Лекция 1
Основы Python

12 февраля 2015 г.

О языке Python

История

Автор: Гвидо ван Россум.
(Сейчас он является BDFL — Benevolent Dictator For Life)

1994г. — Python 1.0
2000г. — Python 2.0
2008г. — Python 3.0

Особенности

- Понятность кода
  - Легко писать
  - Легко отлаживать
  - Легко читать и модифицировать

Особенности

- Понятность кода
  - Легко писать
  - Легко отлаживать
  - Легко читать и модифицировать
- Низкая эффективность
  - *Чистый* Python 5-100 раз медленнее C++

Эффективность разработки и эффективность исполнения

Когда эффективность разработки важнее?

- Исследовательское программирование
- Прототипы
- “Служебные” программы

Интеграция с другими языками

90% времени выполнения тратится на 10% кода.

Интеграция с другими языками

90% времени выполнения тратится на 10% кода.

Остальные 90% кода не нужно оптимизировать, но отлаживать все равно нужно.

Интеграция с другими языками

90% времени выполнения тратится на 10% кода.

Остальные 90% кода не нужно оптимизировать, но отлаживать все равно нужно.

Хороший подход:

- Медленная часть — на быстром языке
- Сложная часть — на простом языке

Дзен Питона

```
import this
```

Beautiful is better than ugly.

...

Simple is better than complex.

Complex is better than complicated.

...

Readability counts.

...

Версии

Python 2.x

- 2.5
- 2.6
- **2.7**

Python 3.x (несовместим с 2.x)

- 3.2
- 3.3
- 3.4

Работа с интерпретатором

Обычный режим работы

```
$ python main.py
```

Интерпретатор + исходный код.
Запускается новый процесс с программой.

Интерактивный режим

```
$ python
```

Интерпретатор без кода.
Исполнение кода в онлайн-режиме.

Интерактивный режим

```
$ python
```

Интерпретатор без кода.
Исполнение кода в онлайн-режиме.

Выход: Ctrl+D (Unix) или Ctrl+Z+Enter (Windows).

Помощь: `help(X)`, где `X` — то, по чему нужна помощь.
Выход из помощи: q.

Основные типы данных

Python как калькулятор

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> 8 / 5.0
1.6
>>> 5 ** 2
25
```

Деление

```
>>> 17 / 3
5
>>> 17 / 3.0
5.666666666666667
>>> 17 // 3.0
5.0
>>> 17 % 3
2
```

Логические выражения

```
>>> 1 == 1
True
>>> 2 * 2 != 4
False
>>> False or 5 > 2 and True
True
>>> not 1 < 2 < 10
False
```

Переменные

```
>>> x = 1
>>> y = 2
>>> x * y
2
```

Переменные

```
>>> x = 1
>>> y = 2
>>> x * y
2
```

```
>>> x
1
>>> x = 1.0
>>> x
1.0
```

Динамическая типизация. Жесткая типизация.

Изменение “на месте”

```
>>> x = 1
>>> x += 2
>>> x
3
>>> x /= 2.0
>>> x
1.5
```

Строки

```
>>> s = 'hello, world'
>>> s
'hello, world'
```

Строки

```
>>> s = 'hello, world'
>>> s
'hello, world'

>>> s + '!'
'hello, world!'
>>> s[0]
'h'
>>> s[1:6]
'ello,'
>>> len(s)
12
```

Срезы

```
>>> word = 'Python'
>>> word[-1]
'n'
>>> word[-2]
'o'
```

Срезы

```
>>> word = 'Python'
>>> word[-1]
'n'
>>> word[-2]
'o'

>>> word[:2]
'Py'
>>> word[4:]
'on'
>>> word[-2:]
'on'
```

Кавычки в строках

```
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

Что нельзя делать со строками

```
>>> word = 'Python'
>>> word[42]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Что нельзя делать со строками

```
>>> word = 'Python'
>>> word[42]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range


>>> word[0] = 'J'
  ...
TypeError: 'str' object does not support
   item assignment
```

Вывод на экран

```
'Hello, world!'
```

Вывод на экран

```
'Hello, world!'
```

```

```

```
print 'Hello, world!'
```

```
Hello, world!
```

Вывод на экран

```
'Hello, world!'
```

```

```

```
print 'Hello, world!'
```

```
Hello, world!
```

```
print '1 + 1 =', 2
```

```
1 + 1 = 2
```

Переносы строк

```
print 'first line'
print
print 'third line'
```

```
first line

third line
```

Переносы строк

```
print 'first line'
print
print 'third line'
```

```
first line

third line
```

```
print 'first\nsecond'
```

```
first
second
```

Тройные кавычки

```
print """Usage: thingy [OPTIONS]
   -h - Display this usage message
   -H hostname - Hostname to connect to
"""
```

```
Usage: thingy [OPTIONS]
   -h - Display this usage message
   -H hostname - Hostname to connect to
```

Пользовательский ввод

```
s = raw_input("Enter something.\n> ")
print "You entered", s
```

```
Enter something.
> hi
You entered hi
```

Интерпретация ввода

```
>>> n = raw_input("Enter a number.\n> ")
Enter a number.
> 10
>>> n
'10'
```

Интерпретация ввода

```
>>> n = raw_input("Enter a number.\n> ")
Enter a number.
> 10
>>> n
'10'

>>> int(n)
10
>>> float(n)
10.0
```

Списки

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

Списки

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

>>> l[0]
1
>>> l[-2:]
[2, 3]
>>> l + [4, 5]
[1, 2, 3, 4, 5]
>>> len(l)
3
```

Изменение списков

```
>>> l = [1, 2, -1, 4, 5]
>>> l
[1, 2, -1, 4, 5]
>>> l[2] = 3
>>> l
[1, 2, 3, 4, 5]
>>> l.append(6)
>>> l
[1, 2, 3, 4, 5, 6]
```

Другие операции

```
>>> l = [1, 2, 3, 4]
>>> l[1:3] = [0, 0]
>>> l
[1, 0, 0, 4]
>>> l[1:3] = [-1]
>>> l
[1, -1, 4]
```

Другие операции

```
>>> l = [1, 2, 3, 4]
>>> l[1:3] = [0, 0]
>>> l
[1, 0, 0, 4]
>>> l[1:3] = [-1]
>>> l
[1, -1, 4]

>>> len([])
0
>>> l[1:2] = []
>>> l
[1, 4]
```

Вложенные структуры

```
>>> l = ['a', 10]
>>> l[0]
'a'
```

Вложенные структуры

```
>>> l = ['a', 10]
>>> l[0]
'a'


>>> l = [[1, 2], 'hi', [[]]]
>>> l[0][0]
1
>>> l[2][0]
[]
```

Составные операторы

Условный оператор if

```
if x < 0:
    sign = -1
    print "Negative"
elif x == 0:
    sign = 0
    print "Zero"
else:
    sign = 1
    print "Positive"
```

Цикл while

```
x = 4
while x > 0:
    x -= 1
    print "Current value:", x
```

```
Current value: 3
Current value: 2
Current value: 1
Current value: 0
```

Цикл for

```
words = ['cat', 'dog', 'python']
for x in words:
    print x, len(x)
```

```
cat 3
dog 3
python 6
```

Функция range

```
>>> range(3)
[0, 1, 2]
>>> range(2, 7)
[2, 3, 4, 5, 6]
```

Функция range

```
>>> range(3)
[0, 1, 2]
>>> range(2, 7)
[2, 3, 4, 5, 6]

>>> range(2, 7, 2)
[2, 4, 6]
>>> range(5, 1, -1)
[5, 4, 3, 2]
```

for и range

```
for i in range(1, 5):
    print i
```

```
1
2
3
4
```

xrange

```
for i in xrange(1, 5):
    print i,
```

```
1 2 3 4
```

xrange

```
for i in xrange(1, 5):
    print i,
```

```
1 2 3 4
```

```
x = 0
for i in xrange(10**10):
    x += 1
print x
```

```
10000000000
```

xrange

```
>>> xrange(5)
xrange(5)
```

Что это такое разберем позже.
(тема “Итераторы”).

Изменение аргумента for

```
for i in something:
    ...
```

Правило: в `...` пока нельзя менять `something`. (тема “Модель памяти”).

Функции

Определение функций

```
def salute():
    print "Hello, world!"

salute()
```

```
Hello, world!
```

Определение функций

```
def salute():
    print "Hello, world!"

salute()
```

```
Hello, world!
```

```
def salute_string():
    return "Hello, world!"

>>> salute_string()
'Hello, world!'
```

Аргументы функций

```
def plus(a, b):
    return a + b

>>> plus(1, 2)
3
```

Аргументы функций

```
def plus(a, b):
    return a + b

>>> plus(1, 2)
3


>>> plus('abcd', 'qwerty')
'abcdqwerty'
```

Выход из функции

```
def do_smth(n):
    if n > 0:
        return
    else:
        print 'Not positive'
```

```
>>> do_smth(1)
>>> do_smth(-1)
Not positive
```

Другое

Полезные функции

```
>>> max(1, 2, 10)
10
>>> max([1, 2, 10])
10
```

(Аналогично `min`)

Полезные функции

```
>>> max(1, 2, 10)
10
>>> max([1, 2, 10])
10
```

(Аналогично `min`)

```
>>> sum([1, 2, 3])
6
>>> abs(-4)
4
```

Функции строки

```
>>> 'hello'.title()
'Hello'
>>> 'hello'.upper()
'HELLO'
>>> 'Hello'.lower()
'hello'
```

Названия типов данных

Числовые типы: `int`, `float`.

Строка: `str`.

Список: `list`.

Названия типов данных

Числовые типы: `int`, `float`.

Строка: `str`.

Список: `list`.

```
help(str)
help(list.append)
```

Комментарии

```
# this is a comment
x = 1
# x += 1
print x
```

```
1
```