**Exercise 2: Writing a program**

In the previous exercise, you typed commands in one at a time. If you had to do things more than once, this would quickly become annoying. The usual way to do Python is to put all the commands into a text file, and then tell Python “go do those commands”. Such a text file is called a “program”.

Instead of typing those commands in again and again each time we want to plot a mathematical function, we could make a program to do the same thing. We do this by creating a new text file containing those commands, in their proper order. We then save the file with a meaningful name, such as “plot1.py”. (Do not put spaces in your file name, and end the name with “.py”) In Spyder, we can then press the F5 key and Spyder will run the program. You can make changes to the program and run it again, as needed.

Try it! Copy the Python code from exercise 1 and paste it into a new text editor window in Spyder, save the program, and press F5. Try changing things: change sin() to cos(), change the title, whatever… just play with it a bit and see what happens. Any time you want to check how things are going, save and press F5 again to run the program.

**FUNCTIONS**

One of the very useful tricks in Python is the ability to write functions. The sin(t) in the above program, for example, is a function. In the case of sin(), that function is one of the things defined when you import pylab so you don’t have to tell Python how to calculate sine. You can write your own functions with the ‘def’ (define) command. This allows you to simplify your programs into manageable chunks. Here’s an example of how it works, using a rather silly and useless function.

 def silly(x):

 y = sqrt(x\*\*2 + 9)

 return y

There are several key things to note on that example. First, the colon and the indentation are part of the code! The colon tells Python to expect an indented chunk of code. All of the indented lines are part of the definition. Next, the x in the definition is a place-holder. You can call silly(42), or silly(q), or silly(*anything*), as long as *anything* is something that can be squared (x\*\*2) in the second line of the definition.

Finally, the “return” value is what goes back to the rest of the program when you call this function. Once a function is defined in a program, we can use it just like we would use any more standard mathematical function. For example, the following code asks the user for a number and then prints “silly” of that number.

 n = input("Enter a number: ")

 print("Silly of that number is", silly(n))

(Oh, and by the way, input() and print() are simple ways to put numbers into a program and get numbers out. They are also Python functions, and are built-in.) This code works with arrays of numbers also:

 a = linspace(0, 10, 11)

 print(silly(a))

 plot(a, silly(a), 'go')

 plot(a, silly(a) + 1, 'rx')

 show()

**ASSIGNMENT**

If you drop an anvil from a great height, it accelerates downward at about 9.8 m/s2. Write a program containing two functions: one that returns v(t) and another that returns y(t) for the anvil. Your program should then plot both these functions on one graph, for time ranging from 0 to 5 seconds. Have the program label the axes appropriately and put your name in the title. You could do this just using individual commands, as with the first exercise, but to increase your chance of surviving the rest of the exercises do it as a program, with functions.