Objective: This worksheet provides a quick overview for how to write user defined functions. The user defined functions can be called multiple times in the code possibly using the different argument values. The functions provides modularity and the codes also becomes easier to read and maintain.

A Simple Function

- The structure of writing a function is comparable to ***if, for*** and ***while***. Except that the function name succeeds a keyword ***def*** that means the definition.
- The following example shows a simple function ***max()*** that takes two arguments ***a*** and ***b*** and returns the greater of two.
- The following example also shows that usage of the keyword ***return***. It is used to return a value to the caller code that called the function.

Functions

```
In [1]: def max (a, b):
            if a > b:
                return a
            else:
                return b
```

- Only the function definition is written in the above cell. If the cell is run, nothing visibly happens but the function is defined and it can be used in other cells. This is shown in the example below:

```
In [2]: print(max(12, 65))
        65
```

- The function can be used in other code also. For example, the two values can be accepted from the user input and then the function can be called to find the greater value.

```
valueX = input("Enter the first value: ")
valueX = float(valueX)
valueY = input("Enter the second value: ")
valueY = float(valueY)
maxValue = max (valueX, valueY)
print(maxValue)
```

```
Enter the first value: 3.14
Enter the second value: 3.13
3.14
```

Are you just reading? Take a pause and implement the above functionality yourself!

Exercise:

1. Implement a function ***fib (n)*** to print ***n*** Fibonacci terms as 1st term = 1, 2nd term = 1, 3rd term = 2 and so on. Call this function from the Python code that accepts the value of ***n*** from the user.

Function with Default Arguments

- Assume, there is a requirement to implement a function which would be called with two arguments. The first argument is the name of the person and the second argument is a greeting message.
- The first argument is mandatory but in case if the second argument is not provided, the function will simply take the greeting message as ***" how are you?"***
- The required functionality can be implemented using a mechanism that is called the ***default argument*** as shown below:

```
def printMessage (name, msg = " how are you?"):
    print(name + msg)
```

- Notice that the default value of the greeting message is provided with the second argument ***msg***. When this function is called with and without the second argument, observe the output:

```
printMessage ("Mahesh,", " Happy Birthday!")
printMessage ("Mahesh,")
```

```
Mahesh, Happy Birthday!
Mahesh, how are you?
```

Exercise:

1. What would happen if the value of the default argument is not provided in the above function and the function is called without any second argument?

Recursion

- If a function calls itself to implement a functionality, it is called ***recursion***. The following example conveys the idea.

```
def fib(term):
    if term == 1 or term == 2:
        return 1
    else:
        return fib(term-1) + fib(term-2)
```

- Notice that if the value of the argument term is either 1 or 2, the value 1 would be be returned. Otherwise, the sum of the previous two fibonacci terms will be returned.

Exercise:

1. Implement a recursive function to calculate the value of a factorial? Note that n! =n(n-1)(n-2)(n-3)..1