**Home Work 5**

**CMSE 501**

**Submitted by:**

**Farhan Azeem Afridi**

**20131558**

**Submission Date: 27-11-2013**

**Objective:**

Develop class **Polynomial**. The internal representation of a Polynomial is a **dictionary of terms**. Each term is a key-value pair that contains an exponent and a coefficient. The term **2x4 had the coefficient 2 and the exponent 4**. For simplicity, assume the polynomial contains only **non-negative exponents**. Develop the class with a dictionary based interface for accessing terms that includes the following elements:

1. **The class's constructor accepts a dictionary of exponent: coefficient pairs.**
2. **Coefficient values in a Polynomial are accessed by exponent keys (e.g., polynomial[exponent]=coefficient). If a polynomial does not have a coefficient for a specified exponent, the expression polynomial[exponent] evaluates to zero.**
3. **The length of a Polynomial is the value of its highest exponent.**
4. **Define method \_\_str\_\_ for representing a Polynomial as a string with terms of the form cxy.**
5. **Include an overloaded addition operator (+) to add two Polynomials.**
6. **Include an overloaded subtraction operator (-) to subtract two polynomials.**

**CODE:**

import copy **# copy is used for making shallowcopy and deepcopy**

class Polynomial: **# defining class Polynomial**

def \_\_init\_\_(self,dicta={0:0}): **# Initializing the class, where dicta is an empty dictionary**

self.dicta = dicta **# the class accepts a dictionary of key:value pairs**

**# where key = exponent and value = coefficient**

def \_\_len\_\_(self): **# the length of a polynomial is the is the value of its highest exponent**

return max(self.dicta) **# here it is returning the highest exponent or the degree of the polynomial**

def \_\_str\_\_(self): **# defining the mehtod \_\_str\_\_ to represent the polynomial objects as strings**

self.string = " "

for key, value in self.dicta.iteritems(): **# iterating on the key:value(exponent:coefficient) pair of the polynomial dictionary**

if value>=0: **# if the coefficient is greater than or equal to zero add plus sign**

if key==0:

self.string += ("%dx^%d" % (value , key))

else:

self.string += " + "

self.string += ("%dx^%d" % (value , key))

else:

self.string += " - " **# otherwise if it is less than 0 means negative add minus infront of it**

self.string += ("%dx^%d" % (-(value) , key))

return self.string

def \_\_neg\_\_(self): **# not part of the assignment {what will be the outcome of -(2x^2+3x+1)}**

newdictneg = copy.deepcopy(self.dicta) **# here the neg method helps us in changing the whole polynomial into a negative one**

for key,value in newdictneg.iteritems(): **#here we are using deepcopy to produce a copy of the dictionary we are going to operate on**

newdictneg[key] = -value **#so that the original items in the dictionary wont be altered**

return Polynomial(newdictneg) **# deep copy is used in the rest of the class for the same purpose**

def \_\_add\_\_(self, other): **# here we are modifying the + operator, it takes two arguments self and** other

newdictadd = copy.deepcopy(self.dicta) # which means it is overloaded

for key, value in other.dicta.iteritems():

if key in newdictadd:

newdictadd[key] += value **# if the exponents of two polynomials match add their coefficients**

else:

newdictadd[key] = value **# otherwise if they dont put in the value as it is**

return Polynomial(newdictadd)

def \_\_sub\_\_(self, other): **#here we are declaring the function for a (-) overloaded operator**

newdictsub = copy.deepcopy(self.dicta) **#where self and other are two objects of the class Polynomial**

for key, value in other.dicta.iteritems():

if key in newdictsub:

newdictsub[key] -= value **#if exponents of both the polynomials match, subtract the value of other from self(second from first)**

else:

newdictsub[key] = -value **#if an exponent in not present add it with a minus sign**

return Polynomial(newdictsub)

def calculate(self,x=float): **#not included in the assignment, part of midterm**

cal = 0 **#it calculates the value of the polynomial for any variable x**

for key,value in self.dicta.iteritems():

cal += value\*(x\*\*key)

return cal

# Main Program

print "Welcome to Polynomial Generator \n"

dict1 = {} **#Empty Dictionary**

terms1 = int(raw\_input( "please enter the number of terms in the first polynomial \n"))

exponent1=0

while(exponent1<=terms1-1): **#Loading the exponents and Coefficients in to dict1**

coefficient1 = int(raw\_input("please enter the coefficient for exponent %d \n" % exponent1))

dict1[exponent1] = coefficient1

exponent1=exponent1+1

print dict1 **#repeating the same procedure as above for making another dictionary**

dict2 = {}

terms2 = int(raw\_input( "please enter the number of terms in the second polynomial you want \n"))

exponent2=0

while(exponent2<=terms2-1):

coefficient2 = int(raw\_input("please enter the coefficient for exponent %d \n" % exponent2))

dict2[exponent2] = coefficient2

exponent2=exponent2+1

print dict2

poly1 = Polynomial(dict1) **#Polynomial Object 1**

poly2 = Polynomial(dict2) **#Polynomial Object 2**

print "the first polynomial is \n" , poly1 **#Printing Polynomial Object 1 i.e poly1, \_\_str\_\_ method will be called**

print "the second polynomial is \n" , poly2 **#Printing Polynomial Object 2 i.e poly2, \_\_str\_\_ method will be called again**

poly3 = poly1+poly2 **# now making a third polynomial object which is actually the addition of the two objects declared above**

print "the addition of both the polynomials is \n" , poly3 **# \_\_add\_\_ method will be called when + operator is used for the addition of the objects**

poly4 = poly1-poly2 **#similarly a fourth object which carries out the subtraction, \_\_sub\_\_ will be called**

print "the first polynomial minus the second is \n" , poly4

poly5 = poly2-poly1

print "the second polynomial minus the first is \n" , poly5

print "degree of poly1 is " , len(poly1) **#now printing the length of the polynomials, usually for dictionaries it should be**

print "degree of poly2 is " , len(poly2) **#the number of items in the dictionary but for this class, len() will return highest exponent**

print "degree of poly3 is " , len(poly3) **# i.e the degree of the polynomial.**

print "degree of poly4 is " , len(poly4) **# \_\_len\_\_ is called here**

print "for x=2, poly1 is " , poly1.calculate(2) **#calculate method is called here**

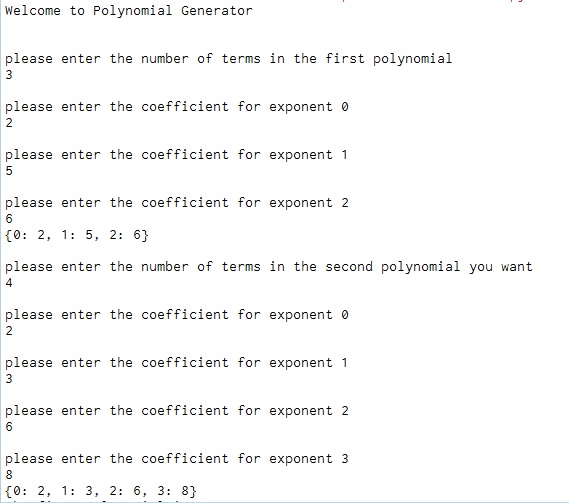
print "-(polnomial one) = ", -(poly1) **# \_\_neg\_\_ is called here**

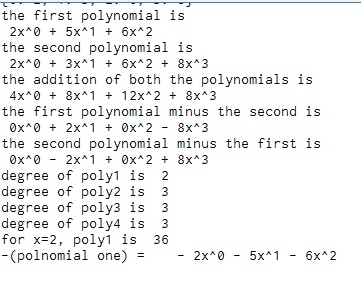
**Please find the direct print from Canopy at the end of the assignment**

**Sample Output and Alterations:**

**Case 1: For the code given in the assignment**

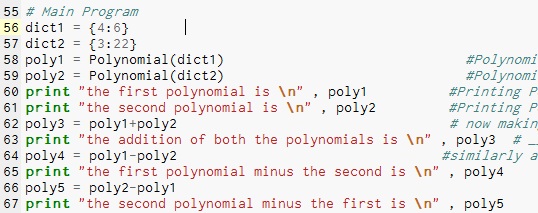
**Output:**



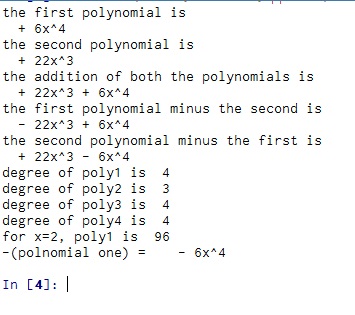


**Case 2: (Single term Polynomial and its degree)**

**Program:**

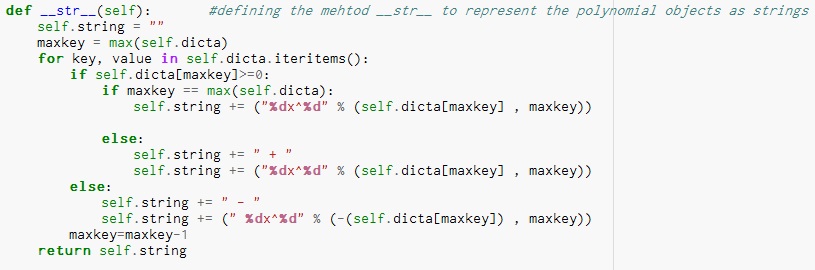


**Output:**



**Case 3: (Printing the Polynomial as a Descending order String, starting from the highest value till the lowest)**

**\_\_Str\_\_ Function**



**Program and Output:**

