Lesson 2

Modules and Functions

If you have a series of commands you want to use (defining variables, slicing lists or character strings, printing (out results … and much more to come) it's nice if you can keep them all in one file to run with a single command, and modify as needed instead of typing them all over again. A series of commands kept in a text file is called a *module*, and a set of commands that can accept input and produce output is called a *function.* Often several functions will be contained in a module. You will learn how to create these below, as they can save a lot of work!

Below a file is denoted by a pair of horizontal lines with the file name at the top, and commands in between. Note that all modules MUST have the extension '.py'. You should write this module using wordpad++ or a similar pure text editor. DO NOT use a word processor like Msword. Save it to your python folder, the same directory that you switched to in the command line before running python.

---
my_first_script.py

```
# This module demonstrates how a set of commands can be run all at once.
# run in spyder Shift+Enter or the run bottom
#
# (You should always put explanatory comments in your code)

print(  "Hello World!" )

LUE = 6 * 7

print( "The answer to life, the universe and everything is", LUE  )
```

Now that you have created this module, run it both from the command line and from the spyder as described in the comments. It's always a good idea to describe how your modules and functions are to be used inside them.

Now we will create a second script to run three functions that we will also create. Type in and save the following two modules to your python folder:

---
second_script.py

```
# see the first module for how to run a module.
```

 # demonstrate importing and using a function with no input or output variables

```
from first _module import NoIn_NoOut  # METHOD 1 TO IMPORT FUNCTION #
NoIn_NoOut()
```

 # demonstrate importing a whole module then running functions from it

```
import first _module
```

 # run a function that greets you (note the dot to import a function from a module)

```
name = 'Said'
first_module.sayhello(name)
```

 # run function that adds three variables and provides the sum in an output variable

```
a = 20
b = 30
c = -8
import first _module as fm
mysum = fm.sum3(a,b,c)  # METHOD 2 TO IMPORT FUNCTION #
print( "my sum =", mysum )
```

------
first_module.py

```
# this module holds three functions

def NoIn_NoOut():
        print( "Hello World!" )

def sayhello(name):
        print( "Hello %s!" %name )

def sum3(a,b,c):
        total = a + b + c
        return total
```

When you type the first module, notice the indents. These are crucial, and must all be the same. The best way to ensure they are the same is to use tabs instead of spaces. You'll get error messages if you mix tabs and spaces, so only use tabs!

Now that you have both files saved exactly as you see them above, from the command line (in the python directory of course) run your module that will run the three functions like this:

Prompt> python first _module.py

If you get error messages, don't panic. Start at the end of the error messages and work up, fixing the problems that you understand, then running again. You may have to do this several times.

Note the two ways to import then use functions:

Approach 1: import <function> from <module>
function(input)

Approach 2: import <module> as <shortmod>
shortmod.function(input)

The first approach is best if you will only be using one or two functions from the module. The second approach is best if you will be using multiple functions from the same module.

You should also absorb the idea of breaking whatever work you will be doing into smaller tasks called functions, then use a module of commands to run these functions in whatever sequence is needed. This is crucial if you will be doing the same task more than once.

These functions may seem so simple they are silly, but we are saving energy to get to the main event of reading and displaying satellite data. All we care about now is getting the basic ideas down so they are ready to use when we write the more complex and useful functions.

Summary: Scripts, Functions, Modules

A ***script*** is a set of commands saved in a file:

scriptname.py

```
command1
command2
command3
....
```

A ***function*** is a set of commands that receives input variables and/or produces output variables. It is defined like this:

```
def function_name( input_variable1, input_variable2…):
        command1
        command2
        ....
        return output_variable
```

The function is run as follows:

```
output_variable = function_name(input_variable1, input_variable2…):
```

A ***module*** is a set of functions saved in a file:

modulename.py

```
def function1
        commands
def function2
        commands
....
```

A function from a module must be imported before being used, either by importing the module first:

```
import mymodule as mm
output_var = mm.function2(input_var)
```

Or import one function alone:

```
from mymodule import function2
output_var = function2(input_var)
```

Boolean Logic and Decisions

A computer program can be used to make decisions based on data. What happens with the script below is so obvious is needs no further explanation. Write it, save it, and predict what it will do before you run it. Note that we have introduced the *input* function to provide a way to enter data from the command line.

---
decision_script.py

```
#  This series of commands demonstrates how python can be used
# to make decisions based on data.

Bob_age = 18
Maria_age = 24
Your_age = int(input("enter your age")) # input alone saves a string character, therefore, you will need to use int() to specify that the number that you are saving is an integer

if  (Your_age == Bob_age):
        Print( "you are the same age as Bob" )

# (why did we need to use "==" instead of "="? )

if  (Your_age >= Bob_age) and (Your_age <= Maria_age):
        print( "You are about Bob's and Maria's age" )

comment the above lines before jump to this example.

I would like to know if I am over Maria's age and under Bob's age, if not the average of those two guys:
if  (Your_age < Bob_age) or (Your_age > Maria_age):
        print( "You are not part of the group." )
else:
        print( "You are part of the group" )
        avg_age = (Bob_age + Maria_age)/2
        print( "with average age of", avg_age )
```

As usual, run this either from the command line ("python decision_script.py") or from the python prompt ("import decision_script" ; "decision_script")