Functions:

1. To create a function in python we use def keyword, it defines the function
2. We pass function name to identify function and followed by ()parentheses
3. We pass : (colon) at the end of function as its scope and it is indented
4. Inside the function we can have modularity code
5. As our program becomes complex it makes easier and manageable

There are mainly two types of functions.

User-define functions :

1. The user-defined functions are those define by the user to perform the specific task.

Built-in functions :

1. The built-in functions are those functions that are pre-defined in Python.

Function Definition:

```
def d1():
   print("d1 function")
```

Function call

```
d1()
```

IndentationError

```
def d1():
print("d1 function")
d1()
```

IndentationError: expected an indented block

```
# Create a function with no arguments
def d1():
   print("d1 function")

def d2():
   print("d2 Function")

d1() # d1 function
d2() # d2 Function
```

```
# Create a function with arguments
# Once we create a function we use multiple times
def d1(eid, ename):
   print(eid, ename)

d1(101, "Hari") # 101 Hari
d1(102, "Manoj") # 102 Manoj
d1(103, "Jagadesh") # 103 Jagadesh
d1(104, "Vinod") # 104 Vinod
```

```
# Function using key and value
def emp(eId, eName):
   print(eId, eName)

emp(eId=101, eName="Hari") # 101 Hari
emp(eId=102, eName="Manoj") # 102 Manoj
emp(eId=103, eName="Jagadesh") # 103 Jagadesh
emp(eId=104, eName="Vinod") # 104 Vinod
```

```
# Passing default arguments in function
# We can set default arguments in function to reuse the value/element
def books(book = "Book1"):
    print(book)

books() # Book1
books("Book2") # Book2
books("Book3") # Book3
books("Book4") # Book4
```

```
# Create a arbitrary arguments Using *args
# If we don't know the number of arguments in advance use arbitrary arguments
# Once function called it will return tuple of values
def orderFood(*foodNames):
    print(foodNames) # ('pizza', 'burger', 'sandwich', 'biryani')

orderFood("pizza", "burger", "sandwich", "biryani")
```

```
# **Kwargs It will return Dictionary of Values
def d1(**foodNames):
    for i in foodNames.items():
        print(i);
d1(oderId=101, foodOne ="pizza", foodTwo ="burger")

Output
('oderId', 101)
('foodOne', 'pizza')
('foodTwo', 'burger')
```

```
# Function using with return value
def d1():
   a,b = 5,10
   return  a,b
result = d1()
print(result) # (5, 10)

# Note: Suppose, if we are using return type in python, we need to return the function to a variable

result = d1()
print(result) # (5, 10)
or
print(d1()) # (5, 10)
```

```
# Return Data Structures
def sequences():
   return {"NameOne", "NameTwo"}, ["NameOne", "NameTwo"], (1,2)
print(sequences())
({'NameTwo', 'NameOne'}, ['NameOne', 'NameTwo'], (1, 2))

def dictt():
   return {1:"Vinod", 2:"Manoj", 3:"Hari"}
print(dictt()) # {1: 'Vinod', 2: 'Manoj', 3: 'Hari'}
```

```
# Boolean Validations using function
def name(userName):
   if userName:
      print("True")
   else:
      print("False")

name({"NameOne"})
name(("NameTwo")) # True
name(["NameThree"]) # True
name({101:"NameFour"}) # True
name([]) # False
name({}) # False
```

```
# function using index, slicing
def bikeNames(*bikes):
   print(bikes[0:3], "\n", bikes[0], "\n", bikes[3])
bikeNames("BikeOne", "BikeTwo", "BikeThree", "BikeFour")

Output
('BikeOne', 'BikeTwo', 'BikeThree')
 BikeOne
 BikeFour
```