CSC148 - SUMMER 2017(Instructor: Sadia)

Chen Sun

Abstract data types and data structures for implementing them.

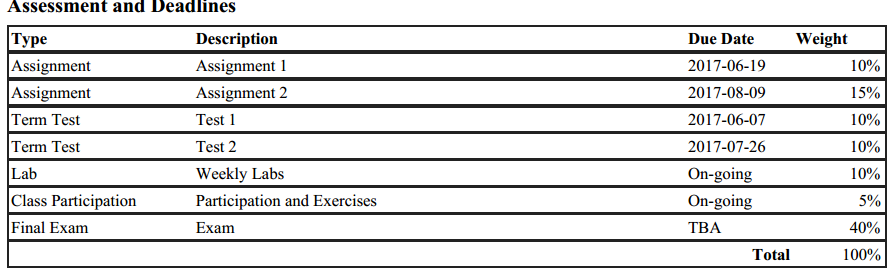
Linked data structures, queue, stack, tree, BST…

Encapsulation and information-hiding.

Object-oriented programming.

Specifications. Analyzing the efficiency of programs. Big O analysis

Recursion.



----------------------------

Lab – pair programming 1% each

Week 1

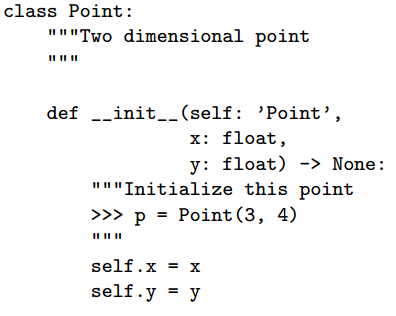
What is Object?

Everything in Python is an Object – like list, string and dictionary

Object have **attributes** and **methods**

Define a new Class

Noun as attribute, verb as methods



Object Oriented Analysis

* Find Object : Nouns
* Object Behavior : Methods
* Object States : instance variables
* Determine Object Interactions

Inheritance

* Copying-and-pasting from the old class, and then making a few changes, is a bad idea
* Inheritance allows a new class to specialize an existing class by specifying only what is different between them
* Inheritance models ‘is-a’ relationships : cat is an animal – animal is a superclass, cat is a subclass

To inherit from another class, add the superclass name in parentheses class InvisibleShip(Ship):

If the method exists in InvisibleShip, it is called, otherwise the one in ship is called

Magic methods for python classes:

* Comparing two instances of a class using >,<,==
  + Only if we have define \_\_lt\_\_, \_\_gt\_\_,\_\_le\_\_,\_\_eq\_\_...
  + To see if p1 less than p2, p1.\_\_lt\_\_(p2)
  + Similarly for \_\_add\_\_ and \_\_mult\_\_
* 新建Class : Define a Class −
* class Employee:
* 'Common base class for all employees'
* empCount = 0
* def \_\_init\_\_(self, name, salary):
* self.name = name
* self.salary = salary
* Employee.empCount += 1
* def displayCount(self):
* print ( ' Total Employee number is {} '.format(self.empCount) )
* def displayEmployee(self):
* print ( ' Name {} has a salary {}'.format(self.name, self.salary) )
* The variable *empCount* is a class variable whose value is shared among all instances of a this class. This can be accessed as *Employee.empCount* from inside the class or outside the class.
* The first method \_\_init\_\_() is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.
* 新建Instance Object: creating an instance −
* emp1 = Employee("Zara", 2000)
* emp2 = Employee("Manni", 5000)
* emp1.displayEmployee()
* emp2.displayEmployee()
* print "Total Employee %d" % Employee.empCount
* When the above code is executed, it produces the following result −
* Name : Zara ,Salary: 2000
* Name : Manni ,Salary: 5000
* Total Employee 2
* you can add, remove, or modify attributes of classes and objects at any time −
* emp1.age = 7 # Add an 'age' attribute.
* emp1.age = 8 # Modify 'age' attribute.
* del emp1.age # Delete 'age' attribute.

Class Inheritance

class Parent: # define parent class

parentAttr = 100

def \_\_init\_\_(self):

print (“Calling parent constructor”)

def parentMethod(self):

print ('Calling parent method')

def setAttr(self, attr):

Parent.parentAttr = attr

def getAttr(self):

print ("Parent attribute : {}".format(Parent.parentAttr))

class Child(Parent): # define child class

def \_\_init\_\_(self):

print "Calling child constructor"

def childMethod(self):

print 'Calling child method'

c = Child() # instance of child

c.childMethod() # child calls its method

c.parentMethod() # calls parent's method

c.setAttr(200) # again call parent's method

c.getAttr() # again call parent's method

class A: # define your class A

.....

class B: # define your class B

.....

class C(A, B): # subclass of A and B

.....

Overriding Methods

You can always override your parent class methods. One reason for overriding parent's methods is because you may want special or different functionality in your subclass.