Chp05 User Defined Functions

References

1. https://www.w3schools.com/python/python_functions.asp w3school Python Functions

2. https://www.tutorialspoint.com/python3/python_functions.htm Tutorialspoint (Python 3 Functions)

3. *https://docs.python.org/3/tutorial/index.html* sections 4.7, 4.8 *(4.8.1-4.8.3). This can also be obtained from Python manual*

4. *Learning Python, 5th, Chapter 16, 17 and 18*

5. *Python for Everybody – Chapter 4 - Functions*

- So far we have used built-in functions: max, min, sum, len, etc
- Used also functions defined in other modules such as math, cmath,, statistics, etc
- It is also possible to develop user-defined functions
- Once we define a function, we can reuse the function over and over throughout our program
  - This reduces the size of the code and also improves readability.

Defining a function - I

Syntax:

```
def functionName([param1, param2,…, paramn]):
    body_of_function
```

- The first line starting with def is known as the function header. It marks the beginning of the function definition.
- The colon at the end of function header should be there. Parameters are optional.
- The **body** consist of statements which are executed when a function is called.
- The body of the function should be indented.
- Function name rules are similar to variables names

Defining a function - II

Note: The body of the function must have contents. If you don't want to put contents, just put the word **pass** as contents. pass keyword means do nothing and is ignored by the Python interpreter.

This can be useful during the design of your program. Example:

```
def funcname():
    pass
```

Calling a function

To call a function, specify the name of the function followed by arguments in (). Function must be defined first before it is called. Example:
chp05ex01

```
def displayHelloWorld():
    print('Hello world!')


#call a function
displayHelloWorld() #() must be there
#Results: Hello world!
```

- Leave a double space between functions to improve code readability (if you have two or more functions)
- Leave a double space between functions and codes calling functions to improve code readability

Function with Parameters -I

You can add as many parameters as you want, just separate them with a comma.**Example:** chp05ex02

```
def sumTwoNumbers(num1,num2):
    print(f'{num1} + {num2} = {num1+num2}')


sumTwoNumbers(12,34.5)
sumTwoNumbers(25,50)
```

.....

Important: The order of arguments matters when a function is called (`positional arguments`)

Arguments vs parameters

Difference between argument(s) and parameter(s).

- A parameter is the variable listed inside the parentheses in the function definition.
- An argument is the value that is sent to the function when it is called.
- Some authors uses them interchangeably!

In example chp05ex02:

num1 and num2 are **parameters** while 12 and 34.5 are **arguments**

return statement - I

Use `return` keyword to return values. Can return any data type (number, string, list, tuple, dictionary, boolean, etc.). Example: chp05ex03

```
def numVowels(str):
    vowelStr='aeiou'
    count = 0
    for char in str.lower():
        if char in vowelStr:
        count += 1
    return count
```

return statement - II

```
inStr=input('Enter String: ')
print(f'Entered string is: {inStr}')
print(f'There are {numVowels(inStr)} vowels in
the entered string')


#Results:
Enter String: Programming in Python
Entered string is: Programming in Python
There are 5 vowels in the entered string.
```

return statement - III

Upon encountering a **`return`** statement, Python evaluates the expression and immediately transfers control back to the caller of the function. The value of the expression is also sent back to the caller. If a function contains no return statement, Python transfers control to the caller after the last statement in the function's body is executed, and the special value None is automatically returned.

Note: **return** statement with no return value is the same as return None. Some authors places the return without a return value at the end of the function.

Returned value can be assigned to a variable. Refer example chp05ex04

Passing a list, tuple, dictionary, etc as argument

You can send any data types of parameter to a function (string, number, list, tuple, dictionary etc.), and it will be treated as the same data type inside the function.
Example of passing a list:

Example: chp05ex05

```
def listAsArg(countries):
    for country in countries:
        print(country, end=' ')


countriesNames = ['Tanzania', 'Kenya',
'Uganda', 'Rwanda', 'Burundi']
listAsArg(countriesNames)
```

Predicate functions

Predicate functions return True or False. In strings we have functions such as `isalpha`, `isalnum`, etc

Example: chp05ex06

function **isOdd** returns True if its argument is odd otherwise it return False. Make sure the argument is checked before it is passed to a function. This example also illustrates that you can have multiple **returns** but only one will be used to return a value.

```
def isOdd(num):
    if num % 2 == 1:
        return True
    else:
        return False
```

Example chp05ex07 shows how you call a function a number of times

Keyword arguments -I

- Keyword arguments allow to match by name, instead of by position using the *parameter = value* syntax.
- This way the order of the arguments does not matter.

Example: chp05ex08

```
def power(num, x):
    return num ** x


y1=power(4,3) # num=3, x=3 Postion arguments
y2=power(x=3, num=4) #num and x are specified
print(f'power(4,3) ={y1}') #64
print(f'power(x=3, num=4) ={y2}') #64 - same
```

Keyword arguments -II

Important: positional arguments can be mixed with keyword arguments during a function call, **BUT** keyword arguments must come after positional arguments.

```
def power(num, x):
    return num ** x


y3=power(3, x=4) # x is specified. OK
y4=power(num=3, 4) #position arg follow
keyword # NOT ALLOWED. Results into error
```

Note: For functions which takes numerical values, better use keyword arguments. Example:

```
calcCost(total=50, shipping=5, discount=0.1)
```

Default Parameter Value -I

A default parameter is a parameter that assumes a default value if a value is not provided in the function call for that parameter. Example: chp05ex09

```
def power(num, x=1): #x=1 is default value
    return num ** x


y1=power(4) # num=4, x=1
y2=power(4,3) #num=4, x=3
print(f'power(4) ={y1}')  #4
print(f'power(4,3) ={y2}') #64
```

Default Parameter Value -II

Important:

- Any number of parameters in a function can have a default value. BUT once we have a default parameter, all the parameters to its right must also have default values.
- This means to say, non-default arguments cannot follow default arguments
- That means, following declaration is wrong:

```
def power(x=1, num): #error
```

Arbitrary Arguments (Variable length arguments) - I

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition. This tuple remains empty if no additional arguments are specified during the function call

```
def multiArgs(name,*args):
    function body
return …….
```

- The * means you can pass any number of values (separated by comma).
- The **args** is treated as **tuple** and iterator can be used to get the values.
- It is recommended the ***args** to be the last parameter.

Variable length arguments - II

Example: chp05ex10

```
def multiArgs(name,*nums):
    result=0
    for item in nums:
        result=result+item
    return name+' '+str(result)


print(f'multiArgs("Asha")= {multiArgs("Asha")}') #
Asha 0
print(f'multiArgs("Juma",1,2)=
{multiArgs("Juma",1,2)}') # Juma 3
print(f'multiArgs("John",1,2,3,4)=
{multiArgs("John",1,2,3,4)}') #John 10
print(f'multiArgs("Asha",1,2,3,4,5)=
{multiArgs("Asha",1,2,3,4,5)}') # Asha 15
```

Passing Arguments by Object Reference

- In Python, arguments are passed to functions by assignment (object reference)
- Let us see how mutability and immutability works in functions

Example: chp05ex11 & chp05ex12

To avoid the effect on mutable objects, make a copy of the mutable object before you call a function.

```
numbersList = [10,20,30]
numbersListTemp = numbersList.copy()
# call the function with numbersListTemp
numbersList2= changeList(numbersListTemp)
```

global vs local variables - I

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions.

```
global variables
function definition
    function body  # can access global variables

Script # within script you can access
       # also global variables
```

global vs local variables - II

- The global statement tells Python that a function plans to change one or more global variables.
- If global value is not intended to change within the function then it can be used without the use of the word global.
- The global statement consists of the keyword global, followed by one or more names separated by commas.
- The statement global VarName tells Python that VarName is a global variable.

Example: `chp05ex13` shows how to use the word global to change the global variable.

Returning Multiple Values

Refer to: Python for Data Analysis - Data Wrangling...Pandas, NumPy, and IPython,2nd(McKinney Oreiily 2017) - Chp03 Built-in Data Structures, Functions, and Files.

Python function can return multiple values (e.g. list, tuple, dictionary)

Refer to examples: chp05ex14 and chp05ex15

Multiple Functions: Can have multiple functions within a file. A function can be called within another function.

```
def func1(param1, param2,...):
    body of func1


def func1(param1, param2,...):
    body of func2


# Here insert code which utilizes func1 and
func2
```

Example: chp05ex16

Modules

References

- https://www.w3schools.com/python/python_modules.asp
- https://www.tutorialspoint.com/python/python_modules.htm
- Learning Python, 5th (Mark Lutz Oreilly 2013) – chapter 22 (Modules : The big picture), chapter 23 (Module coding Basics)
- https://docs.python.org/3/tutorial/modules.html Modules - python documentation

- A module is a file consisting of python code
- A module can define functions, variables, classes, etc
- A module allows you to organize your python code
- Grouping related code into a module makes the code easier to understand and use
- The name of the module is the same as the python filename
- Used with modular programming, which is to separate a program into parts.

So far used built in modules in python such as math, cmath, statistics, etc.

```
import math
print(f'pi={math.pi} sqrt(6)={math.sqrt(6)}')

import statistics as stat #using alias
tp=(12,45,78,23)
print(f'Mean(tp)= {stat.mean(tp)} stdev(tp)=
{stat.stdev(tp)}')

from math import pi, sqrt
print(f'pi={pi} sqrt(6)={sqrt(6)}')
```

For simplicity, myUtilities.py exists in the same folder as the notebook. Functions and variables are in the file myUtilities.py. Demo: Open in notebook

```
def power(num,x):
    return num ** x


def isOdd(x):
    if x % 2 == 1: return True
    else: return False
```

……..

Using myUtilities.py - I

Example: chp05ex17

```
import myUtilities #do NOT include extension .py
print(f'Sum of 5 and 7 is:
{myUtilities.addTwoNumbers(5, 7)}')
print(f'2 power 4 is: {myUtilities.power(2,
4)}')
print(f'Is 5 an odd number?
{myUtilities.isOdd(5)}')
print(f'Value of x is {myUtilities.x}')
print(f'Value of variable countries is:\n
{myUtilities.countries}')
```

Using myUtilities.py - II

Results of chp05ex17

```
Sum of 5 and 7 is: 12
2 power 4 is: 16
Is 5 an odd number? True
Is 6 an odd number? False
Value of x is 25
Value of variable countries is:
['Tanzania', 'Kenya', 'Uganda',
'Rwanda', 'Burundi', 'Somalia']
```

Reloading a module -I

- A module is loaded only once, regardless of the number of times it is imported.
- The import statement checks if the module is already in memory and does the actual importing **ONLY IF** this is the first time the module is used
- After importing a module, if something changes in the module or you want to refresh in case some data have changed, you can reload the module. This is mostly required when using jupyter notebook

1. Restart the kernel if you are using jupyter notebook

OR

2. Use: `importlib.reload(moduleName)`

Reloading a module -II

```
import importlib
import moduleName
……
#re-load the module
importlib.reload(moduleName) #The module must
have been successfully imported before.
```

Example: **chp05ex18**

```
import importlib
import myUtilities #module name
print(myUtilities.x)
importlib.reload(myUtilities) #reload
print(myUtilities.x)
```

dir function

The `dir()` built-in function returns a sorted list of strings containing the names defined by an object.

Example: chp05ex19 – showing all attributes in math module

Example: chp05ex20 shows how to exclude attributes which starts with __ (double underscore) for a number of objects.

Module search path

The module search path is stored in the system module **sys** as the **sys.path** variable. The `sys.path` variable contains the current directory, PYTHONPATH, and the installation-dependent default.

Example: chp05ex21

Demo: How to set PYTHONPATH in windows O/S to include your folders where python can look for modules:

==End of chapter 5 ==

Chp06 Working with Files (Students to prepare their own notes). Assignment to be provided BUT won't be submitted

Next chapter: Chp07 Exceptions Handling