Python for Informatics

Assignment 2

"Functions"

Preparation:

Please run "Anaconda Navigator", and then install and launch *Spyder*. The *Spyder* application is the Integrated Development Environment (IDE) that you will be using for all of the remaining assignments in this course. *Spyder* provides a main Editor pane in which you can type or paste Python statements that are saved as a *.py* Python program/script file. Below the Editor pane, you'll find a Python interpreter pane in which you can type and execute Python commands as part of a Python interpreter *interactive session*. Additionally, a debugger for step-wise running/tracing of your *.py* Python program/script files.

Assignment 2 requires you to use *Spyder* to create and submit a Python program/script (*.py*) file. You create a notebook within JupyterLab by selecting File→New file…, and then subsequently selecting File→Save as… to specify a meaningful filename (e.g., "functions.py").

Background:

A string literal is a special syntax that is used to specify a value that literally defines a string within your program. For example, 'Fred' is a string literal. It is literally the sequence of characters F, r, e, and d. However, the string literal 'Fred' cannot be converted into an *int* value. By comparison, consider the string literal '42'. It is literally the sequence of characters 4 and 2. In this case, the string literal '42' can be converted into an *int* value. This assignment does not require you to validate the string literal values—your code should assume that they have been properly defined as strings that can be converted into *int* values. You, the programmer, will properly define and place the string literal values in the appropriate positions.

As a notational convention, when you are told to create a function named ***to_number(str)***, that means that the name of the function will be ***to_number***, and that function will expect to receive a parameter value of type ***str***. Therefore, since ***str*** is the name of an object type, you should not name your function argument to be ***str***. To gain more insight into this, type the following into a Python interpreter pane:

***type*('Fred')**

The interpreter will dutifully tell you that that '**Fred**' is an object of type ***str***. So, instead of naming your argument ***str***, you should use a name that is not already reserved to specify a built-in object type. A good name would be something like ***inp_str*** or ***num_str***. I would prefer to use ***num_str***, as it tells the user of my function that the argument that they supply must be a number that is of type ***str***, i.e. '**42**'.

If you ensure that you understand the above background information, you will avoid the most common mistakes made by students with this assignment.

Description:

Create a Python .py file that performs the following:

1. Define a function named ***to_number(str)*** that takes a string value as a parameter, converts it to an **int** value, and returns that **int** value.
2. Define a function named ***add_two(n1, n2)*** that takes two ***ints*** as parameters, sums them, and then returns that ***int*** sum value.
3. Define a function named ***cube(n)*** that takes a numeric value as a parameter, cubes that value, and then returns that resulting numeric value.
4. Use the above functions to compose one (***only*** one) statement to specify two ***string literals***, convert them to ***ints***, add them together, cube the result, and print the cubed value.

Note: Step 4 above is not requiring you to define an additional function, as that would require more than one statement. Moreover, defining an additional function would not ***do*** anything unless you also ***call*** that function, which would then be yet another statement. Since defining and calling a function is more than one statement, doing so would not meet the requirements of this assignment. Do not define an another function! The idea with this assignment is for you to demonstrate how nested function calls can compress a great deal of functionality into only one statement. For example, given functions named ***foo()*** and ***bar()***, a statement such as ***foo(bar())*** can execute a substantial amount of functionality in only one statement. The foregoing is just an example—please don't name any of your functions ***foo*** or ***bar***. You should use the names that are specified in the assignment description steps above.

Deliverable:

One Python .py (not .pynb) file, submitted as an attachment at our course shell assignment page.
Please ensure that your full name is specified at the top of your .py file within a Python comment.

Submission Deadline:

Please see the course schedule in our syllabus for all assignment submission deadlines. Please see the Peerwise reminder below.

Peerwise Reminder:

How many multiple choice Peerwise questions have you created and submitted so far? You need to submit 32 by the end of the course! A good strategy would be to submit at least 4 per week.