**CodroidHub summer training**

Title :

* Break & Continue
* Predefined function & User defined functions
* Find simple interest using user defined functions & Sum of 2 numbers
* Default arguments
* Variable length
* Recursive function
* Return type function
* Package
* Class & Object
* Constructor

**Submitted by: DRISHTI GUPTA**

**ROLL NO: (2322825)**

**BRANCH: AI&ML**

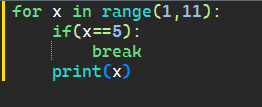
**SUBMITTED TO: Mr. DEVASHISH SIR**

**(FOUNDER, CodroidHub Private Limited)**

Break & Continue

* In Python, the break statement is used to terminate the execution of a loop prematurely. When a break statement is encountered inside a loop (either for or while), the loop stops running, and the control flow jumps to the statement immediately following the loop.

Example :



Output :

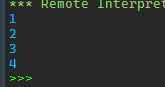
x = 1, x is not 5, so print(x) outputs 1.

x = 2, x is not 5, so print(x) outputs 2.

x = 3, x is not 5, so print(x) outputs 3.

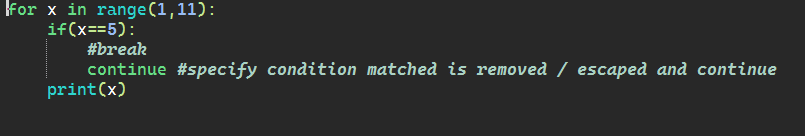
x = 4, x is not 5, so print(x) outputs 4.

x = 5, x equals 5, so the break statement is executed, terminating the loop.

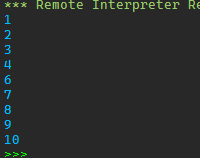


* In Python ,the Continue statement is used to skip the rest of the code inside a loop for the current iteration and move on to the next iteration of the loop. Unlike break, which terminates the loop entirely, continue only skips the current iteration.

Example :



Output :

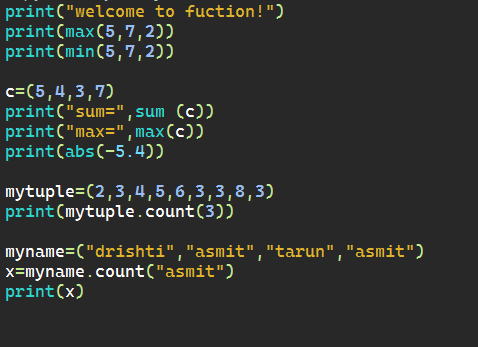


Predefined function & User defined functions

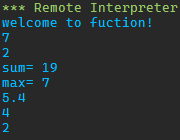
**Predefined Functions**

Predefined functions, also known as built-in functions, are functions that are already defined in Python's standard library. These functions are readily available for use without any additional coding or importing of libraries.

Example :



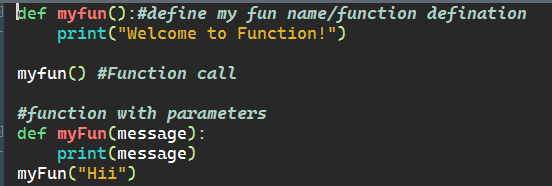
Output :



### User-Defined Functions

User-defined functions are functions that you define yourself to perform specific tasks. You create these functions using the def keyword, followed by the function name, parentheses (), and a colon :. The code block within the function is indented.

Example :

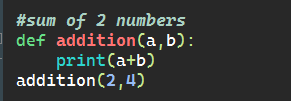


Output :



Find simple interest using user defined functions & Sum of numbers

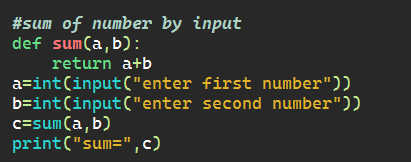
Code : sum of numbers



Output :



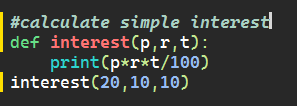
Code : sum of number by input method



Output :



Code : simple interest



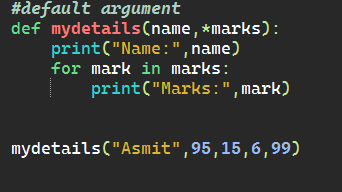
Output :



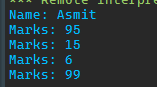
Default arguments

Default arguments in Python are values provided in a function definition that are used if no corresponding argument is passed during the function call. Default arguments are specified in the function definition by assigning a value to the parameter. When a function with default arguments is called, the default values are used if no explicit value is provided for those parameters.

Example :



Output :



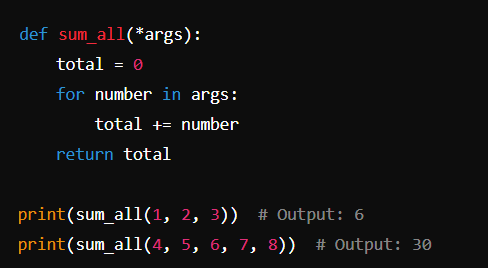
Variable length

In Python, you can define functions that accept a variable number of arguments using special syntax. This allows your functions to handle more flexible inputs. There are two main ways to achieve this: using \*args for non-keyword variable-length arguments and \*\*kwargs for keyword variable-length arguments.

### 1. \*args (Non-keyword Variable-length Arguments)

The \*args syntax allows you to pass a variable number of non-keyword arguments to a function. Inside the function, args is a tuple containing all the arguments passed to the function.

**Example:**



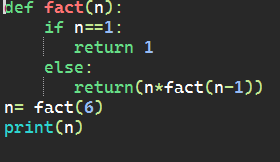
Recursive function

A recursive function in Python is a function that calls itself in order to solve a problem. Recursive functions break down a problem into smaller subproblems of the same type. Each recursive call works on a smaller version of the original problem, and the recursion typically has a base case to stop the recursion.

**Key Components of a Recursive Function**

1. **Base Case**: The condition under which the recursion stops. Without a base case, the function would call itself indefinitely, leading to a stack overflow.
2. **Recursive Case**: The part of the function where it calls itself with a smaller or simpler input.

Example: Factorial Function

****

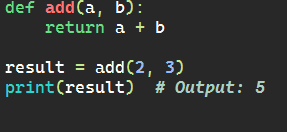
Output :



Return type function

In Python, the return type of a function is the type of value that the function returns. Functions can return various types of values, including integers, floats, strings, lists, dictionaries, tuples, or even other functions. The return statement is used to exit a function and return a value from the function to the caller.

Example :



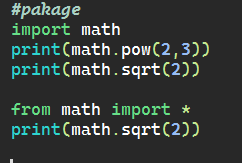
Output :



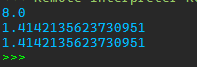
Package

In Python, a package is a way of organizing related modules into a directory hierarchy. A package is essentially a directory that contains a special file named \_\_init\_\_.py (which can be empty), along with other module files and sub-packages. Packages allow you to structure your Python code into manageable sections and promote modularity and reusability.

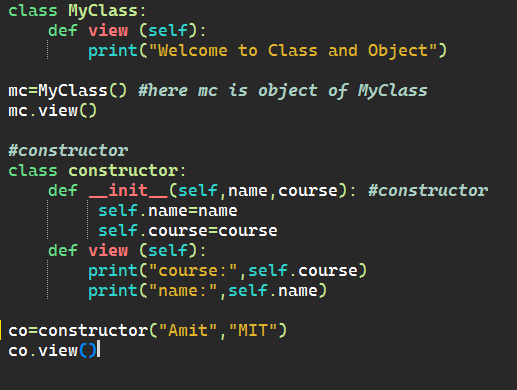
Example :



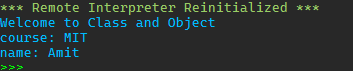
Output :



Class & Object



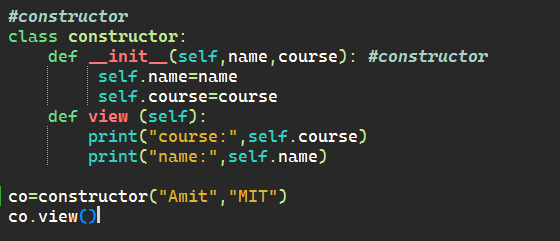
Output :



Constructor

In Python, a constructor is a special method that is automatically called when an object of a class is created. The constructor method in Python is named \_\_init\_\_. The primary purpose of the constructor is to initialize the object's attributes and perform any setup or initialization tasks required.

Example :



Output :

