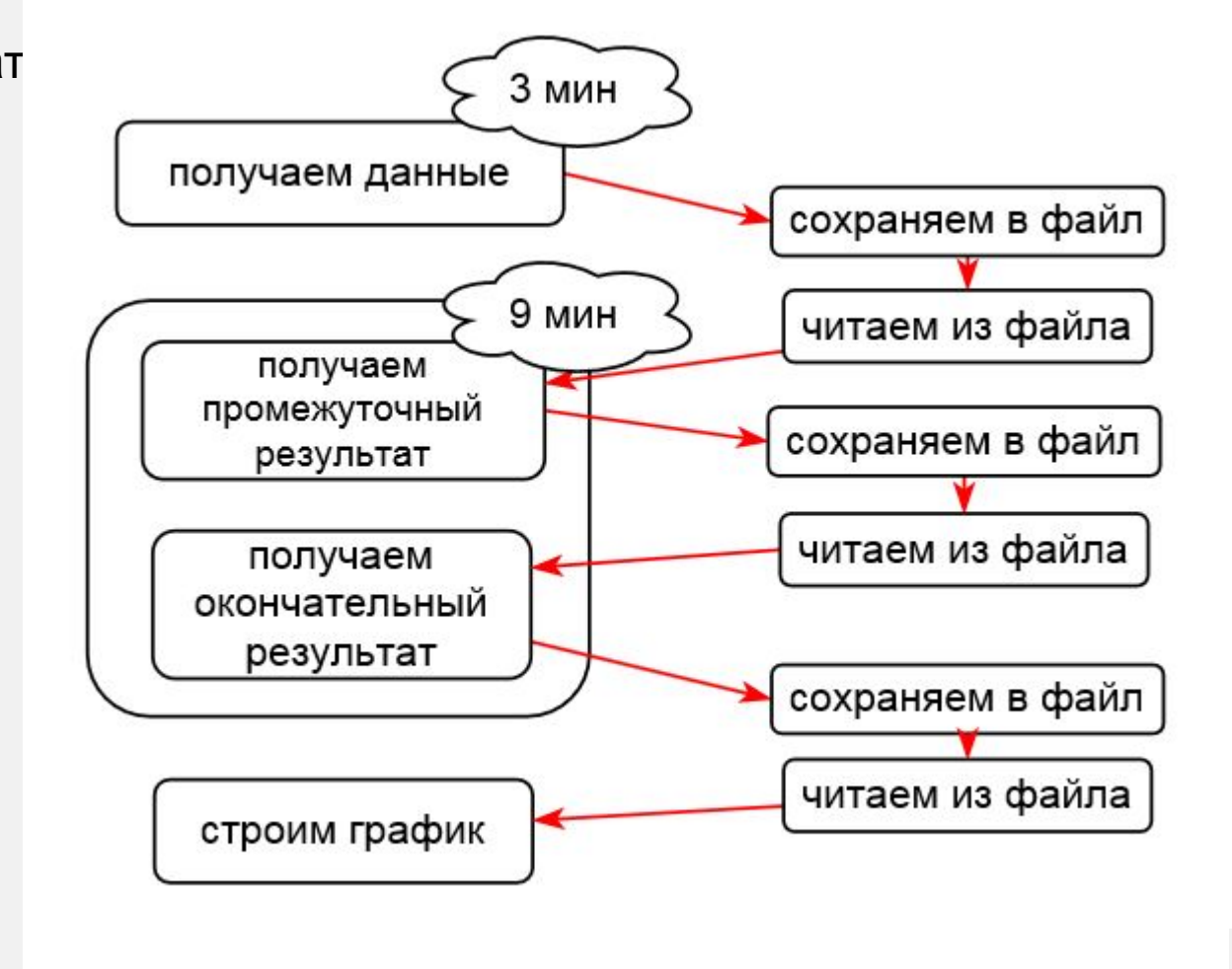
Язык питон в научных вычислениях

Максимов Лев Викторович

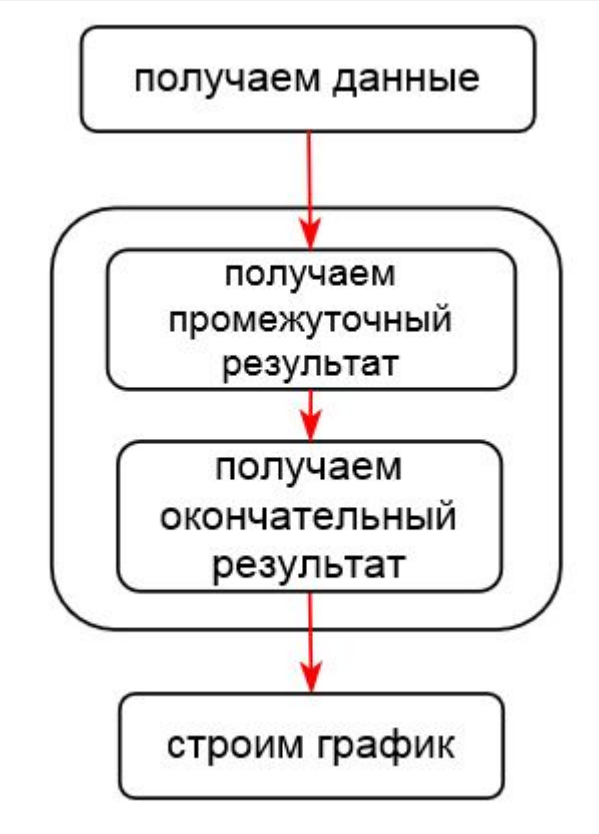
Интерпретатор vs компилятор

- **Получаем** данные
- **Обрабатываем** их
- **Отображаем** результат



Интерпретатор vs компилятор

- **Получаем** данные
- **Обрабатываем** их
- **Отображаем** результат



Интерпретатор vs компилятор

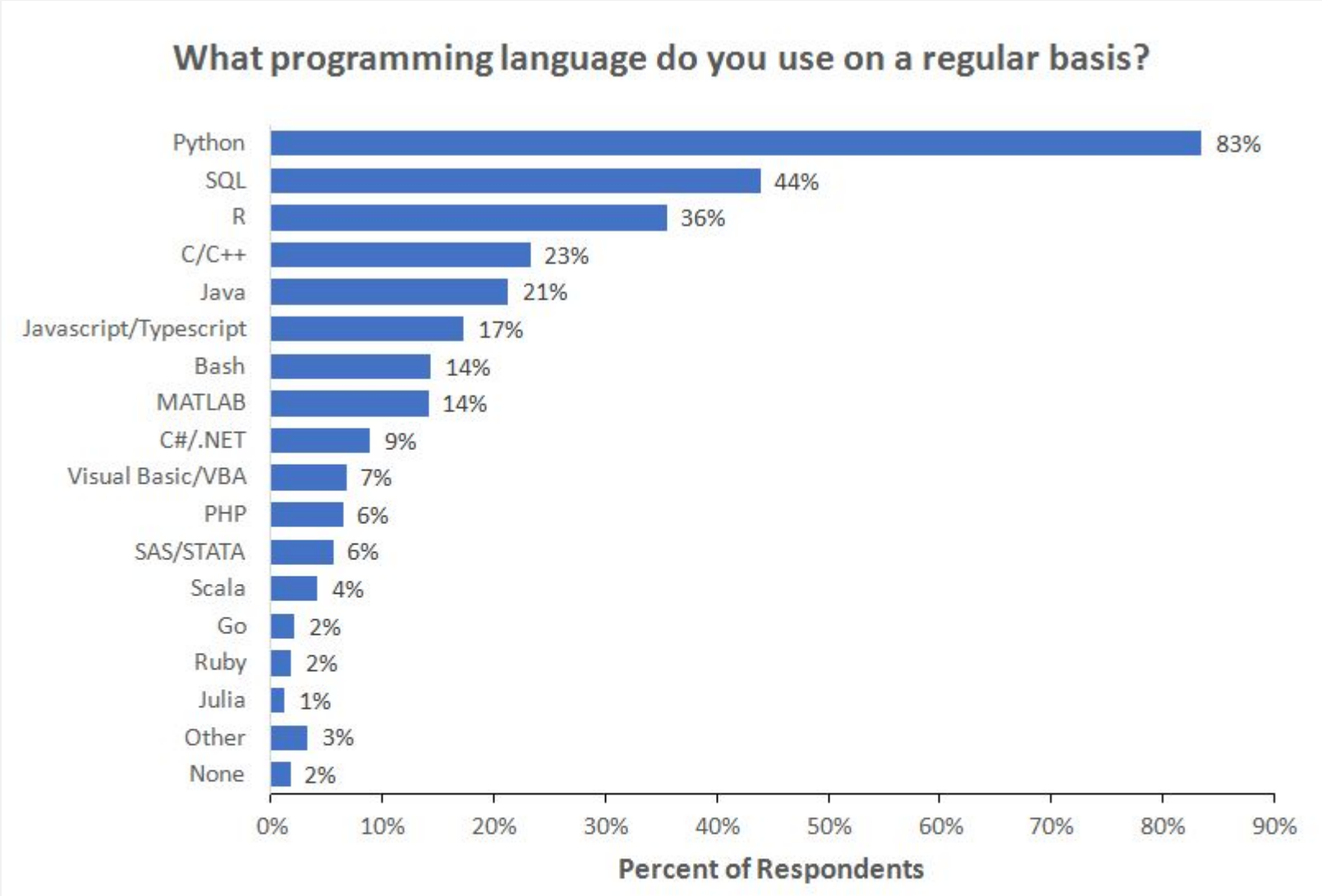
Компиляторы

- C / C++
- Fortran
- LabView
- Java

Интерпретаторы

- Matlab
- Mathematica
- R
- Python

Языки для machine learning и data science



Опрос Kaggle, 2018

Популярность языков программирования на stackoverflow

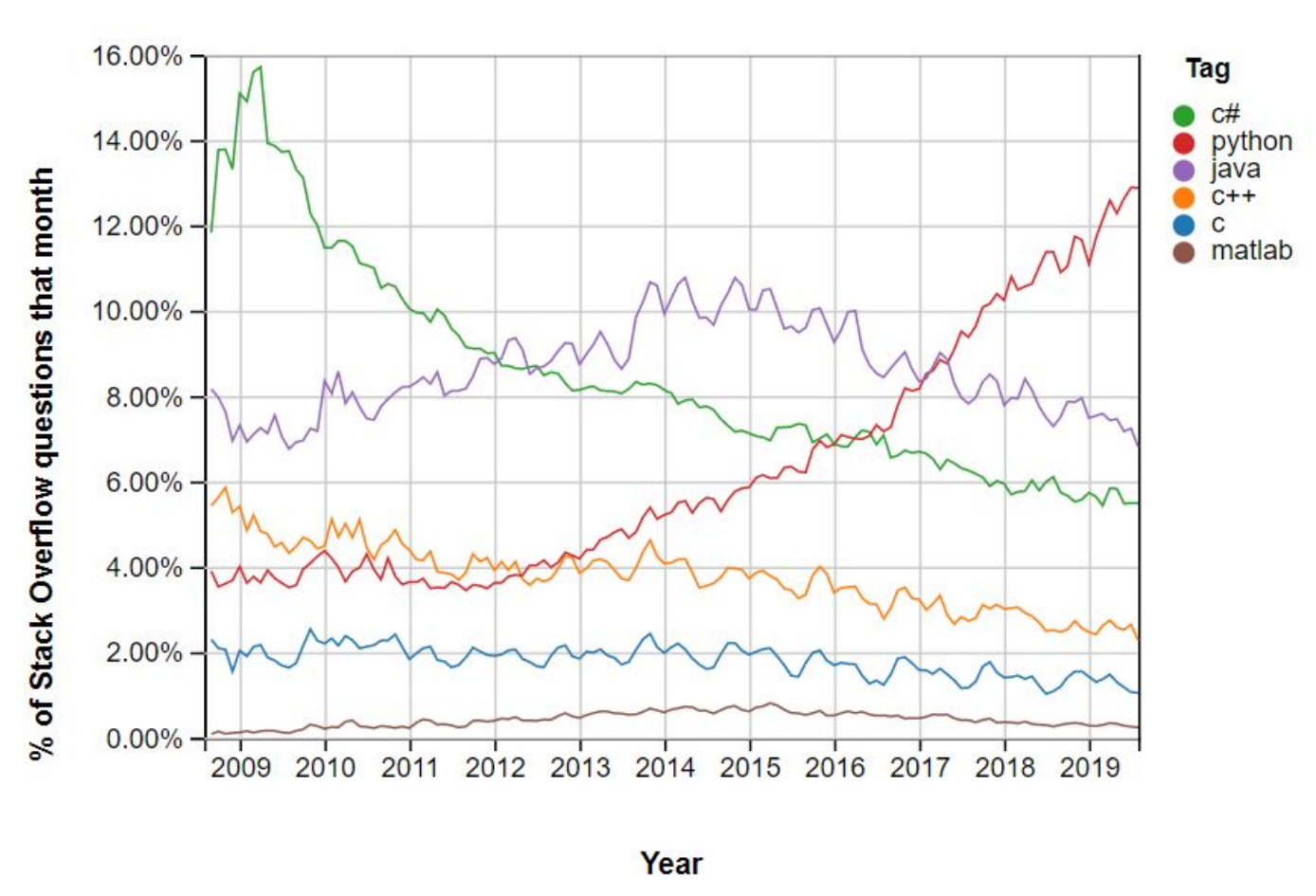


Диаграмма популярности языков программирования

Популярность на **Stack Overflow** (по кол-ву тегов)

Популярность на **GitHub** (по кол-ву проектов)

Языки для научного программирования

= языки, на которых удобно работать с формулами и матрицами:

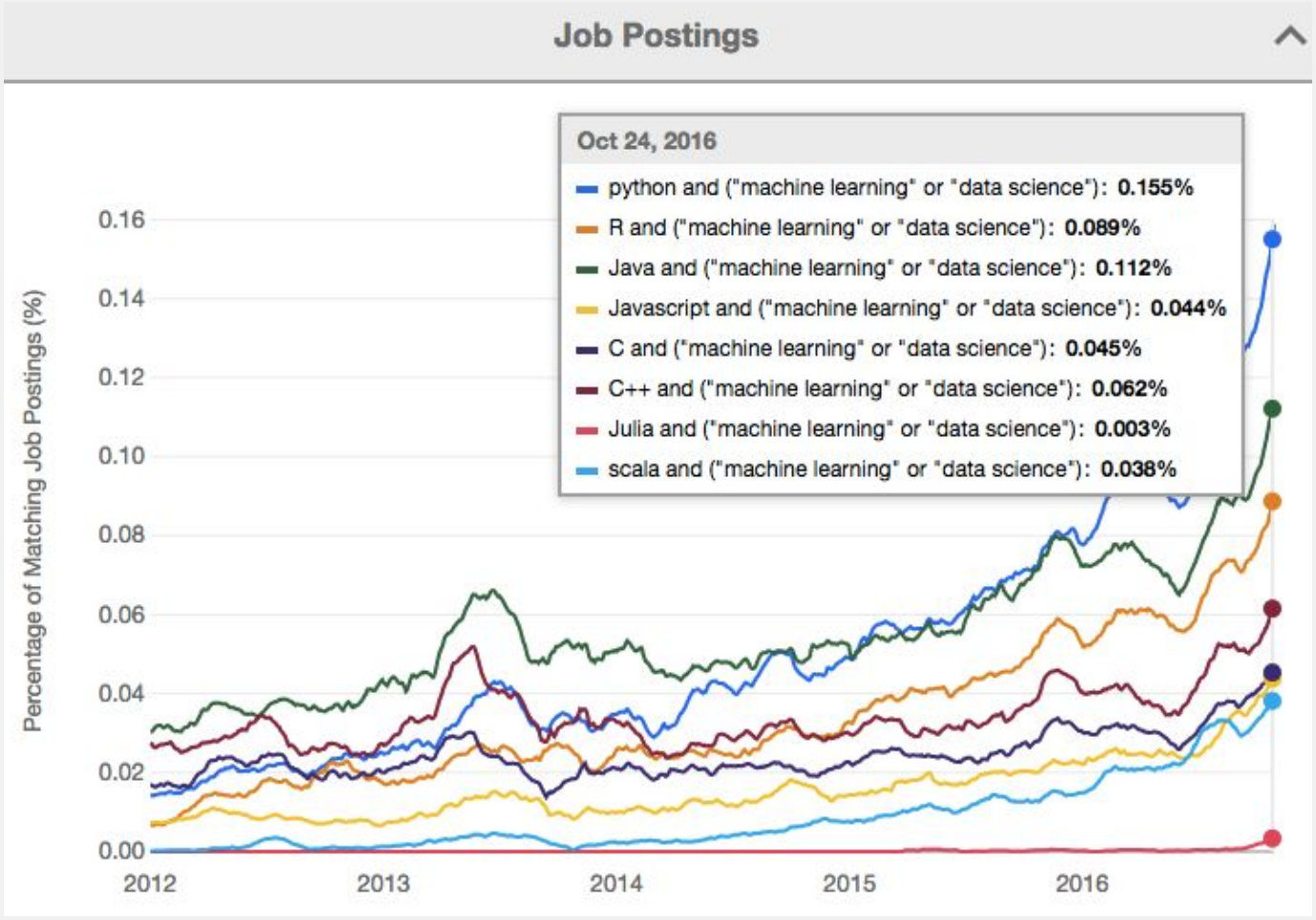
- исторически это:
  - 1957 Algol (algorithmic language) на нем публиковали новые алгоритмы
  - 1958 **Fortran** (formula translator) быстро считает формулы
  - 1964 APL (a programming language) матрицы
  - 1972 **C,** 1983 **C++** системное программирование

- на сегодня это:
  - 1984 **Matlab** (matrix laboratory) матрицы, графики
  - 1991 **Python**, универсальный скриптовый язык
  - 1993 **R**, статистика и графики
  - 1995 **Java**, «идёт на всём»
  - 2012 **Julia**, синтез matlab и python

- узкоспециализированные:
  - 1986 **LabView** железо, визуализация
  - 1988 **Mathematica**, интегралы, уравнения

Кто использует python в работе

* **физики, биологи, экономисты**, для вычислений, вместо Matlab/Mathcad:
  – numpy операции с матрицами, scipy алгоритмы обработки данных



Кто использует python в работе

* **физики, биологи, экономисты**, для вычислений, вместо Matlab/Mathcad:
  – numpy операции с матрицами, scipy алгоритмы обработки данных

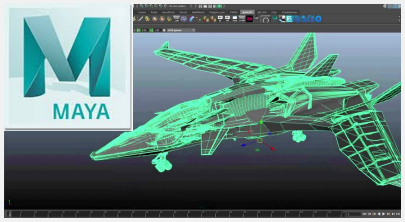
* **веб-разработчики**, фреймворки для создания сайтов, online игр:
  – django, универсальный, наиболее популярный фреймворк на python

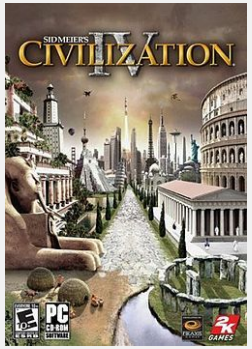
Instagram Dropbox Pinterest

* **системные администраторы**, скрипты для автоматизации своей работы:
  – fabric, выполнение программы на множестве компьютеров сразу

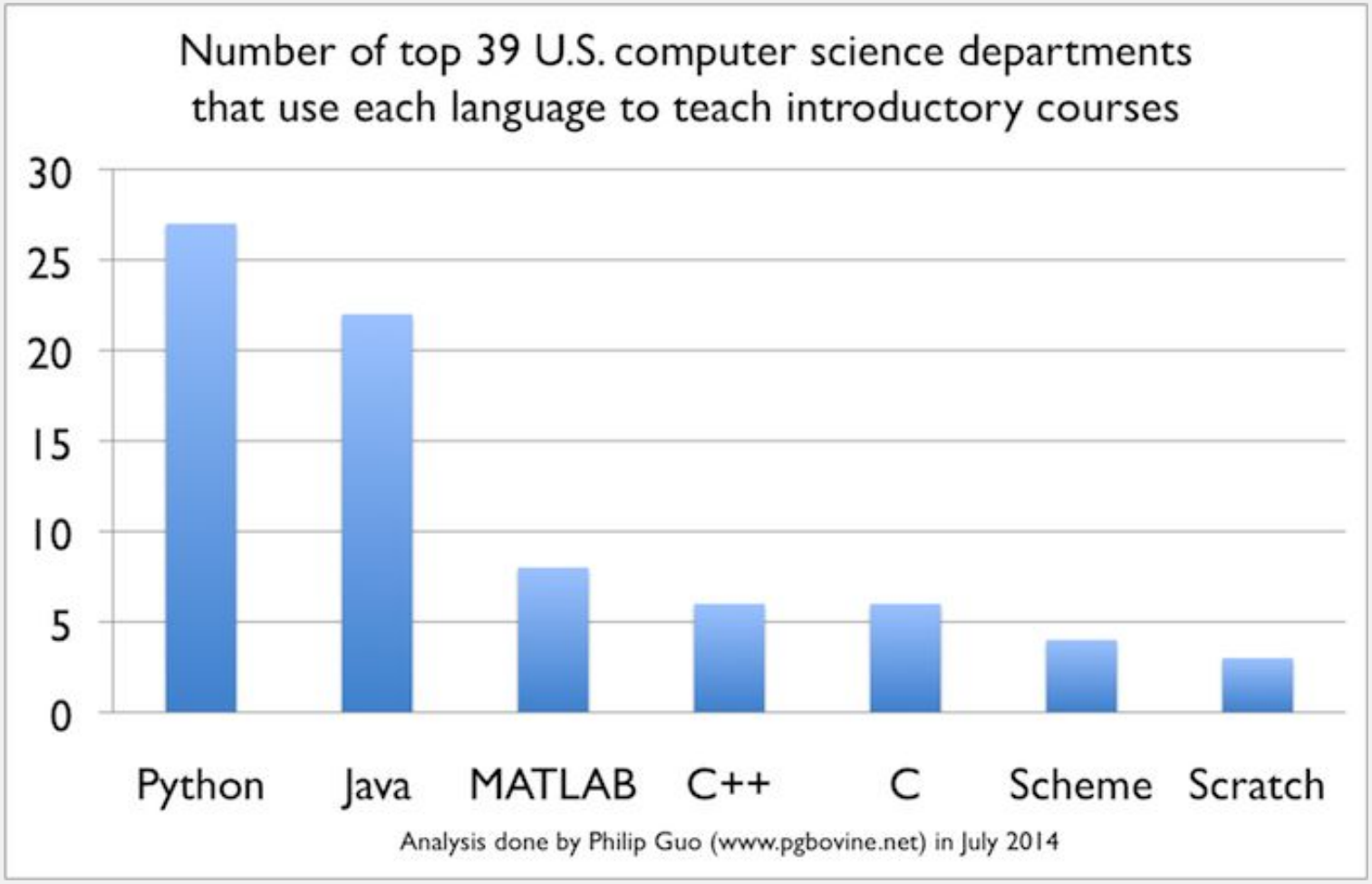
* **программисты**, для создания gui-приложений; в качестве языка макросов:
  – есть биндинги для qt, wx, gtk виджетов

GIMP WORLD OF TANKS

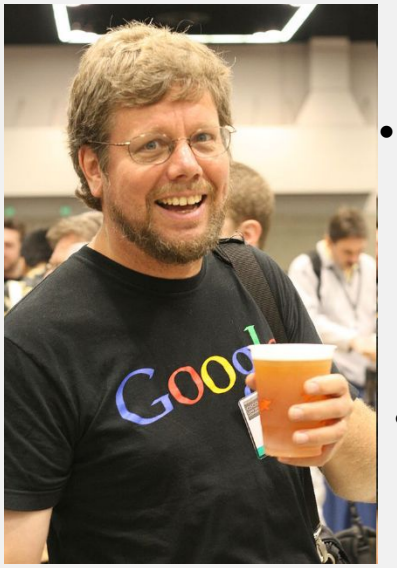




Язык курса «Введение в программирование»



История развития python



- **Гвидо ван Россум** создал python в **1991**
  - начал работать над ним в 1989
  - любитель скетчей Monty Python Flying Circus
  - на 1991 год в python уже были классы и наследование

- **python 2.0** вышел в 2000 (garbage collector, unicode support), стал community-backed.
  - все версии 2.x обратно совместимы

C++

```
for(int i=0; i<10; i++)
    printf("%d", i);
```

python2

```
for i in range(10):
    print("%d" % i)
```

python2

```
for i in xrange(10):
    print("%d" % i)
```

- **python 3.0** вышел в 2008: сброс «балласта» (местами обратно-несовместимое упрощение синтаксиса за счет избавления от устаревших конструкций, накопленных по пути с v2.0 до v2.7)

python3

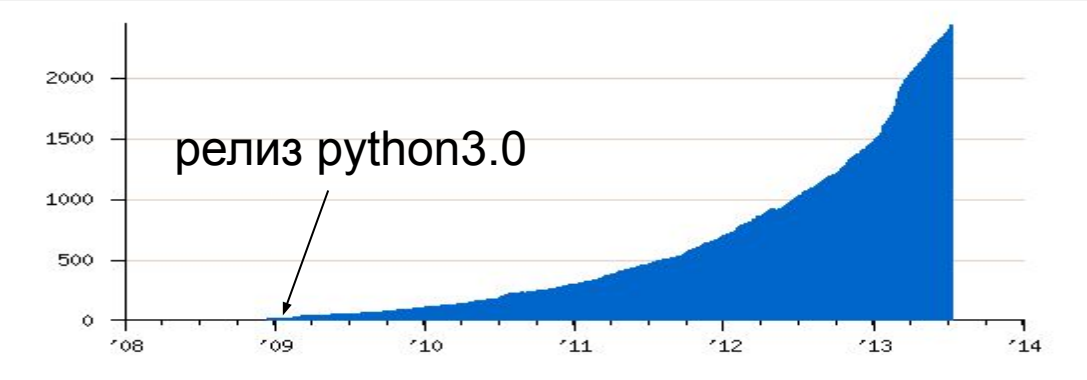
```
for i in list(range(10)):
    print("%d" % i)
```

python3

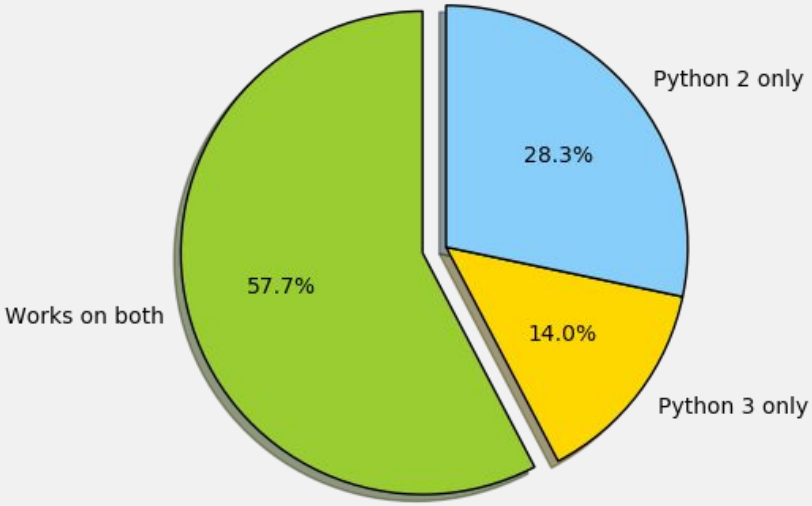
```
for i in range(10):
    print("%d" % i)
```

Python 2 vs Python 3

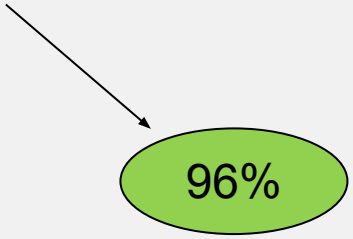
- Кол-во библиотек под python3:



- На сегодня:
  - Из 6000 «стабильных» пакетов:



  - под python3 работают 345 из 360 самых скачиваемых пакетов



- Поддержка python2 заканчивается 31.12.2020

Readability counts: пробелы

C/C++

```
if (a>b){
    printf("a
больше\n");
    printf("чем b\n");
}
printf('вот так\n')
```


python

```
if a>b:
____print('a
больше')
____print('чем b')
print('вот так')
```

- Стандарт PEP8 (pep = python enhancement proposal):
  - **Spaces** are the preferred indentation method over **tabs**.
  - Use **4 spaces** per indentation level.
  - В python2 допускаются (но не приветствуются) **смешанные** отступы, при этом 1 Tab = 8 пробелов. В python3 они запрещены.

- Настройте ваш редактор так, чтобы по нажатию Tab вставлялось 4 пробела

Readability counts: пробелы

C/C++

```
if (a>b){
    printf("a
больше\n");
    printf("чем b\n");
}
printf('вот так\n')
```


python

```
if a>b:
␣␣␣␣print('a
больше')
␣␣␣␣print('чем b')
else:
␣␣␣␣print('b
больше')
print('вот так')
for a in range(100):
␣␣for b in range(100):
␣␣␣␣for c in range(100):
␣␣␣␣␣␣for d in range(100):
␣␣␣␣␣␣␣␣for d in range(100):
␣␣␣␣␣␣␣␣␣␣print(a+b*c+d*e)
```

Readability counts: syntactic sugar

c++	matlab	python

`if (x=>0 && x<10)`
`  printf("one digit");`
	`if x >= 0 && x < 10,`
	`  disp 'one digit';`
	`end`
		`if x => 0 and x < 10:`
		`  print('one digit')`
		`if 0 <= x < 10:`
		`  print('one digit')`

`if (x==1 || x==2 ||`
`  x==3 || x==5)`
`  printf("prime");`
	`if any([1, 2, 3, 5]==x),`
	`  disp 'prime';`
	`end`
		`if x in (1, 2, 3, 5):`
		`  print('prime')`

`m[4][3][5]`	`m(4, 3, 5)`	`m[4, 3, 5]`
`c++ (либо ++c либо c+=1)`	`c = c + 1`	`c += 1`
`h=a; a=b; b=a`	`h=a; a=b; b=a`	`a, b = b, a`

Динамическая типизация

Полиморфные функции

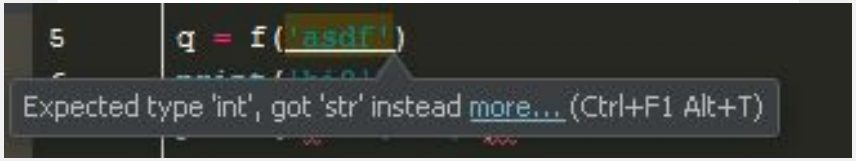
```
def f(a, b):
    return a + b
```

```
> f(3, 4)
7
> f('car', 'toon')
'cartoon'
```

«duck typing»

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck”

```
def f(a: int, b: int) -> int:
    return a + b
```



Изменение класса «на лету»

```
class A(object):
    def f(self, x):
        return x*2

def g(self, y):
    return y*3

a = A()
print a.f(10)
A.f = g
print a.f(10)

20
30
```

«monkey patching»

guerrilla patch
gorilla patch
monkey patch

- тестирование
- исправление сторонних библиотек

без модификации их кода

Способы ускорения python кода

Pypy – альтернативная реализация python с JIT (just-in-time compiling)
Numba – библиотека для JIT-оптимизации
Cython – компилятор, гибридный синтаксис между C/python
Ctypes, cffi – интерфейс к .dll / .so файлам

Достоинства и недостатки python

- **низкий порог** вхождения
- грамотно спроектирован
- легко читаемый **синтаксис**
- наличие огромного количества **библиотек** с кодом на любой случай жизни
- **переносимость**: Windows, Linux, MacOS, Arduino, Raspberry Pi и пр.
- **скорость** исполнения:
  - критические по времени исполнения функции можно ускорять
  - большая часть библиотек (numpy, scipy) уже ускорены

Структура курса

- Введение в питон:
  - базовые типы,
  - ООП,
  - исключения,
  - модули,
  - ввод-вывод,
  - регулярные выражения

- Пакеты для научных вычислений:
  - **numpy**, **sympy**: матрицы, формулы
  - **matplotlib**, **bokeh**: визуализация
  - интегралы, уравнения: **sympy**
  - стат.обработка данных **pandas**
  - арифметика произв.точности **mpmath**
  - минимизация ф-й **iminuit**
  - создание gui приложений **pyqt5**

- Технические вопросы:
  - как пользоваться документацией
  - «графичская консоль» **jupyter**
  - виртуализация (virtualenv, conda create)
  - ср-ва отладки (ipdb, пр.),
  - фреймворки для тестирования (nose, **pytest**)
  - профилирование кода
  - ускорение кода (cython, pypy, **numba**)

Страница курса

vk.com/python_nsu_2019

Установка python

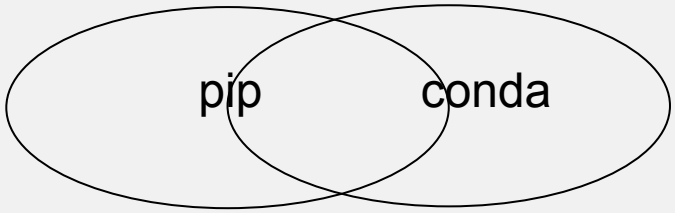
• «просто» питон: python.org

pip install jupyter

• «научный» питон: anaconda.com

conda install jupyter

GUI



• Anaconda - «всё включено», дистрибутив >1Gb

• Miniconda – только сам питон и conda

«Пуск»: conda install menuinst
GUI: conda install anaconda-navigator