Functions in Python

The concept of function comes to decrease the reusablity of code suppose after some operation or task we have to call a program again and again after some operation. Just like take an example of ATM where you input your data and information like information of your account and PIN Code then it gives you money and receipt after that at last it print message **Thanks for using our ATM**. This is all possible due to function. Here, they use first function for taking second function for processing and giving money and third function for thanking users. This will goes continue when any user come and use the ATM Machine. This is how function is used in the real world problem. Function actually takes input and process the input then after yields the output after processing.

In [6]:
```
def func():
    print("Hello Amrit Keshari")
```

Function is a line of code which is in the particular code block that starts with the keyword named **def** then we write name of the function and then use parentheses and after that ends with colon then goes to new line and write the program of the task we want to do in our computer by using indentation. After colon in next line first you have to take either four space or one tab then start writing program. By this way, we can define the function in python. This indentation shows that your code is in the code block of particular funtion and it runs only if when its particular function is called.

In [9]:
```
func()
```

```
Hello Amrit Keshari
```

To call or invoke the function, we just have to write the name of the function and after that write the parentheses if function having some parameter then put values into the argument and after that you should call or invoke but if the function having no parameter then you do not need to pass any argument during the invoking or calling a function.

Parameter Versus Arguments

Parameters are the variables of function that is passed in the parentheses of the function during defining of the function. When we define the function then we pass some variable that shows function need some input which required to perform the task of the function. Whereas arguments are the variables that is passed in the paretheses of the function but during function call or invoke. At that time of function call we pass number of arguments required for the function. Note that the number of parameters of the particular function is same as the the number of particular or the very function arguments.

Function Without Parameter

Function without parameter is defined by writing firstly **def** keyword and after writing name of the function and then parentheses and after parentheses we put colon we do not pass any required variable for function in the parentheses of the function. After this we just write the program of the function in the function code block by following the rule of indentation. Function without parameter is mainly used in **greeting messages** to display in the computer screen.

In [19]:
```
#define function without parameter
def welcomeSir():
    print("Welcome! Dear Sir")
#call function
welcomeSir()
```

```
Welcome! Dear Sir
```

Function With Parameter

Function with parameter is defined by writing firstly **def** keyword and after that the name of the function and then write the parentheses and then we put colon. But in this function we pass some variables in the paretheses of the function that means function is taking some input to process it then it yields output as a result. Function with parameters are mainly used in the function that perform some mathematical and logical operations on input data and values. Just like calculating bank balance after crediting or debiting money for that a function with parameter is required to calculate the total amount after withdrawl or deposit of money. We can add as many parameter as we want by separating each of them with comma. If the function takes **n** number of arguments only if it has **n** numbers of parameter. It is noted that number parameters equals to number of arguments.

In [26]:
```
#define function with parameter
money = 100
def credit(x):
    print(f"Amount after credit:{money + x}")
def debit(x):
    print(f"Amount after debit:{money - x}")
#define function without parameter
def amount():
    print(f"Amount:{money}")
```

Now see the calling or invoking of function with arguments and function without arguments

```
In [30]: #calling function with arguments
         credit(4)
         debit(7)
         #calling function without argument
         amount()
```

```
Amount after credit:104
Amount after debit:93
Amount:100
```

Default Parameter in Function

As we all know that parameters of a function is the variable of function that is passed in the parentheses of the function during function definition. So there is a concept of default parameter that says that during function defination that variable that is passed in the parentheses of function is also defined. If someone calls the function without argument then that function variable takes the value of default parameter as argument of the function by default. We can give as many as default parameter in a function. We can combine both normal parameter and default parameter in the parentheses of the function when it is needed.

```
In [3]: #default parameter
        def myBook(author="Amrit"):
            print(f"Author:{author}")
```

```
In [5]: #function calling with argument
        myBook("Aman Gupta")
        myBook("Raj Agarwal")
        #function calling without argument
        myBook()
```

```
Author:Aman Gupta
Author:Raj Agarwal
Author:Amrit
```

Keyword Arguments in Function

Keyword arguments is the concept in python in which when we pass arguments during function call, in the parentheses of a function then it is called positional arguments in function because its arguments are values or datas only, but when we pass arguments in the form of key value pair in the parentheses of function it means **key="value"** then that argument is called keyword argument of the function.

```
In [10]: #define function
```

```
def amritFunction(friend):
    print(f"My friend is {friend}")
    print(f"How are you {friend}")
```

In [12]:
```
#keyword argument
amritFunction(friend="Aditya")
amritFunction(friend="Deepak")
```

```
My friend is Aditya
How are you Aditya
My friend is Deepak
How are you Deepak
```

In a function in python, we can pass both positional arguments and keyword arguments and combination of positional arguments and keyword arguments. But when you use **(*,variableName)** asterick sign comma variableName in the parentheses of the function. Then it is mandatory to use only keyword argument if you do not use keyword argument or instead of using keyword argument you use positional argument then you get error. When you use **(variableName,/)** variable name comma backslash in the parentheses of the function. Then it is mandatory to use positional argument. If you use keyword arguments instead of positional arguments then you will get error.

By using (*,variableName) Mandatory for keyword arguments

In [15]:
```
#keyword arguments only allowed
def amritFunction(*,x):
    print(f"X:{x} balls")
```

In [17]:
```
#function call with positional arguments
amritFunction(8)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 2
      1 #function call with positional arguments
----> 2 amritFunction(8)

TypeError: amritFunction() takes 0 positional arguments but 1 was given
```

See it throws error when i use positional argument in (*,)

In [19]:
```
#function call with keyword arguments
amritFunction(x=5)
```

```
X:5 balls
```

By using (variableName,/) Mandatory for positional arguments

In [23]:
```
#positional arguments only allowed
```

```
def amritFunction(y,/):
    print(f"Y:{y} Bats")
```

In [25]:
```
#function call with positional arguments
amritFunction(7)
```

```
Y:7 Bats
```

In [27]:
```
#function call with keyword arguments
amritFunction(y=9)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 2
      1 #function call with keyword arguments
----> 2 amritFunction(y=9)

TypeError: amritFunction() got some positional-only arguments passed as keyword
arguments: 'y'
```

See it throws error when i use keyword argument in (,/)

Arbitary Arguments in Function

Arbitary Arguments is the concept in python in which if we unknown about the number of parameters of the function then we use arbitary arguments. During function calling or invoking, as we do not know the number of parameter in the function then we can pass one argument, two arguments and multiple arguments as your wish number of arguments depends. During function definition, in parameter of the function **(* variableName)** single asterick sign with variable name is written which shows unknown about the number of parameters in the function. So, we can easily put one or more than one argument in the function after seeing this.

In [48]:
```
#define function
def amritFun(*a):
    print(a)
```

In [50]:
```
#calling function with arbitary arguments
print("First Call: With 3 arguments")
amritFun(23,45,90)
print("Second Call: With 2 arguments")
amritFun(65,65)
```

```
First Call: With 3 arguments
(23, 45, 90)
Second Call: With 2 arguments
(65, 65)
```

See it prints all the values that we pass as an argument during function call. But when it comes to print all the elements one by one using print function it print all the elements and it can be possible using loop statements easily. Lets have a look.

```
In [55]: def amritFun(*a):
             for i in a:
                 print(i)
         amritFun(1,2,3,4,5,6,7)
```

```
1
2
3
4
5
6
7
```

```
In [58]: def amritFun(*a):
             print(a[0])
             print(a[1])
             print(a[2])
         amritFun(1,2,3,4)
```

```
1
2
3
```

Keyword Arbitary Arguments in Function

Keyword Arbitary Argument is the concept in python especially for keyword arguments only. During function definition in python, when we pass **(** variablename)** two astericks sign with variable name that it shows number of keyword argument is unknown in this particular function. So, during function calling or invoking we pass one keyword argument, two keyword arguments and multiple keyword arguments, it is upto you that how much arguments you want to pass in this particular function.

```
In [63]: #define function
         def amritFun(**a):
             print(a)
```

```
In [65]: #function call with 5 key argument
         amritFun(x=1,y=2,z=3,w=5)
         #function call with 3 key argument
         amritFun(x=4,y=8,z=9)
```

```
{'x': 1, 'y': 2, 'z': 3, 'w': 5}
{'x': 4, 'y': 8, 'z': 9}
```

List as an Argument

Yes, we can pass list as an argument of the function during function call. When we pass list as an argument then we can perform any operation with the list by using loop statement inside the function code block.

In [74]:
```
#define function
def amritFun(list1):
    for i in list1:
        print(f"Element:{i}")
```

In [76]:
```
#calling function
listAmrit = [68, "Amrit", "Keshari",True]
amritFun(listAmrit)
```

```
Element:68
Element:Amrit
Element:Keshari
Element:True
```

Return & Pass Statement

As we know that, function return data or values as a output in the program. For returning datas or values we use return statement that directly return the value. We can return any value from the function by using **return** keyword. Now, about pass statement, it is mainly used when we define a function but not writing any code inside its code block in that case to execute that function we use **pass** keyword.

In [86]:
```
#define function
def firstDouble(list1):
    return list1[0]*2
```

In [90]:
```
#function call
listAmrit = [1,2,3,4,5]
x = firstDouble(listAmrit)
print(x)
```

```
2
```

Pass Statement Program

In [93]:
```
#define function
def amritFunction():
    pass
#call function
amritFunction()
```

After using pass statement the function runs easily without any error.