Lecture 6

Complex Data Representations

Steven Ludtke
N420, sludtke@bcm.edu

Despair (don't)

1) "I'M NOT LEARNING ANYTHING, I CAN'T DO THE HOMEWORK !
If you're spending time trying to figure out the homework, you're learning more than you think.

2) "I'M GOING TO FAIL !"
As long as you keep trying, and turn in your attempts to figure it out, you will stay above the dreaded C.

Metric for Success?

- Objective measures? What skills do you have?
- If you leave the class with more programming skills than you entered the class with, I will be satisfied
- I would MUCH rather see an honest, but unsuccessful attempt to solve a homework problem than a successful program you copied online or got from a friend!

Intellectual Property, Plagiarism and Copyright in Programming

- Don't be afraid to use the web to find conceptual answers to your problems!
- Of course it is plagiarism to present someone else's code as your own. Add a comment containing the URL, and author's name if available.
- In general, people expect code posted publicly will be copied, but, if you are copying code verbatim, check the copyright.

Team Learning 5

```
import numpy as np

x=np.arange(0,pi*2.0,0.05)
y=sin(x)
print(x)
print(y)
```

Team Learning 5

```
x=np.arange(0,np.pi*2.0,0.05)
y=np.sin(x)
a=np.loadtxt("curve.txt")

from bokeh.plotting import figure, output_notebook, show

output_notebook()      # output to Jupiter

p = figure()   # create a new plot
p.line(x, y, legend="y=sin(x)")
p.circle(x, y,fill_color="white", size=5)
show(p)
```

extra credit: See if you can plot the curve from the file "b" and sin(x) covering the same range as "b" on the same plot.

Homework Review

- Use BioPython to write a program that prompts the user for a protein sequence, runs a BLAST on the sequence, then prints the "RecName" for the top 5 hits (if there are that many).

```
from Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML

sequence=input("Sequence:")
result=NCBIWWW.qblast("blastp","swissprot",sequence)
blast_record = NCBIXML.read(result)
result.close()
```

Homework Review

```
for i in range(5):
    title=blast_record.descriptions[i].title
    recname1=title.split("RecName: ")[1]
    recname=recname1.split(";")[0]
    print(recname)
```

Homework Review

```
for i in range(5):
    print(blast_record.descriptions[i].title.split("RecName: ")[1].split(";")[0])
```

Homework Review

- Use BioPython to write a program that prompts the user for a PubMed query (like the "Ludtke SJ[Author]" from lecture). Print the number of matches, and the Title, First Author Name, Last Author Name for the first 5 returned results

Homework Review

```
from Bio import Entrez
from Bio import Medline

query=input("Pubmed query: ")
Entrez.email = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term=query, retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

handle=Entrez.efetch(db="pubmed",id=ids[0],rettype="medline")
records = list(Medline.parse(handle))
handle.close()
```

Homework Review

```
from Bio import Entrez
from Bio import Medline

query=input("Pubmed query: ")
Entrez.email = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term=query, retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

records=[]
for id in ids[:5]
    handle=Entrez.efetch(db="pubmed",id=id,rettype="medline")
    records = list(Medline.parse(handle))
    handle.close()
    records.append(record[0])
```

Homework Review

```
from Bio import Entrez
from Bio import Medline

query=input("Pubmed query: ")
Entrez.email = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term=query, retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

handle=Entrez.efetch(db="pubmed",id=ids[0:5],rettype="medline")
records = list(Medline.parse(handle))
handle.close()
```

Homework Review

```
from Bio import Entrez
from Bio import Medline

query=input("Pubmed query: ")
Entrez.email = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term=query, retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

handle=Entrez.efetch(db="pubmed",id=ids[0:5],rettype="medline")
records = list(Medline.parse(handle))
handle.close()

for record in records:
    print(record["TI"])
    print(record["AU"][0],"\t",record["AU"][-1])
```

Dictionaries

- keys must be immutable, values can be any type
- { k1:v1, k2:v2, k3:v3, ... }

  Example:

  a={ 1:2,2:3,”a”:”b”,2.0:3.2,(1,2):”really?” }

  a[1] -> 2

  a[(1,2)] -> “really?”

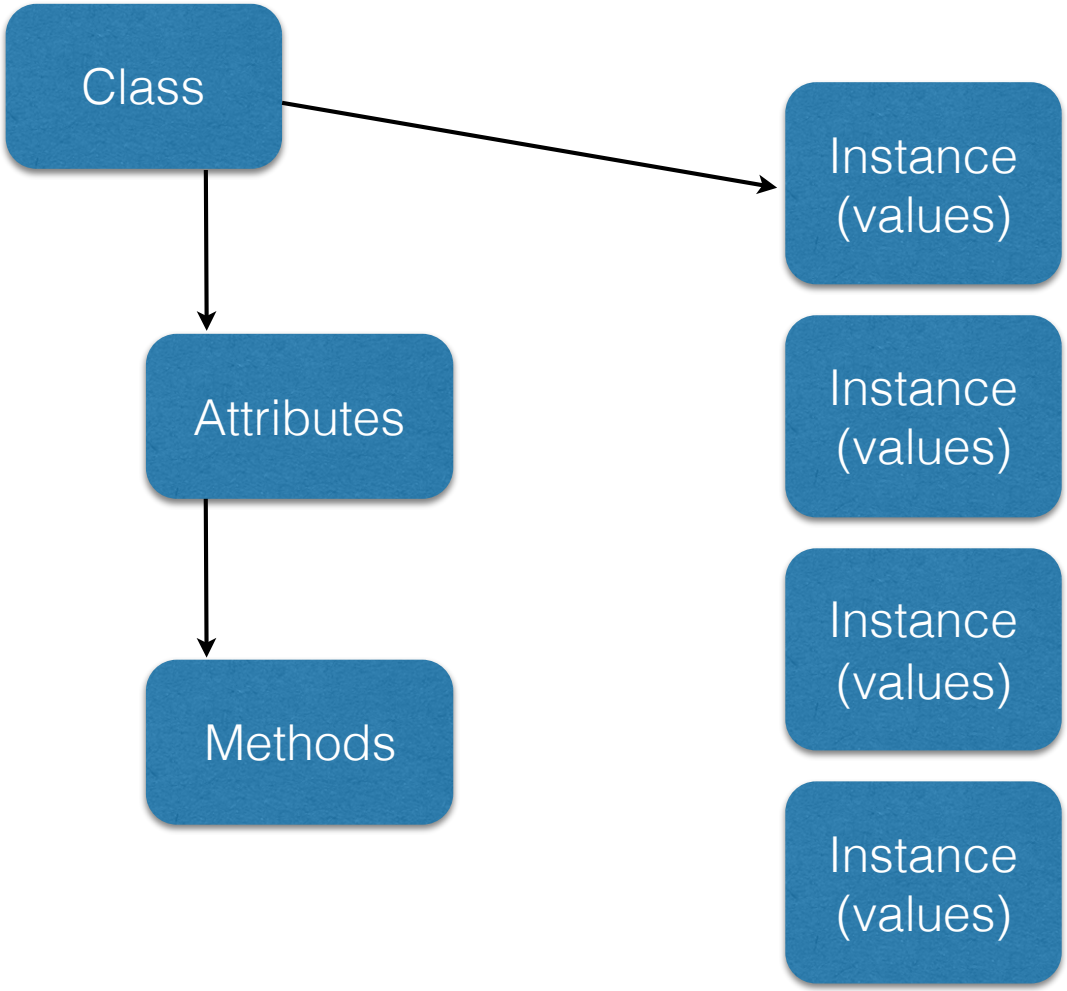
  a[2] -> 3.2
- Methods:
  - keys, values, items
  - set_default

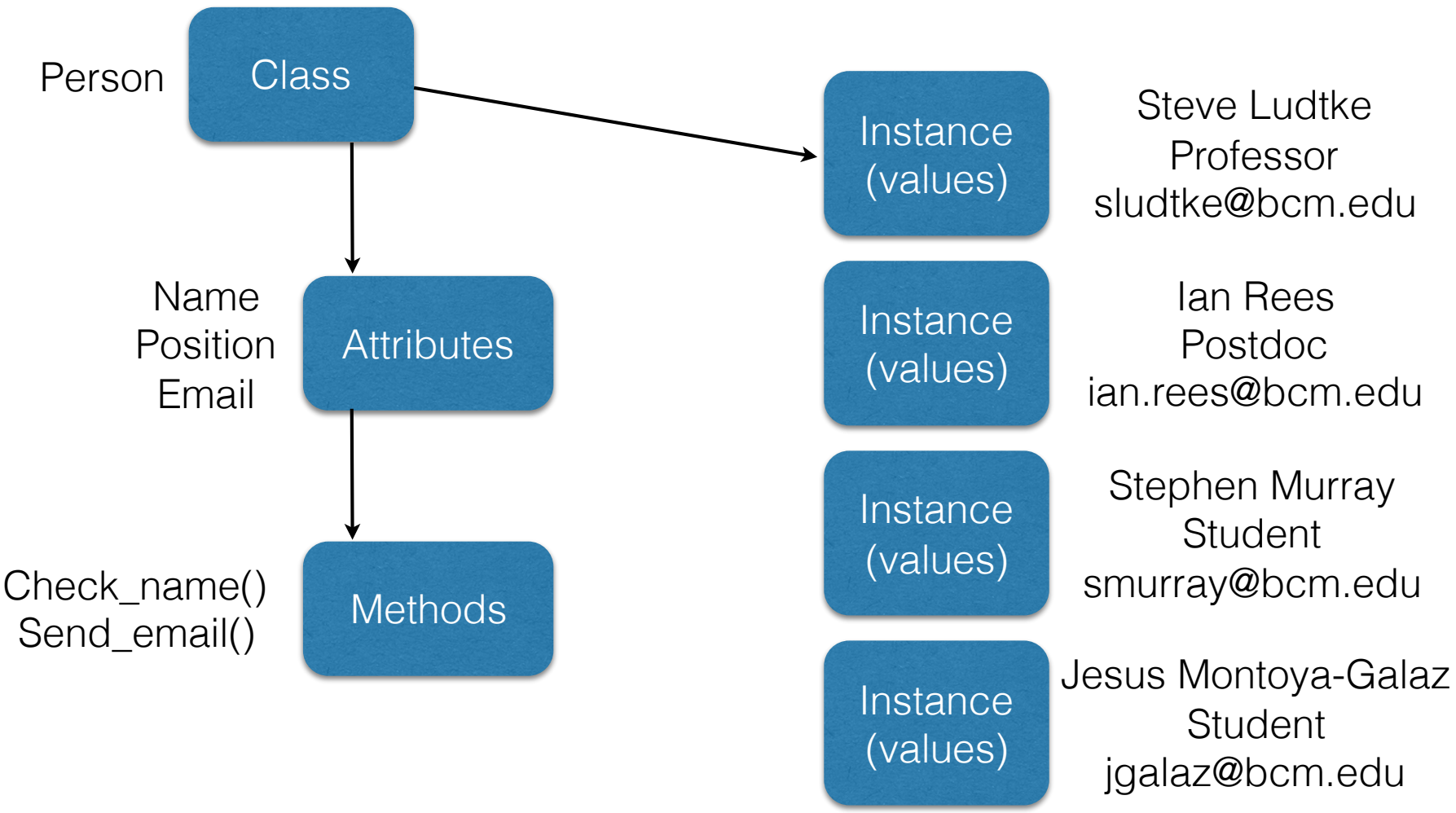
Object Oriented Programming (A Quick Introduction)

- 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
```

Scope

What will this produce ?

```
class abc:
  xyz=10
  def __init__(self) : self.qqq=5

  def f(self):
    # Which of the next 3 lines is correct?
    print(self.xyz)
    print(abc.xyz)
    print(xyz)

x=abc()
y=abc()
x.f()
x.xyz=15
y.f()
print(x.xyz,y.xyz,abc.xyz)   # What will this give?
```

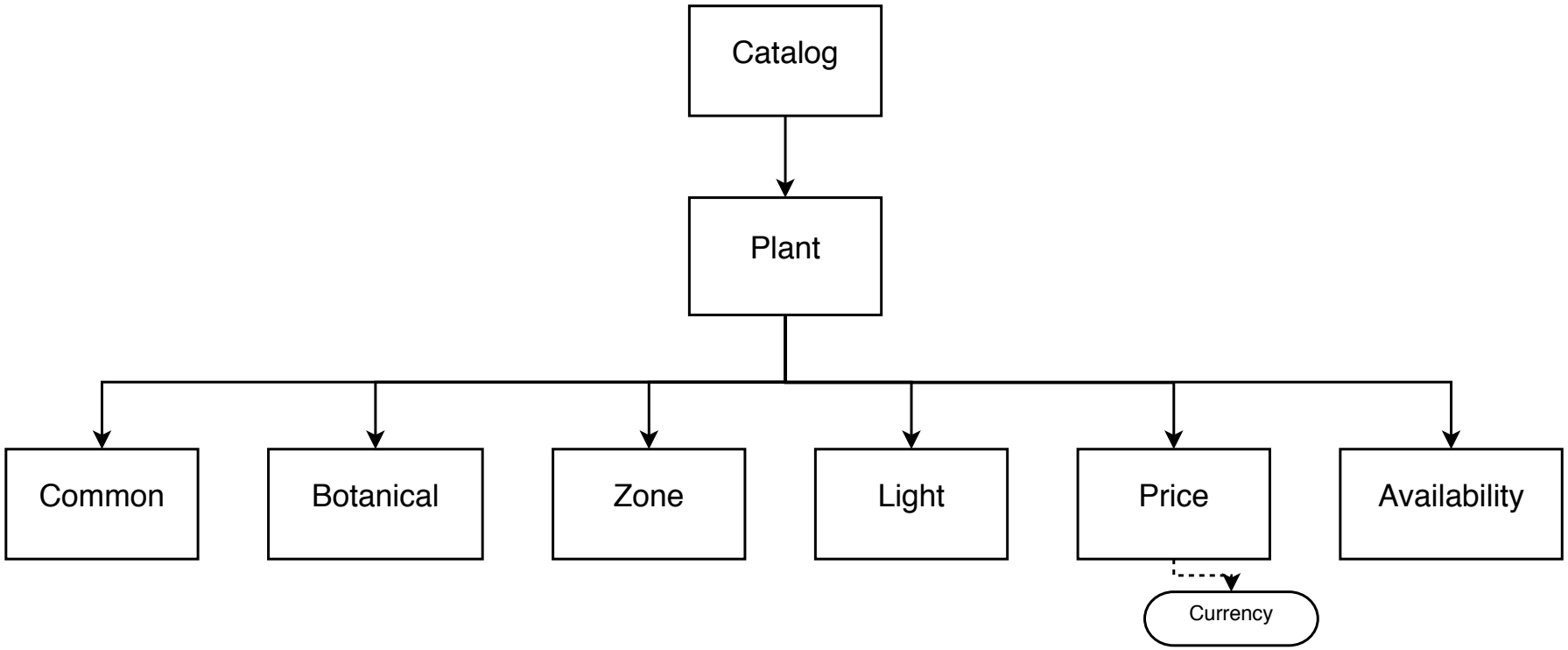
Things we skipped

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

XML

- <?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")
e=et.getroot()          # The root object in the XML file
e=xml.etree.ElementTree.fromstring("XML CODE")
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

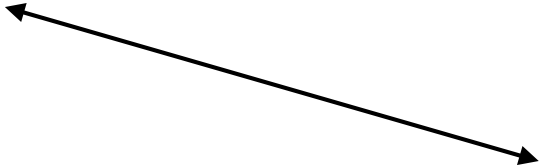
- How could we represent this data in a computer?
- Hierarchical or Flat?

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

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>

HTML

<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 make text bold, or even both together. ta da

</BODY>

</HTML>

urllib

Web Scraping:

```
from urllib.request import urlopen

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

for line in conn: print(str(line,"utf-8"))
```

urllib

```
from urllib.request import urlopen
conn=urlopen("https://www.rcsb.org/pdb/rest/describePDB?
structureId=4hhb")
data=conn.read()
conn.close()

import xml.etree.ElementTree
et=xml.etree.ElementTree.fromstring(data)
```

Homework 6

- In a separate email from the normal homework, email me (just me, not the TAs) a description of your planned class project. The description should include:
- Overall goal/motivation of the program. Why can't you do it with existing tools. Target ~3-5 sentences.
- inputs - What inputs or data sources will the program use, and what type of data/file is it?
- outputs - What will the (useful) output of the program be

Homework 6

- 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). Add a Markdown Cell to your notebook explaining what you retrieved and how/why you did what you did with the result