# Python

осень 2016
“техническое” занятие + введение

Антон Алексеев // ПОМИ, СофИТ лабс
anton.m.alexeyev+hse@gmail.com

print(“Hello HSE!”)

# История Python

- Guido van Rossum - Benevolent Dictator For Life
- Язык ABC
- Распределённая ОС Amoeba
- 1991 - анонс
- The Zen of Python

# Python и мир

- много реализаций: PyPy, IronPython, Jython, Python.NET,...
- **CPython** вместо стандарта
- что такое PEP
- PEP8

# Основные характеристики

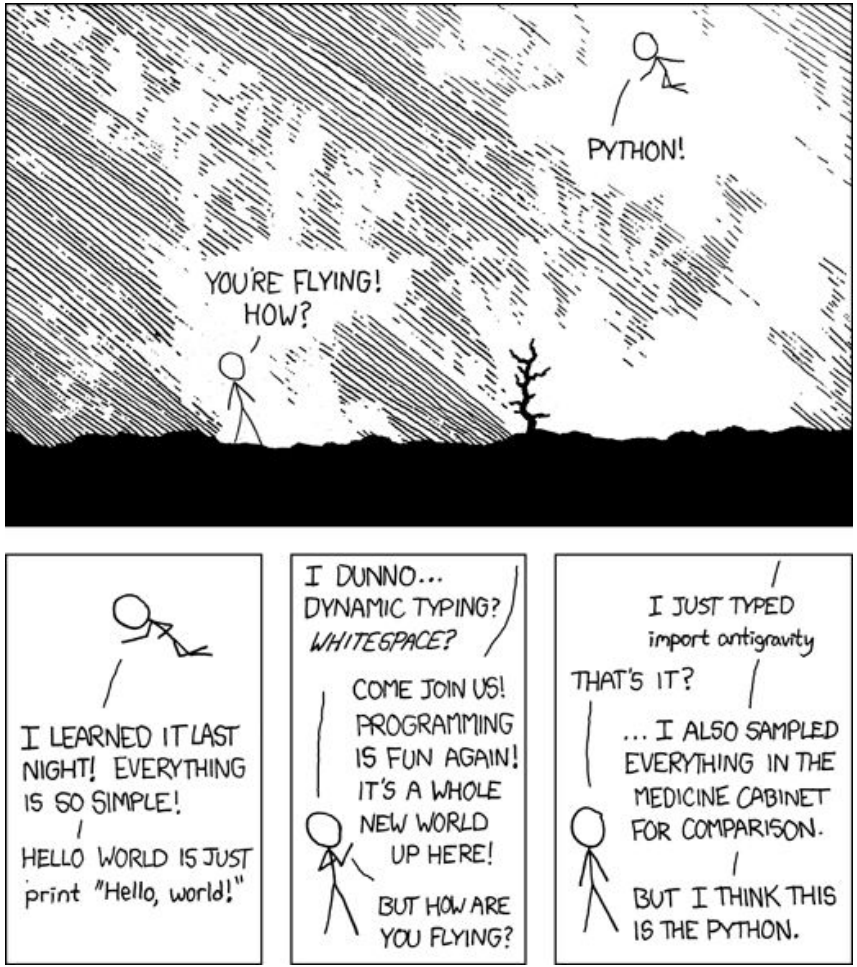
- Динамически типизированный
- Интерпретируемый, исполняется построчно (ну, почти), REPL
- Сборка мусора
- Минималистический синтаксис
- Отступы!
- ООП, лямбды и прочая красота

# Python и наука или Зачем мы собрались

- Прост в изучении
- Просто поддерживать, т.к. сложно написать непонятно
- “Платформонезависим”
- Легко прикрутить библиотеку на C/C++
- Много *богатых* библиотек и инструментов
- Большое сообщество учёных-питонистов

# Культурные особенности

- Назван в честь “Летающего цирка Монти Пайтона”
- Обучающие примеры с цитатами из шоу
- Холивар **Python** vs **R**
- Шутки… тысячи их!


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

# Let’s get our hands dirty: REPL

$ python3 --version

$ python3

>> import this

>> import numpy

>> numpy.__version__

>> print(numpy.__version__)

# Let’s get our hands dirty: exec file

Рекомендую Notepad++

$ python3 your_file.py

```
# -*- coding: utf-8 -*-

print(“Python, baby, do you speak it?”)
```

# Переменные, именование

a = 42

~~abstractMagicSingletonProxyFactoryBean~~

abstract_magic_singleton_proxy_factory_bean

дороти3

# Простые типы: bool

```
>>> print("True") if 0 < 1 else print("False")
True
>>> print("True") if None else print("False")
False
>>> print("True") if 2128506 else print("False")
True
>>> print("True") if -1 else print("False")
True
>>> print("True") if 22.4 else print("False")
True
```

```
>>> print("True") if 0.0 else print("False")
False
>>> print("True") if "" else print("False")
False
>>> print("True") if " " else print("False")
True
>>> print("True") if [] else print("False")
False
>>> print("True") if [1] else print("False")
True
```

*короче это можно проверить так:
>>> print(bool([]))
False

# Простые типы: bool - testing

```
if x == 12 or y <= 15 and not z > 20 and not w:
  print(“condition satisfied”)
elif x < 12 < y :
  print(“second condition satisfied”)
else:
  print(“not a single condition was satisfied that day”)
```

# Простые типы: int, float, long

| Operation | Result |
| --- | --- |
| `x + y` | sum of *x* and *y* |
| `x - y` | difference of *x* and *y* |
| `x * y` | product of *x* and *y* |
| `x / y` | quotient of *x* and *y* |
| `x // y` | floored quotient of *x* and *y* |
| `x % y` | remainder of `x / y` |
| `-x` | *x* negated |
| `+x` | *x* unchanged |

# Простые типы: int, float, long

| | |
|---|---|
| `abs(x)` | absolute value or magnitude of *x* |
| `int(x)` | *x* converted to integer |
| `float(x)` | *x* converted to floating point |
| `complex(re, im)` | a complex number with real part *re*, imaginary part *im*. *im* defaults to zero. |
| `c.conjugate()` | conjugate of the complex number *c* |
| `divmod(x, y)` | the pair `(x // y, x % y)` |
| `pow(x, y)` | *x* to the power *y* |
| `x ** y` | *x* to the power *y* |

# *Простые типы, для целочисленных

| Operation | Result | Notes |
|---|---|---|
| `x \| y` | bitwise *or* of *x* and *y* | |
| `x ^ y` | bitwise *exclusive or* of *x* and *y* | |
| `x & y` | bitwise *and* of *x* and *y* | |
| `x << n` | *x* shifted left by *n* bits | (1)(2) |
| `x >> n` | *x* shifted right by *n* bits | (1)(3) |
| `~x` | the bits of *x* inverted | |

# Цикл for, минимальный пример

range(from, to, step)

range(from, to) ⇔ range(from, to, 1)

range(to) ⇔ range(0, to, 1)

```
for i in range(from, to, step):
    print (i)
```

# Пример программы: факториал

```
# fact(k) = 1 * … * k, fact(0) = 1
# => fact(i) := fact(i - 1) * i при i > 1

n = 3
accumulator = 1

for i in range(1, n + 1):
  accumulator = accumulator * i

print(accumulator)
```

# Напутствие

- RTFM: Read The Fantastic Manuals!
- Zen of Python
- PEP8

# Рекомендуемые материалы

- Dive into Python (3), M.Pilgrim
  http://www.diveintopython3.net/
  http://ru.diveintopython.net/
- RT[F]M!
  https://docs.python.org/

# Python

осень 2016
“техническое” занятие + введение

# *Only happy when it’s complicated

```
# -*- coding: utf-8 -*-
"""
    Комментарии к нашему модулю
"""
import argparse # подключение библиотеки

def add_indent(text, indent):
    """
        Комментарии к функции
    """
    return "_" * indent + text

if __name__ == '__main__':  # __name__ - имя этого файла без пути к нему или __main__
    parser = argparse.ArgumentParser()
    parser.add_argument("--text", type=str, default="", help="[default:''] text for output")
    parser.add_argument("--indent", type=int, default=0, help="[default:0] indentation")
    args = parser.parse_args()

    # комментарии в теле программы
    print(add_indent(args.text, args.indent))
```

```
$ python3 indent.py --text ”I’m an alligator” --indent 3
```

# *Упражнение 1

Числа Фибоначи (1,1,2,3,5,8,13,...), определение:
f(i) = f(i - 1) + f(i - 2)
f(0) = 1, f(1) = 1

**Дано**

fibo_number_id

**Требуется**

0 - используя цикл for, вычислить fibo_number_id-ое число Фибоначчи
1 - воспользоваться argparse и подавать на вход fibo_number_id как аргумент [см. пример выше], убедиться, что работает без выпадения питоновских ошибок, в том числе на некорректном вводе
2 - обернуть “логику” вычисления i-го числа Фибоначчи в функцию [см. пример выше]

# *Реализации

- CPython
  - ‘default python’
  - здорово и быстро
  - вместо стандарта
- Jython
  - компилируется в байт-код JVM
  - интеграция с Java
- IronPython
  - компилируется под .NET / Mono
- PyPy
  - питон на питоне

# *Если вам всё же нужны .exe-шники...

- Заворачивайте в них код вместе с PVM!
  - py2exe
  - PyInstaller
  - freeze
- Но вам не нужны *.exe-шники