

# Лекция 2

## Еще основы Python

19 февраля 2015 г.

# Элементы функционального программирования

# Распаковка списков

```
>>> x, y = 1, 'a'
```

```
>>> x
```

```
1
```

```
>>> y
```

```
'a'
```

# Распаковка списков

```
>>> x, y = 1, 'a'
```

```
>>> x
```

```
1
```

```
>>> y
```

```
'a'
```

```
>>> l = [1, 2, 3]
```

```
>>> x, y, z = l
```

```
>>> print x, y, z
```

```
1 2 3
```

# Swap в Python

```
>>> x = 'a'
>>> y = 'b'
>>> print x, y
a b
>>> x, y = y, x
>>> print x, y
b a
```

# Функция zip

```
x = ['a', 'b', 'c']
```

```
y = ['q', 'w', 'e']
```

“Параллельный” проход по двум спискам.

# Функция zip

```
x = ['a', 'b', 'c']  
y = ['q', 'w', 'e']
```

“Параллельный” проход по двум спискам.

```
for a, b in zip(x, y):  
    print a, b
```

```
a q  
b w  
c e
```

# Функция enumerate

```
x = ['a', 'b', 'c']
```

Проход и по индексам и по значениям.



# Функция enumerate

```
x = ['a', 'b', 'c']
```

Проход и по индексам и по значениям.

```
for index, value in enumerate(x):  
    print index, value
```

```
0 a  
1 b  
2 c
```

# Функция reversed

```
x = ['a', 'b', 'c']
```

Проход в обратном порядке.

# Функция reversed

```
x = ['a', 'b', 'c']
```

Проход в обратном порядке.

```
for val in reversed(x):  
    print val
```

```
c  
b  
a
```

# Функция sorted

```
x = [2, 1, 3]  
print sorted(x)
```

```
[1, 2, 3]
```

# Функция sorted

```
x = [2, 1, 3]  
print sorted(x)
```

```
[1, 2, 3]
```

```
x = ['hello', 'world', 'hi']  
print sorted(x)
```

```
['hello', 'hi', 'world']
```

## Опции sorted

```
>>> sorted([1, 3, 2], reverse=True)
[3, 2, 1]
```

## Опции sorted

```
>>> sorted([1, 3, 2], reverse=True)
[3, 2, 1]
```

```
def get_second(x):
    return x[1]
```

```
l = [['a', 2], ['c', 1], ['b', 3]]
```

```
>>> sorted(l)
[['a', 2], ['b', 3], ['c', 1]]
```

```
>>> sorted(l, key=get_second)
[['c', 1], ['a', 2], ['b', 3]]
```

# Лямбда-функции

Создание безымянных функций.

```
>>> f = lambda x: x + 1
```

```
>>> f(1)
```

```
2
```

```
>>> g = lambda a, b: a - b
```

```
>>> g(3, 5)
```

```
-2
```



# Лямбда-функции

Создание безымянных функций.

```
>>> f = lambda x: x + 1
```

```
>>> f(1)
```

```
2
```

```
>>> g = lambda a, b: a - b
```

```
>>> g(3, 5)
```

```
-2
```

(Так использовать не рекомендуется)

# Лямбда-функции и sorted

```
>>> l = [['a', 2], ['c', 1], ['b', 3]]
```

```
>>> sorted(l)
[['a', 2], ['b', 3], ['c', 1]]
```

```
>>> sorted(l, key=lambda x: x[1])
[['c', 1], ['a', 2], ['b', 3]]
```

```
>>> sorted(l, key=lambda x: x[1],
           reverse=True)
[['b', 3], ['a', 2], ['c', 1]]
```

# Списковые выражения

```
sq = []  
for x in range(10):  
    sq.append(x ** 2)  
  
print sq
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

# Списковые выражения

```
sq = []  
for x in range(10):  
    sq.append(x ** 2)  
  
print sq
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
sq2 = [x ** 2 for x in range(10)]  
print sq2
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

# Сложные списковые выражения

```
print [(x, y) for x in [1,2,3]
        for y in [3,1,4] if x != y]
```

```
[(1, 3), (1, 4), (2, 3), (2, 1),
 (2, 4), (3, 1), (3, 4)]
```

```
combs = []
for x in [1,2,3]:
    for y in [3,1,4]:
        if x != y:
            combs.append((x, y))
```

# Вложенные списковые выражения

```
matrix = [  
    [1, 2, 3, 4],  
    [5, 6, 7, 8],  
    [9, 10, 11, 12],  
]
```

```
print [[row[i] for row in matrix]  
       for i in range(4)]
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11],  
 [4, 8, 12]]
```

# Еще типы данных

# Словари

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel
{'sape': 4139, 'jack': 4098}
>>> tel['jack']
4098
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'jack': 4098,
 'guido': 4127}
```



# Операции со словарями

```
>>> del tel['sape']
>>> tel
{'jack': 4098, 'guido': 4127}
>>> tel['jack'] = 1234
>>> tel
{'jack': 1234, 'guido': 4127}
```

# Операции со словарями

```
>>> del tel['sape']
>>> tel
{'jack': 4098, 'guido': 4127}
>>> tel['jack'] = 1234
>>> tel
{'jack': 1234, 'guido': 4127}

>>> tel.keys()
['jack', 'guido']
>>> 'jack' in tel
True
```

## Цикл со словарями

```
tel = {'jack': 1234, 'guido': 4127}
```

```
for k in tel:  
    print k, tel[k]
```

```
jack 1234  
guido 4127
```

## Цикл со словарями

```
tel = {'jack': 1234, 'guido': 4127}
```

```
for k in tel:  
    print k, tel[k]
```

```
jack 1234  
guido 4127
```

(Порядок произвольный)

# Элементы словаря

```
tel = {'jack': 1234, 'guido': 4127}
```

```
for key, value in tel.items():  
    print key, value
```

```
jack 1234  
guido 4127
```

# Элементы словаря

```
tel = {'jack': 1234, 'guido': 4127}
```

```
for key, value in tel.items():  
    print key, value
```

```
jack 1234  
guido 4127
```

(Порядок произвольный)

## Упорядочивание словарей

```
d = {'a': 2, 'b': 1, 'c': 3}
for k, v in d.items():
    print k, v
```

```
a 2
```

```
c 3
```

```
b 1
```

## Упорядочивание словарей

```
d = {'a': 2, 'b': 1, 'c': 3}
for k, v in d.items():
    print k, v
```

```
a 2
c 3
b 1
```

```
for k, v in sorted(d.items()):
    print k, v
```

```
a 2
b 1
c 3
```



# Упорядочивание словарей

```
for k, v in sorted(d.items(),  
                  key=lambda x: x[1]):  
    print k, v
```

```
b 1  
a 2  
c 3
```

## Что может быть ключом словаря?

```
>>> d = {}  
>>> d['a'] = 1  
>>> d[5] = 'hi'  
>>> d  
{'a': 1, 5: 'hi'}
```

## Что может быть ключом словаря?

```
>>> d = {}
>>> d['a'] = 1
>>> d[5] = 'hi'
>>> d
{'a': 1, 5: 'hi'}
```

```
>>> d[[1, 2]] = 0
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'list'

```
>>> d[{'a': 'b'}] = 0
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'dict'

# Туплы

`tuple` = неизменяемый `list`

```
>>> t = (1, 2, 3)
```

```
>>> t
```

```
(1, 2, 3)
```

```
>>> t[0] = 0
```

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

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: 'tuple' object does not  
    support item assignment
```

# Туплы

`tuple` = неизменяемый `list`

```
>>> t = (1, 2, 3)
```

```
>>> t
```

```
(1, 2, 3)
```

```
>>> t[0] = 0
```

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

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: 'tuple' object does not  
    support item assignment
```

```
>>> list(t)
```

```
[1, 2, 3]
```

# Таплы как ключи

```
>>> d[(1, 2)] = 0
>>> d
{'a': 1, (1, 2): 0, 5: 'hi'}
>>> d[(1, 2)]
0
```

# Множества

```
>>> s = set([1, 2, 3])
>>> s
set([1, 2, 3])
>>> s.add(1)
>>> s.add(4)
>>> s
set([1, 2, 3, 4])
>>> 2 in s
True
>>> 'hi' in s
False
```

# Файлы



# Файлы

```
>>> f = open('test.txt', 'w')
>>> f
<open file 'test.txt' ...>
>>> f.write('hello, world')
>>> f.close()

>>> f2 = open('test.txt')
>>> f2.read()
'hello, world'
>>> f2.close()
```

# Файлы и with

```
with open('test.txt') as f:  
    f.read()
```

После выхода из блока `with` файл гарантированно закрыт.

Рекомендуемый способ работы с файлами.

# Чтение файлов

Содержимое файла

```
First line
```

```
Second line
```

```
>>> f.read()
```

```
'First line\nSecond line\n'
```

# Чтение файлов

Содержимое файла

First line

Second line

```
>>> f.read()
```

```
'First line\nSecond line\n'
```

```
for line in f:
```

```
    print '---', line,
```

```
--- First line
```

```
--- Second line
```

# Модули

# Подключение библиотеки

```
>>> import math
>>> math.floor(5.5)
5.0
>>> math.ceil(5.5)
6.0
```

# Подключение библиотеки

```
>>> import math
>>> math.floor(5.5)
5.0
>>> math.ceil(5.5)
6.0
```

Помощь по модулю:

```
>>> help(math)
```

# Подключение функций

```
>>> from math import factorial  
>>> factorial(3)  
6
```



# Подключение функций

```
>>> from math import factorial
>>> factorial(3)
6
```

```
>>> math.factorial(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'math' is not defined
```

# Модуль sys

Файл test.py

```
import sys  
print sys.argv
```

```
$ python test.py
```

```
['test.py']
```

```
$ python test.py hello world 1 2
```

```
['test.py', 'hello', 'world', '1', '2']
```

# Модуль sys

`sys.stdin` - “файл” ввода с клавиатуры

`sys.stdout` - “файл” вывода на экран

```
>>> import sys
```

```
>>> name = sys.stdin.read()
```

```
Alexey
```

```
>>> sys.stdout.write(name)
```

```
Alexey
```

# Работа со строками

# split

```
>>> s = 'First sentence. 2nd sentence.'  
>>> s.split(' ')  
['First', 'sentence.', '2nd',  
'sentence.']  
>>> s.split('.')  
['First sentence', ' 2nd sentence', '']
```

# split

```
>>> s = 'First sentence. 2nd sentence.'
>>> s.split(' ')
['First', 'sentence.', '2nd',
'sentence.']
>>> s.split('.')
['First sentence', ' 2nd sentence', '']

>>> 'aaaa'.split('a')
['', '', '', '', '']
```

# join

```
>>> ' '.join(['abc', 'de', 'f'])  
'abc de f'  
>>> '_'.join('hello')  
'h_e_l_l_o'
```

# join

```
>>> ' '.join(['abc', 'de', 'f'])  
'abc de f'
```

```
>>> '_'.join('hello')  
'h_e_l_l_o'
```

```
>>> '_'.join(['a', 'b', 'c']).split('_')  
['a', 'b', 'c']
```

```
>>> '_'.join('a_b_c'.split('_'))  
'a_b_c'
```



# Накапливание строк

```
lines = []  
for i in range(5):  
    lines.append(str(i))  
print '\n'.join(lines)
```

```
0  
1  
2  
3  
4
```

## Еще операции со строками

```
>>> 'hello '.rstrip()
'hello'
>>> ' hello'.lstrip()
'hello'
>>> ' hello\n'.strip()
'hello'
```

## Еще операции со строками

```
>>> 'hello '.rstrip()
'hello'
>>> ' hello'.lstrip()
'hello'
>>> ' hello\n'.strip()
'hello'

>>> 'a'.isalpha()
True
>>> 'a'.isdigit()
False
```

# Разное

# Управление циклами

```
x = 0
while True:
    x += 1
    print x,
    if x > 5:
        break
```

1 2 3 4 5 6

# Управление циклами

```
x = 0
while x < 6:
    x += 1
    if x == 2 or x == 3:
        continue
    print x,
```

1 4 5 6

# Пустые блоки

```
for i in range(5):  
    pass
```

```
def foo(a, b):  
    pass
```

# Функция main

```
def foo(a, b):
```

```
    ...
```

```
def bar(x):
```

```
    ...
```

```
def main():
```

```
    foo(1, 2)
```

```
    ...
```

```
    bar('a')
```

```
main()
```