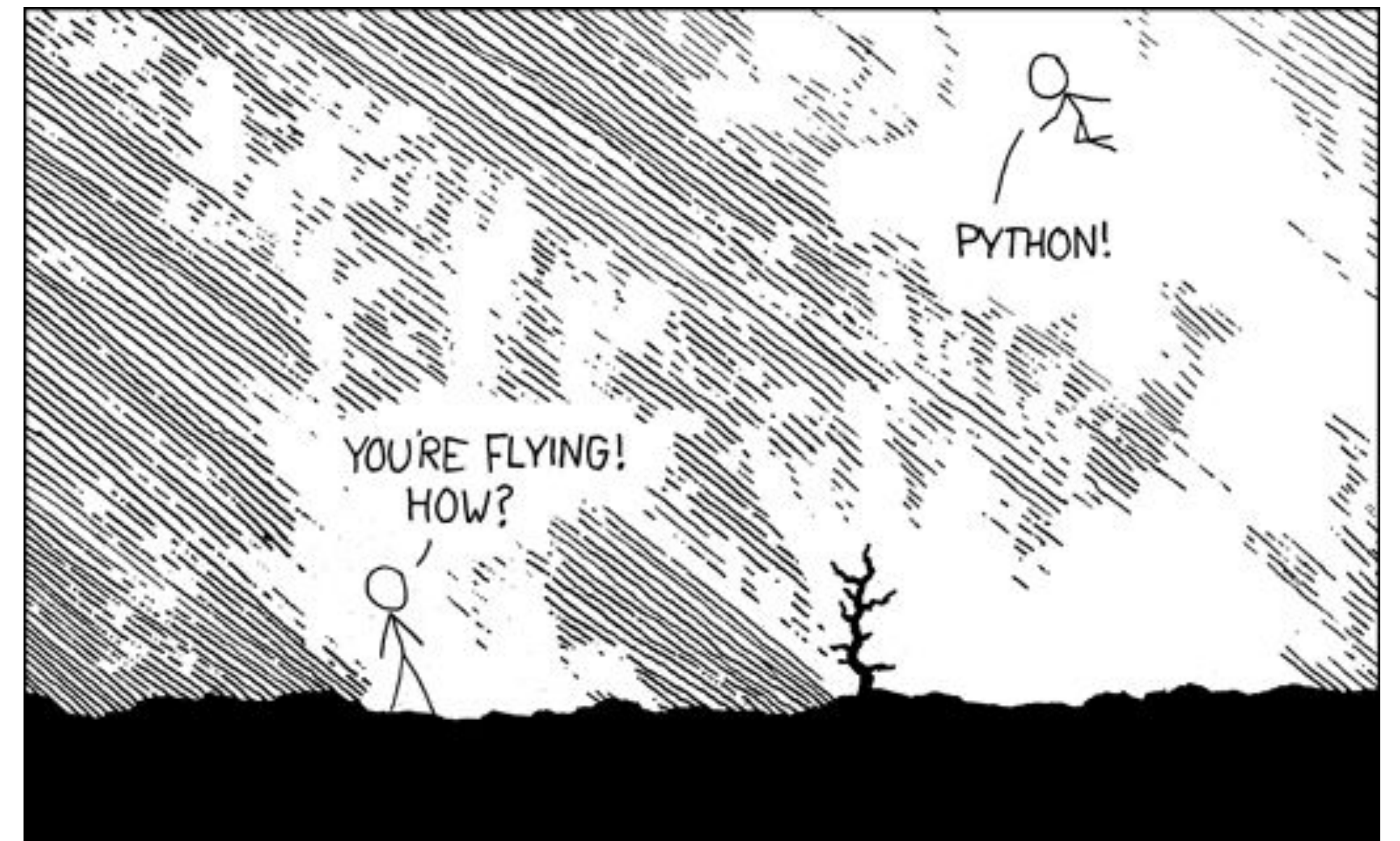


Python Scripting - Part 1

Spring 2022
PCfB Class 4
February 4, 2022

webs



Outline

- Why Python?
- Data types
- Variables
- Methods

Why Python?

Enhanced readability

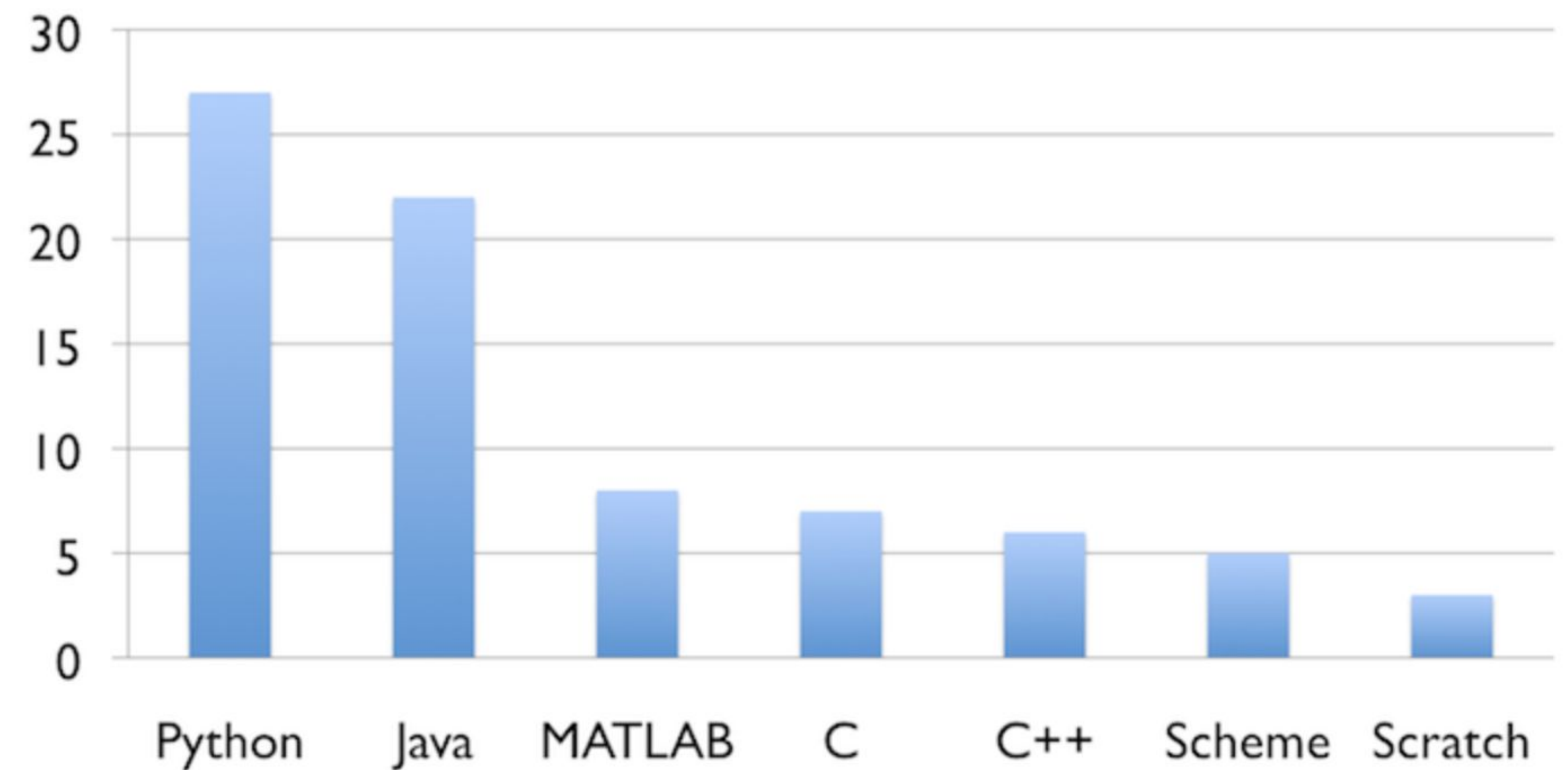
PYTHON

```
print('hello world')
```

JAVA

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("hello world");  
    }  
}
```

Number of top 39 U.S. computer science departments that use each language to teach introductory courses



Analysis done by Philip Guo (www.pgbovine.net) in July 2014, last updated 2014-07-29

Still very powerful

NETFLIX

Google



Dropbox



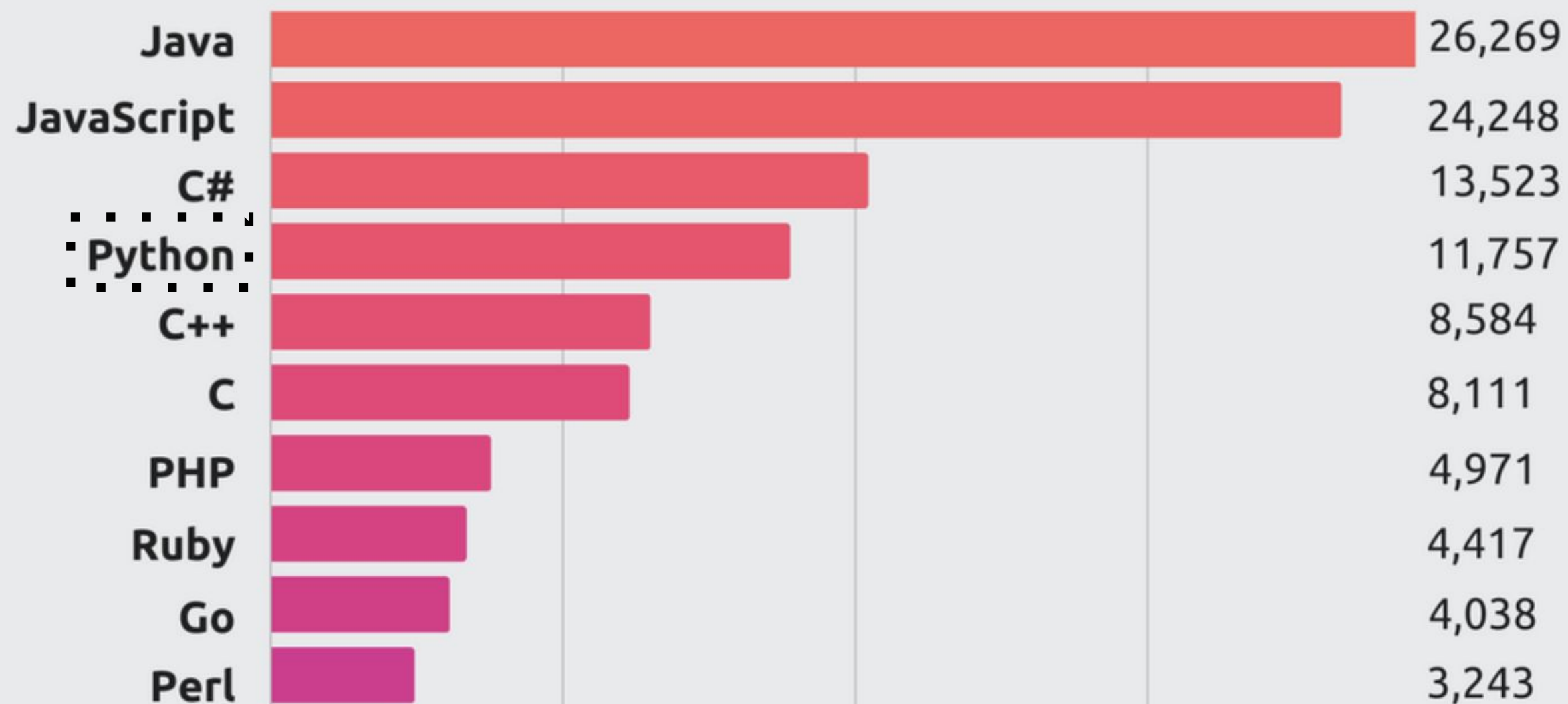
Instagram



Very popular

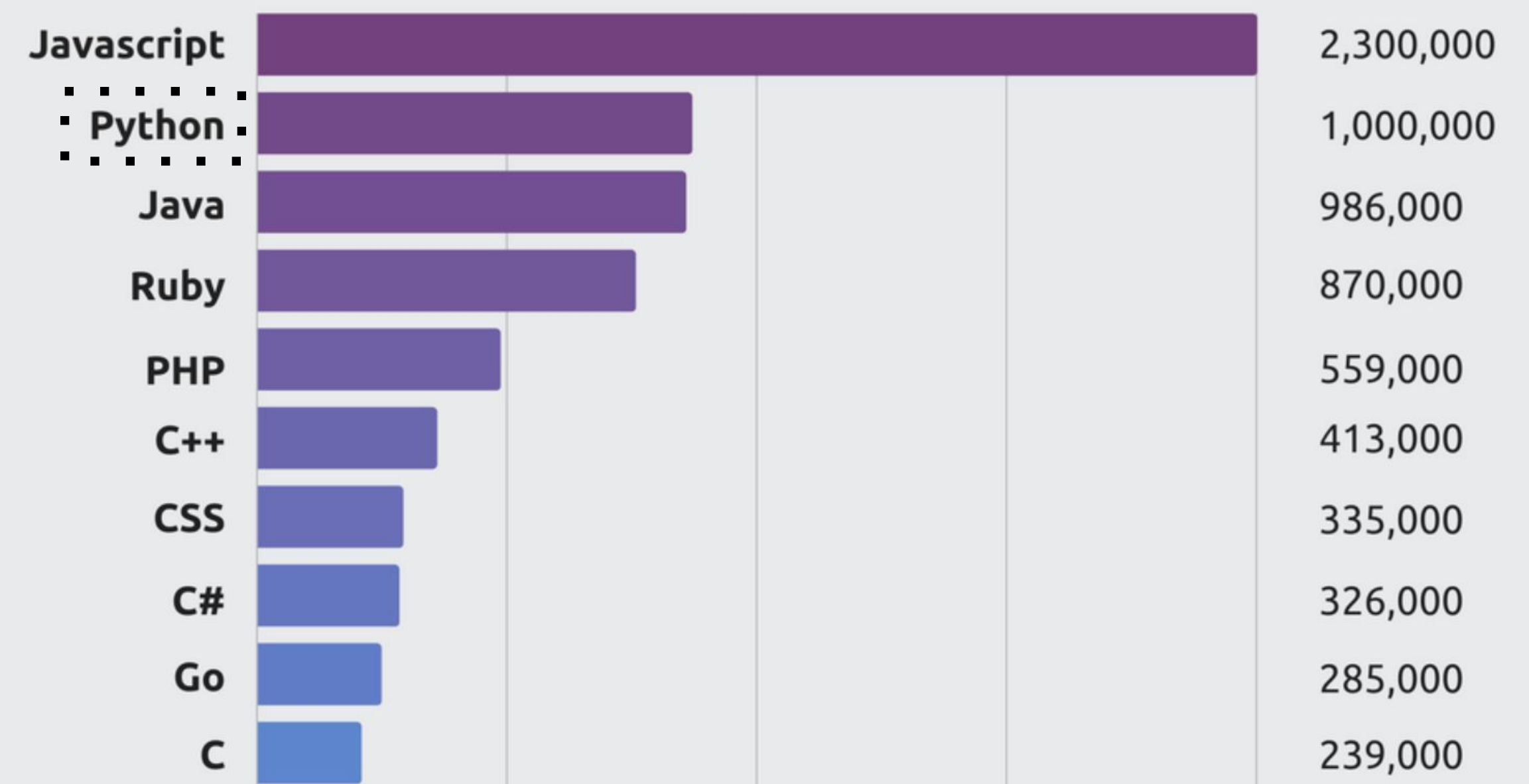
Most In-Demand Languages

Indeed Job Openings - Dec. 2017



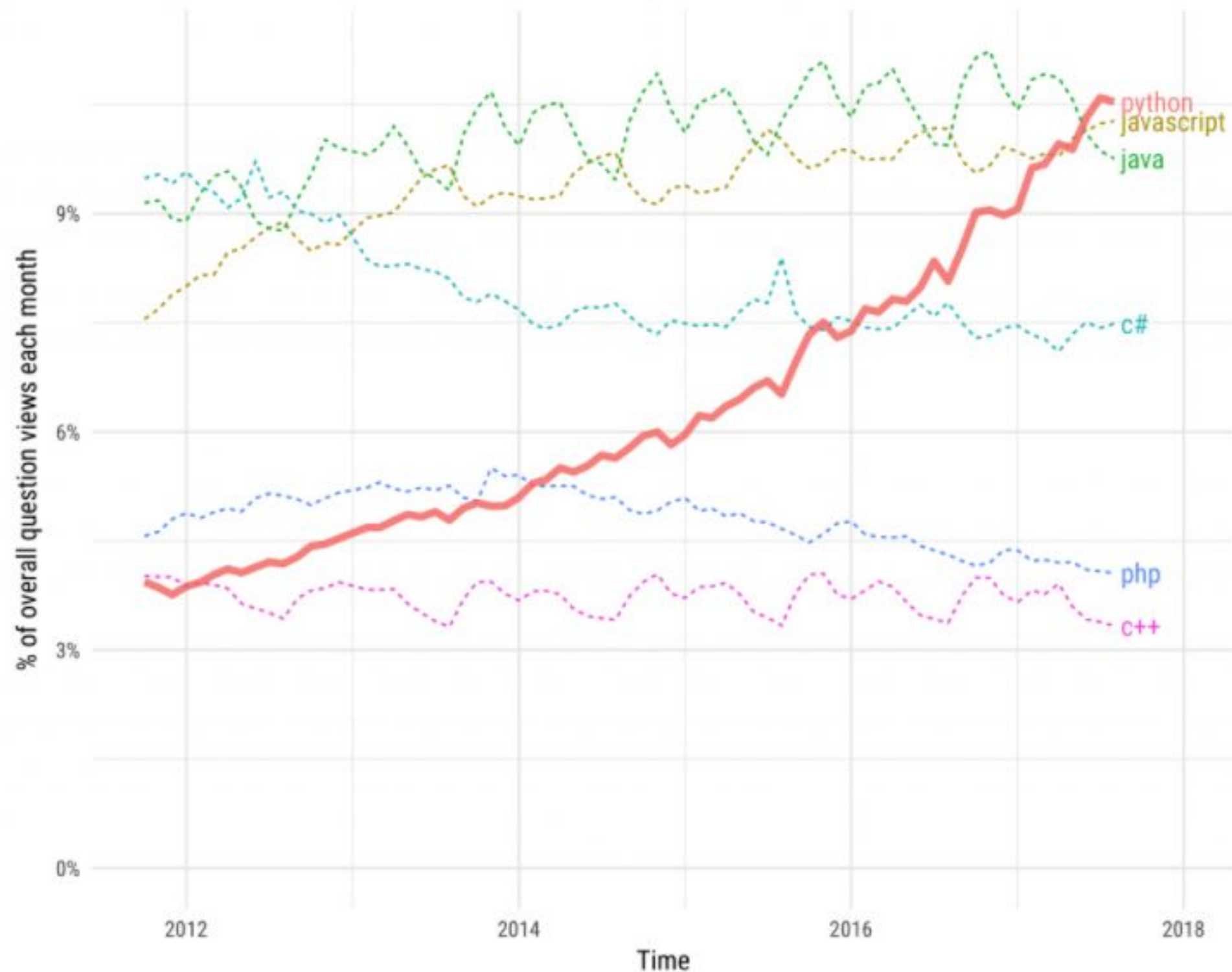
Most Pull Requests 2017

GitHub



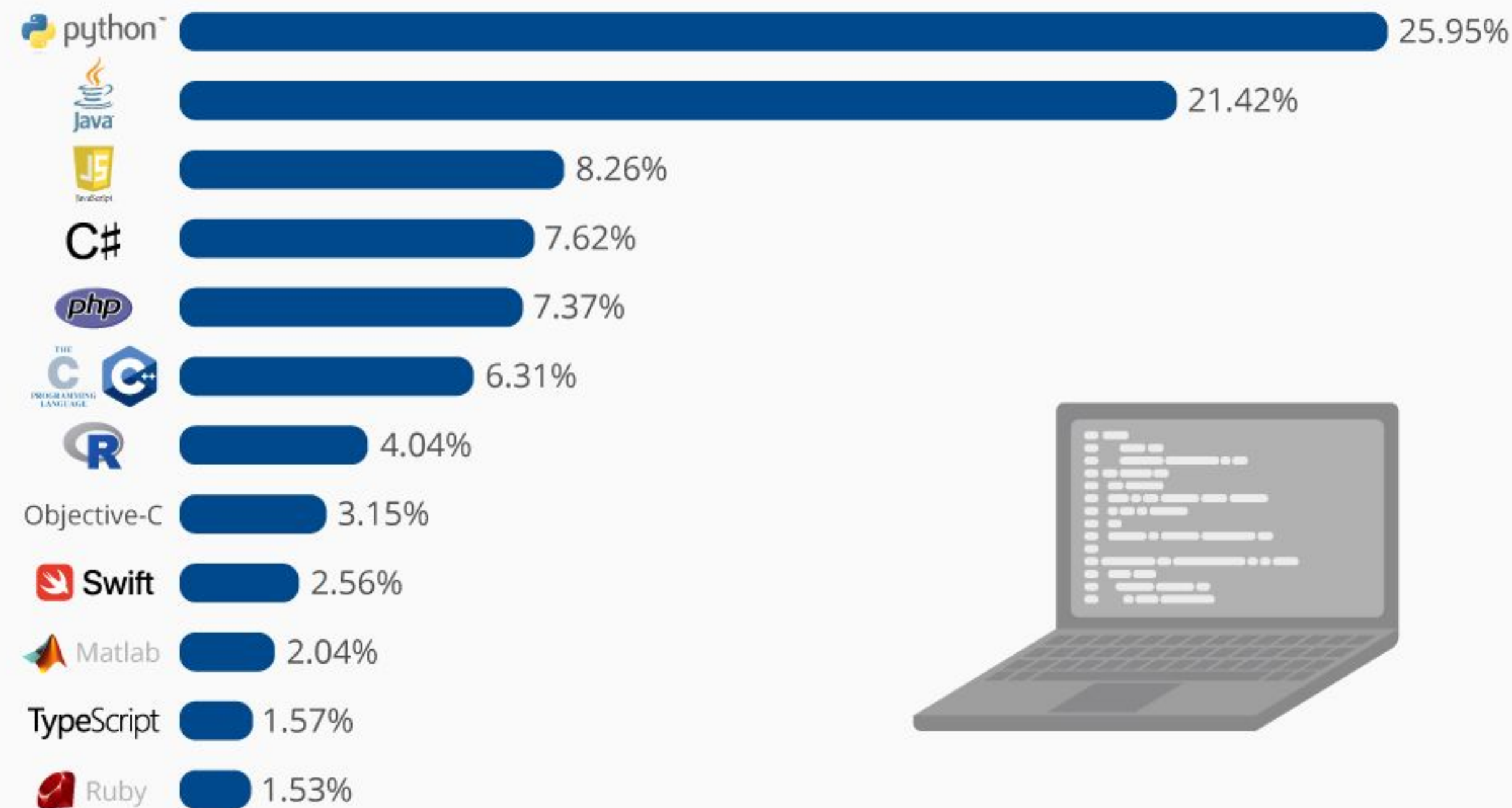
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



The Most Popular Programming Languages

Share of the most popular programming languages in the world*



* Based on the PYPL-Index, an analysis of Google search trends for programming language tutorials.
Source: PYPL



vs.



PYTHON 2

← Legacy

It is still entrenched in the software at certain companies

2 Library

Many older libraries built for Python 2 are not forwards-compatible

0100 0001 ASCII

Strings are stored as ASCII by default

≈ 5/2=2

It rounds your calculation down to the nearest whole number

print "hello"

Python 2 print statement

PYTHON 3

Future →

It will take over Python 2 by 2020

Library 3

Many of today's developers are creating libraries strictly for use with Python 3

Unicode 0000 0000 0100 0001 Text strings are Unicode by default

5/2=2.5 =

The expression 5 / 2 will return the expected result

print ("hello")

The print statement has been replaced with a print () function

PYTHON 2.X



PYTHON 3.X

```
>>> print "Hello World!"  
Hello World!  
>>> print 3/2  
1  
>>> variable = 123456789  
>>> print (type(variable))  
<type 'int'>
```

```
>>> print ("Hello World!")  
Hello World!  
>>> print (3/2)  
1.5  
>>> variable = 123456789  
>>> print (type(variable))  
<class 'int'>
```

Ways to use Python

1. Stand-alone scripts

- Code saved in text file, executed on command line
- As described in PCfB book

2. Interactive mode via command line

- Enter commands 1-by-1 on command line
- Good for testing

3. Jupyter notebook

- Rich, web-based interface; results presented inline
- Good for teaching purposes and sharing code

Interactive development environments (IDEs)

The screenshot displays the Spyder Python IDE interface. The main window is divided into several panes:

- Project explorer:** Shows the file structure of the current project, including folders like 'Data', 'spyder', and 'tests'.
- Editor:** Contains the Python code being edited. The code defines a function to generate data, perform calculations, and plot results. It includes comments and uses libraries like numpy, scipy, and matplotlib.
- Outline:** Provides a hierarchical view of the code structure, showing functions and classes.
- Variable explorer:** Displays the current state of variables in the workspace, including their names, types, sizes, and values.
- IPython console:** Shows the execution of code snippets, including a 3D surface plot and a polar plot.

The code in the editor is as follows:

```
6
7 import pylab
8 from numpy import cos, linspace, pi, sin, random
9 from scipy.interpolate import splprep, splev
10
11 %% Generate data for analysis
12
13 # Make ascending spiral in 3-space
14 t = linspace(0, 1.75 * 2 * pi, 100)
15
16 x = sin(t)
17 y = cos(t)
18 z = t
19
20 # Add noise
21 x += random.normal(scale=0.1, size=x.shape)
22 y += random.normal(scale=0.1, size=y.shape)
23 z += random.normal(scale=0.1, size=z.shape)
24
25
26 %% Perform calculations
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 %% Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
49
50 pylab.subplot(2, 2, 2)
51 data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
52 fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
53 pylab.legend()
54 pylab.xlabel('x')
```



Data types

Data types

String

Integer

Floating point

Boolean

Converting between types

String

Integer

Floating point

Boolean

Data containers

List

[1, '1', 'one', [1, 2]]

Dictionary

{1: 'one', 2: 'two', 3: 'three'}

Variables

Methods

Dot notation

dir()

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',  
 '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__',  
 '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__',  
 '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',  
 '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__',  
 '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',  
 'pop', 'remove', 'reverse', 'sort']
```

#Comment, #comment, #comment

- Used to:
 - Guide others through your script
 - Indicate assumptions being made
 - Document changes made across versions
- You really can't have too many comments!
- Most will probably be more useful to YOU than others

Demo