Function in Python

A function in Python is a block of code that performs a specific task. Functions allow you to encapsulate a piece of code and reuse it multiple times throughout your program. Functions can take input in the form of parameters and can return an output in the form of a return value. In Python, you can define your own functions using the "def" keyword, followed by the name of the function, a set of parentheses that may contain parameters, and a colon after which body of function begins.

In [1]:
```
#lets create a function
def myFunction():
    print("This is my first custom-function")
#function wil only executed when it is invoked (called), a function can be invoked using functionName followed
#by parenthesis as myFunction()
myFunction()
```

```
This is my first custom-function
```

In [2]:
```
#function with parameters
def myFunction(name):
    print("Welcome Mr.{}".format(name))
#function call
myFunction("Mohammad Nasir")
```

```
Welcome Mr.Mohammad Nasir
```

```
There is a lot of discussion about what is parameter and what is arguments. the parameters are variables defines
in signature of function as myFunction(name) here name is a parameter which can take some value and argument is
the value which is passed when a function is inovked as myFunction("Muhammad Nasir")
```

Parameters are Fixed in Function

In [11]:
```
#In python the parameters can be defined as many as required but at function call it is necessory to provide
#value for each parameter otherwise error will be generate,
def myFun(val1,val2,val3):
    print("Average of given numbers is:{}".format((val1+val2+val3)/3))
myFun(4,5)#it will generate error as expected parameters are three but we passed two
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 5
      3 def myFun(val1,val2,val3):
      4     print("Average of given numbers is:{}".format((val1+val2+val3)/3))
----> 5 myFun(4,5)

TypeError: myFun() missing 1 required positional argument: 'val3'
```

In [12]:
```
#lets call above function with three arguments
myFun(4,6,5)
```

```
Average of given numbers is:5.0
```

Unknown number of parameters / Optional Positional Arguments

In [13]:
```
#Python is best for some reason it allow us any number of parameters we want using a single * with parameter
#it defines that this function can have any number of parameters and parameters inside method can be accessed
#via tuple as function provide passed argument as tuple object, let see an example
def myFun2(*parameters):
    print(type(parameters))
myFun2()
```

```
<class 'tuple'>
```

```
as shown above even we don't pass any argument it return empty tuple
```

In [19]:
```
#example 2 unknown parameters
def calculateSum(*numbers):
    if(numbers.count!=0):
        return sum(numbers)
    else:
        return 0
print(calculateSum(1,23,4,98,87))
print(calculateSum(1,23,4,98,87,99,129,234))
print(calculateSum(1,23,4))
```

```
213
675
28
```

Named Arguments

In [22]:
```
#we can also pass named agurments by which the sequence of arguments does not matter, let see an example,
def myFunction(name,roll,age):
    print("StudentInfo:\nRoll:{}\tName:{}\tAge:{}".format(roll,name,age))
#invoke the function
myFunction(roll="SP20M2BB030",name="Mohammad Nasir",age=23)
```

```
StudentInfo:
Roll:SP20M2BB030        Name:Mohammad Nasir     Age:23
```

In above function call we paseed argument using their name instead of passing them in a sequence as they are

Parameters with Default Value

```
we can also pass a default value of specific parameter which will only be used when the value of parameter
is not passed when function is invoked for example,
```

In [36]:
```
def myFunction(itemName,itemUnitPrice,itemCount=1):#in this function first two parameters are mendatory but
    #the third one itemCount is optional and when this is not passed it means item count is 1 simply
    total = itemUnitPrice * itemCount
    print('''OrderInfo:\nItemName:{}\nUnitPrice:{}
ItemCount:{}\nTotalValue:{}'''.format(itemName,itemUnitPrice,itemCount,total))
myFunction(itemName="Iphone14",itemUnitPrice=400000)
```

```
OrderInfo:
ItemName:Iphone14
UnitPrice:400000
ItemCount:1
TotalValue:400000
```

In [37]:
```
#Example2
myFunction(itemName="AppleWatch",itemUnitPrice=5000,itemCount=10)
```

```
OrderInfo:
ItemName:AppleWatch
UnitPrice:5000
ItemCount:10
TotalValue:50000
```

Passing List as Argument

```
We can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated
as the same data type inside the function.
```

In [38]:
```
def printListItem(myList):
    for value in myList:
        print(value)

#lets inovke function
printListItem(["Nasir","Jillani","Jibran","GM","Awais","Jahantab"])
```

```
Nasir
Jillani
Jibran
GM
Awais
Jahantab
```

In [41]:
```
#passing dictionary in function
def printDict(dictionary):
    for key,value in dictionary.items():
        print(key,value)
#lets inovke function
printDict({"Name":"Nasir","Roll":"SP20M2BB030","Class":"BSCS 7THE"})
```

```
Name Nasir
Roll SP20M2BB030
Class BSCS 7THE
```

Returning Value from Function

```
We can return any type of value from function
```

In [43]:
```
def getSum(*values):
    sum = 0;
    for value in values:
        sum+=value
    return sum

sum = getSum(1,3,5,7,9)
print(sum)
```

```
25
```

Function with empty body

```
Normally we can't create a function in python which don't have body, even loop, and conditional statment
at least require one statment and when we want to create a function with empty body we can use pass keywrod
which tells framework that listen bro there is nothing to execute go and execute next statment,
```

In [44]:
```
def myFunction():
    pass
print(myFunction())
```

```
None
```

```
None printed because above function returns None
```

None printed because above function returns None

Return multiple values from function

In [47]:
```
#we can return multiple values from function via tuple and can assign these values to multiple objects as,
def calculateSumAverageMod(value1,value2):
    sum = value1+value2
    avg = (value1+value2)/2
    mod = value1%value2
    return (sum,avg,mod)
#lets invoke function
sum,avg,mod = calculateSumAverageMod(26,5)
print("sum:{}".format(sum))
print("avg:{}".format(avg))
print("mod:{}".format(mod))
```

```
sum:31
avg:15.5
mod:1
```

Recursive Function

In [55]:
```
#A function which call / invoke it self untill a specific condtion met is known as recursive function, for example,
def calculateFactorial(value):
    factorial=1;
    #if value is greater than zero multiply that value in factorial and inovke same function with decrementing 1 fro
    if(value>0):
        factorial = value * calculateFactorial(value-1)
        #at end return the value
    return factorial
#calling function to calculate factorial
print(calculateFactorial(6))
```

```
720
```

Function Description

```
We can also provide a function description inside a function and anyone can use that to check what actually
function is about using functionName.__doc__
```

In [60]:
```
def getSum(val1,val2):
    "A function which will take 2 values and return their sum"
    return val1+val2
print(getSum.__doc__)#it will print the function detail
```

```
A function which will take 2 values and return their sum
```

What is Method?

```
A function is a standalone block of code that performs a specific task, while a method is a function that is
associated with an object and has access to the properties and behaviors of that object.
```

```
Function is like a global variable can be accessed anywhere in program while a method is assosiated with specific
class and can only be access using its class object or if method is static which is defines using @staticmethod
rest procedure is same will be accesed using class name
```

In [65]:
```
class Student:
   def __init__(self,name,age,roll):
    self.name=name
    self.age=age
    self.roll=roll
st1 = Student("Nasir",23,"SP20M2BB030")
print(st1.roll)
```

```
SP20M2BB030
```

```
In Python, self is a special parameter that is automatically passed to methods defined in a class. It refers to the
instance of the class itself, and is used to access the properties and behaviors of that instance.
```

In [69]:
```
#lets create a method in our class and see how we can use
#non-static method with object
class Student:
    def __init__(self,name,age,roll):
        self.name=name
        self.age=age
        self.roll=roll
#here I create a method which return string, returning fields and their value stored in current
#object of this class, self is a must parameter in python
    def toString(self):
        return '''
Name:{}
Roll:{}
Age:{}'''.format(self.name,self.roll,self.age)


st1 = Student("Nasir",23,"SP20M2BB030")
print(st1.toString())
```

```
Name:Nasir
Roll:SP20M2BB030
Age:23
```

Age:25

It is necessary to include self as the first parameter in a method within a class in Python. self refers to the instance of the class that the method is being called on, and allows the method to access the properties and behaviors of the instance.

In [ ]: