



Python Basic

Урок 1

Знакомство с python



Регламент курса

- ✓ Курс состоит из двух частей
- ✓ Каждая часть - 8 уроков по 2 часа
- ✓ Домашние задания
- ✓ Видеозапись
- ✓ Методичка и исходные коды





Цели курса

- Изучить основы Python
- Проникнуться философией python
- Научиться писать правильный python-код
- Набить руку на задачах различной сложности





Зачем нам Python?

- Причины популярности
- Сравнение с другими языками
- Востребованы ли профессионалы, умеющие работать с ним?
- Насколько он сложен в освоении?
- Сфера применения





Вопросы к слушателям

1. Какие цели вы ставите, придя на курс?
2. Какие языки программирования знаете?
3. Работали с данным языком?



План курса

Часть 1	
Урок 1	Знакомство с python
Урок 2	Встроенные типы и операции с ними
Урок 3	Функции и работа с файлами
Урок 4	Полезные инструменты



План курса

Часть 2	
Урок 5	Модули и библиотеки
Урок 6	Объектно Ориентированное Программирование
Урок 7	Интерфейсы
Урок 8	Несколько полезных паттернов





Организационные вопросы

- Пишите в комментарии к уроку. Я буду отвечать на них каждый день;
- Личные сообщения не рекомендуются - обсуждение может быть полезно другим;
- Материалы к уроку буду выкладывать в день урока (самое позднее - на следующий день)





Домашние задания

- Задания в отдельном проекте (ссылка в методичке);
- Старайтесь делать easy и normal задания;
- В начале каждого урока - обсуждаем ДЗ;
- Выполненные задания не позднее 3-ех часов до начала;



Шаг за шагом к вершинам мастерства!



Пройдем этот путь вместе...



Марк Лутц – Изучаем Python

Марк Лутц – Программирование на Python (в 2 т.)

Дэвид Бизли – Python. Подробный справочник



The Zen of Python

`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!



Великодушный
Пожизненный
Диктатор

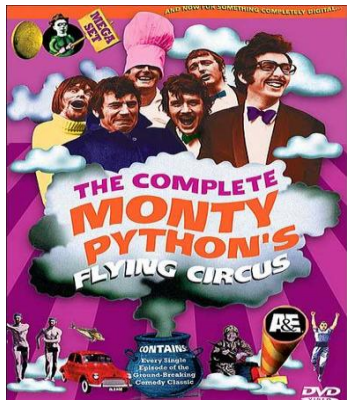


Guido van Rossum



python
1991

Что в имени тебе моем?



ABC
Modula-3



2.x ? 3.x
03.12.2008



Где код?



>>> прямо в интерпретаторе

- IDLE (Windows)
- iPython



>>> exit() # для выхода

> python имяскрипта.py

\$ имяскрипта.py

В начале файла:

`#!/usr/bin/python3`

IDE
там, где удобно и приятно



Notepad++



Sublime Text



Хочу кодить!



Приступим...

строка_кода **# комментарий**

```
''' Многострочный  
Красивый  
Комментарий  
'''
```

Строгая типизация

~~a = 'str' + 1~~

a = 'str' + str(1)

имя_переменной = значение **# объявление переменной**

Полезные функции

- dir()
- help()
- type()

Блоки кода

def func1():

 a = 0

 return a

Отступы

Двоеточие





Типы данных

Изменяемые

Список (`list`) **# [1,2,3,4]**

Словарь (`dict`) **# {1:'a', 'b':True, None:112}**

Множество (`set`) **# {1, True, 'mystr'}**

Неизменяемые

Число (`int, double`) **# 1 2.5**

Строка (`str`) **# 'stroka'**

Булево (`True, False`)

None

Кортеж (`tuple`) **# (1,2,3,4)**

Неизменяемое множество `frozenset`



Переменные

Тип переменной определяется при исполнении

Переменная задается присвоением:

```
a = 1  
b = 'stroka'  
c = True
```

Переменная, которой не задано значение, не существует:

```
>>> z
```

Traceback (most recent call last):

File "<pyshell#30>", line 1, in <module>

z

NameError: name 'z' is not defined



Числа (`int`, `float` и др.)

Числа неограниченной длины

Литералы чисел (некоторые):

<code>1</code>	# целые
<code>1.57</code>	# вещественные
<code>0x6F</code>	# 16-ричная с.с.
<code>3+5j</code>	# комплексные

Операции над числами (основные):

`+` `-` `*` `/` `//` `%` `**` `&` `|` `^`



Строка (str)

Литералы строк:

```
>>> 'stroka'
'stroka'
>>> "tozhe stroka"
'tozhe stroka'
>>> ''' Multi-line
string
'''
' Multi-line\nstring\n'
```

Нет отдельного символьного типа:

's' # строка из одного символа

Операции над строками:

+ *

Строковые методы (некоторые):

find, replace, upper, strip, join, format



Список (list)

Изменяемая последовательность элементов

Списки:

```
>>> [] # пустой список
[]
>>> [1,2,True,None,'stroka',[3,4,5]] # разные типы в списке
[1, 2, True, None, 'stroka', [3, 4, 5]]
```

Операции над списками:

+ * in del

Некоторые методы списков:

append, extend, insert, pop, reverse, sort



Условный оператор

```
if условие_1:  
    код_1  
elif условие_2:  # необязательно  
    код_2  
...  
else:             # необязательно  
    код_n
```

Логические операторы: ==, !=, >, <, <=, >=, (not) is, (not) in

```
if a == 0:  
    print 'Zero!'  
elif a == 1:  
    print 'One'  
else:  
    print 'Don't know'
```

```
if a == 0:  
    print 'Zero!'  
else:  
    print 'Don't know'
```

```
if a == 0:  
    print 'Zero!'
```



Оператор цикла **while**

```
while условие:      # выполняется, пока True
    код_1
else:              # необязательно
    код_2            # если не было break
```

```
x = y // 2           # Для значений y > 1
while x > 1:
    if y % x == 0:    # Остаток
        print y, 'has factor', x
        break        # Перешагнуть блок else
    x -= 1
else:               # Нормальное завершение цикла
    print y, 'is prime'
```



Иструкции **break, continue, pass**

```
break           # ВЫХОД ИЗ ЦИКЛА  
  
continue        # переход к началу цикла  
  
pass            # пустышка/заглушка
```

```
a = 8  
while True:  
    a = a - 3  
    if a > 0:  
        continue    # продолжить  
print 'Finish'
```

```
while True:  
    a = input('Enter number:')  
    if a == 0:  
        break        # выход  
print 'Finish!'
```

```
if a == 0:  
    pass # пустая инструкция  
else:  
    b = c / a
```



PEP 8

<https://www.python.org/dev/peps/pep-0008/>

<http://pep8.ru/doc/pep8/>

булево ~~==~~ **True**
булево ~~==~~ **False**

отступ



строка_кода...
79



is (is not)

None

~~=~~

имена: ~~! Q !~~

имя модуля
ИмяКласса
имя_функции
имя_метода
_внутри_метод
УХ_КОНСТАНТА
имя_переменной

func1
... } 2 строки
...
func2

1 пробел

a = b + c - d * e / f

между операторами



Для пользы дела



Python Tips, Tricks, and Hacks -

http://idzaaus.org/static/files/articles/Python_Tips,_Tricks,_and%20Hacks_%28rus%29.pdf

100 вопросов по Python на собеседованиях - <https://www.dezyre.com/article/100-data-science-in-python-interview-questions-and-answers/188>

PyTricks - <https://github.com/brennrm/PyTricks>



Видео-лекции

<http://programming086.blogspot.ru/2015/12/python-2015.html> - Сергей Лебедев

<http://www.youtube.com/playlist?list=PLPErILqzuTQqXEIjiN6gwFzV1yRuwReR0> - Георгий Курячий

