Python: intro

budnyjj@pirates.by

Темы курса

- Базовые типы
- Контейнеры
- Функции
- Классы
- Исключения
- IO

Сегодня

- Обзор языка
- Hello, world!
- Базовые типы данных
- Условный оператор
- Операторы цикла

Особенности языка Python

- Интерпретируемый
- Высокоуровневый
- Динамически типизируемый
- Поддерживает ООП

Версии Python

"Python 2.x is legacy,
Python 3.x is the present and future of the language" [1]

[1] https://wiki.python.org/moin/Python2orPython3

Getting help

- https://docs.python.org
- opennet.ru/docs/RUS/python/
- help()

Hello, world!

- Interactive

```
python
>>> print "Hello, world!"
```

- Script hello.py:

```
1 #!/usr/bin/python2
2 print "Hello, world!"
3
```

```
chmod +x hello.py
./hello.py
```

or

```
python2 hello.py
```

import this

```
python -c "import this"
```

Zen of Python

The Zen of Python, by Tim Peters.
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one — and preferably only one — obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea — let's do more of those!

Идентификаторы

- A-Z, a-z, 0-9, _
- Case sensitive

Стандартные типы данных

- Boolean
- Numeric: int, float, long, complex
- String
- List
- Tuple
- Dictionary[2]

[2] Существуют различия в типах данных в разных версиях Python:
https://docs.python.org/2.7/library/stdtypes.html
https://docs.python.org/3.4/library/stdtypes.html

Базовые типы данных

```
1  #!/usr/bin/python
2
3  logic   = True          # A boolean assignment
4  counter = 103           # An integer
5  miles   = 1000.0        # A floating point
6  cmplx   = 1 + 1j        # A complex
7  name    = "John"        # A string
8
9  print logic, not Logic
10 print counter, counter * miles
11 print miles, miles / counter, miles // counter
12 print cmplx, cmplx.conjugate()
13 print name, "|".join(name)
```

Преобразование типов

```
1  type(x)
2  int(x[, base])
3  long(x[, base])
4  float(x)
5  complex(real[,imag])
6  str(x)
7  repr(x)
8  eval(x)
9  chr(x)
10 unichr(x)
11 ord(x)
```

Операторы

```
+, -, *, /, %, **, //      # arithmetical
==, !=, <, >, >=, <=       # comparison
and, or, not               # logical
in                         # membership
is                         # identity
```

Пара слов про контейнеры

- List

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

- Dict

```
1 d = {"one": 1, "two": 2,}
2 d["three"] = 3
3 print d
4 del(d["two"]
5 print d
6
```

Примечания

- Boolean, Numeric, Complex, String, Tuple are **immutable**
- List, Dictionary are **mutable**
- `'foo' == "foo"`
- Docstring:

```
"""
This is a docstring example.

It is useful mainly for documentation purposes.
"""
```

Условный оператор

```
1  x = int(raw_input("Please enter an integer: "))
2  if x < 0:
3      x = 0
4      print "Negative changed to zero"
5  elif x == 0:
6      print "Zero"
7  elif x == 1:
8      print "Single"
9  else:
10     print "More"
11
```

Оператор цикла for

```
1  words = ["cat", "window", "defenestrate"]
2  for w in words:
3      print w, len(w)
4
5  # range(5)            == [0, 1, 2, 3, 4]
6  # range(3,10)         == [3, 4, 5, 6, 7, 8, 9]
7  # range(3, 10, 2)     == [3, 5, 7, 9]
8  # range(3, 10, -2)    == [9, 7, 5, 3]
9
10 for i in range(len(words)):          # ugly
11     print i, words[i]
12
13 for i, word in enumerate(words):    # much better
14     print i, word
```

Оператор цикла for

```
1 for n in range(2, 10):
2     for x in range(2, n):
3         if n % x == 0:
4             print n, "equals", x, "*", n/x
5             break
6     else:
7         print n, "is a prime number"
8
```

Оператор цикла for

```
1 for num in range(2, 10):
2     if num % 2 == 0:
3         print "Found an even number", num
4         continue
5     print "Found a number", num
6
```

Оператор цикла while

```
1 while True:
2     pass
3
```

import math

```
1 import math
2 dir(math)
3 help(math)
```

Пример решения задачи «найти n-тое число Фибоначчи»[3]

$$F_n = \begin{cases} 0 & \text{при n = 0;} \\ 1 & \text{при n = 1;} \\ F_{n-1} + F_{n-2} & \text{при n > 1.} \end{cases}$$

```
1  def fib(n):
2  """ Return n-th number in Fibonacci sequence. """
3      if n == 0: return 0
4      elif n == 1: return 1
5      else: return fib(n-1) + fib(n-2)
6
7  if __name__ == "__main__":
8      reqNumber = int(raw_input(
9      "Enter index of requested Fibonacci number: "))
10     print("Your number is:", fib(reqNumber))
```

[3] http://stackoverflow.com/questions/494594/how-to-write-the-fibonacci-sequence-in-python

Задача «n-тое число Фибоначчи»

- Что будет, если мы попытаемся вычислить fib(-1)?
- Можно ли оптимизировать процесс вычислений? Как это сделать?

Спасибо за внимание!

Продолжение следует...