# Lecture 7

OOP Introduction
XML
Network Introduction

Prof. Steven Ludtke
N410.07, sludtke@bcm.edu

# Homework Review

```
import numpy as np
import matplotlib.pyplot as plt
from sys import argv

data=np.loadtxt(argv[1]).transpose()

plt.plot(data[0],data[1])

plt.show()
```

# Homework Review

```
import numpy as np
import matplotlib.pyplot as plt

filename=input("input file: ")
outfile=input("output file (use .pdf or .png): ")
xcol=int(input("x column: "))
ycol=int(input("y column: "))
style=input("style ('-' line, 'o' point): ")
xlabel=input("X Label: ")
ylabel=input("Y Label: ")
title=input("Title: ")

data=np.loadtxt(filename).transpose()

plt.plot(data[xcol],data[ycol],style)
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.title(title)

if outfile[-4:] in (".png",".pdf") : plt.savefig(outfile)
else: plt.show()
```

# Intellectual Property & Plagiarism

- Don't be afraid to use the web to find answers to your problems!
- Of course it is plagiarism to present someone else's code as your own. Just add a comment containing the URL
- In general, people don't post code publicly and expect it not to be copied. Still, if you are copying code verbatim, check the copyright info on the website.

# Representations of Complex Data

- Colony of mice
- Multiple cages
- Multiple mice in each cage
- Need to record weight of each mouse and the time it can spend on a rotarod each day

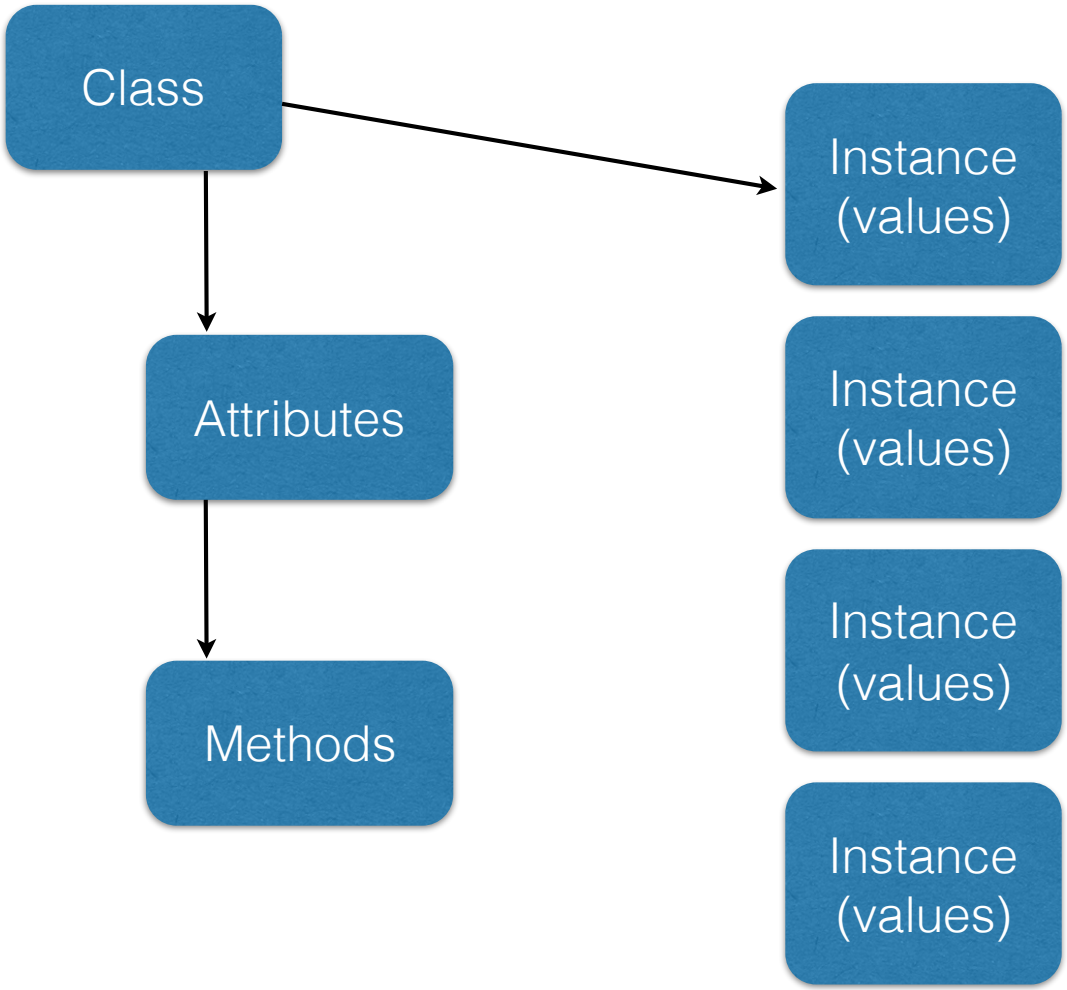
- How could we represent this data in a computer?
- Hierarchical or Flat?

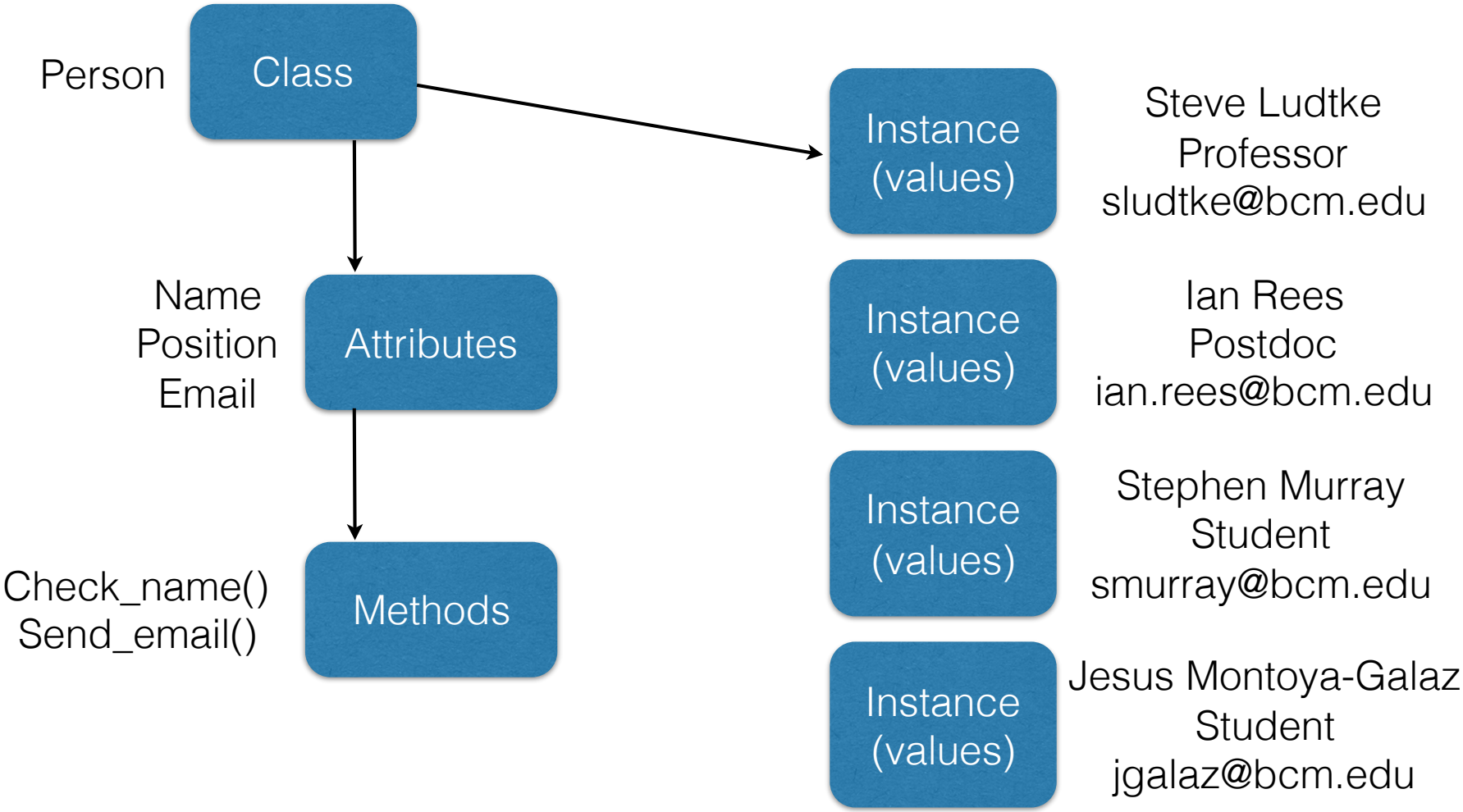
# Object Oriented Programming (A Quick Introduction)

- A Class is a grouping of associated data elements and optionally code elements (methods).

class person:

"This class describes a person"


Class
Attributes
Methods
Instance (values)
Instance (values)
Instance (values)
Instance (values)


Person
Class
Name
Position
Email
Attributes
Check_name()
Send_email()
Methods
Instance
(values)
Steve Ludtke
Professor
sludtke@bcm.edu
Instance
(values)
Ian Rees
Postdoc
ian.rees@bcm.edu
Instance
(values)
Stephen Murray
Student
smurray@bcm.edu
Instance
(values)
Jesus Montoya-Galaz
Student
jgalaz@bcm.edu

# Just Like a Dictionary

```
a={}

a[“firstname”]="Steve"
a[“lastname”]="Ludtke"
a[“address”]="1 Baylor Plaza"

print "{} {}\n{}".format(a[“firstname”],a[“lastname”],a[“address”])
```

# Attributes

```
class Person:
    “This class represents a person”

a=Person()

a.firstname="Steve"
a.lastname="Ludtke"
a.address="1 Baylor Plaza"

print "{} {}\n{}".format(a.firstname,a.lastname,a.address)
```

# In C++/Java

```
class person {
   string firstname;
   string lastname;
   string address;
   string city;
   char state[2];
   int zipcode;
   int phone;
};
```

# Methods

```
class Person:
    def __init__(self,firstname=None,lastname=None,address=None,
city=None, state=None, zipcode=None, phone=None):
        self.firstname=firstname
        self.lastname=lastname
        self.address=address
        self.city=city
        self.state=state
        self.zipcode=zipcode
        self.phone=phone

    def __repr__(self):
        return "{}, {}\n{}\n{}, {}  {}\n{}\n".format(self.lastname,
self.firstname,self.address,self.city,self.state,self.zipcode,self.phone)

    def fixcase(self):
        self.firstname=self.firstname.title()
        self.lastname=self.lastname.title()
```

# Setters & Getters

```
class Person:
…
        def set_name(self,first,last):
                self.firstname=first
                self.lastname=last
                self.fixcase()

        def get_name(self): return "{},
{}".format(self.lastname,self.firstname)

        def get_lastname(self): return self.lastname

        def get_firstname(self): return self.firstname
```

# Inheritance

```
class BCM_Person(Person):
    def set_bcmid(self,bcmid):
        self.bcmid=int(bcmid)

    def get_bcmid(self): return self.bcmid
```

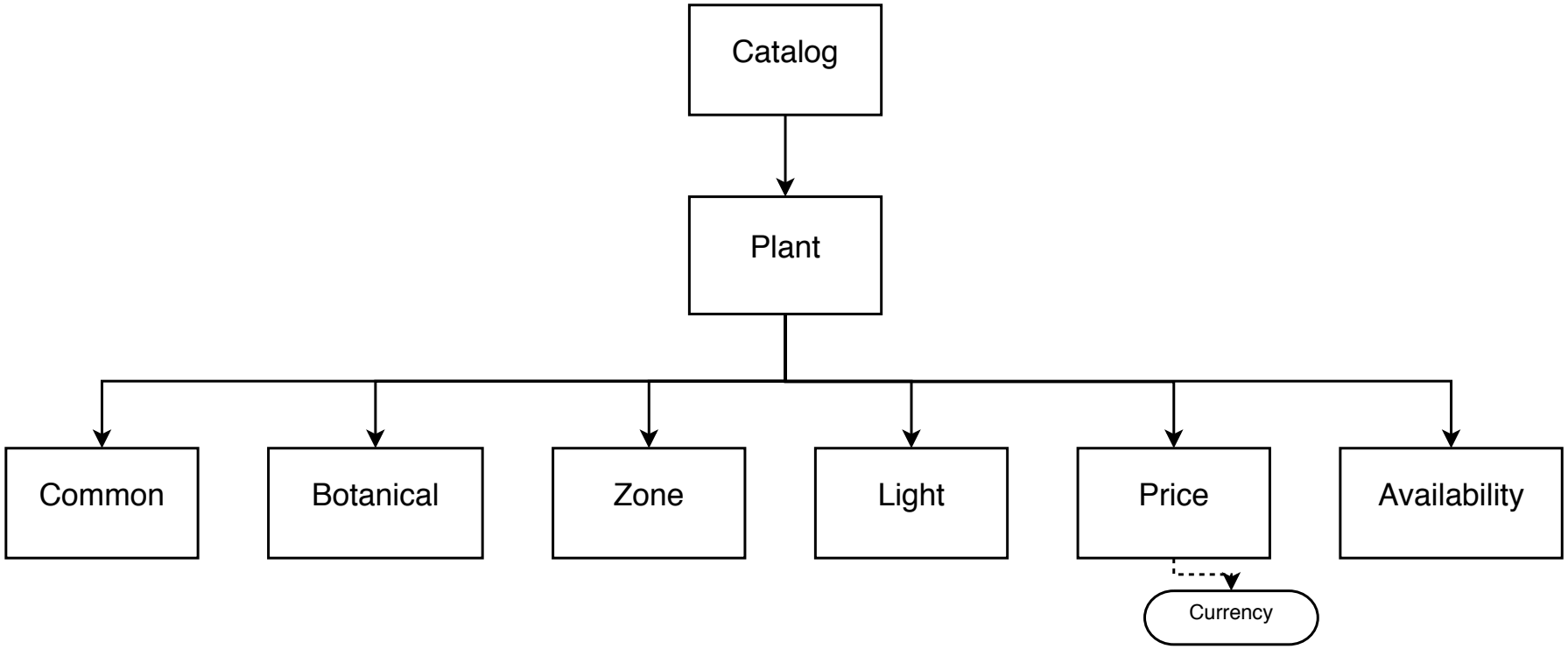
# Things we skipped

- Scope rules for classes
- Class attributes and methods
- Changes to classes vs instances
- Special methods to emulate, lists, dicts, etc.

# XML Basics

- <?xml version="1.0" encoding="UTF-8" ?>
- Tags
  - <tag> content </tag>  or <tag />
- Attributes
  - <tag attr1=”value” attr2=”value2”> </tag>
- Nesting
  - <tag1>content <tag2>nested</tag2></tag1>
- Case sensitive! (unlike HTML)

# Data Representation



# XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
   <PLANT>
      <COMMON>Bloodroot</COMMON>
      <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
      <ZONE>4</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE CURRENCY=”dollar”>2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
   </PLANT>
   <PLANT>
      <COMMON>Columbine</COMMON>
      <BOTANICAL>Aquilegia canadensis</BOTANICAL>
      <ZONE>3</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE CURRENCY=”dollar” >9.37</PRICE>
      <AVAILABILITY>030699</AVAILABILITY>
   </PLANT>
</CATALOG>
```

# XML in Python

- xml.sax
  - Simple API for XML (W3C)
  - Parse XML files sequentially, callbacks
- xml.dom
  - Document Object Model (W3C)
  - View XML as a single hierarchical document
- xml.etree
  - Python specific, similar to DOM
  - Easier to use !

# Using ElementTree

```
import xml.etree.ElementTree
et=xml.etree.ElementTree.parse("xml_example.xml")
et=xml.etree.ElementTree.fromstring("XML CODE")
e=et.getroot()          # The root object in the XML file
e[n]                    # Children of this element
e.items() or e.attrib   # An element’s attributes
e.text   # Unused text between the start and end tags
e.tag    # The element's tag as a string
```

# Representations of Complex Data

- Colony of mice
- Multiple cages
- Multiple mice in each cage
- Need to record weight of each mouse and the time it can spend on a rotarod each day

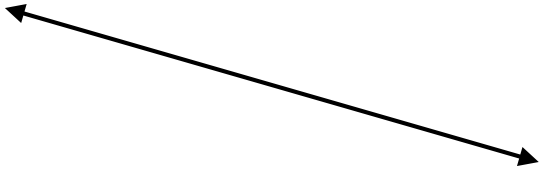
- How could we represent this data in a computer?
- Hierarchical or Flat?

# Flat

| mouse tag | strain | gender | colony | cage | 1/12/15 mass | 1/12/15 rotarod | 1/13/15 mass | 1/13/15 rotarod | … |
|---|---|---|---|---|---|---|---|---|---|
| 19-834 | YAC128 | M | HD123 | 7 | 10.3 | 45 | 10.4 | 53 | |
| 19-835 | YAC128 | M | HD123 | 7 | 9.8 | 47 | 9.8 | 51 | |
| 19-836 | BACHD | M | HD123 | 8 | 8.4 | 59 | 8.5 | 49 | |
| 19-837 | BACHD | M | HD123 | 8 | 9.2 | 56 | 9.4 | 51 | |
| … | | | | | | | | | |

# Relational

| mouse tag | strain | gender | colony | cage |
|---|---|---|---|---|
| 19-834 | YAC128 | M | HD123 | 7 |
| 19-835 | YAC128 | M | HD123 | 7 |
| 19-836 | BACHD | M | HD123 | 8 |
| 19-837 | BACHD | M | HD123 | 8 |
| … | | | | |



| mouse tag | meas. date | mass | rotarod |
|---|---|---|---|
| 19-834 | 1/12/15 | 10.3 | 45 |
| 19-835 | 1/12/15 | 9.8 | 47 |
| 19-836 | 1/12/15 | 8.4 | 59 |
| 19-837 | 1/12/15 | 9.2 | 56 |
| 19-834 | 1/13/15 | 10.4 | 53 |
| 19-835 | 1/13/15 | 9.8 | 51 |
| … | | | |

# Hierarchical (XML)

```
<colony name="HD122">
   <cage number="7">
      <mouse>
         <tag>19-834</tag>
         <strain>YAC128</strain>
         <gender>M</gender>
         <measurement date="1/12/15">
            <mass units="g">10.3</mass>
            <rotarod units="s">45</rotarod>
         </measurement>
         <measurement date="1/13/15">
            <rotarod>53</rotarod>
            <mass>10.4</mass>
         </measurement>
      </mouse>
      <mouse>
         …
      </mouse>
      …
   </cage>
   <cage number="8"> <mouse> … </mouse> … </cage>
</colony>
```

# Object Oriented

```
class colony:
  def __init__(self,name=None):
    self.name=name
    self.cages={}

class cage:
  def __init__(self):
    self.mice={}

class mouse:
  def __init__(self,strain=None,gender=None):
    self.strain=strain
    self.gender=gender
    self.measurements={}

class measurement:
  def __init__(self,mass,rotarot):
    self.mass=mass
    self.rotarod=rotarod
```

# Object Oriented

```
mycolony=colony("HD122")
mycolony.cages[7]=cage()
mycolony.cages[8]=cage()
mycolony.cages[7].mice["19-834"]=mouse("YAC128","M")
mycolony.cages[7].mice["19-834"].measurements["1/12/15"]=measure
ment(10.3,45)
mycolony.cages[7].mice["19-834"].measurements["1/13/15"]=measure
ment(10.4,53)
…
```

# XML Schemas

- Schemas Specifications
  - DTD
  - XML Schema
  - RELAX NG
- Specific Schemas/Ontologies
  - http://www.bioontology.org
  - http://en.wikipedia.org/wiki/List_of_XML_markup_languages

# RESTful Servers

- Generally return XML
- Client-Server - Servers store data, clients display data
- Stateless - URL uniquely identifies display
- Cacheable - Pages must declare whether their data is

# Real World Example

- www.pdb.org/pdb/software/rest.do
  - Several interfaces provided

# urllib

```
import urllib.request

f=urllib.request.urlopen("http://blake.bcm.edu/dl/test.html")

for i in f: print(i)
```

# RSS

- Small XML files containing frequently updated information. Link to webpage.
- 2 variants, also Atom.
- Required elements (2.0): title, link, description
- Optional elements: language, copyright, managingeditor, webmaster, pubdate, lastbuilddate, category, generator, docs, cloud, ttl, image, rating, textinput, skiphours, skipdays

# HTML

- http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html
- Declarative language
- HTML is a type of XML, XHTML obeys XML rules more completely
- Python HTMLParser module
- ‘commands’ in HTML are denoted by
  <command option=value option=value>text</command>
- For example:

  <HTML>

  <HEAD><TITLE>My Page</TITLE></HEAD>

  <BODY>

  <H3>Hi Everyone</H3>

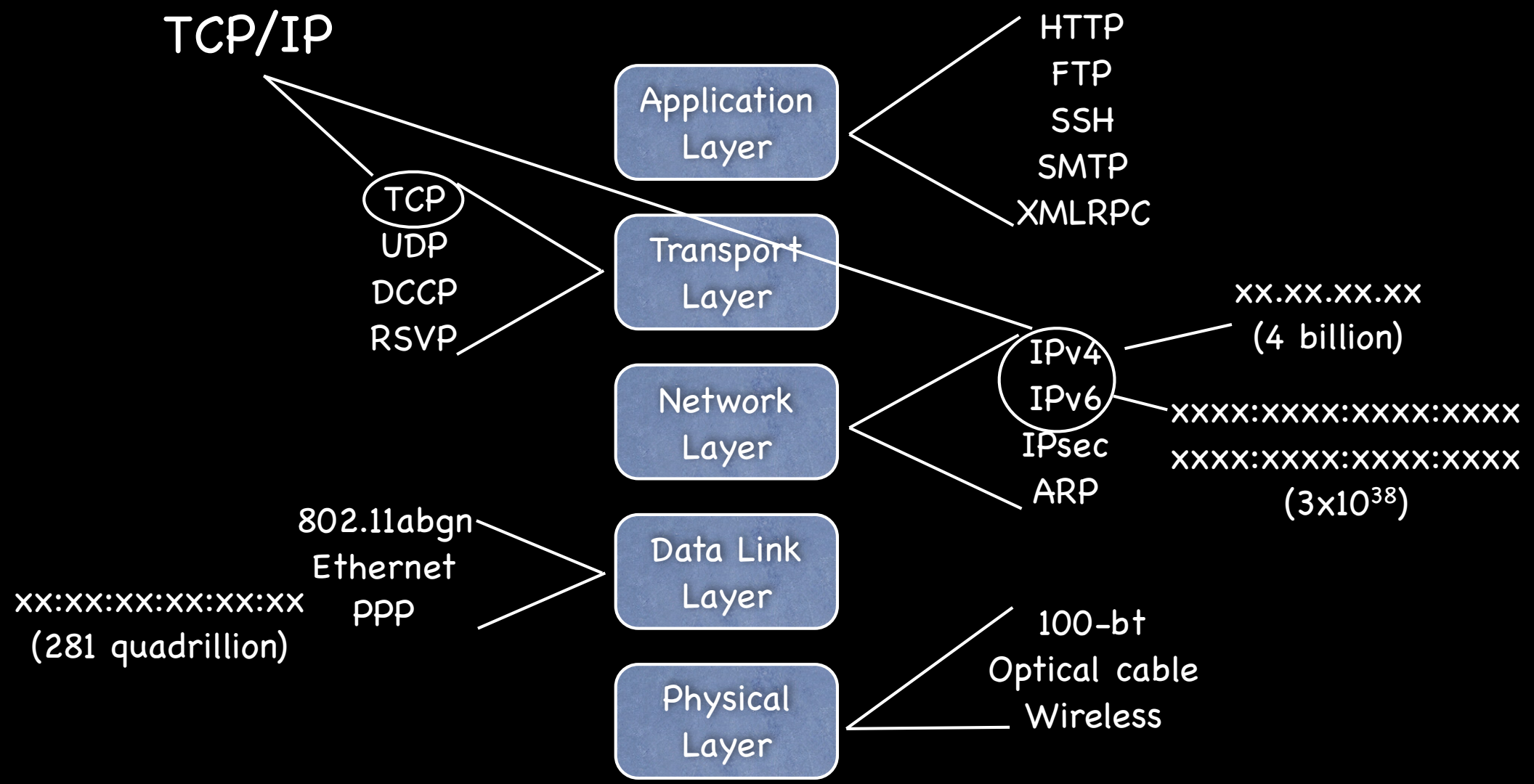
  <P>This is really just some test text to demonstrate how HTML works. I can do interesting things like italicize or make text bold, or even both together. ta da

  </BODY>

# CSS

- Cascading Style Sheet
  - Used to present websites in a uniform way
  - Define how a page looks, then use the definition on many pages
  - <link rel="stylesheet" type="text/css" charset="utf-8" media="all" href="/moin_static185/modern/css/common.css">
  - 
  - 

# Networking


TCP/IP
Application Layer
HTTP
FTP
SSH
SMTP
XMLRPC
TCP
UDP
DCCP
RSVP
Transport Layer
xx.xx.xx.xx
(4 billion)
IPv4
IPv6
xxxx:xxxx:xxxx:xxxx
xxxx:xxxx:xxxx:xxxx
(3x10^38)
Network Layer
IPsec
ARP
802.11abgn
Ethernet
PPP
xx:xx:xx:xx:xx:xx
(281 quadrillion)
Data Link Layer
100-bt
Optical cable
Wireless
Physical Layer

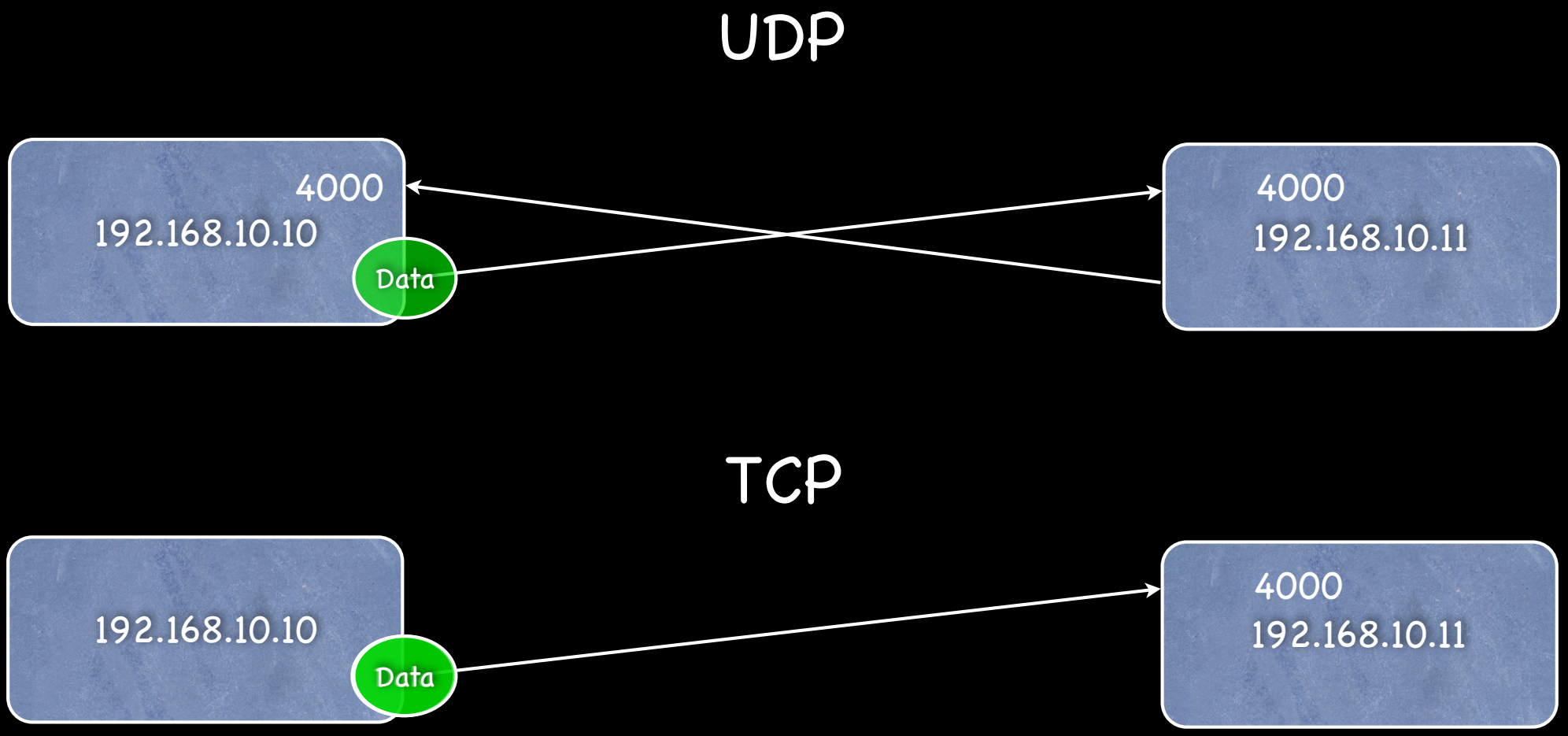
# IPv4 Network Parameters

- IP Address - Computer's unique* address  (x.x.x.x)
- Netmask - defines local 'subnet', machines the computer can speak to 'directly'
- Router - Address used to contact machines outside subnet
- DNS Server - Address where names can be mapped to addresses
- Port - For a specific connection 0-65535, 0-1023 reserved for system services

# Common Services

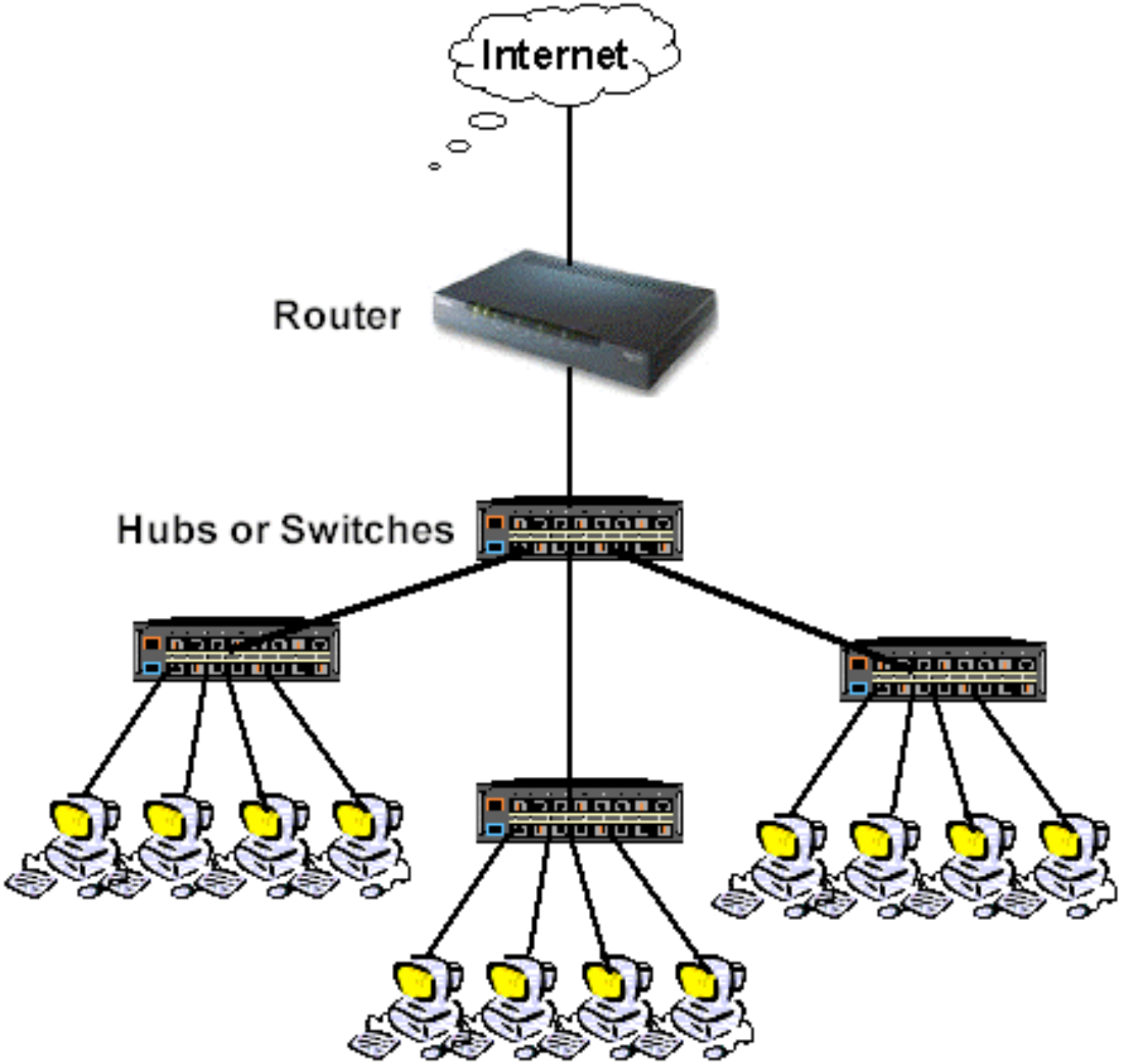
| port | service |
|---|---|
| 21 | ftp |
| 22 | ssh |
| 23 | telnet |
| 25 | smtp (mail) |
| 79 | finger |
| 80 | http (web) |
| 110 | pop3 (email retrieval) |
| 123 | ntp (time) |
| 137-139 | Windows file sharing |
| 143 | imap (email retrieval) |
| 443 | https (secure http) |

# Sockets (TCP/UDP)



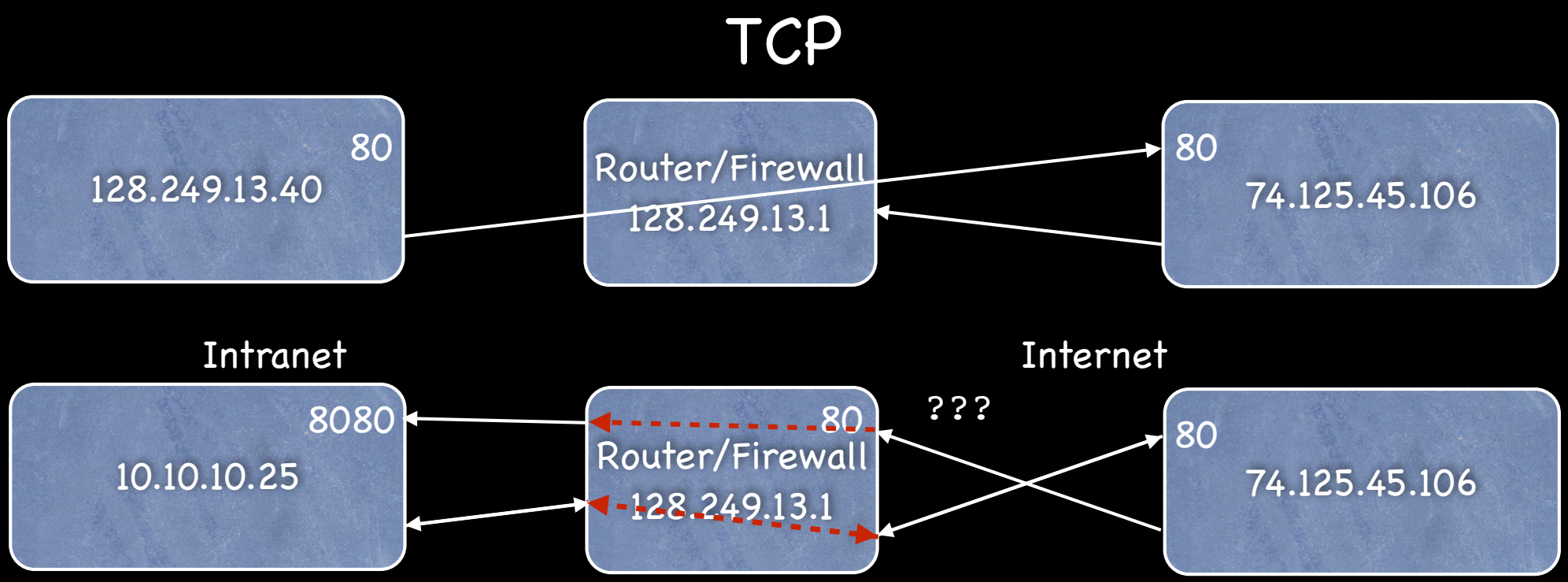
# IPv4 Network Parameters

- IP Address - Computer's unique* address  (x.x.x.x)
- Netmask - defines local 'subnet', machines the computer can speak to 'directly'
- Router - Address used to contact machines outside subnet
- DNS Server - Address where names can be mapped to addresses
- Port - For a specific connection 0-65535, 0-1023 reserved for system services




http://www.practicallynetworked.com/networking/port_expand.htm

# NAT


TCP
80
128.249.13.40
Router/Firewall
128.249.13.1
80
74.125.45.106
Intranet
Internet
8080
10.10.10.25
80
Router/Firewall
128.249.13.1
???
80
74.125.45.106

# Glossary

- HTTP - Hypertext Transport Protocol
- HTML - Hypertext Markup Lanugage
- XML - Extensible Markup Language
  - SGML - Parent of XML & HTML
- JavaScript - Python-like language on webpages (NOT Java)
- JSON - JavaScript Object Notation
- AJAX - Asynchronous JavaScript and XML (2005 Google Maps)
- AJAJ - Asynchronous JavaScript and JSON
- CSS - Cascading Style Sheet
- RSS - Really Simple Syndication

# Homework 7

- Write a program that retrieves something from the web and processes it in some useful fashion (easier if you find an XML or RSS site). Turn in the program and a brief description of what it retrieves and what you do with the results.