Assignment 7 1 - 'Subroutine'

```
#note that here we deviate from the booklet in objective and implementation.
#python is not a language for which a 'subroutine', in the fortran sense, is a natural construct.
#functions are inordinately more prevalent, and are more flexible than fortran functions, in that they can easily return more than one result.
#technically, this makes it arguable as to whether they are 'functions' in the true sense of the word; they certainly aren't in the mathematical sense.
#presented here is the construction of a python function to perform a similar process as the subrountine in the assign_7_1 assignment; along with an example of how it would be called and used in the file 'week7prog.py', which operates differently to the 'week7prog.o' written in fortran.

#note this different version of the line importing numpy; this is not because of the functions/etc, but simply to show how it works.
#if a module being imported has a long or inconvenient name, we can rename how we refer to it (only for the purpose of within our program) using this format.
#in this case, we will refer to 'numpy' as just 'np'. So 'numpy.sqrt()' would instead be 'np.sqrt()'; a little more convenient
import numpy as np

def sevenfunc(a,b,c):
  disc = b*b - 4.0*a*c #calculate discriminant
  if (disc > 0.0):
    #two solutions when greater than zero
    x1 = (-b + np.sqrt(disc))/(2.0*a)
    x2 = (-b - np.sqrt(disc))/(2.0*a)
    x_sol = 2
  elif (disc == 0.0):
    #one when equal to it
    x1 = x2 = -b/(2.0*a)
    x_sol = 1
  else:
    #none when less than it
    x1 = x2 = 0.0
    x_sol = 0
  #all variables listed after the 'return' statement are passed back to the calling program/routine by the function
  return x_sol, x1, x2
```

Assignment 7 1 - Main Program

```
#here we import our other code as a module, much as we might import numpy or similar
import assign_7_1

#set some in-program values for the coefficients
prog_a = 4.0
prog_b = 1.2
prog_c = -2.5

#call the actual routine, note the use of assign_7_1.sevenfunc in order to call the function
#all three variables are listed, comma separated, before the equals, as being the values returned from the function
num, xa, xb = assign_7_1.sevenfunc(prog_a,prog_b,prog_c)

```

```
#output any information to the screen
print 'The solution for x of a quadratic equation of the form %0.2f x^2 + %0.2f x + %0.2f = 0 is:'%(prog_a,prog_b,prog_c)
if (num == 0):
  print 'No real solutions'
elif (num == 1):
  print 'One solution, x = %f'%(xa)
elif (num == 2):
  print 'Two solutions, x = %f and %f'%(xa,xb)
else:
  print 'Solving function returned an invalid number of solutions (%d)'%(num)
```

Assignment 7 1 - Output

```
The solution for x of a quadratic equation of the form 4.00 x^2 + 1.20 x + -2.50 = 0 is:
Two solutions, x = 0.654674 and -0.954674
```

Assignment 7 2 - Main Program

```
#import our other function via their file (minus the '.py' extension); state that it will be referred to as 'mymod'
import assign_7_2_mod as mymod

#input the radius to use from the user
radius = float(raw_input('Specify radius: '))

#create output lines, calculating values for properties in-line using the relevant functions
output  = 'For a radius of %f:\n'%(radius)
output += 'The area of a circle is %f\n'%(mymod.circ_area(radius))
output += 'The perimeter of a circle is %f\n'%(mymod.circ_perim(radius))
output += 'The surface area of a sphere is %f\n'%(mymod.sphere_area(radius))
output += 'The volume of a sphere is %f\n'%(mymod.sphere_vol(radius))
output += 'The difference in surface area of a sphere to area of a circle is %f\n'%(mymod.sphere_area(radius) - mymod.circ_area(radius))
output += 'The volume of a sphere whose radius is the perimeter of a circle is %f\n'%(mymod.sphere_vol(mymod.circ_perim(radius)))

print output #write to screen
fout = open('assign_7_2.out','w')
fout.write(output) #write to file
fout.close()
```

Assignment 7 2 - Module

```
from numpy import pi #import only this value from numpy; does not need to be referenced with 'numpy.'

#area of a circle, pi r^2
def circ_area(r):
  return pi * r * r

#perimeter of a circle, 2 pi r
def circ_perim(r):
  return 2.0 * pi * r

```

```
#surface area of a sphere, 4 pi r^2
def sphere_area(r):
  return 4.0 * pi * r * r

#volume of a sphere 4/3 pi r^3
def sphere_vol(r):
  return (4.0/3.0) * pi * r * r * r
```

Assignment 7 2 - Output

```
For a radius of 1.000000:
The area of a circle is 3.141593
The perimeter of a circle is 6.283185
The surface area of a sphere is 12.566371
The volume of a sphere is 4.188790
The difference in surface area of a sphere to area of a circle is 9.424778
The volume of a sphere whose radius is the perimeter of a circle is
   1039.030304
```