CSC 110: Week 8, Lecture 1

Dr. Bodine

Last week…

- Lists, lists and more lists….
- Storing sequential data with arrays/lists
- List: methods and operations
- Dictionaries for non-sequential data
    - Key-value pairs

This week..

- Objects
- Object-oriented programming
- Classes
- Encapsulation
- (Inheritance & polymorphism)

Objects

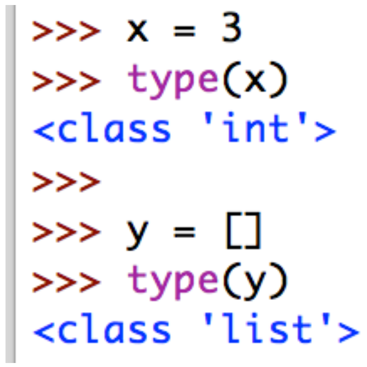
- Can think of **objects** as “Active data type”
  - Collection of related data
  - Set of operations to manipulate that data

Objects continued…

- Attributes of the object
  - Instance variables
    - Information that is specify to that instance
  - Methods
    - Functions that manipulate the information

Object examples

- EVERYTHING in python is an object
- Ints, Floats, Strings, Lists, Dictionaries, ...
- What is this "class" business???
  - We will get there later today...



```
>>> x = 3
>>> type(x)
<class 'int'>
>>>
>>> y = []
>>> type(y)
<class 'list'>
```

What is Object-oriented Programming (OOP)?

- Programmer defines not only the structure of data that is to be stored but also what operations can be done on the data

- How do we specify this information?
  - Depends on the language!

Classes and OOP

- **Classes** are common in OOP languages (but not universal!)
  - Python, Java, C++, PHP, and many more

- Classes comprise the information to store and procedures to manipulate the information

- Objects are *instances of a class*

A simple example

- Consider the lists we have been using.

```
mylist = [ 1 , 2 , 3 ]
```

- What does this line do for us?
  - Creates a list type object

- Now we can manipulate the information by accessing the list methods

```
mylist.append(4)
```

A closer look…

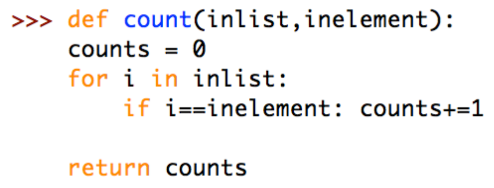
```
mylist = [ 1 , 2 , 3 ]
mylist.append(4)
```

- The object mylist is an instantiation of the list class
- append is a method--a function that lives inside the class

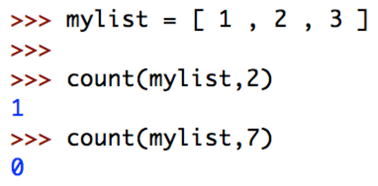
Methods

- Last week we did some coding to replicate list methods

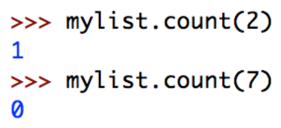
- Example: replacing count(x) with a function that took the list as a parameter



```
>>> def count(inlist,inelement):
    counts = 0
    for i in inlist:
        if i==inelement: counts+=1

    return counts
```

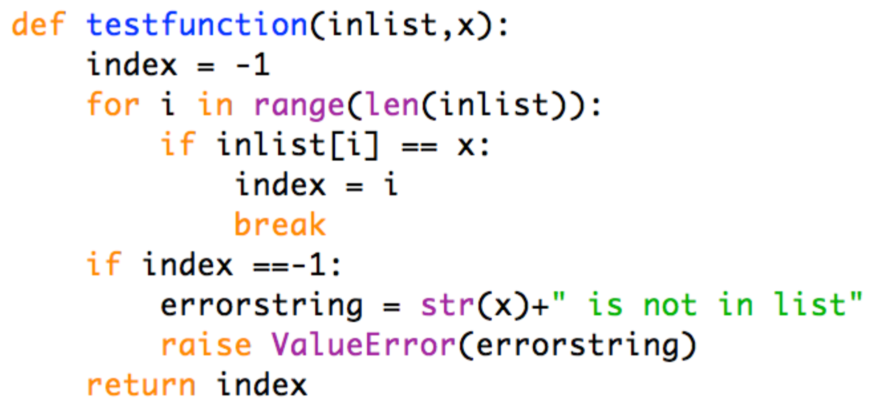


```
>>> mylist = [ 1 , 2 , 3 ]
>>>
>>> count(mylist,2)
1
>>> count(mylist,7)
0
```



```
>>> mylist.count(2)
1
>>> mylist.count(7)
0
```

Think-pair-share



```python
def testfunction(inlist,x):
    index = -1
    for i in range(len(inlist)):
        if inlist[i] == x:
            index = i
            break
    if index ==-1:
        errorstring = str(x)+" is not in list"
        raise ValueError(errorstring)
    return index
```

- Can you identify what built-in method it mimics?
- How would you use this function? How would you use the method?
- How does using this function differ from using the method?

Our previous coding…

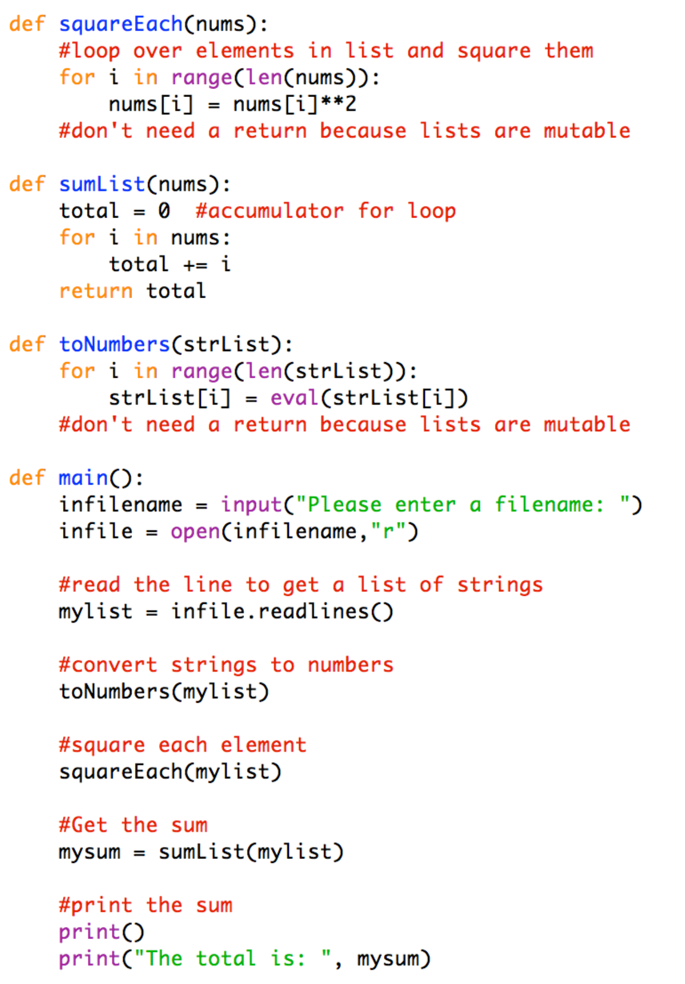
- We have been writing **modules** for our homework and in class
  - Run from the editor or import in shell

- Sometimes contained multiple functions
  - Think of our toNumbers, squareEach, sumList assignment
- Could use the functions we defined inside by using the dot

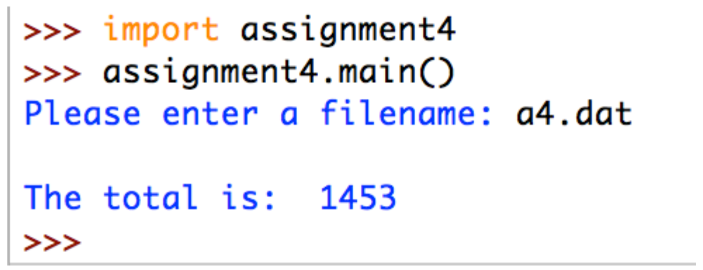
```
<module name>.<function name>
```

Details from assignment 4

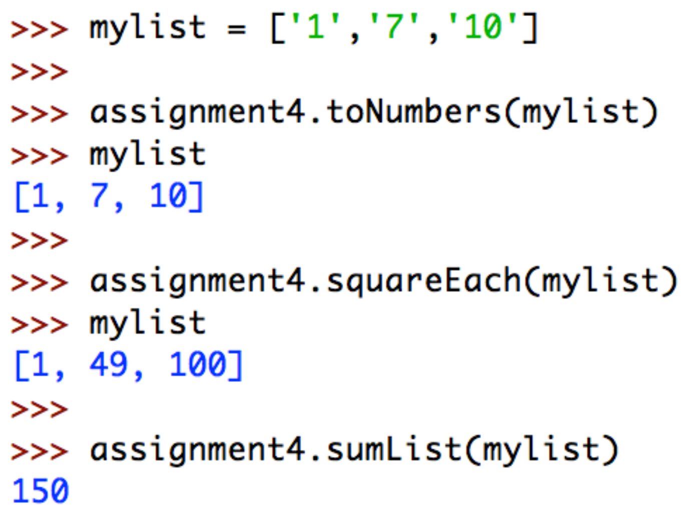
- Say we saved our code as assignment4
- We had multiple functions defined
    - toNumbers(strList)
    - squareEach(nums)
    - sumList(nums)
- We called them inside of our main function
- If we import the module, we can also access them by themselves to act on lists we are manipulating.



```python
def squareEach(nums):
    #loop over elements in list and square them
    for i in range(len(nums)):
        nums[i] = nums[i]**2
    #don't need a return because lists are mutable

def sumList(nums):
    total = 0  #accumulator for loop
    for i in nums:
        total += i
    return total

def toNumbers(strList):
    for i in range(len(strList)):
        strList[i] = eval(strList[i])
    #don't need a return because lists are mutable

def main():
    infilename = input("Please enter a filename: ")
    infile = open(infilename,"r")

    #read the line to get a list of strings
    mylist = infile.readlines()

    #convert strings to numbers
    toNumbers(mylist)

    #square each element
    squareEach(mylist)

    #Get the sum
    mysum = sumList(mylist)

    #print the sum
    print()
    print("The total is: ", mysum)
```



```
>>> import assignment4
>>> assignment4.main()
Please enter a filename: a4.dat

The total is:  1453
>>>
```



```
>>> mylist = ['1','7','10']
>>>
>>> assignment4.toNumbers(mylist)
>>> mylist
[1, 7, 10]
>>>
>>> assignment4.squareEach(mylist)
>>> mylist
[1, 49, 100]
>>>
>>> assignment4.sumList(mylist)
150
```

Modules refresher

- Helpful for organizing code
  - Modular design, anyone….
- Import to get shell access to:
  - Functions
  - Variables declared outside of functions (what about inside?)
- Access information inside the module with the dot operator (.)

Python classes: definitions

```
class <classname>(<object>):
   <a statement>
   ...
   <another statement>
```

Python classes: definition example

```python
class MyClass(object):

    def __init__(self,name):
        self.name = name

    def hello(self):
        hellostring = "Hello, "+format(self.name)+"!"
        print(hellostring)
```

What is this `self` business?

- When we want the methods to have access to the information stored by the object when need to tell it that it can.

- Hence our def __init__(self) says the class has access to the information contained in itself. That may sound trivial…but it isn’t.

Python classes: instantiations

- What happens when we make a new object?
- For example, when we make a new MyClass object?
- The funcion __init__ is special—it always runs when we instantiate
- Do other functions run? Do they run in modules?

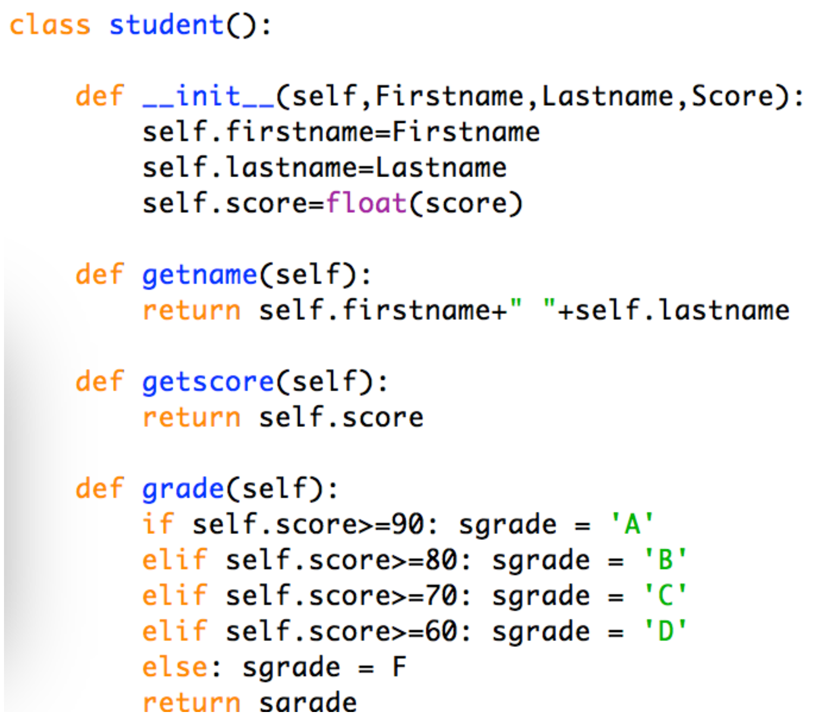
Updating our scores assignment

- Last week we saw what can happen when we try to store related data in lists—get out of sync
- Could fix this instance with a dictionary but what if we wanted to do something more complex?
- Let’s make a student class

Making our student class

- Need to know what information we want to store
  - Instance variables
  - First Name, Last Name, Score
- Need to know what we want to do with the information
  - Methods (functions)
  - Do we want to be able to get return the values?
  - Do we want to assign grades based on the values?
- Need to define an __init__ function

Our student class



```python
class student():

    def __init__(self,Firstname,Lastname,Score):
        self.firstname=Firstname
        self.lastname=Lastname
        self.score=float(score)

    def getname(self):
        return self.firstname+" "+self.lastname

    def getscore(self):
        return self.score

    def grade(self):
        if self.score>=90: sgrade = 'A'
        elif self.score>=80: sgrade = 'B'
        elif self.score>=70: sgrade = 'C'
        elif self.score>=60: sgrade = 'D'
        else: sgrade = F
        return sgrade
```

Now what do we do with the student object?

- We might want to store the information so we want do things with them like compute averages

- Can we just throw these students into a list?
  - [student_1, student_2, … student_n]