Introduction to Python for Science and Engineering

The following exercises were reproduced from David Pine's Python Manual: chapter 2: Launching Python and adapted and extended by Paul McNulty and Marc Gershow

previous version of manual, accessible without library access

previous version of chapter 2

```
##RUN THIS CODE BLOCK FIRST - it brings in functions like sin, cos, log, ...
#in the future, we'll do things slightly differently to make your code more robust and portab
from numpy import *
```

2.12.1

A ball is thrown vertically up in the air from a height h_0 above the ground at an initial velocity v_0. Its subsequent height h and velocity v are given by the equations

$h = h_0 + v_0 t - \frac{1}{2} g t^2$

$v = v_0 - gt$

where $g = 9.8$ is the acceleration due to gravity in m/s^2. Write code that finds the height h and velocity v at a time t after the ball is thrown. Start by setting $h_0 = 1.2$ (meters) and $v_0 = 5.4$ (m/s) and have your code print out the values of height and velocity (see *Note about printing*). First print the height and velocity after 0.5 seconds. Then add more code to find them after 2.0 seconds.

When you execute the code block you write below, there should be 4 numbers output, each on its own line. If you want to be fancy, you could make the script write something like

h(0.5) = 2.7

but

2.7

would also be fine, as would

2.675000000

In other words, we don't care how your answer is formatted (for now). You will probably find it's easiest to cut and paste a large portion of your answer to the t = 0.5 section to calculate the t = 2.0

answer. *What code do you have to recopy and what stays the same?* This is OK for now, but at the end of the assingment you'll learn a better way to make your code reusable

```
##USE THIS CODE BLOCK TO COMPLETE EXERCISE 2.12.1##

h0 = 1.2
v0 =  5.4
g= 9.8

h1 = h0 + (v0 * .5) - .5*g*.5**2
print(h1)

h2 = h0 + (v0 * 2) - .5*g*2**2
print(h2)

v1 = v0 - g*.5
print(v1)

v2 = v0 - g*2
print(v2)
```

```
2.6750000000000003
-7.600000000000001
0.5
-14.200000000000001
```

2.12.2

Write a script that defines the variables $V_0 = 10$, $a = 2.5$, and $z = 4\frac{1}{3}$, and then evaluates the expression

$$V = V_0 \left(1 - \frac{z}{\sqrt{a^2+z^2}}\right)$$

Then find V for $z = 8\frac{2}{3}$ and print it out (see *Note about printing*). Then find V for $z = 13$ by changing the value of z in your script.

Again, you don't have to worry about how your answer is formatted

```
##USE THIS CODE BLOCK TO COMPLETE EXERCISE 2.12.2##

v0 = 10
a = 2.5

v1 = v0 * (1 - (8 + 2/3)/((a**2 + 8 + 2/3**2)**.5))
print(v1)
```

```
v2 =  v0 * (1 - (13)/((a**2 + 13**2)**.5))
print(v2)
```

```
-12.78161362559282
0.17993553019352682
```

2.12.3

In the code block below, write code that calculates the following expressions:

1. $a = \frac{2+e^{2.8}}{\sqrt{13}-2}$
2. $b = \frac{1-(1+\ln 2)^{-3.5}}{1+\sqrt{5}}$
3. $c = \sin\left(\frac{2-\sqrt{2}}{2+\sqrt{2}}\right)$

After running this code block, typing a, b, or c in the next code block, followed by shift+return should yield the value of the expressions in (a), (b), or (c), respectively.

```
##USE THIS CODE BLOCK TO COMPLETE EXERCISE 2.12.3##

import numpy
import math

a = (2 + numpy.e**2.8)/(13**.5 - 2)
print(a)

b = (1 - (1 + math.log(2, math.e)**-3.5)/(1 + 5**.5))
print(b)

c = numpy.sin((2 - 2**.5)/ (2 + 2**.5))
print(c)
```

```
11.488045914800644
-0.42355081168503705
0.17073234104506005
```

```
##USE THIS CODE BLOCK TO TEST YOUR ANSWER##
```

2.12.4

A quadratic equation with the general form

$ax^2 + bx + c = 0$

has two solutions given by the quadratic formula

$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

1. Given a, b, and c as inputs, write code that gives the numerical values of the two solutions. Write the constants a, b, and c as floats, and show that your script gives the correct solutions for a few test cases when the solutions are real numbers, that is, when the discriminant $b^2 - 4ac \geq 0$. Use the `print` function in your script, discussed at the end of Section 2.8.1 *Scripting Example 1*, to print out your two solutions.
2. Written this way, however, your script gives an error message when the solutions are complex. For example, see what happens when $a = 1$, $b = 2$, and $c = 3$. You can fix this using statements in your script like $a = a + 0j$ after setting a to some float value. Thus, you can make the script work for any set of real inputs for a, b, and c. Again, use the `print` function to print out your two solutions.

```
##USE THIS CODE BLOCK TO COMPLETE EXERCISE 2.12.4##
a = 1
a = a+0j #make the number complex to allow for complex solutions
b = 2
c = 3

##write your code to print the results
import numpy
x1 = (-b + numpy.sqrt(b**2 - 4*a*c))/(2*a)
print(x1)

x2 = (-b - numpy.sqrt(b**2 - 4*a*c))/(2*a)
print(x2)
```

```
(-1+1.4142135623730951j)
(-1-1.4142135623730951j)
```

Double-click (or enter) to edit

LOOKING AHEAD: FUNCTIONS

It sure is a pain having to cut and paste or retype the same code over and over again, and it makes it really easy to make mistakes; what if you didn't type in a value of a in 2.12.4? The computer would happily use the value you calculated in 2.12.3 earlier.

Functions make your code reusable! The Pine manual introduces functions very late (in chapter 7) compared to most books.

Let's look at redoing 2.12.4 using a function

```
import numpy

def printQuadraticSolution(a,b,c):


  a = a+0j #make the number complex to allow for complex solutions

  x1 = (-b + numpy.sqrt(b**2 - 4*a*c))/(2*a)

  x2 = (-b - numpy.sqrt(b**2 - 4*a*c))/(2*a)

  print("The solutions are {" + str(x1) + "and" + str(x2) + "}")
 #return x1, x2

#print(printQuadraticSolution(1,2,3))
#print(printQuadraticSolution(1, 2, 1))

printQuadraticSolution(1,2,3)
printQuadraticSolution(1, 2, 1)
```

```
The solutions are {(-1+1.4142135623730951j)and(-1-1.4142135623730951j)}
The solutions are {(-1+0j)and(-1+0j)}
```

The function definition looks like this

1. the keyword def (for define) that tells you you're about to write a function
2. the name of the function ("printQuadraticSolution")
3. the arguments to the function (a, b, c) - what you will "pass" to the function to give it the information it needs to know to run
4. a colon : which has to be there as the last part of the first line
5. the rest of the code, which is all **indented**. Python uses indentation to say what is part of the function and what isn't

   Now please fix the line "x2 = ..." above to calculate the other solution, then test it out with a few new values below

```
# I don't see what is wrong with x2
```

Functions can return results

When you type y = sin(x), you are calling a function (sin). You pass a value to the function (x) and you get something back, the sine of x, in return. When a function **returns** a value, you use the

keyword **return** at the **end** of the function to say what it is

In Python a function can return multiple values.

For instance, instead of printing out the two solutions to the quadratic equation, we could write a function that **returns** them to us to use later

```
import numpy as np
def getQuadraticSolution(a,b,c):
  a = a+0j #make the number complex to allow for complex solutions

  x1 = (-b + np.sqrt(b**2 - 4*a*c))/(2*a)
  x2 = (-b - np.sqrt(b**2 - 4*a*c))/(2*a)
  return (x1,x2)

(x1,x2) = getQuadraticSolution(1,5,6)
print("The two solutions returned by getQuadraticSolution were x1 = {} and x2 = {}".format(x1
```

```
The two solutions returned by getQuadraticSolution were x1 = (-2+0j) and x2 = (-3+0j)
```

Test your knowledge: See if you can use a function to redo the first problem without so much cutting and pasting. I'll get you started

```
#h=h0+v0t−1/2gt**2
#v=v0−gt


def ballHeightAndVelocity (h0, v0,g, t):
  ## your code here
  h = h0 + (v0 * t) - .5*g*t**2
  v = v0 - g*t

  return (h,v)
```