Assignment 9 1 - Program

```
import numpy

#note that most of the difficulty with this assignment in python comes from the issue that the number format that the data file is in:
#'(real, imag)'
#is ideal for Fortran, but

values = [] #initialise empty array

fin = open('complex_data.dat','r') #open the relevant data file

fdata = fin.read().split('\n') #read in entire file (.read()), then split it at all of the line breaks (.split('\n'))

fin.close()

for line in fdata:
  #if line is not empty...
  if (line != ''):
    #reformat the line, removing the brackets and commas which Fortran uses but which python does not
    modline = line.replace('(','').replace(')','').replace(',','')
    bits = modline.split() #split up the two numbers
    value = complex(float(bits[0]), float(bits[1])) #parse each bit as real + imaginary
    values.append(value) #extend the list with this new value

text = str(numpy.sum(numpy.abs(values))) #calculate sum of the moduli
text += '\n'
text += str(values[0] * values[-1]) #calculate product of first and final values
text += '\n'

print text
fout = open('assign_9_1.out','w')
fout.write(text)
fout.close()
```

Assignment 9 1 - Output

```
253.424241671
(34.6754823036+36.7993545838j)
```

Assignment 9 2 - Main Program

```
import numpy
from assign_9_2_mod import Vector

#open data file and read in all lines
fin = open('vectors.dat','r')
fdata = fin.read().split('\n')
fin.close()

#initialise empty array
myvectors = []

```

```
for line in fdata:
  #for each line, if not empty...
  if (line != ''):
    x,y,z = line.split() #split the line into the separate components (note implicit expansion of three element list into three comma separated variables)
    myvectors.append(Vector(float(x),float(y),float(z))) #make vector and add it to the list

#perform calculations. Note the use of indices offset by 1 from those stated in the booklet. This is due to the zero'th element default array indexing in python
string = str(myvectors[0] + (myvectors[1] - myvectors[2])) + '\n' # v_1 + (v_2 - v_3)
string += str(Vector.dot(myvectors[1],myvectors[3])) + '\n' # v_2 dot v_4
string += str(Vector.cross(myvectors[0],myvectors[4])) # v_1 cross v_5

#output to screen and file
print string
fout = open('assign_9_2.out','w')
fout.write(string)
fout.close()
```

Assignment 9 2 - Module

```
import numpy

#Vector object and methods, all vector forms and operations
#class Vector:
class Vector(object):
  #initialisation
  __slots__ = ('x','y','z') #limit properties to include only x and y components (otherwise random additionals unrelated to vectors can be added dynamically, which could break other parts of the vector operations)
  def __init__(self, x, y, z):
    self.x = x
    self.y = y
    self.z = z
  #method to display contents of vector object (default separated by commas)
  def show(self, mode='comma'):
    if (mode == 'comma'):
      a = ','
    elif (mode == 'space'):
      a = ' '
    elif (mode == 'tab'):
      a = '\t'
    else:
      a = str(mode)
    return '[' + str(self.x) + a + str(self.y) + a + str(self.z) + ']'
  def __repr__(self):
    return self.show()
  def __str__(self):
    return self.show()
  #method which returns the magnitude of a vector object
  def mag(self):
    a = numpy.sqrt(self.x*self.x + self.y*self.y + self.z*self.z)
    return a
  ###Unique methods within vector type
  #method which adds two vector objects together
  def add(a,b):
    x = a.x + b.x
    y = a.y + b.y
    z = a.z + b.z
    return Vector(x, y, z)
  #method which subtracts two vector objects
  def sub(a,b):
    x = a.x - b.x
    y = a.y - b.y
    z = a.z - b.z
    return Vector(x, y, z)
  #scale a vector by a scalar
  def scale(a,b):
    x = a.x * b
    y = a.y * b
    z = a.z * b
    return Vector(x, y, z)
  ###Replacement to default operations when using vectors
  def __add__(self,other):
    return self.add(other)
  def __sub__(self,other):
    return self.sub(other)
  def __mul__(self,other):
    return self.scale(other)
  def __rmul__(self,other):
    return self.scale(other)
  #perform dot product between two vectors
  def dot(a,b):
    x = a.x * b.x
    y = a.y * b.y
    z = a.z * b.z
    return x + y + z
  #perform cross product between two vectors
  def cross(a,b):
    x = (a.y * b.z) - (a.z * b.y)
    y = (a.z * b.x) - (a.x * b.z)
    z = (a.x * b.y) - (a.y * b.x)
    return Vector(x, y, z)
  #return unit vector for one vector
  def unit(a):
    b = a.scale(1.0 / a.mag())
    return b

Null = Vector(0.0,0.0,0.0)
```

Assignment 9 2 - Output

```
[13.0,-3.8,0.0]
22.6
[7.0,-2.0,-17.0]
```