Void Functions

Review of Single Result (Pure) Functions

- To this point, we have defined a function as a mini-program which accepts one or more parameters and returns a single result. These are called **single result** or **pure** functions.
- Single result functions do not have "side-effects" which means they do not affect the rest of the program in any way - ie. print data, change the value of global variables, etc.

Void Functions

- In programming, a function can have a wider definition - a function can be defined as simply a named group of statements that performs a task.
- These functions are often called **void functions** because they do **not** include a return command.
- For now, we will use void functions to::
  1) Process data passed into the function using parameters and then output the results.
  2) Simply output the data passed in from the parameters.

- Void functions therefore often contain the **print ()** function.

Void Functions Do Not Contain the input () Function - Why?

- String, integer or float variables are non-mutable which mens they cannot be changed after being assigned.
- Void functions therefore **cannot** be used to input string, integer or float data into a program
- They also **cannot** change the value of a string, integer or float parameter parameters.
- The **input ()** function will therefore not be used in void functions (until we deal with lists).

Void Functions Called On Their Own

- Since void functions do not return a value, <u>they are called on their own</u> in a program.
- Unlike pure functions, they are **not** assigned to a variable or used with print () when called.

Examples of Creating and Using a Void Function

An example of a void function is the following:

```
def printTwoWordsTogether (wordOne,wordTwo) :
    # A function to concatenate and output words both ways
    # wordOne and wordTwo must be strings
    forward = wordOne + wordTwo
    backward = wordTwo + wordOne
    print ("The words together are",  forward, "and", backward )
```

Since void functions do not return a value, <u>they are used on their own</u> (ie. They are not assigned to a variable or used in conjunction with a command). For example, the function printTwoWordsTogether created above can be called as follows:

```
# Calling Program
printTwoWordsTogether  (“hello”, “there”)  # Calls the function created above
                                           # Actual parameters are the strings“hello” and “there”
first = input (“Enter a word”)             # User’s first word
second = input (“Enter another word”)      # User’s second word
printTwoWordsTogether (first, second )     # Calls the function created above a 2nd time
                                           # Actual parameters are the variables first and second
```

A more complex example of void function is below. This function can be used in multiple programs:

```
def timesTable ():
    # Outputs a 5 by 5 multiplication table
    for row in range (1, 6):
        for col in range (1, 6):
            print (str (row * col).rjust (5), end ="")  # What does the .rjust (5), end ="" do?
        print ()        # Moves the line down for the next time table

# Calling Program
print (“The five times table is:”)
timesTable () # Calls the function.
```

Using Void Functions to Break Your Program Into Smaller Parts

Programmers use void functions to simplify code by breaking it up into smaller parts. There are a number of advantages of breaking a large program into a number of functions including:

- Functions can make a program smaller by eliminating repetitive code. Later, if you make a change, you only have to make it in one place.
- Programs divided into functions are easier to read and understand .
- Dividing along program into functions allows you to debug the parts one at a time and then assemble them into a working whole.
- Well-designed functions can often be used in many different programs. Once you write and debug a void function, you can reuse it.

Menu Driven Programs

A menu driven program is one which presents the user with a menu and allows the user to choose to do one or more of the menu items. Void functions are ideal to make menu driven programs easy to create as shown below.

```
''' An example of a simple menu driven program. '''
def menuItemOne ():
    '''Function executed if the 1st menu item is selected. It only outputs the message “This is menu item
    one” but obviously a function could do anything the programmer wishes to do '''
    print ("**********************")
    print ("This is menu item one")
    print ("**********************")
    print ()        # Outputs a blank line for spacing purposes
```

```
def menuItemTwo ( aParameter):
        # Function executed if the 2nd menu item is selected. It has a parameter
        print ("######################")
        print ("This is menu item two ")
        print ("This is the parameter -", aParameter)
        print ("######################")
        print ()

def menuItemThree ():
        # Function executed if the 3rd menu item is selected.
        print ("^^^^^^^^^^^^^^^^^^^^^^^^")
        print ("This is menu item three")
        print ("^^^^^^^^^^^^^^^^^^^^^^^^")
        print ()

def showMenu ():
        # This void function displays the menu.
        print ()        # Outputs a blank line for spacing purposes
        print ("Select an item from the menu")
        print ("1. Menu item one")
        print ("2. Menu item two")
        print ("3. Menu item three")
        print ("4. Exit the program")

#Mainline (main program which calls the functions)
userWord = input ("Please enter a word")    # User inputs a word for menu item #2
while True:    # Will continue looping until the break command is called
        showMenu ()  # Calls the void function which displays the menu
        userChoice = input ("Enter a menu choice: ")        # Inputs menu choice made by the user
        if userChoice == "1" :          # Selection structure to call appropriate function depending on choice
                menuItemOne ()
        elif userChoice == "2" :
                menuItemTwo (userWord )             # Passes in the actual parameter userWord
        elif userChoice == "3" :
                menuItemThree ()
        elif userChoice == "4" :        # Exits the loop
                break
        else :              # Gives the user a message if incorrect data is inputted
                print ("Input a menu choice between 1 and 4")

print ("Thank you for using this program")
```

Void Functions Cannot Be Used to Input Data:

Study the following small program:

```
def anInputFunc (aParameter):
    ''' This function will change the formal parameter but not the actual parameter'''
    aParameter = input ("Enter a new string: ")
    print ("The output from the parameter aParameter in the function is", aParameter)

#Main Program
userData = "Original String"
anInputFunc (userData)    # Calls the function with the variable userData as the actual parameter
print ("The output from the global variable userData in main program is", userData)
```

Remember in Python, a function cannot change the value of global variables that are passed in as actual parameters. This means that when the user inputs a value for *aParameter* in the function *anInputFunc*, that value will be outputted when the function is executed because it is treated as a local variable. However, the value of the actual parameter userData will not be changed when the function is called and therefore it will userData will still have the value "Original String" when it is outputted.

To summarize, assuming that the user inputted the string “Hello There” when prompted in the function, the program will output:

The output from the parameter aParameter in the function is Hello There
The output from the global variable userData in main program is Original String