# CSC 103 – Spring 2017 Running-Notes

* \_\_repr\_\_()
  + Object(%r,%r) % (self.value, self.value1)
  + This returns a representation of the object that is Python Syntax Friendly
* \_\_str\_\_() ifabfe
  + Object(%s,%s\_ %s (self.value,self.value1)
  + Used to return a String representation of that object’s details
* String method comes before the representation method
* \_\_eq\_\_() 🡪 Need to compare the type of self and type(other), as well as values of self and other.
* Program Design Recipe
  + Data Definition: Define each input’s type (location is a string)…written as a comment above and in the function.
  + Purpose statement, signature/contract, header
    - (Household int int 🡪 String) #Signature: …
      * Show the input to output logic-line
    - (Return the income category) #purpose: …
      * Write out the return value of the function.
    - First line of the function with pass #header:…
      * Literally first line.
  + Test Cases important as well

Def class (unittest.TestCase):

Def test auhef

**Lists**

* Recursive representation of lists
  + Basically link them to each other and have “mt” as the value of the last node

class Pair:  
 def\_\_init\_\_(self,first,rest):

self.first = first

self.rest = rest

* Programming with lists
  + Example: Write a function that determines whether the word “dark” appears in a list of string in StrList.
* Functions on lists
* Recursive representation of trees
* Functions on trees

**Trees**

List of 3 nodes = Pair(2, Pair(27, Pair(22, “mt”)))