Python Networking and Threading

Daniel Zappala

CS 360 Internet Programming
Brigham Young University

Why Use Python?

- many high-level abstractions available
- because of all the other great features of Python (easy string parsing, simple threading, dynamic typing, overriding builtin methods), you can very quickly and easily build powerful network programs
- also provides direct access to the same socket API you use with C
  - simple, but powerful
  - socket addressing easier, buffer allocation done for you

Python Modules for Network Programs

▸ Python Documentation

see Sections 17 - 20

Python Requests

Requests

▸ Requests: HTTP for Humans

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

Example Code

▸ Python Networking and Threading

Threading

Thread Classes

```
import threading

class Hello(threading.Thread):
    """ A thread that says hello. """
    def __init__(self):
        threading.Thread.__init__(self)

    def run(self):
        print "Hello from thread",self.getName()
```

- create a subclass of threading.Thread
- call the parent class init method
- the run() method is called when the thread is created

Creating and Running Threads

```python
threads = []
for i in range(0,10):
    t = Hello()
    threads.append(t)
for t in threads:
    t.start()
for t in threads:
    t.join()
```

- create an instance of the class
- calling the start() method creates the thread and invokes the run() method
- call the join() method to wait for the thread to finish

Example Code

▸ Python Networking and Threading