CLASS PROJECTS

If you are taking the class for a grade, class projects must be submitted to me (not the TA) by email following the *exact instructions* on the class website, by 11:59pm on Saturday, Feb 28. Please read the instructions this week !

Class projects will be presented on
Monday, March 2 from 9:00 - 11:30 am

We will do all of the presentations in this one extended session. Much like a national meeting, you will have 1 minute to get your laptop working with the projector (I suggest testing it ahead of time), and 5 minutes to present. We may have time for a question or two.

The graduate school does not permit auditors to participate in exams. You are welcome to observe the presentations if you like, though.

What's left…

- Today - Image Processing, Networking, XML
- Monday 2/9 - Lab - Networking
- Friday 2/13 - GUI development
- Monday 2/16 - No Class, Holiday
- Friday 2/20 - Optimization, profiling, debugging
- Monday 2/23 - Lab - Making
- Friday 2/27 - Databases

- Saturday 2/28, 11:59 pm - Class projects due!
- Monday 3/2 - Class project presentations

Lecture 9

Network Programming
XML
PIL (Image Processing)

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

Lab 3 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()
```

Lab 3 Review

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

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

Homework Review

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

data=np.loadtxt(argv[1]).transpose()
x=data[1]
y=data[2]
A = np.vstack([x, np.ones(len(x))]).transpose()
m,b = np.linalg.lstsq(A, y)[0]

plt.plot(x,y,"o")
plt.plot(x,x*m+b,"-",label="fit: y={:1.3g}x+{:1.3g}".format(m,b))

plt.legend()
plt.show()
```

http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.lstsq.html

Homework Review

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

data=np.loadtxt(argv[1]).transpose()
x=data[1]
y=data[2]
m,b,r,p,err = stats.linregress(x,y)

plt.plot(x,y,"o")
plt.plot(x,x*m+b,"-",label="fit: y={:1.3g}x+{:1.3g}".format(m,b))

plt.legend()
plt.show()
```

http://mathworld.wolfram.com/LeastSquaresFitting.html

Homework Review

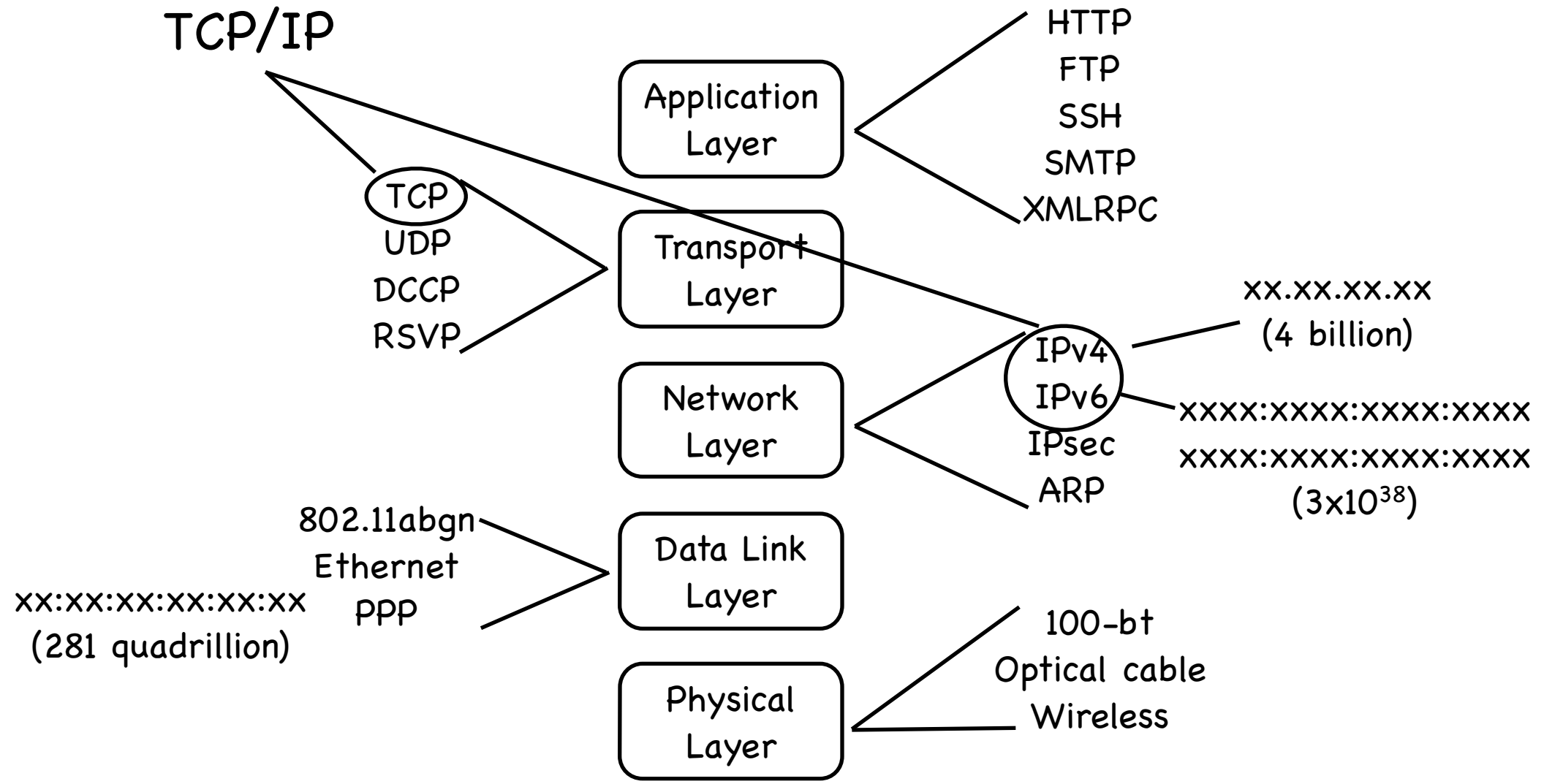
```
x & y are vectors, note how you can do math with them easily

xm=x.mean()
ym=y.mean()
sxx=sum((x-xm)**2)
sxy=sum((x-xm)*(y-ym))

m=sxy/sxx
b=ym-m*xm
```

http://mathworld.wolfram.com/LeastSquaresFitting.html

Homework Review

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

data=np.loadtxt(argv[1]).transpose()
x=data[1]
y=data[2]

xm=x.mean()
ym=y.mean()
sxx=sum((x-xm)**2)
sxy=sum((x-xm)*(y-ym))
m=sxy/sxx
b=ym-m*xm

plt.plot(x,y,"o")
plt.plot(x,x*m+b,"-",label="fit: y={:1.3g}x+{:1.3g}".format(m,b))

plt.legend()
plt.show()
```

Networking


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

IPv4 Network Parameters

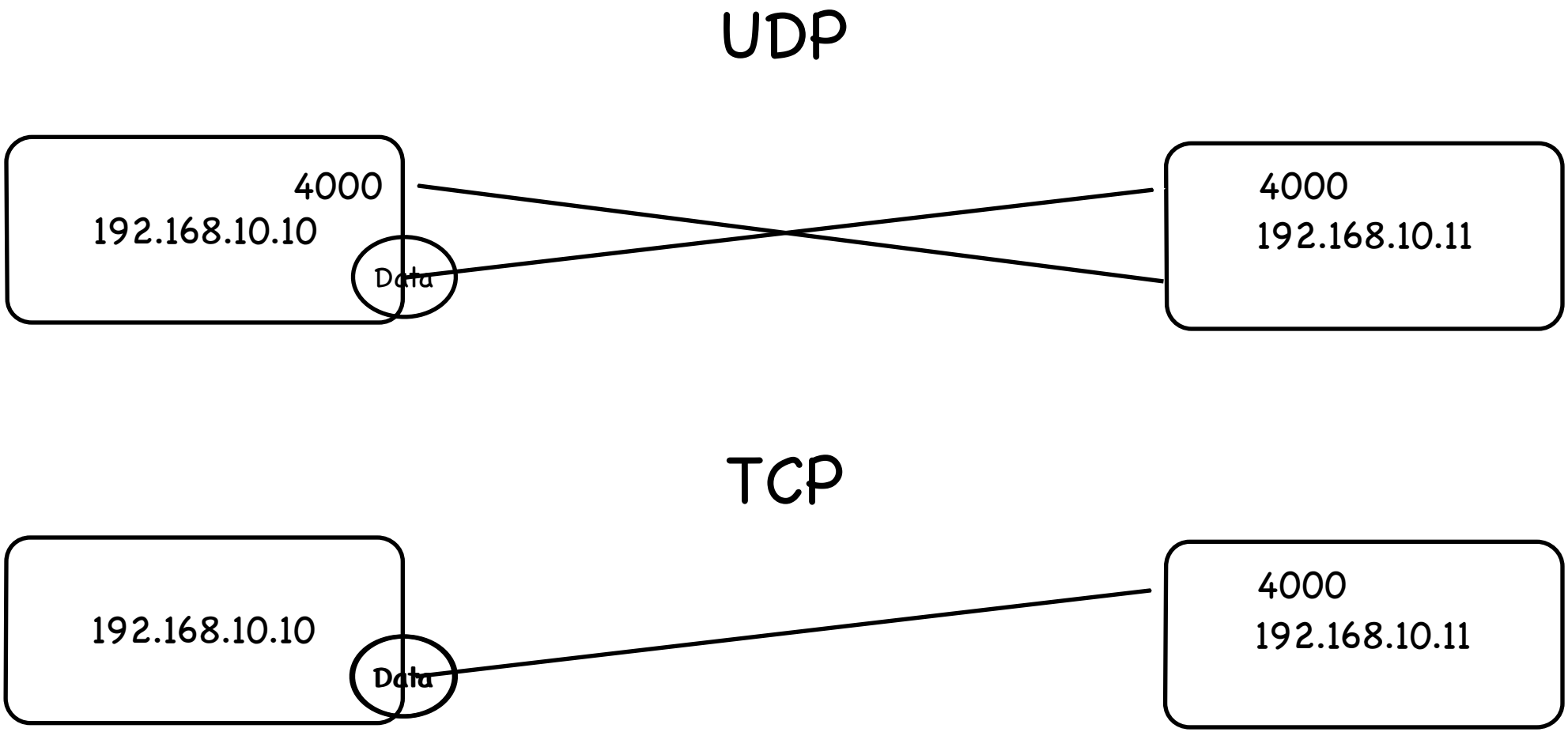
- IP Address - Computer's unique* address (x.x.x.x)
- Netmask - defines local 'subnet', machines the computer can speak to 'directly'
- Router - Address used to contact machines outside subnet
- DNS Server - Address where names can be mapped to addresses
- Port - For a specific connection 0-65535, 0-1023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

* - Some addresses are for 'private networks'. These are 10.*.*.*, 172.16.*.*–172.31.*.*, and 192.168.*.*

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)

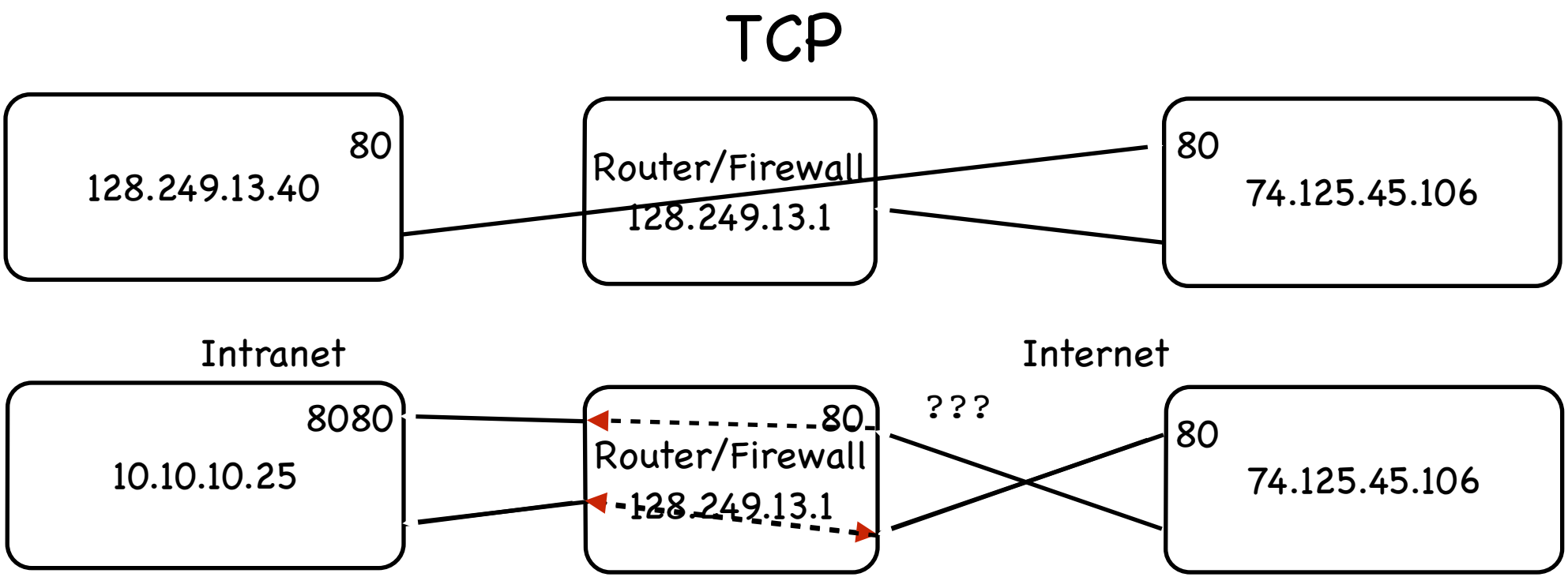


IPv4 Network Parameters

- 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

* - Some addresses are for 'private networks'. These are 10.*.*.*, 172.16.*.*–172.31.*.*, and 192.168.*.*

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

urllib2

```
import urllib2

f=urllib2.urlopen("http://blake.bcm.edu/dl/test.html")

for i in f: print i
```

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>

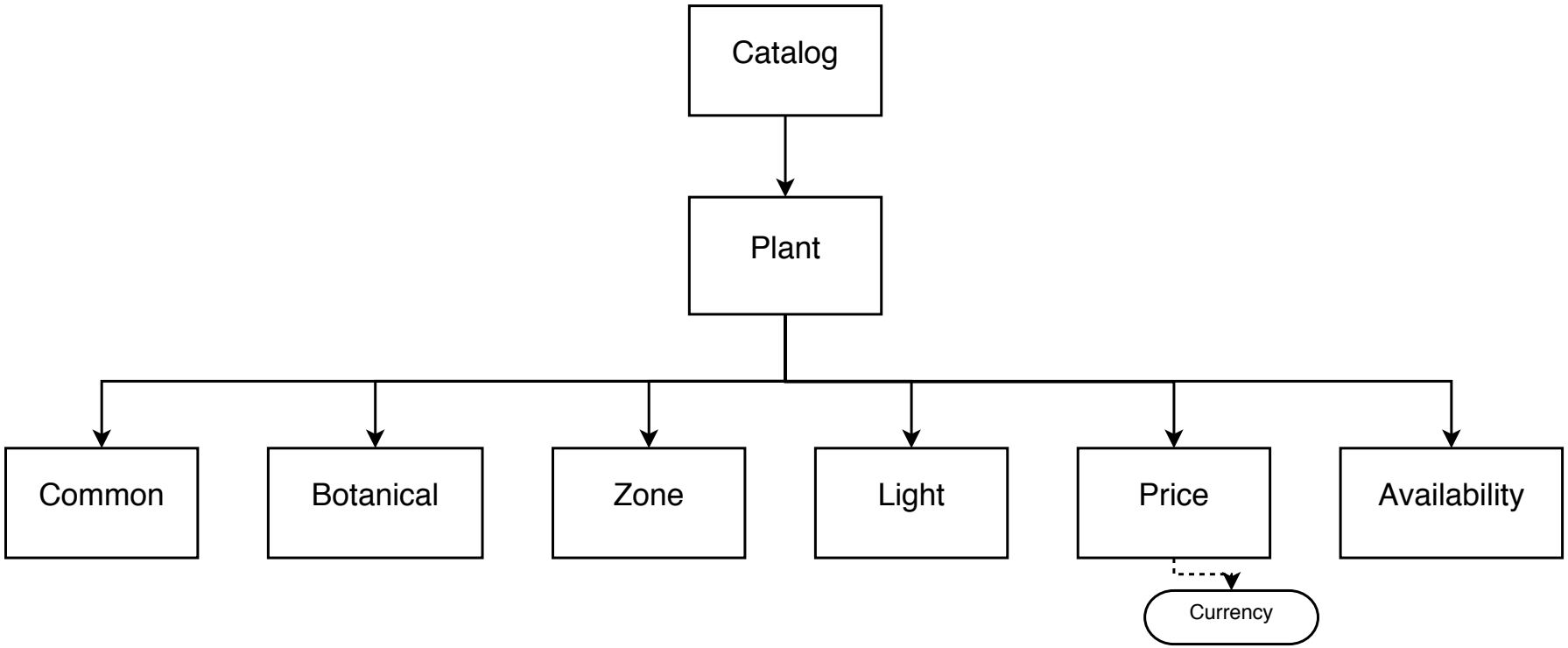
  <P>This is really just some test text to demonstrate how HTML works. I can do interesting things like italicize or make text bold, or even both together. ta da

  </BODY>

XML Basics

- <?xml version="1.0" encoding="UTF-8" ?>
- Tags
  - <tag> content </tag> or <tag />
- Attributes
  - <tag attr1=”value” attr2=”value2”> </tag>
- Nesting
  - <tag 1>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.cElementTree   (or xml.etree.ElementTree)
et=xml.etree.cElementTree.parse("xml_example.xml")
et=xml.etree.cElementTree.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
```

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

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

General Image Processing

- PIL/PILLOW (today)
- SciPy 'ndimage' module
  - Apply NumPy capabilities to image processing
- OpenCV
  - Computer vision library with Python bindings
- EMAN2
  - Greyscale quantitative image processing
  - Links to structural biology (CyroEM)
- GIMP
  - Free Photoshop-like, cross-platform
  - Python based 'modules'

PIL/PILLOW

- Python Imaging Library
- Open-source/commercial
- PIL unofficially dead, PILLOW is the replacement
- Reads-writes many standard formats
- Generic image processing

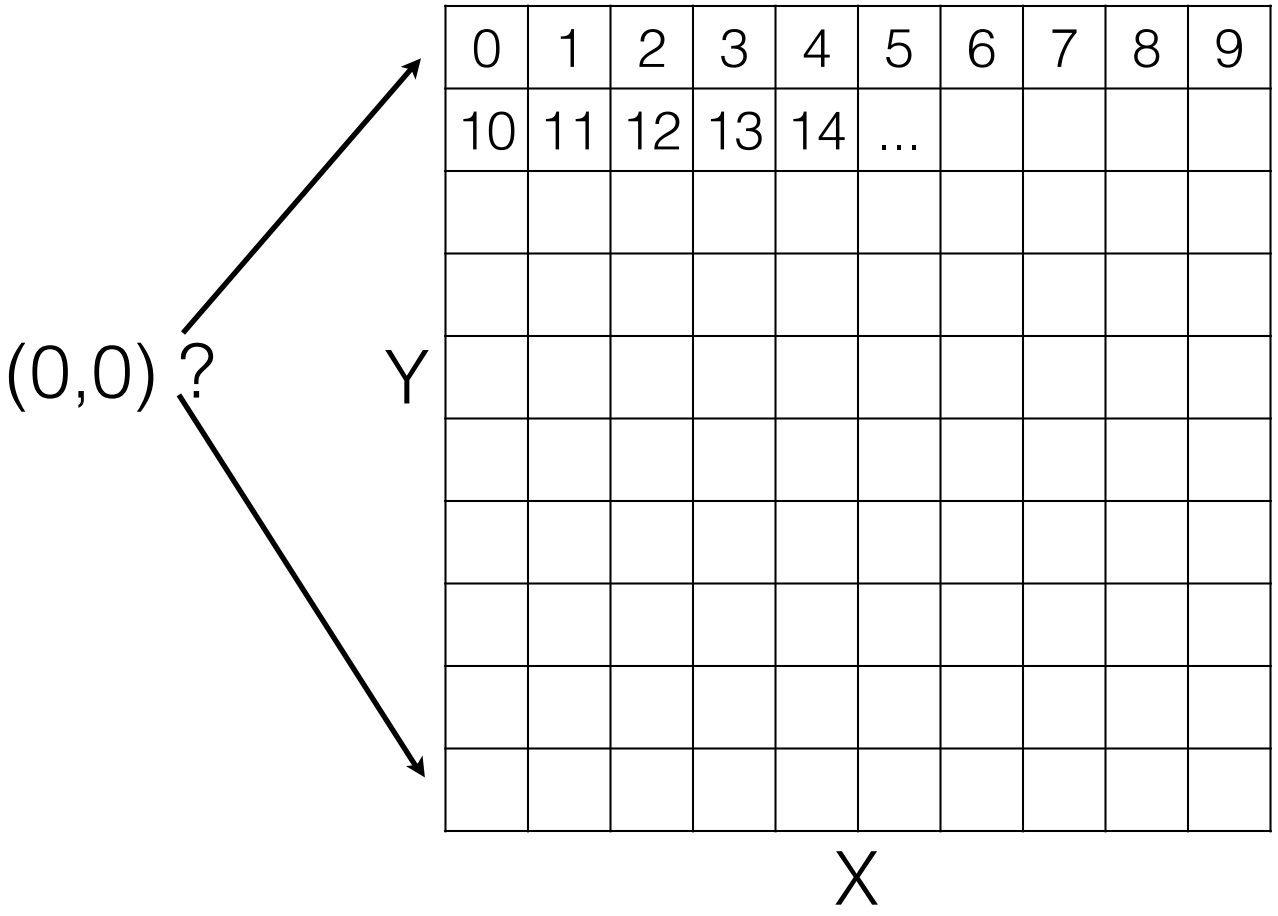
Installing PIL

- https://pypi.python.org/pypi/Pillow
- Windows:
  - Standard installer packages should work
- Linux
  - Should be in package installer (may need to look for python imaging)
- Mac
  - Make sure you have Xcode & command-line tools installed
  - Install libjpeg (http://www.ijg.org/)
    - http://snippets.dzone.com/posts/show/38
  - sudo easy_install pillow
  - If this doesn't work, you may want to try the instructions on either the pillow or the pil website.

- to test: 'import PIL'

Images

Pixel stored at location x+nx*y (row major)
or y+ny*x (column major, less common)



Color Images

X	X	X
X	X	X
X	X	X

?

Planar, 3x3 image, row-major

R	R	R
R	R	R
R	R	R
G	G	G
G	G	G
G	G	G
B	B	B
B	B	B
B	B	B

Interleaved, 3x3 image, row-major

R	G	B	R	G	B	R	G	B
R	G	B	R	G	B	R	G	B
R	G	B	R	G	B	R	G	B

PIL - File Formats*

Fmt	Bits	Loss	Cmpr	Notes
BMP	8 or less			
GIF	8 total, cmap	X	X	
IM	all modes !			LabEye & IFUNC
JPEG	8	X	X	
PCX	8 or less			
PNG	8		X	
PPM	8			PBM,PGM,PPM
TIFF	8	R	R	
EPS				Needs GS to read most
PDF				Write only

* - Only the most common ones

Image Modes

- 1 (1-bit pixels, black and white, stored with one pixel per byte)
- L (8-bit pixels, black and white)
- P (8-bit pixels, mapped to any other mode using a colour palette)
- RGB (3x8-bit pixels, true colour)
- RGBA (4x8-bit pixels, true colour with transparency mask)
- RGBX (3x8-bit pixels, true colour with padding byte)
- CMYK (4x8-bit pixels, colour separation)
- YCbCr (3x8-bit pixels, colour video format)
- I (32-bit signed integer pixels
- F (32-bit floating point pixels)

PIL

```
from PIL import Image
im=Image.open("file.jpg")
data="\0"*(128*128*4)        # string of zero pixels
from array import array
data=array("c",data)         # convert to an array object
im=Image.frombuffer("RGBX",(128,128),data,"raw","RGBX",0,1)
im.show()                    # machine specific display
pix=im.load()                # for pixel access
pix[x,y]                     # access pixel at x,y
im.save(filename,[format],[options])
```

Using Numpy

```
from numpy import *

from PIL import Image

a=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(128,128))

im=Image.fromarray(a)

im.show()                # Image is black !?!?

a+=1

a*=127

im2=Image.fromarray(a)

im2.show()
```

PIL

```
im=Image.open("hh.jpg")

a=array(im)

a[0,0]

im2=Image.fromarray(a)
```

PIL

```
a=fromfunction(lambda x,y:(127+sin(x/100.)*127., 127+cos(y/
100.)*127.,127+sin(x/500.)*127.),(256,256))

b=dstack(a)

c=b.astype(uint8)

im=Image.fromarray(c)

im.show()
```

PIL

- ImageChops - invert(a), lighter(a,b), darker(a,b), add(a,b), subtract(a,b), difference(a,b), screen(a,b)
- ImageDraw
  - a=Image.new(“RGBA”,(128,128))
  - draw=ImageDraw.Draw(a)
  - draw.line((x0,y0,x1,y1),fill=”red”), point, rectangle, arc, chord, ellipse, text
- ImageFilter - BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN
- etc.

PIL Attributes

- im.format
- im.mode
- im.size
- im.info

Images in MatPlotLib

```
ipython -pylab

im=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(64,64))

imshow(im)

imshow(im,cmap=cm.gray)

savefig("a.png")
```

Homework 5

- 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 it does with it