

Python

базовый тренинг

Ружин Алексей
ruzin@me.com

Функции

Определение

- определение:

```
def add(x, y):  
    return x + y
```
- определение функции с параметром по умолчанию:

```
def split(line, delimiter=','):  
    return line.split(delimiter)
```

- ```
a = 10
def foo(x=a):
 return x
a = 5
foo()
```

```
def foo(x, items=[]):
 items.append(x)
 return items
foo(1)
foo(2)
foo(3)
```

```
def foo(x, items=None):
 if items is None:
 items = []
 items.append(x)
 return items
```

# переменное число аргументов

- ```
import sys
def fprintf(file, fmt, *args):
    file.write(fmt.format(*args))

fprintf(sys.stdout, "{} {} {}", 42, "hw", 2.71)
```
- ```
def printf(fmt, *args):
 fprintf(sys.stdout, fmt, *args)
```

# явное указание переменных

- ```
def foo(w, x, y, z):  
    pass
```

```
foo(z=[], x=3, y=33, w='hello')
```

словарь "остальных" аргументов

- ```
def foo(data, **params):
 color = params.get('color', 'black')
 bg = params.get('background', 'white')
 #...
```
- Комбинация всех вариантов:  

```
def foo(*args, **kwargs)
 pass
```

# Передача параметров и возвращаемые значения

- ```
a = [1, 2, 3, 4, 5]
def square(items):
    for i, x in enumerate(items):
        items[i] = x * x
square(a)
print(a)
```
- ```
def foo()
 return 1, 2
```



# Правила видимости

- локальные
- глобальные
- не локальные
- доступность на чтение  
недоступность на изменение

# Функции как объекты и замыкания

- функции - объекты первого класса

```
def callf(func):
 return func()
```

- Замыкания

```
x = 37
def helloworld():
 print("Hello, world, {}".format(x))

callf(helloworld)
```

# Пример

- ```
def countdown(n):  
    def next():  
        nonlocal n  
        r = n  
        n -= 1  
        return r  
    return next
```

```
x = countdown(10)  
x()
```

Декораторы

- `@trace`
`def square(x):`
 `return x*x`

`def square(x):`
 `return x*x`
`square = trace(square)`
- декоратор с параметрами

Генераторы и инструкция yield

- ```
def countdown(n):
 print("Обратный отсчет, начиная с {}".format(n))
 while n > 0:
 yield n
 n -= 1
 return
```

```
for x in countdown(10):
 print(x)
```

# Сопрограммы и выражения yield

- ```
def receiver():  
    print("ready")  
    while True:  
        n = yield  
        print("got {}".format(n))
```

```
r = receiver()  
r.__next__()  
r.send(1)
```

Генераторы списков

- [expression for item1 in iterable1 if condition1
for item2 in iterable2 if condition2
...
for itemN in iterableN if conditionN]

```
s = []  
for item1 in iterable1:  
    if condition1:  
        for item2 in iterable2:  
            if condition2: ...  
                for itemN in iterableN:  
                    if conditionN:  
                        s.append(expression)
```

Выражения-генераторы

- (expression for item1 in iterable1 if condition1 ...)

```
x = (x*x for x in range(10))  
print(x.__next__())  
for a in x:  
    print(a)
```


Lambda-функции

- `add = lambda x, y: x + y`
`print(add(2, 3))`

`names.sort(key=lambda n: n.lower())`

Строки документирования

- ```
def func(a):
 """
 some description
 :param a: Описание
 :type a: int
 :returns: возвращает
 :rtype: int
 """

 pass

help(func)
```

# Атрибуты функций

- ```
def func():  
    pass
```

```
func.any = 1
```

Задача 1

- Напишите декоратор, который кэширует результат вызова функции и возвращает его, если последующие вызовы этой функции производятся с теми же значениями параметров в течение 5 минут, но не более 10 раз.

Задача 2

- Прочитайте из файла ключевые слова (по одному в строке) и выведите на экран те из них, которые встречаются более одного раза, в порядке уменьшения частоты вхождений в исходный файл. Следует игнорировать регистр букв (прописные/заглавные).

Пример на входе:

```
microsoft  
apple  
Microsoft  
Apple  
security  
Microsoft  
Internet
```

Пример на выходе:

```
microsoft: 3  
apple: 2
```