|  |  |
| --- | --- |
| **Python Function:**   * A function is a group of related statements that performs a specific task. * Python functions that make our program more organized and manageable by dividing our code into smaller and modular chunks.   Syntax : def function name():  statements  Example:  def greet():  print("Hello")  print("How do you do?")  greet() | OUTPUT  Hello  How do you do? |
| **Function Arguments**   * Function arguments are passed inside the parenthesis during the function call.   Program:  def greet(a,b):  r=a+b;  greet(1,2)  print("Hello", name)  print("How do you do?")  greet("Brijal") | Hello Brijal  How do you do? |
| **Passing Multiple Arguments**   * If we need to pass multiple arguments to a function, we can separate them by commas.   Program  def add\_numbers(n1, n2):  result = n1 + n2  print("The sum is", result)  number1 = 5.4  number2 = 6.7  add\_numbers(number1, number2) | The sum is 12.100000000000001 |
| **Return Value from Function**  Program  def add\_numbers(n1, n2):  result = n1 + n2  print("The sum is", result)  add\_numbers(5.4, 6.7)  #result = add\_numbers(5.4, 6.7)  #print("The sum is", result)  **def sum(n1,n2):**  **result = n1 + n2**  **return result**    **result = sum(10,20)**  **print(result)** |  |
| **Types of functions**  There are two types of functions:   1. Built-in functions - Functions that are built into Python. 2. User-defined functions - Functions defined by the users themselves.   Program  marks = [55, 64, 89, 33, 44]  length = len(marks)  print(“Length is”, length)  marks\_sum = sum(marks)  print(“The total marks you got is : “) | Length is 5  The total marks you got is : 306 |
| AIM: Suppose You want to find the average marks you obtained in the exam.  Based on the average marks you want to find your grade as:   * You will get Grade A if the average marks is equal to or above 80 * You will get Grade B if the average marks is equal to or above 60 and less than 80 * You will get Grade C if the average marks is equal to or above 50 and less than 60 * And if the average marks is less than 50, you will get Grade F   Program  #find the average marks and return it  def find\_average\_marks(marks):  sum\_of\_marks = sum(marks)  number\_of\_subjects = len(marks)    average\_marks = sum\_of\_marks/number\_of\_subjects    return average\_marks  # compute grade and return it  def compute\_grade(average\_marks):  if average\_marks >= 80.0:  grade = 'A'  elif average\_marks >= 60:  grade = 'B'  elif average\_marks >= 50:  grade = 'C'  else:  grade = 'F'    return grade  marks = [55, 64, 89, 23, 75]  average\_marks = find\_average\_marks(marks)  grade = compute\_grade(average\_marks)  print("Your average marks is", average\_marks)  print("Your grade is", grade) |  |
| **Do yourself**  Can you create a program to add and multiply two numbers?  For this, create two functions add\_numbers() and multiply\_numbers(). These functions should compute the result and return them to the function call and should print from outside the function.  ANSWER🡪 |  |
| **Python Function Arguments**   1. **Positional Arguments:**  * Positional arguments are the arguments that need to be passed to the function call in a proper order.   def add\_numbers(n1, n2):  result = n1 + n2  return result  result = add\_numbers(5.4, 6.7)  print(result)   * These arguments we passed to the function are called positional arguments. * It is because the first argument 5.4 is assigned to the first parameter n1 and the second argument 6.7 is assigned to the second parameter n2. These arguments are passed based on their position.   def add\_numbers(n1, n2):  result = n1 + n2  return result  result = add\_numbers(5.4)  print(result) |  |
| 1. **Default Arguments:**  * Function arguments can have default values in Python. * We can provide a default value to an argument by using the assignment operator =.   For example,  def add\_numbers(n1 = 100, n2 = 1000):  result = n1 + n2  return result  result = add\_numbers(5.4)  print(result)   * If we run the function without any arguments:   def add\_numbers(n1 = 100, n2 = 1000):  sum = n1 + n2  return sum  result = add\_numbers()  print(result) | 1005.4  Output  1100 |
| 1. **Keyword Arguments(KWargs):**  * In Python, we can not only pass arguments to a function based on position, but also using their parameter name. * During the second function call, we have passed arguments by their parameter name. * Python allows functions to be called using keyword arguments. When we call functions in this way, the order (position) of the arguments can be changed. | OUTPUT:  Hello Jack  What's going on?  Hello Jill  Howdy? |
| Do yourself  Write a Function to swap values of a pair of integers. | def swapping(x,y):  temp = x  x = y  y = temp  print("X : ", x)  print("Y: ",y)  swapping(10,20)  # create a temporary variable and swap the values |
| WAP to find N! Using function. | def factorial(n):  if n < 0:  return 0  elif n == 0 or n == 1:  return 1  else:  fact = 1  while(n > 1):  fact \*= n  n -= 1  return fact    num = 5;  print("Factorial of",num,"is",  factorial(num)) |
| WAP to print Fibonacci series of n numbers, where n is given by the programmer. | def Fibonacci(n):  if n<= 0:  print("Incorrect input")  # First Fibonacci number is 0  elif n == 1:  return 0  # Second Fibonacci number is 1  elif n == 2:  return 1  else:  return Fibonacci(n-1)+Fibonacci(n-2)    print(Fibonacci(0)) |
| A Lambda Function in Python programming is an anonymous function or a function having no name. It is a small and restricted function having no more than one line. Just like a normal function, a Lambda function can have multiple arguments with one expression.  In Python, lambda expressions (or lambda forms) are utilized to construct anonymous functions. To do so, you will use the lambda keyword (just as you use def to define normal functions). Every anonymous function you define in Python will have 3 essential | adder = lambda x, y: x + y  print (adder (1, 2))  (lambda x: x + x)(2) |

**SUMMARY**

**About Functions**

* A Function is a block of code that performs a specific task.
* We use the def key word to define a function.
* To bring the function into action, we need to call the function. The same function can be called as many times as we want.
* We can pass values to a function. These values passed to a function are called arguments or parameters.
* The return statement can be used inside a function. It returns a value from a function & exits the function.

**About Function Arguments**

* The arguments that are passed Based on their position are known as positional arguments.
* If we give name to arguments, they are keyword arguments. The order of argument doesn’t matter for keyword arguments.
* We can also provide default values to arguments. These default values are used if we do not pass arguments during a function call.