Network Programming with Python
First: A Little Theory

- Client/Server model
- Network sockets
- TCP/IP protocol suite
Clients and Servers
Clients and Servers

• Machines vs. Programs
• Examples of Server programs:
  – Apache
  – Tornado
• Examples of Client programs:
  – Mozilla Firefox
  – Google Chrome
Network Sockets

- Used to identify particular processes (programs) on particular machines.
- Socket is composed of two numbers:
  - IP address: machine identifier
  - Port number: process identifier
- Berkeley Sockets most common approach to sockets.
- A connection between two computers can be represented as two sockets: one for the client machine and program, one for the server machine and program.
Port Numbers

- Well-known ports
  - 0-1023
  - Examples:
    - 25: SMTP (email), 80: HTTP (web), 110: POP3 (email), 443: HTTPS (secure web)

- Registered ports
  - 1024-49151

- Private/Dynamic ports
  - 49151-65535
TCP/IP

- Layers of protocols (link, internet, transport, application).
- Mainly, we're concerned with TCP.
- One alternative to TCP is UDP.
NOW FOR CODE!!!!!
Steps for the server

- Create socket object
- Bind socket object to a particular socket
- Listen
- Program loop:
  - Accept connections from clients
  - [Do program stuff]
  - Close socket
Loopback Interface

- IP Address: 127.0.0.1
- Refers to the machine the program is running on.
- Not really networking (more like interprocess communication), but good for starting network programming.
Steps for the client

- Create socket object
- Ask to connect to a particular socket (THE SERVER'S SOCKET)
- [Do program stuff]
- Close the socket
Extra stuff

- recvall() method
  - Python provides sendall() method
- Reuse address option
Sending a single message

- Time to add program code!
Awesome E-Books

- CS e-book databases on the UW Libraries website:
- Python presentations from Autumn 2011:
  - http://www.cs.washington.edu/education/courses/cse