3 def sum(*args):
4     '''Function returns the sum of all values'''
5     r = 0
6     for i in args:
7         r += i
8     return r
9
10 print sum
11 print sum.__doc__
12 print sum(1, 2, 3)
13 print sum(1, 2, 3, 4, 5)
14
```


Ключевые аргументы

```
1 def display(**details):  
2     for i in details:  
3         print "%s: %s" % (i, details[i])  
4  
5 display(name="Lary", age=43, sex="M")  
6
```

Особенности передачи аргументов

```
1 def func(a,b,c,d=False,*args,**kwargs):  
2     print a, b, c, d, args, kwargs  
3  
4 func(*[1,2,3,4,5], **{'6':7})  
5 func(*[1,2,3,], **{'d':7})  
6 func(1, 2, *[3,], **{'d':7})  
7
```

Область видимости

```
1 name = "Jack"
2
3 def f():
4     # global name
5     name = "Robert"
6     print "Within function:", name
7     print locals()
8     print globals()
9
10 print "Outside function:", name
11 f()
12 print "Outside function:", name
```

“Объектами первого класса в контексте конкретного языка программирования называются сущности, которые могут быть переданы как параметр, возвращены из функции, присвоены переменной”².

²http://en.wikipedia.org/wiki/First-class_citizen

Функция как объект первого класса

```
1 def square(x):  
2     return x ** 2  
3  
4 s = square  
5 print s(5)  
6  
7 def ff(f, x):  
8     return f(f(x) - 1)  
9  
10 print ff(s, 5)
```

Анонимные функции

```
1 lowercase = lambda x: x.lower()
2 print_assign = lambda name, value:\
3     name + '=' + str(value)
4 adder = lambda x, y: x+y
5
6 print lowercase("THETA")
7 print print_assign("two", 2)
8 print adder(2, 3)
```

Simple is better than complex

```
items = [("one", 1), ("two", 2), ("three", 3)]  
# 1 + 2 + 3 = ?
```

```
▶ total = reduce(lambda a, b:\n                    (0, a[1] + b[1]),items)[1] 1  
2
```

```
▶ def combine (a, b): 1  
    return 0, a[1] + b[1] 2  
total = reduce(combine, items)[1] 3
```

```
▶ total = 0 1  
for a, b in items: 2  
    total += b 3
```

```
▶ total = sum(b for a,b in items) 1
```

Декораторы

```
def func(...)  
...  
def wrapper(...)  
...  
f = wrapper(func)
```

VS

```
def wrapper(...)  
...  
@wrapper  
def func(...)  
...
```


Пример декоратора

```
1 import time
2
3 def timer(f):
4     def tmp(*args, **kwargs):
5         t = time.time()
6         res = f(*args, **kwargs)
7         print "Time of function evaluation: \
8             {:f}".format(time.time()-t)
9         return res
10    return tmp
11
12 @timer
13 def func(x, y):
14     return x + y
15
16 print func(1, 2)
```

Модуль fibo.py

```
1 # Fibonacci numbers module
2
3 def fib(n):    # write Fibonacci series up to n
4     a, b = 0, 1
5     while b < n:
6         print b,
7         a, b = b, a+b
8
9 def fib2(n): # return Fibonacci series up to n
10    result = []
11    a, b = 0, 1
12    while b < n:
13        result.append(b)
14        a, b = b, a+b
15    return result
16
17 if __name__ == "__main__":
18     import sys
19     fib(int(sys.argv[1]))
```

Импорт модуля

- ▶ `import fibo`
- ▶ `import fibo as f`
- ▶ `from fibo import fib`
- ▶ `from fibo import *`

Использование fibo.py

- ▶ В качестве модуля:

```
1 import fibo
2 dir(fibo)
3 fib = fibo.fib(500)
4
```

- ▶ В качестве скрипта:

```
1 python2 fibo.py 50
2
```

import sound.effects.echo

```
sound/                # Top-level package
__init__.py           # Initialize the sound package
formats/              # Subpackage for file format conversions
    __init__.py
    wavread.py
    wavwrite.py
    aiffread.py
    aiffwrite.py
    ...
effects/              # Subpackage for sound effects
    __init__.py
    echo.py
    surround.py
    reverse.py
    ...
filters/              # Subpackage for filters
    __init__.py
    equalizer.py
    vocoder.py
    karaoke.py
    ...
```

Обзор содержимого стандартной библиотеки

Организация ВП

- ▶ **re**
- ▶ difflib
- ▶ datetime
- ▶ calendar
- ▶ **collections**
- ▶ heapq
- ▶ bisect
- ▶ array
- ▶ Queue
- ▶ mutex
- ▶ math, cmath
- ▶ random
- ▶ fraction
- ▶ pprint
- ▶ **itertools**
- ▶ functools
- ▶ hashlib
- ▶ threading

Ввод и вывод

- ▶ pickle
- ▶ gdbm
- ▶ zlib
- ▶ csv
- ▶ email
- ▶ json
- ▶ HTMLParser
- ▶ xml.*
- ▶ webbrowser
- ▶ urllib[2]
- ▶ httpplib

Поддержка, взаимодействие с платформой

- ▶ os
- ▶ os.path
- ▶ argparse
- ▶ gettext
- ▶ locale
- ▶ pydoc
- ▶ logging
- ▶ unittest
- ▶ pdb

- ▶ <http://zetcode.com/lang/python/functions/>
- ▶ <https://docs.python.org/2/library/functions.html>
- ▶ <https://docs.python.org/2/howto/functional.html>
- ▶ <https://docs.python.org/2/tutorial/modules.html>
- ▶ <https://docs.python.org/2.7/library/>

Через неделю — классы и исключения:

- ▶ Атрибуты, методы объектов
- ▶ Инкапсуляция
- ▶ Наследование
- ▶ Генерация и обработка исключений