[2] Functions in Python

December 31, 2022

1 Lab Manual-02

2 Functions in Python

2.1 1. Functions

You may have come across functions in other computer languages, where they may have been called subroutines or procedures.

Functions serve two primary development roles: * Maximizing code reuse and minimizing redundancy. * Procedural decomposition by splitting systems into pieces that have well-defined roles.

Functions reduce our future work radically and if the operation must be changed later, we only have to update one copy in the function, not many scattered copies throughout the code.

In Python, **"def"** creates a function object and assigns it to a name. Let's start with a simple example of our first function!

[1]:
```python
# A simplest example of a function is:
def function_name(pram1):
    """
    Body: Statements to execute
    """
    print(pram1) # this will print the pram1 only

    # We can concatenate two string together with + sign.
    print(pram1 +', this is Python') # this will print the concatenated string
```

So, we have defined/created our first function in the above code cell with its name **"function_name"**. Once, the function is defined, we can call that function with its name in our code.

In this case, our created function **"function_name"**, needs a parameter "pram1". We need to pass a parameter when calling the function **"function_name"**.

Let's do this!

[2]:
```python
# function call with its name
function_name('Hello world')
```

```
Hello world
Hello world, this is Python
```

[3]:
```
# function call without parameter
function_name()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2264\4173951694.py in <module>
      1 # function call without parameter
----> 2 function_name()

TypeError: function_name() missing 1 required positional argument: 'pram1'
```

To avoid this error, we can use a default value for the parameter **"pram1"**.

[4]:
```
def function_name(pram1 = 'Default Value'):
    print(pram1)
    # Let's concatenate two string together with + sign.
    print(pram1 +', this is Python')
```

Now if we don't pass the parameter during the function call, it will print the Default Value.

[5]:
```
# Function call without the parameter
function_name()
```

```
Default Value
Default Value, this is Python
```

If we pass the parameter, it will replace the Default Value.

[6]:
```
function_name('Hi') # or function_name(pram1='Hi')
```

```
Hi
Hi, this is Python
```

Good to know!

The terms "parameter and argument" may have different meanings in different programming languages. Sometimes they are used interchangeably, and the context is used to distinguish the meaning. The term parameter (sometimes called formal parameter) is often used to refer to the variable as found in the function definition, while argument (sometimes called actual parameter) refers to the actual input supplied at function call. For example, if one defines a function as ***def f(x):*** *..., then* ***x*** *is the parameter, and if it is called by* ***a = ...; f(a)*** *then* ***a*** *is the argument.*

Lets create a function that takes two parameters (type number in this case) and **returns** the multiplication of the numbers.

[7]:
```
# function returns the result
def my_func(num1, num2):
    num = num1*num2
    return num
# we can write "return num1**num2" in a single line as well
```

[8]:
```
# Function call
print(my_func(2,3))
```

```
6
```

We can get the results from our function in a variable and then print that variable, **let's try!**

[9]:
```
out = my_func(2,3)
print(out)
```

```
6
```

Functions can have documentation strings (docstring) enclosed b/w " ""doc"" ". *I want to say, a function should have a document string, they are very useful.* Let's create a docstring for our example function.

[10]:
```
def my_func(num1, num2):
    """
    These docstrings are very useful
    Jupyter has a great feature for these strings
    write function name, and click shift + tab
    my_func -- shift + tab
    You will see these doc string
    """
    return num1**3
```

[11]:
```
#my_func - then press shift+tab to see the doc string
my_func

print(my_func.__doc__)  # to print docstring
```

```

    These docstrings are very useful
    Jupyter has a great feature for these strings
    write function name, and click shift + tab
    my_func -- shift + tab
    You will see these doc string

```

Again, DocStrings are very useful, it is always encouraged to write a proper documentation of your custom function.

You can find the documentation of all the built-in functions very useful. If we type range and click < shift+tab > (in jupyter notebook environment), we will see the documentation of range, we don't need to memorize this all! We will be using this feature lots of time in this course!

[12]:
```
#some built in functions

print(max(3,4,1))
print(min(2,1,5))
print(max('a','A','b','Z', 'z'))
```

```
4
1
z
```

[13]:
```
print(round(4.764))
print(round(4.324))
print(round(4.764,1))
print(round(4.764,2))
```

```
5
4
4.8
4.76
```

[14]:
```
#
#Let's write an if condition in a function
#
def function_1():
    x,y =2,8

    if(x < y):
        st= "x is less than y"
    print(st)


function_1()    #calling a function named "funtion_1"
```

```
x is less than y
```

[15]:
```
# Function to compute square
def square(num):
    return num**2
```

[16]:
```
# Function call using its name 'square' and a required parameter
print(square(4))
```

```
16
```

Let's re-arrange the above function "square" in a single line.

[17]:
```
def square(num):return num**2
print(square(4))
```

```
16
```

3 Exercise

3.1 Mass and Weight

Scientists measure an object's mass in kilograms and its weight in newtons. If you know the amount of mass of an object in kilograms, you can calculate its weight in newtons with the following formula: **

weight = mass * 3 9.8

** Write a program that asks the user to enter an object's mass, and then calculates its weight. If the object weighs more than 500 newtons, display a message indicating that it is too heavy. If the object weighs less than 100 newtons, display a message indicating that it is too light.

3.2 Hot Dogs Cookout Calculator

Assume that hot dogs come in packages of 10, and hot dog buns come in packages of 8. Write a program that calculates the number of packages of hot dogs and the number of packages of hot dog buns needed for a cookout, with the minimum amount of leftovers. The program should ask the user for the number of people attending the cookout and the number of hot dogs each person will be given. The program should display the following details: * The minimum number of packages of hot dogs required * The minimum number of packages of hot dog buns required * The number of hot dogs that will be left over * The number of hot dog buns that will be left over

3.3 Miles Per Gallon

Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several tankfuls of gasoline by recording miles driven and gallons used for each tankful. Develop a sentinel-controlled-repetition script that prompts the user to input the miles driven and gallons used for each tankful. The script should calculate and display the miles per gallon obtained for each tankful. After processing all input information, the script should calculate and display the combined miles per gallon obtained for all tankfuls (that is, total miles driven divided by total gallons used).

- Enter the gallons used (-1 to end): 12.8
- Enter the miles driven: 287
- The miles/gallon for this tank was 22.421875
- Enter the gallons used (-1 to end): 10.3
- Enter the miles driven: 200
- The miles/gallon for this tank was 19.417475
- Enter the gallons used (-1 to end): 5
- Enter the miles driven: 120
- The miles/gallon for this tank was 24.000000
- Enter the gallons used (-1 to end): -1
- The overall average miles/gallon was 21.601423

3.4 Software Sales

A software company sells a package that retails for $99. Quantity discounts are given according to the following table:

Quantity	Discount
10–19	10%
20–49	20%
50–99	30%
100 or more	40%

Write a program that asks the user to enter the number of packages purchased. The program should then display the amount of the discount (if any) and the total amount of the purchase after the discount.

3.5 Shipping Charges

The Fast Freight Shipping Company charges the following rates:

Weight of Package	Rate per Pound (Dollar)
2 pounds or less	1.50
Over 2 pounds but not more than 6 pounds	3.00
Over 6 pounds but not more than 10 pounds	4.00
Over 10 pounds	4.75

Write a program that asks the user to enter the weight of a package and then displays the shipping charges.

3.6 Body Mass Index

Write a program that calculates and displays a person's body mass index (BMI). The BMI is often used to determine whether a person is overweight or underweight for his or her height. A person's BMI is calculated with the following formula:

**

BMI = weight * 703 / height^2

**

where weight is measured in pounds and height is measured in inches. The program should ask the user to enter his or her weight and height and then display the user's BMI. The program should also display a message indicating whether the person has optimal weight, is underweight, or is overweight. A person's weight is considered to be optimal if his or her BMI is between 18.5 and 25. If the BMI is less than 18.5, the person is considered to be underweight. If the BMI value is greater than 25, the person is considered to be overweight.

[ ]:

```

```