Python

Распаковка позиционных аргументов

Вместо позиционных аргументов при вызове функции можно передать кортеж или список, тогда перед ним ставится *

```
1  def get_sum(x, y):
2      return x + y
3
4  args1 = (2, 3)
5  args2 = [4, 11]
6  assert get_sum(*args1) == 5
7  assert get_sum(*args2) == 15
8
```

Распаковка именованных аргументов

Вместо именованных аргументов при вызове функции можно передать словарь, тогда перед ним ставится **

```
1  def get_sum(a, b):
2      return a + b
3
4  kwargs = {'a': 2, 'b': 3}
5  assert get_sum(**kwargs) == get_sum(a=2, b=3)
6
```

Анонимные функции (lambda)

Анонимные функции - функции, которые создаются с помощью инструкции **lambda** и могут содержать лишь одно выражение, которые они возвращают. Анонимные функции можно сразу же и вызвать.

```
foo1 = lambda x, y: x + y
assert foo1(2, 3) == 5
assert foo1('o', 'k') == 'ok'

foo2 = lambda *args: args
assert foo2(2, 3, 4) == (2, 3, 4)

assert (lambda x, y: x + y)('Py', 'thon') == 'Python'
assert (lambda x: x * x + 1)(3) == 10
```

Область видимости переменных

Область видимости - место где определяются переменные и выполняется их поиск.

Всегда, когда в программе используется какое то имя, интерпретатор создает, изменяет или отыскивает это имя в пространстве имен – в области, где находятся имена.

- Если присваивание переменной выполняется внутри инструкции def, переменная является локальной для этой функции.
- Если присваивание производится в пределах объемлющей инструкции def, переменная является нелокальной для этой функции.
- Если присваивание производится за пределами всех инструкций def, она является глобальной для всего файла.

Область видимости переменных

```
x = 1

def func():
    x = 10
    print('x in func =', x)

func()   # 10
print('x after func execution = ', x)   # 1
```

Область видимости переменных

```
x = 1
y = 1
z = 1

def outer():
    y = 10
    z = 10
    def inner():
        z = 100
        print('x in inner =', x)   # 1
        print('y in inner =', y)   # 10
        print('z in inner =', z)   # 100

    inner()

outer()
print(x, y, z)   # 1 1 1
```

Инструкция global

Инструкция **global** позволяет изменять глобальные переменные внутри функции.

```
x = 1

def func():
    global x
    x = 10
    print("x in func =", x)   # 10

func()
print("x after func execution =", x)   # 10
```

Инструкция nonlocal

Инструкция nonlocal позволяет изменять переменную из замыкания внутри функции (Python3)

```
x = 0

def outer():
    x = 1
    def inner():
        nonlocal x
        x = 2
        print("inner:", x)   # 2

    inner()
    print("outer:", x)   # 2

outer()
print("global:", x)   # 0
```

Module sys

https://docs.python.org/3/library/sys.html

Модуль **sys** обеспечивает доступ к некоторым переменным и функциям, взаимодействующим с интерпретатором python.

```
import sys

a = sys.argv.pop()

def check_sys_argv():
    print(a)
```

Module os

os.environ

os.path

os.system

Module requests

```
>>> r = requests.put("http://httpbin.org/put")
>>> r = requests.delete("http://httpbin.org/delete")
>>> r = requests.head("http://httpbin.org/get")
>>> r = requests.options("http://httpbin.org/get")
```