User-Defined Single Result (or Pure) Functions

A function can be thought of as a self-contained “mini-program" that accepts one or more inputs and processes it into a single output. The input for a function is known as a parameter or an argument and is enclosed in brackets ( ). A single result (pure) function is not **used on its own** - it is most often used with another function (ie. the **print** function) or assigned to a variable.

We have used a number of predefined functions to this point. However, it is difficult and inadvisable for the creators of a programming language to include too many predefined functions (Why?).

A programmer can also create his or her own functions to accomplish <u>any</u> necessary task. The advantage of creating a collection or “library” of these **user-defined functions** is that they can be re-used multiple times in the current program and as well as in any future program.

A user-defined function must include three parts:

1. A header line which has three components:
   - It begins with the programming command **def**,
   - the name of the function (most programmers use the same naming convention as variables which is the starting the name with lower case letter and using a capital to start every subsequent word while others use all capitals to make the user-defined functions stand out)
   - the parameter name(s) enclosed in brackets. These are also called arguments.

2. The <u>body</u> of the function. The body is the code that does the “work” of the function.

3. At least one of the lines of code (usually the last one in the body) must begin with the command **return** followed by the single value that is to be returned by the function to the program which calls the function.

Examples of User-Defined Functions

Two examples of how a user-defined function is declared and documented along with a main program that will call the function are shown below:

```
# Example One

def centRoundOff (amount) :
    '''A function which will accept a value with any number of decimal places
    and will return a value rounded off to the nearest cent (ie. two decimal
    places). '''

    num = amount * 100  #Stores the number being processed. Note this is a local
                        # variable which means that it can only be used in this
                        # function.
    num  = round (num)
    num  = num / 100

    return num  # This value is returned from the function to the calling program.


# Calling program

'''Case 1 - The following line will output 75.88 on the monitor. The function is
called centRoundOff and the parameter or argument for the function is 75.876987
'''

print (centRoundOff (75.876987))

'''Case 2 - The following code will allow the user to input any amount and have it
rounded off the nearest cent. It will then be outputted. '''

dollars = float (input ("Enter a dollar amount: ")) # Amount inputted by the user

rounded = centRoundOff  (dollars)     #The number rounded to two decimal places
                                      # using the centRoundOff function

print ("Your rounded amount is ", rounded)
```

```
# Example Two

def totalNum ( intNum ) :
    '''A function which will accept an integer and return the sum of all of the
    numbers between 1 and that number.
    intNum is a parameter which must be a positive integer value '''

    total = 0    # A variable to store the total of the numbers. This variable is only #
                 used in the function and is therefore referred to as a local variable
    for i in range (1, intNum + 1) :
        total = total + i

    return total # This value is returned from the function to the main program.

# Calling program

'''Case 1 - The following line will output 5050 on the monitor. The function is called
totalNum and the parameter or argument for the function is 100 '''

print (totalNum (100))

'''Case 2 - The following code will allow the user to input any positive integer and
have it rounded off the nearest cent. It will then be outputted. '''

userInt = int (input ("Enter an integer: ")) # Integer inputted by the user

totalOfInt  = totalNum (userInt )        # The total calculated by the user defined #
                                         function totalNum is assigned to the
                                         #variable totalOfInt

print ("The total of the numbers between 1 and ", userInt, “is”, totalOfInt)
```

Assignment

1. Create a folder called User Functions. Copy the two programs above to ensure they work and save them as **UserFunction0A.py** and **UserFunction0B.py**.

2. Improve the totalNum user-defined pure function in UserFunction0B.py so that it can handle both positive and negative integers. Re-save the program as **UserFunction0B.py**