Programming for Economists

Week 9: Introduction to Python

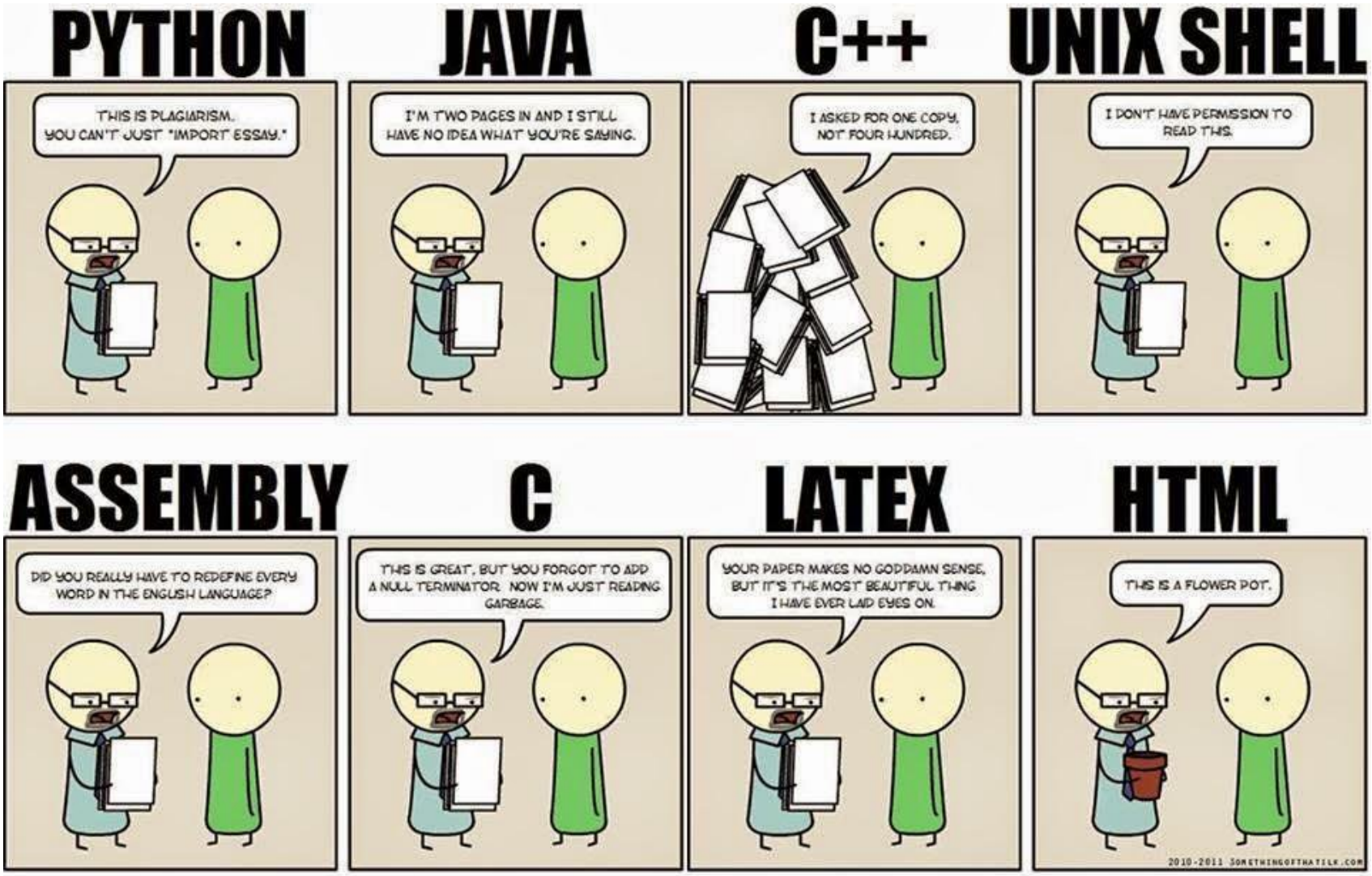
Tyler Abbot

Department of Economics
Sciences Po

Fall 2016

Outline

Python is a programming language I



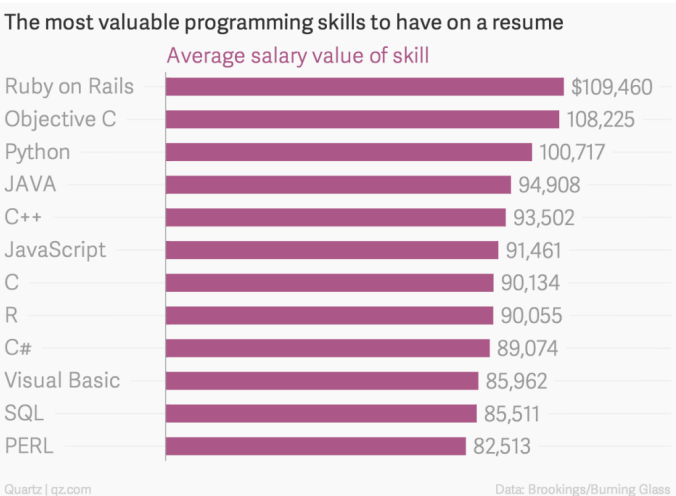
Source: Sebastian Raschka, a Phd student of whom we should all be in awe.

Python is a programming language II

Guido van Rossum, creator of Python. Source: wikipedia

- First implemented in 1989
- Interpretted, object-oriented, high-level programming language
- Favored because of its syntax
- Created by computer scientists
- Many features that we will cover in the coming sessions

Who uses Python?



Source: readwrite.com

- Used by tens of thousands and constantly expanding
- 3100 attendees at PyCon 2015
- The language of choice for education:
  - All MIT intro computer science
  - Six out of top 10 econ departments
  - Many finance programs
  - Lots of industry positions (especially in finance!)

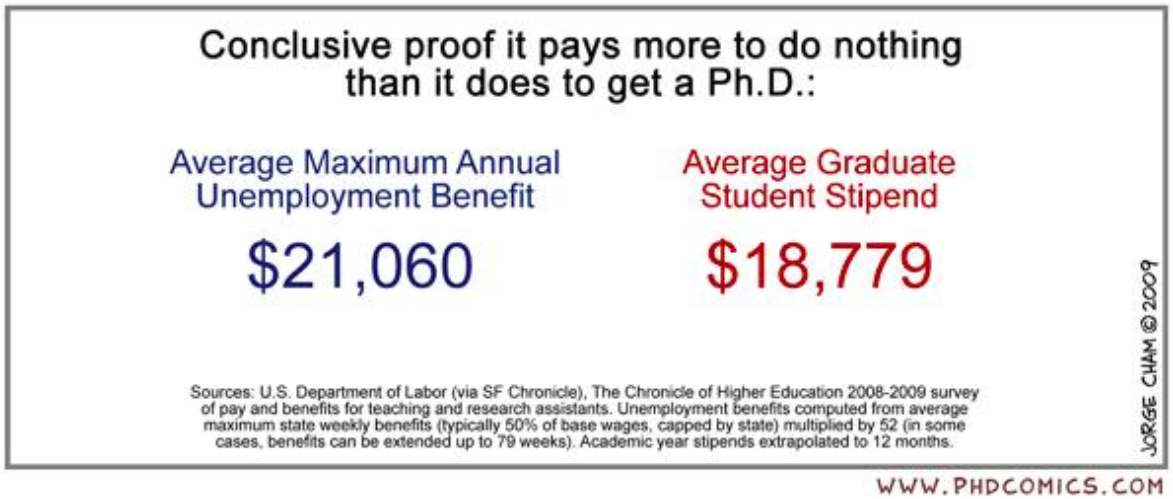
How is Python different from my current tools?

- R, Stata, and Matlab are NOT programming languages
- Python is used outside of academia
- Multithreaded (this is false! but at least multiprocessing is easier)
- Open source
- Syntax. Just type

  ```
  import this
  ```

  into the Python prompt to get a sense of the Zen of Python

Main Features



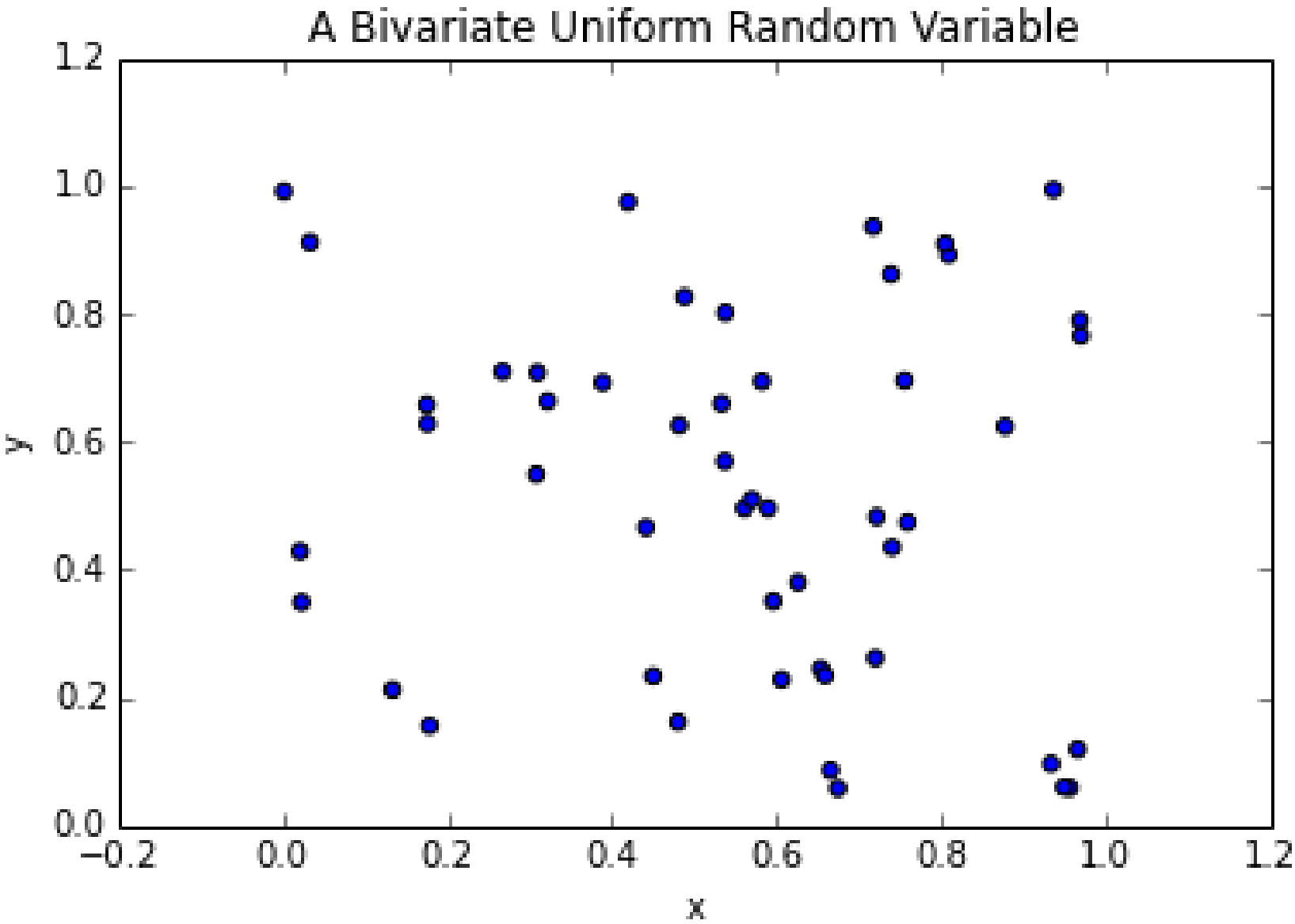
Source: meandmyresearch.blogspot.co.uk

- FREE!!!! You're broke, I know it.
- Interpreted, not compiled
- Object oriented
- Libraries, libraries, libraries

A Simple Python Script

```
1  """
2  Origin: A simple example.
3  Filename: example_scatter.py
4  Author: Tyler Abbot
5  Last modified: 15 September, 2015
6  """
7  import matplotlib.pyplot as plt
8  import random
9
10 x,y = [], []
11 for i in range(0,50):
12     #This is a comment
13     x.append(random.random())
14     y.append(random.random())
15
16 plt.scatter(x, y)
17 plt.xlabel('x')
18 plt.ylabel('y')
19 plt.title('A Bivariate Uniform Random Variable')
20 plt.show()
```

A Simple Python Script Output



Modules and Packages

- Code repositories. Store useful functionality.
- Packages $\subset$ Modules
- PyPI - More than 66,000 packages
- ‘pip‘ - PIP Installs Packages (duh!)

Modules and Packages

- NumPy - Arrays!
- SciPy - Computation!
- MatPlotLib - Plots!
- Pandas - Bears! ... I mean statistics.
- StatsModels - Econometrics
- Requests - HTTP
- Beautiful Soup - Parsing text
- Scrapy - Web scraping
- Sympy - Symbolic algebra

Data Structures

- List
- String
- Tuple
- Set
- Dictionary
- Numpy Array
- Pandas DataFrame

Today, we'll focus on lists... and maybe a bit about loops!

Lists

- A list itself is 72 bytes (big)
- 4G ram = 5.55e7 lists
- Size changes as input changes!! Bigger integers in a list require more memory... this is bad.
- They are native and have nice methods/features

Defining a list

```
1 x = []
2 y = [i for i in range(0,10)]
```

Lists II

List as Stack

```
>>> x = [1, 2, 3]
>>> x
[1, 2, 3]
>>> x.append(4)
>>> x
[1, 2, 3, 4]
>>> x.pop()
4
>>> s
[1, 2, 3]
```

Lists III

List as Stack

```
1 >>> words = ['hello', 'there', 'everyone']
2 >>> for word in words: print word
3 hello
4 there
5 everyone
```

For Loops

- The Python Wiki has a great entry about loops: https://wiki.python.org/moin/ForLoop
- In Python, you most often see for loops
- Most things in Python are iterable
- Even possible to define your own iterable

Loop examples stolen from Python Wiki

```
1  for x in range(0, 3):
2      print "We're on time %d" % (x)
3
4  x = 1
5  while True:
6      print "To infinity and beyond! We're getting close,"\
7              + " on %d now!" % (x)
8      x += 1
9      break
```

added the break just in case...

Strings

- Strings act like lists
- They support slicing
- They have a similar memory footprint
- String specific methods (a lot!)
- Support arithmetic operations

Strings

- Some stuff about strings

Stringy things

```
1  #Create a string
2  x = "You better fill this in at some point..."
3  print x
4
5  #Slice a string
6  print x[:-3]
7
8  #Change the case
9  print x.lower()
10 print x.upper()
11
12 #Find the index of a substring
13 print x.find('fill')
14 print x[11:]
```

Tuple

- Immutable
- Sequence (like a list)
- Support concatenation and repitition
- Useful for passing variables to functions or for constants you don't want to change

Tuple

```
1  #Define a tuple
2  tup = 1, 2, 3
3  tup1 = ("a", "b", "c")
4  print tup
5  print tup1
6
7  #Retrieving tuple data
8  print tup[0]
9
10 #This won't work, since tuples are immutable
11 tup[0] = 1
```

Dictionary

- Like a list, but anything can be the index!
- Iterable in neat ways
- Not super useful in scientific computing (indices are usually present)
- Highly versatile

Tuple

```
1  #Define a dictionary
2  packages = {'NumPy': 'arrays',
3              'SciPy': 'numerical methods',
4              'Pandas': 'statistics'}
5
6  #Get info
7  print packages.items()
8  print packages.get('matplotlib')
9  print packages['NumPy']
10
11 #Loop through the dictionary
12 for package, use in packages.items():
13     print "%s is most useful for %s." %(package, use)
```

Conclusion

At this point you should at least have heard of these things and why they matter for Python:

- ...Python
- The interpreter
- Text editors
- GitHub
- IPython
- A basic python script
- Lists, loops, and strings

Notes are on the website. Do the exercises. I'm available to answer questions. See you next week!