Large-Scale Networks

3b – Python for large-scale networks

Dr Vincent Gramoli | Senior lecturer
School of Information Technologies

THE UNIVERSITY OF SYDNEY

Introduction

THE UNIVERSITY OF SYDNEY

Why Python?

- What to do with Python?
  - To get data (e.g., crawling online social networks)
  - Manipulate and process data
  - Visualize results (as well as understand and communicate results)

- Why Python?
  - Rich collection of already existing building blocks (e.g., NetworkX)
  - Open source free software
  - Easy to learn

What is Python?

- What is Python?
  - An interpreted language
  - Multi-platform tool: available for Windows, Linux, Mac OS X, Android…
  - Can be interfaced with other languages (e.g., C/C++)
- Write directly in the interpreter

```
gramoli@poseidon:~$ python
Python 2.7.3 (default, Feb 27 2014, 19:58:35)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Interpret a python file

```
gramoli@poseidon:~$ python hello-world.py
Hello, world!
```

Getting started

- Start the interpreter

```
>>> print "Hello, world!"
Hello, world!
```

- Do not declare type before assigning its value

```
>>> a = 3
>>> b = 2*a
>>> type(b)
<type 'int'>
>>> print b
6
>>> a*b
18
```

› String operands

```
>>> b = 'hello'
>>> type(b)
<type 'str'>
>>> b+b
'hellohello'
>>> 2*b
'hellohello'
```

Scalar Types

THE UNIVERSITY OF SYDNEY

Scalar types

- Integer

```
>>> 1 + 1
2

>>> a = 4

>>> type(a)
<type 'int'>
```

- Floats

```
>>> c = 2.1

>>> type(c)
<type 'float'>
```

- Casting

```
>>> float(1)
1.0
```

- Complex

```
>>> a = 1.5 + 0.5j

>>> a.real
1.5

>>> a.imag
0.5

>>> type(1. + 0j)
<type 'complex'>
```

- Booleans

```
>>> 3 > 4
False

>>> test = (3 > 4)

>>> test
False

>>> type(test)
<type 'bool'>
```

Divisions

- Integer division

```
>>> 3/2
1
```

- Use floats

```
>>> 3/2.
1.5
```

```
>>> a = 3
```

```
>>> b = 2
```

```
>>> a / b
1
```

```
>>> a / float(b)
1.5
```

- Explicit integer division

```
>>> 3.0 // 2
1.0
```

Lists and Strings

THE UNIVERSITY OF SYDNEY

Lists

- Lists

```
>>> l = ['red', 'blue',
'green', 'black', 'white']
```

```
>>> type(l)
<type 'list'>
```

- Accessing list elements

```
>>> l[2]
'green'
```

```
>>> l[-1]
'white'
```

```
>>> l[3]
'black'
```

```
>>> l
['red', 'blue', 'green',
'black', 'white']
```

- Slicing syntax L[start:stop:stride]

```
>>> l[3:]
['black', 'white']
```

```
>>> l[:3]
['red', 'blue', 'green']
>>> l[::2]
['red', 'green', 'white']
```

- Lists are mutable objects

```
>>> l[2:4] = ['gray',
'purple']
```

```
>>> l
['red', 'blue', 'gray',
'purple', 'white']
```

- The type of list elements may differ

```
>>> l = [3, -200, 'hello']
```

Lists

- Add and remove elements

```
>>> l = ['red', 'blue', 'green',
'black', 'white']
```

```
>>> l.append('pink')
['red', 'blue', 'green',
'black', 'white', 'pink']
```

```
>>> l.pop()
'pink'
>>> l
['red', 'blue', 'green',
'black', 'white']
```

```
>>> l.extend(['pink', 'purple'])
['red', 'blue', 'green',
'black', 'white', 'pink',
'purple']
```

```
>>> l = l[:-2]
```

```
>>> l
['red', 'blue', 'green',
'black', 'white']
```

- Reverse

```
>>> r = l[::-1]
>>> r
['white', 'black', 'green',
'blue', 'red']
```

```
>>> r2 = list(l)
```

```
>>> r2
['red', 'blue', 'green',
'black', 'white']
```

- Sort

```
>>> sorted(r)
['black', 'blue', 'green',
'red', 'white']
```

String

- Different syntaxes

```
>>> s = 'Hello, how are you?'
>>> s = "Hi, what's up?"
>>> s = '''Hello,
how are you'''
>>> s = """Hi
what's up"""
```

- Strings can be indexed/sliced like lists

```
>>> a = "hello"
>>> a[0]
'h'
>>> a = "hello, world!"
>>> a[3:6]
'lo,'
```

- Dictionary

```
>>> tel = {'alice':
1234,'bob':2345}
>>> tel['carole'] = 3456
>>> tel
{'bob': 2345, 'alice': 1234,
'carole': 3456}
>>> tel['bob']
2345
>>> tel.keys()
{'bob', 'alice', 'carole'}
>>> tel.values()
[2345, 1234, 3456]
>>> 'carole' in tel
True
```

Control Flow

THE UNIVERSITY OF SYDNEY

Control flow

- Blocks are delimited by indentation!

- If/elif/else

```
>>> if 2**2 == 4
...     print 'Obvious!'
Obvious!


>>> a = 10
>>> if a == 1:
...     print(1)
... elif a == 2:
...     print(2)
... else:
...     print('a lot')
a lot
```

- For/range

```
>>> for i in range(4):
...     print(i)
0
1
2
3
>>> for word in ('cool', 'powerful', 'simple'):
...     print('Python is %s' % word)
Python is cool
Python is powerful
Python is simple
```

Control flow

- Python supports
  - while
  - break
  - Continue
- Conditional expressions
  - if <object>
    - Evaluates to false:
      - Any number equal to zero (0, 0.0, 0+0j)
      - An empty container (list, tuple, dictionary…)
    - Evaluates to true
      - everything else

- Tests
  - `a == b` tests equality, with logics
  - `a is b` tests identity
  - `a in b` tests containment
  - (If b is a dictionary, containment tests whether a is a key of b)
- Iterator over list/string/keys/ lines…

```
>>> for i in 'powerful':
...     if I in vowels:
...         print(i),
o e u
```

Functions and Modules

THE UNIVERSITY OF SYDNEY

Functions

- Functions are delimited by indentation!

```
>>> def test():
...     print('in test funct')
>>> test()
in test funct
```

- They can return values

```
>>> def test():
...     return 2*2
>>> test()
4
```

- They can take arguments

```
>>> def test(x):
...     Return x*2
>>> test(4)
8
```

- They can take optional args

```
>>> def test(x=2):
...     Return x*2
>>> test()
4
```

Modules

- Importing objects from modules

```
>>> import os
>>> os.listdir('.')
```

- …lists the content of current directory

- Use import * with caution

```
>>> from os import *
```

- …it makes it hard to track symbols

- Importing a single function

```
>>> from os import listdir
```

Modules

- Here is the list of modules we will use:
  - Python-twitter (module twitter) https://pypi.python.org/pypi/twitter
  - NetworkX (module networkx) https://networkx.github.io/
- Version of Python recommended (v2.7)
  - Anaconda http://continuum.io/downloads
  - Canopy https://www.enthought.com/products/epd/
- Other modules of interests:
  - Simplejson: JSON encoder/decoder https://pypi.python.org/pypi/simplejson/
  - Requests: HTTP requests https://pypi.python.org/pypi/requests
  - BeautifulSoup: parse documents and find links: http://www.crummy.com/software/BeautifulSoup/
  - python-instagram: API for instagram https://github.com/Instagram/python-instagram

Conclusion

THE UNIVERSITY OF SYDNEY

Conclusion

- We will use python
  - To obtain network information (e.g., Twitter API)
  - To analyze networks (e.g., NetworkX)

- Because
  - There are many modules
  - Free and open source
  - Python is simple

Going further

- Python scientific:
  - http://scipy-lectures.github.io/_downloads/PythonScientific-simple.pdf

- NetworkX reference:
  - http://networkx.github.io/documentation/latest/_downloads/networkx_reference.pdf

Other Modules

THE UNIVERSITY OF SYDNEY

HTTP headers

- Find the final location of the shortened url: http://bit.ly/GoogleScholar
- ```
  import urllib
  url = "http://bit.ly/GoogleScholar"
  r = urllib.request.urlopen(url) print(r.geturl())
  url = r.geturl()
  r = urllib.request.urlopen(url) print(r.geturl())
  print(r.info())
  print(r.headers.items()) # Remove the "print"
  ```
- ```
  # to display this in a nicer way (in ipython)
  ```

HTML parsing

- Extract the title of Feynman's 100 most cited papers from Google Scholar **import requests**
  from bs4 import BeautifulSoup
  url = "http://scholar.google.com/citations?user=\
- B7vSqZsAAAAJ&hl=fr&cstart=0&pagesize=100"
  headers = { headers = {"User-agent" : *# otherwise google scholar won't answer*
- "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0)\
- Gecko/20100101 Firefox/25.0"} }
  request = requests.get(url, headers = headers)
  soup = BeautifulSoup(request.text)
  table = soup.find("table", attrs={"class" : "cit-table"})
  for paper **in** table.findAll("td", attrs={"id" : "col-title"}):
- **print(paper.a.string)**

JSON

- Download the compressed JSON file: http://data.githubarchive.org/2015-01-01-15.json.gz.
  Write a script that reads the file and extracts all the actors id and repository id pairs.

- **import gzip**
  import sys
  import json
  t = gzip.open(sys.argv[1]).read() \ # *Open the file (in bytes mode)*
- .decode() # *Convert the text to string* d = [json.loads(x) # *Load the data…*
- **for** x **in** t.split('\n') # *… for each line in the file…*
- **if** x] # *… that is not empty* **for** item **in** d:
- **print('%s %s'** % (item['actor']['id'], item['repo']['id']))