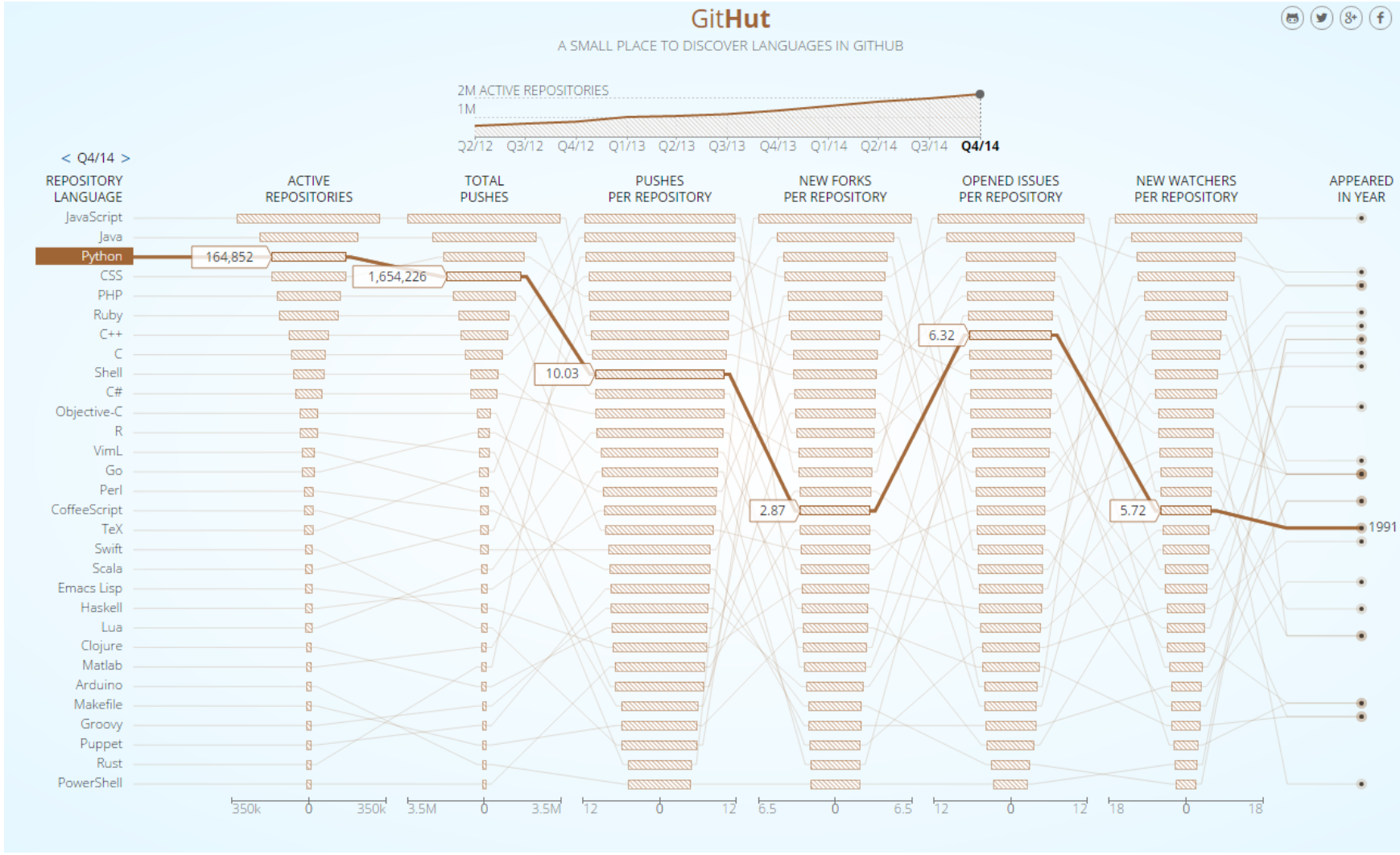
Introduction to Python

Computational Genomics

Weiguang (Wayne) Mao

Significant content courtesy by Marta Wells

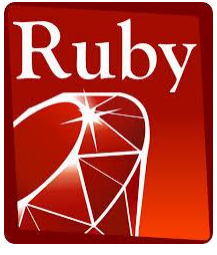

GitHut
A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB
2M ACTIVE REPOSITORIES
1M
Q2/12 Q3/12 Q4/12 Q1/13 Q2/13 Q3/13 Q4/13 Q1/14 Q2/14 Q3/14 Q4/14
< Q4/14 >
REPOSITORY LANGUAGE
ACTIVE REPOSITORIES
TOTAL PUSHES
PUSHES PER REPOSITORY
NEW FORKS PER REPOSITORY
OPENED ISSUES PER REPOSITORY
NEW WATCHERS PER REPOSITORY
APPEARED IN YEAR
JavaScript
Java
Python
CSS
PHP
Ruby
C++
C
Shell
C#
Objective-C
R
VimL
Go
Perl
CoffeeScript
TeX
Swift
Scala
Emacs Lisp
Haskell
Lua
Clojure
Matlab
Arduino
Makefile
Groovy
Puppet
Rust
PowerShell
164,852
1,654,226
10.03
2.87
6.32
5.72
1991
350k 0 350k
3.5M 0 3.5M
12 0 12
6.5 0 6.5
12 0 12
18 0 18

Why or Why not

- Easy to install/learn
- Popular – well documented
- Free, open source
- Supports (72, 747 packages)

Powerful language for scripting

- Legacy
- Your mission
- There's Only Way To Do It!

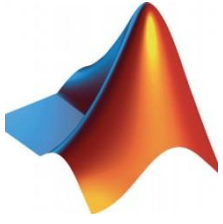


Python vs Matlab

- Data structures (from scratch)
- Web spider, etc

- Simulink
- Computer Vision, etc
- Matrix tradition



http://sebastianraschka.com/Articles/2014_matlab_vs_numpy.html

Versions

- Python 2.7.11 is legacy and the default choice
- Python 3.5.1 is current
- Not fully compatible

Installation

+

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IP[y]:
IPython

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

=

ANACONDA®

How to run python scripts

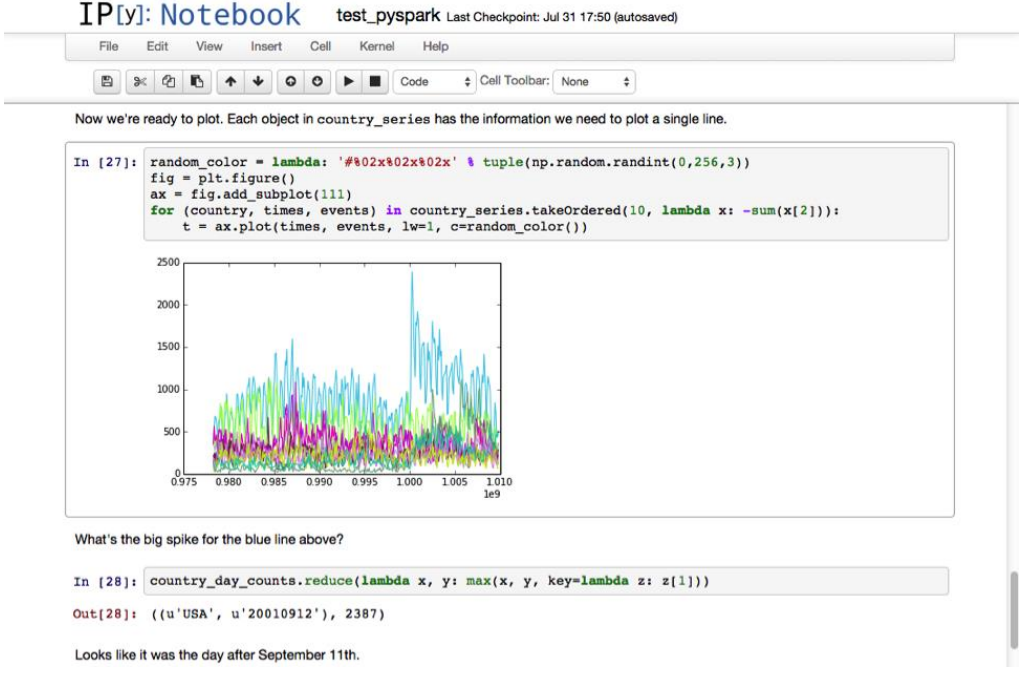
- Install python/anaconda
- Confirm the path (especially multiple pythons)
- python myscript.py

```
print "Hello World"
```

```
F:\>python Recitation.py
Hello World
```

IDE (Integrated Development Environment)

- Eclipse (PyDev)
- iPython (interactive)



0. Indentation

- Need whitespace
- Do not mix tabs + spaces

```
if True:
    print "ok"
```

```
if True:
print "ok"
```

```
IndentationError: expected an indented block
```

1. Types and Variables

- Value
  - Actual data
- Type
  - What kind of data is it
- Variable
  - Name of the data, how to access it

1. Types and Variables

- Variable names
  - Must start with a letter or ‘_’
  - Can contain letters, numbers, and ‘_’
  - No spaces

1. Types and Variables

- Basic types
  - Int
    - Whole numbers
    - Max/min value is ~+/- 2.1 billion
  - Float
    - Decimals
    - Ints are automatically converted if two types are mixed
      - 3/0.5 = 6.0

1. Types and Variables

- Basic types cont.
  - String
    - Text
    - ‘Single’ or “double” quotes
    - Concatenated using +
      - ‘abc’+’def’ = ‘abcdef’
  - Boolean
    - True or false

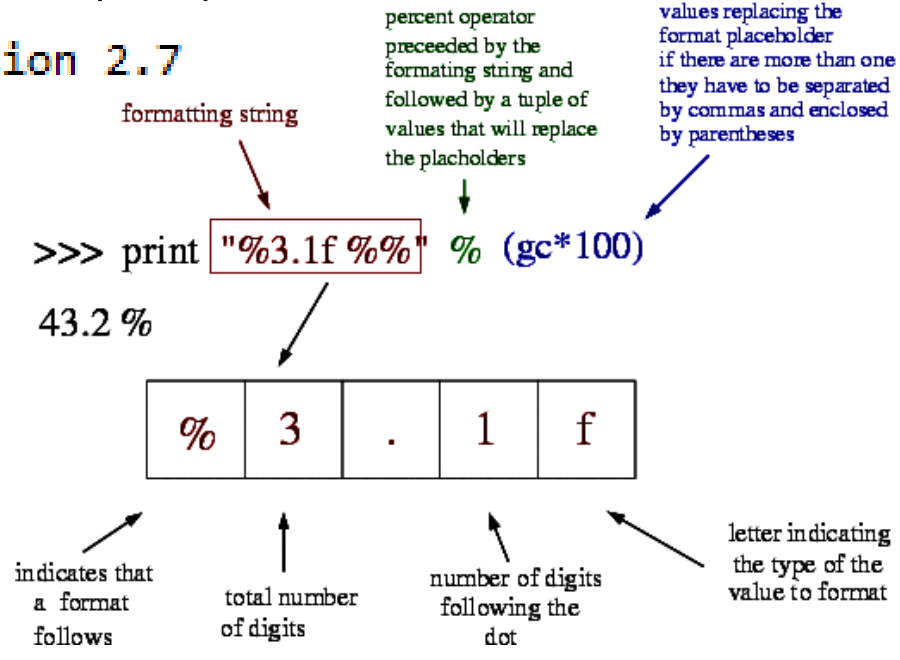
1. Types and Variables

- Use type(x) to check what type the variable is
- Use str(x), int(x), or float(x) to convert between types
  - X = ‘1’
  - Y = 3
  - Z = int(X) + Y

2. String Formatting

- X = 10
- Y = ‘computers’
- Z = 2.7
- Str = ‘%i of the %s are at version %.1f’ % (X,Y,Z)

```
10 of the computers are at version 2.7
```



3. Operators

- Arithmetic
  - (2*5+4)/1-7
    - 7.0
- Exponentiation
  - 2**3
    - 8
- Modulus
  - 7%4
    - 3

- Comparison and Logical
  - 3>=1
    - True
  - (3>=1) and (3!=1)
    - True
  - (3<1) or (3==0)
    - False
  - ‘d’ not in ‘dog’
    - False
  - 3 in [1,2,3]
    - True

4. Data Structures

- Lists
  - myList = [1,2,4,3]
  - myList[0]
    - 1 **Starting from 0**
  - myList[-1]
    - 3
  - myList.append(5)
    - myList = [1,2,4,3,5]
  - len(myList)
    - 5
  - Strings = list of letters

4. Data Structures

- Linked lists

```
class node(object):
    def __init__(self,left,right,dist,name):
        self.left = left
        self.right = right
        self.dist = dist
        self.name = name
```

5. List Operations

- myList.append(X)
  - Add X to list
- myList.count(X)
  - Count number of occurrences of X
- myList.extend(myList2)
  - Append myList2 to myList
- myList.remove(X)
  - Remove X from the list
- myList.sort()
  - Sort list
- myList.index(X)
  - Return the index of X
- myList.insert(I,X)
  - Insert X at position I
- myList.pop(I)
  - Remove item at position I
- myList.reverse()
  - Reverse list elements

6. List Indexing

- myList = [1,2,3,[4,5,6]]
  - myList[0] = 1
  - myList[3] = [4,5,6]
  - myList[3][0] = 4
  - myList[0:2] = [1,2]
  - myList[3][1:] = [5,6]
  - myList[2:] = [3,[4,5,6]]
  - myList[:2] = [1,2]

7. List Comprehensions

- myList = [x**2 for x in range(0,5)]
  - [0, 1, 4, 9, 16]

8. Dictionaries

- myDict = {‘key1’: 5, ‘key2’: 6}
- myDict[‘key1’]
  - 5
- myDict[‘key3’] = 7
  - {‘key1’: 5, ‘key2’: 6, ‘key3’:7}
- myDict.keys(), myDict.values()
  - myDict[key] = value

```
['key3', 'key2', 'key1']
[7, 6, 5]
```

9. Conditionals

```
if x < 0:
        print ‘Negative’
elif x == 0:
        print ‘Zero’
else:
        if x == 1:
                print ‘Single’
        else:
                print ‘Multiple’
```

10. Iteration

```
words = [‘welcome’, ’to’, ’class’]
for w in words:
        print w
```

```
welcome
to
class
```

```
i = 0
while i < 100:
        print i
        i = i +1
```

```
for i in range(100):
        print i
```

0-99

11. Importing

- import numpy
  - numpy.array
- import numpy as np
  - np.array
- from numpy import *
  - array
- Other modules
  - sys, os, math, re, scipy, matplotlib

- How to install packages/modules
  - pip
  - easy_install
  - install from source code

12. Methods

- String methods
  - Str.strip()
    - Removes leading and trailing whitespace
  - Str.split()
    - Splits a string into a list by whitespace

- List methods

12. Methods

- User defined

  def methodname(parameters):

  Statements/calculations

  return value

```
def add(num1,num2):
    return num1+num2

print add(1,2)
```

13. Main Method Example

```
#!/usr/bin/python
# Filename: using_name.py

if __name__ == '__main__':
        print 'This program is being run by itself'
else:
        print 'I am being imported from another module'
```

```
$ python using_name.py
This program is being run by itself

$ python
>>> import using_name
I am being imported from another module
>>>
```

14. Input arguments

```
import sys
print "This is the name of the script: ", sys.argv[0]
print "Number of arguments: ", len(sys.argv)
print "The arguments are: " , str(sys.argv)
```

```
F:\>python Recitation.py --version --parameter 4
This is the name of the script:  Recitation.py
Number of arguments:  4
The arguments are:  ['Recitation.py', '--version', '--parameter', '4']
```

15. Reading Files

```
filename = 'input.txt'
inputfile = open(filename, 'r')
for line in inputfile:
    ### process each line in input.txt

inputfile.close()
```

```
with open (…) as f:
    for line in f:
        #do something with line
```

http://stackoverflow.com/questions/8009882/how-to-read-large-file-line-by-line-in-python

16. Writing Files

```
filename = ‘output.txt’
outputfile = open(filename, ‘w’)
outputfile.write(‘data for output’)
outputfile.close()
```

Look it up!