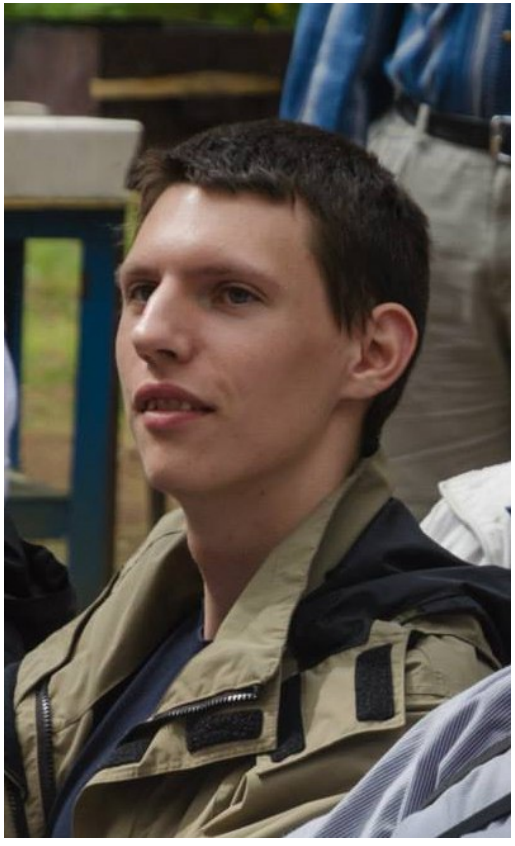
Python: intro

Обо мне



Меня зовут Роман.
Я студент пятого курса БГУИР.
Работаю в Epam Systems.
В свободное время музицирую и играю в настольный теннис.

О языке

Python используется для...

- разработки программного обеспечения
- администрирования компьютерных систем
- научных целей

Python используют Google, IBM, Red Hat, CERN, NASA[1]...
Python есть в каждом дистрибутиве Linux.

[1] Более полный список находится здесь:
https://wiki.python.org/moin/OrganizationsUsingPython

О курсе

Этот курс для тех, кто...

- хочет начать программировать
- хочет освоить еще один язык
- хочет окунуться в мир opensource

Цели курса

По прохождении курса вы...

- изучите синтаксис языка
- будете знать и использовать основные возможности языка
- сможете решать простые прикладные задачи
- будете знать, что делать дальше

Программа курса

- Базовые типы
- Контейнеры
- Функции
- ООП

$\Longrightarrow$

- 8 часов лекций
- задания на дом

Сегодня

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

Python

Python — язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

Название языка

Автор назвал язык в честь британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона».

Особенности языка Python

- Высокоуровневый
- Интерпретируемый
- Динамически типизируемый
- Поддерживает ООП

Версии Python

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

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

Документация

1. help()
2. docs.python.org
3. opennet.ru/docs/RUS/python/

Hello, world!

- REPL:

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

- hello.py:

```
1 #!/usr/bin/env python
2 print("Hello, world!")
3
```

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

python -c "import this"

Красивое лучше, чем уродливое.
Явное лучше, чем неявное.
Простое лучше, чем сложное.
Сложное лучше, чем запутанное.
Плоское лучше, чем вложенное.
Разреженное лучше, чем плотное.
Читаемость имеет значение.
Особые случаи не настолько особые, чтобы нарушать правила.
При этом практичность важнее безупречности.
Ошибки никогда не должны замалчиваться.
Если не замалчиваются явно.
Встретив двусмысленность, отбрось искушение угадать.
Должен существовать один — и, желательно,
только один — очевидный способ сделать это.
Хотя он поначалу может быть и не очевиден, если вы не голландец.
Сейчас лучше, чем никогда.
Хотя никогда зачастую лучше, чем прямо сейчас.
Если реализацию сложно объяснить — идея плоха.
Если реализацию легко объяснить — идея, возможно, хороша.
Пространства имён — отличная штука! Будем делать их побольше!

Литералы

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

Базовые типы данных

- Boolean
- Integer
- Float
- Complex
- String[3]

[3] 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/env 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 float(x)
4 complex(real[,imag])
5 str(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(l)
5 print(l)
6
```

Пара слов про контейнеры

- List

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

- `'foo' == "foo"`
- Comment:

```
# This is a comment example.
```

- Docstring:

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

It is used 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))
```

Функция range

```
1 print(range(5))
2 print(range(3,10))
3 print(range(3, 10, 2))
4 print(range(3, 10, -2))
```

Оператор цикла for

```
1 words = ["cat", "window", "defenestrate"]
2
3 for i in range(len(words)):          # ugly
4     print(i, words[i])
5
6 for i, word in enumerate(words):     # much better
7     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
```

Math module

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

Задача «поиск n-того числа Фибоначчи»

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

Задача «поиск n-того числа Фибоначчи»[4]

```
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 reqNumber = int(raw_input(
8 "Enter index of requested Fibonacci number: "))
9 print("Your number is:", fib(reqNumber))
```

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

Задача «n-тое число Фибоначчи»

- Всегда ли эта программа работает корректно?
- Как оптимизировать процесс вычислений?

Домашнее задание

1. Исправить ошибки
2. Оптимизировать рекурсивную реализацию
3. Написать нерекурсивную реализацию
4. Сравнить производительность

Спасибо за внимание!

- https://github.com/budnyjj/courses_python
- https://vk.com/budnyjj
- budnyjj@gmail.com