
CodroidHub Summer Training

Title:-

Python

FUNCTIONS in Python
PRE-DEFINED FUNCTIONS in Python
USER-DEFINED FUNCTIONS in Python
FUNCTION WITH PARAMETERS in Python
DEFAULT ARGUMENT FUNCTION in
RECURSIVE FUNCTION in Python

FUNCTIONS IN PYTHON

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

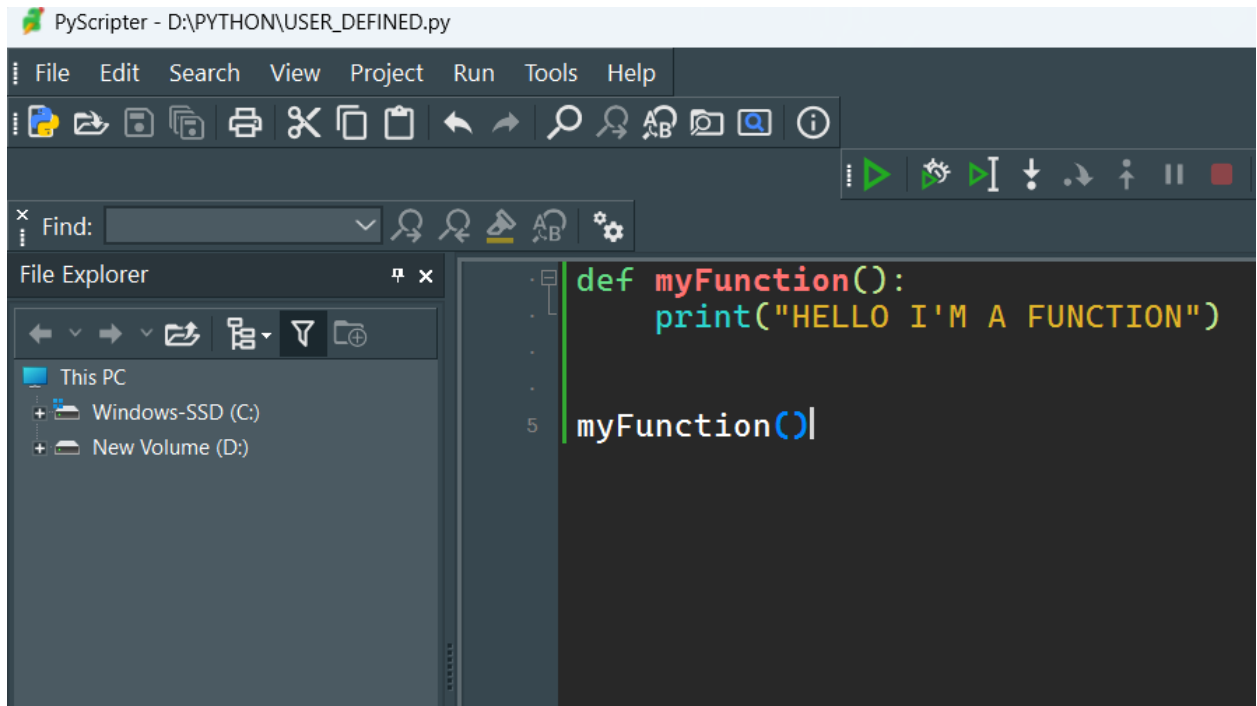
Creating a Function

In Python a function is defined using the def keyword.

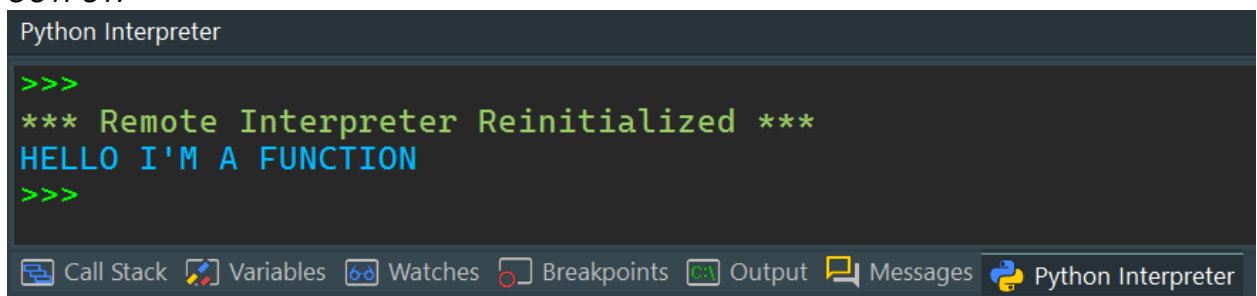
Calling a Function

To call a function, use the function name followed by parenthesis.

EXAMPLE:



OUTPUT:

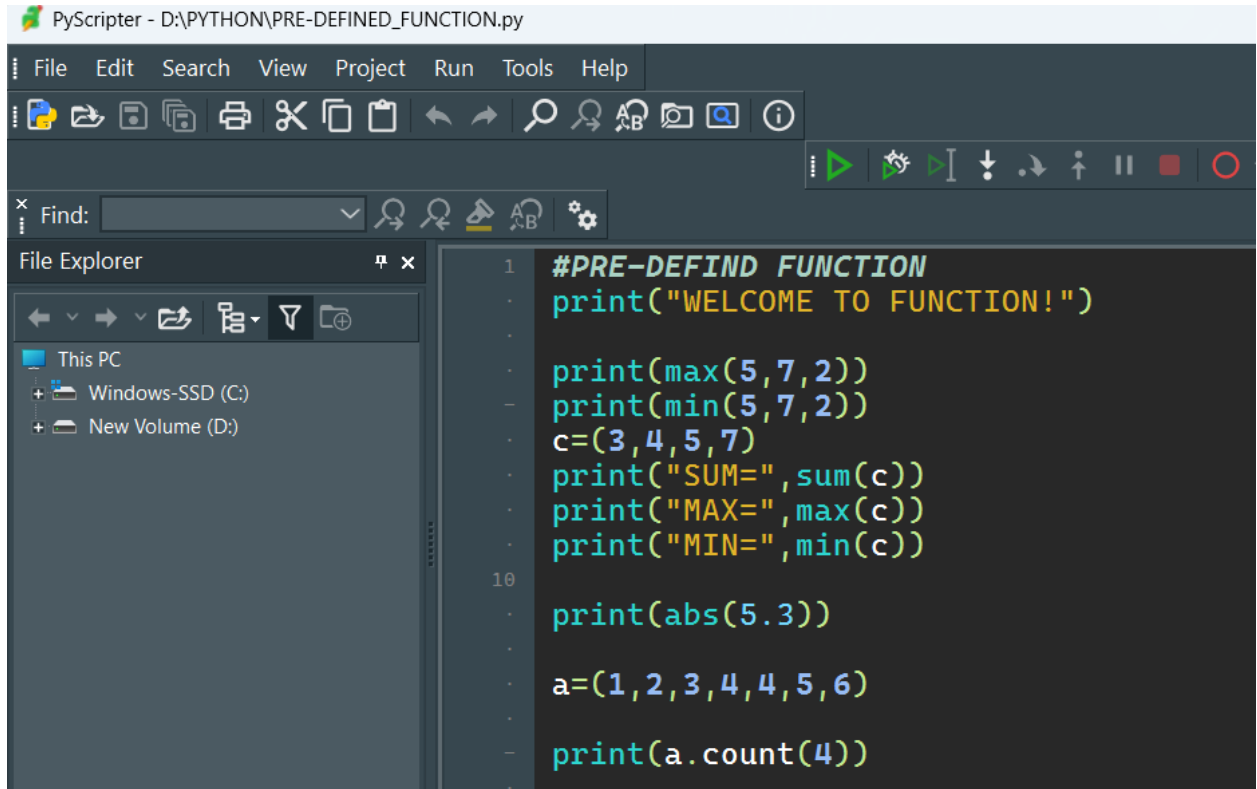


PRE-DEFINED FUNCTIONS IN PYTHON

A predefined function is a function that has already been written in the programming language and can be used by the programmer. A function will return a value that can be stored in a variable or sometimes in a conditional statement

There are many pre-defined or built in functions in python. Total there are 68 built in functions in python programming. For example: print(), list(), abs(), ascii(), all(), any(), max(), min() & many more.

EXAMPLE:

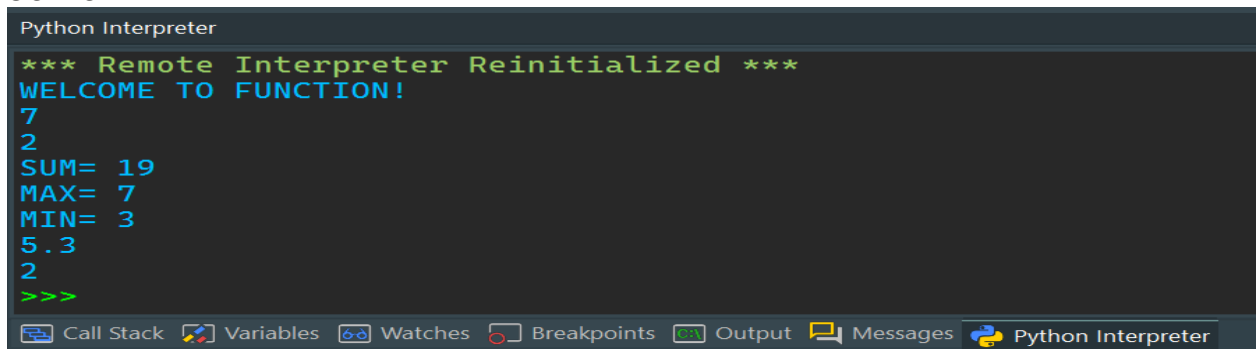


```
PyScripter - D:\PYTHON\PRE-DEFINED_FUNCTION.py

File Explorer
This PC
+ Windows-SSD (C:)
+ New Volume (D:)

1 #PRE-DEFIND FUNCTION
2 print("WELCOME TO FUNCTION!")
3
4 print(max(5,7,2))
5 print(min(5,7,2))
6 c=(3,4,5,7)
7 print("SUM=",sum(c))
8 print("MAX=",max(c))
9 print("MIN=",min(c))
10
11 print(abs(5.3))
12
13 a=(1,2,3,4,4,5,6)
14
15 print(a.count(4))
```

OUTPUT:



```
Python Interpreter

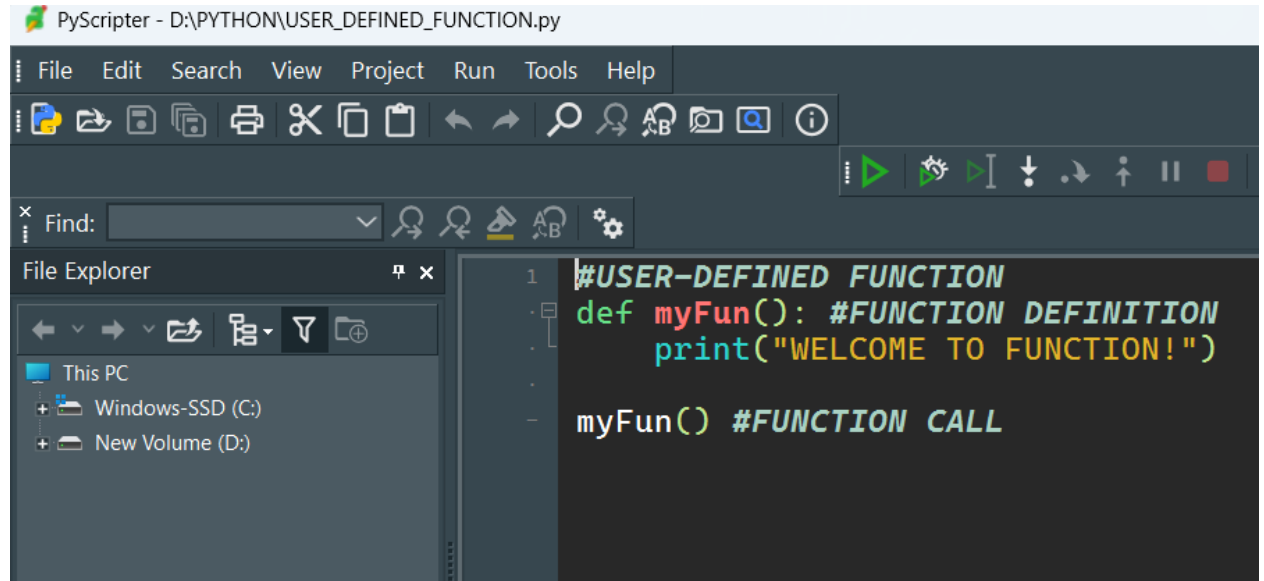
*** Remote Interpreter Reinitialized ***
WELCOME TO FUNCTION!
7
2
SUM= 19
MAX= 7
MIN= 3
5.3
2
>>>
```

USER-DEFINED FUNCTIONS IN PYTHON

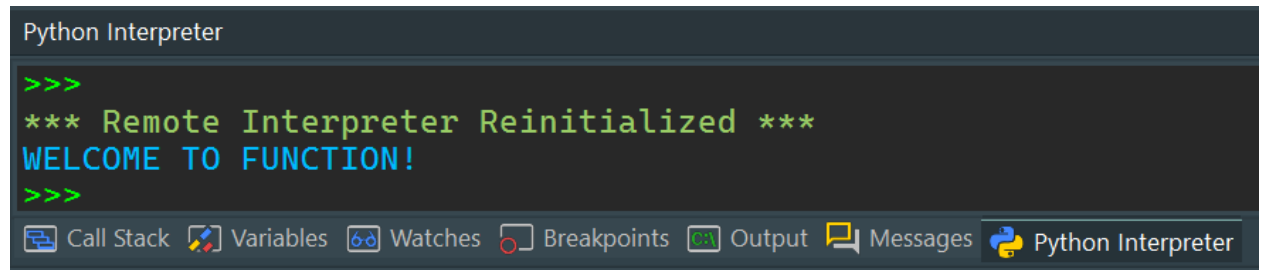
All the functions that are written by any of us come under the category of user-defined functions. Below are the steps for writing user-defined functions in [Python](#).

- In Python, a [def keyword](#) is used to declare user-defined functions.
- An indented block of statements follows the function name and arguments which contains the body of the function.

EXAMPLE:



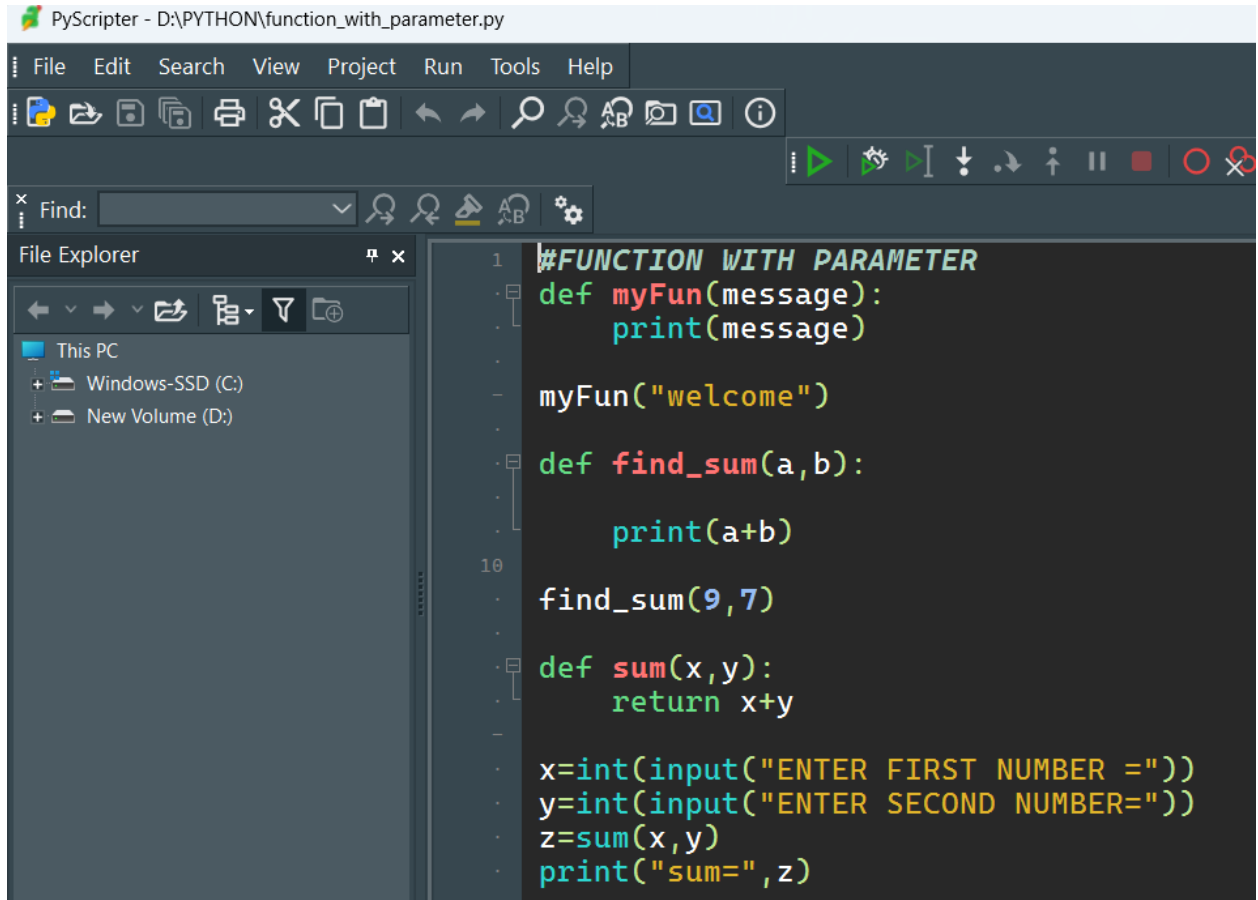
OUTPUT:



FUNCTION WITH PARAMETERS IN PYTHON

If you have experience in C/C++ or Java then you must be thinking about the return type of the function and data type of arguments. That is possible in Python as well .

EXAMPLE:



```
PyScripter - D:\PYTHON\function_with_parameter.py

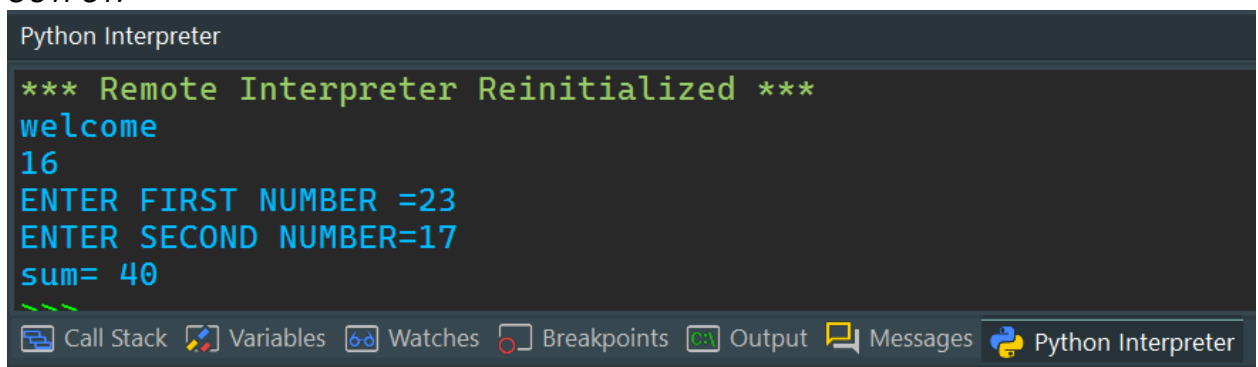
File Edit Search View Project Run Tools Help

Find:

File Explorer
This PC
Windows-SSD (C:)
New Volume (D:)

1 #FUNCTION WITH PARAMETER
2 def myFun(message):
3     print(message)
4
5 myFun("welcome")
6
7 def find_sum(a,b):
8     print(a+b)
9
10 find_sum(9,7)
11
12 def sum(x,y):
13     return x+y
14
15 x=int(input("ENTER FIRST NUMBER ="))
16 y=int(input("ENTER SECOND NUMBER="))
17 z=sum(x,y)
18 print("sum=",z)
```

OUTPUT:



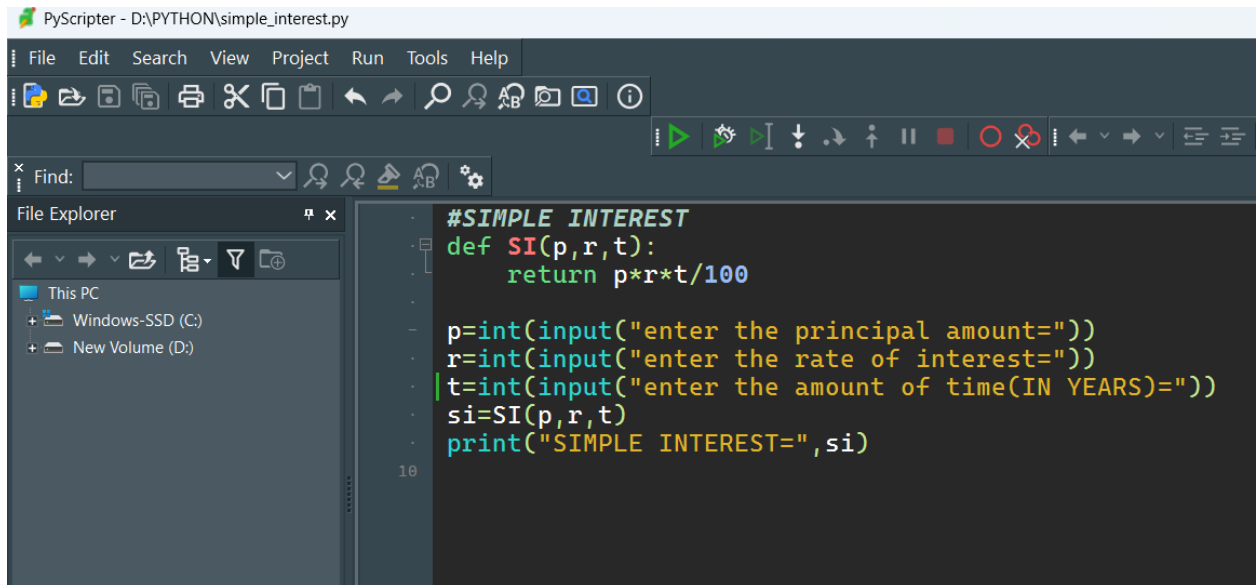
```
Python Interpreter

*** Remote Interpreter Reinitialized ***
welcome
16
ENTER FIRST NUMBER =23
ENTER SECOND NUMBER=17
sum= 40

Call Stack Variables Watches Breakpoints Output Messages Python Interpreter
```

Let us make a program to calculate simple interest using user defined function.

EXAMPLE:

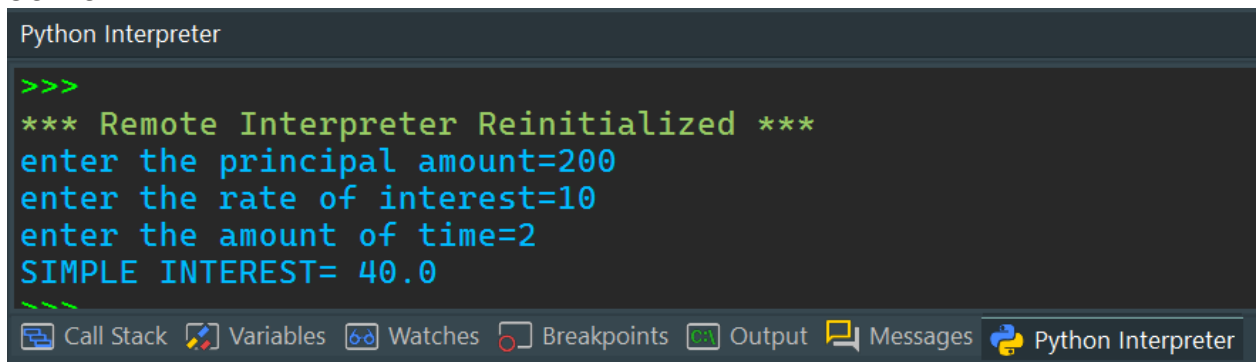


The screenshot shows the PyScripter IDE with a file named 'simple_interest.py'. The code defines a function 'SI' that calculates simple interest based on principal (p), rate (r), and time (t). It prompts the user for these values and prints the result.

```
#SIMPLE INTEREST
def SI(p,r,t):
    return p*r*t/100

p=int(input("enter the principal amount="))
r=int(input("enter the rate of interest="))
t=int(input("enter the amount of time(IN YEARS)="))
si=SI(p,r,t)
print("SIMPLE INTEREST=",si)
```

OUTPUT:



The screenshot shows the Python Interpreter window. It displays the execution of the script, including the function definition and the user input. The output shows the calculated simple interest as 40.0.

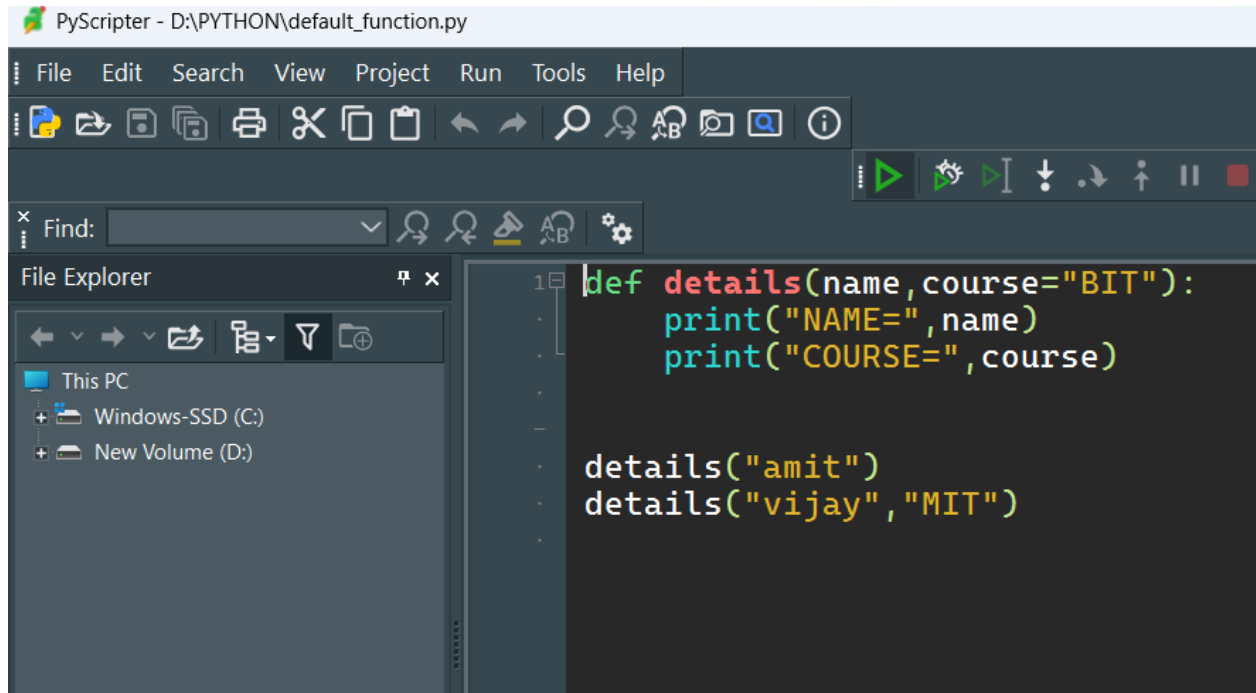
```
>>>
*** Remote Interpreter Reinitialized ***
enter the principal amount=200
enter the rate of interest=10
enter the amount of time=2
SIMPLE INTEREST= 40.0
>>>
```

DEFAULT ARGUMENT FUNCTION IN PYTHON

Python allows function arguments to have default values. If the function is called without the argument, the argument gets its default value.

Python has a different way of representing syntax and default values for function arguments. Default values indicate that the function argument will take that value if no argument value is passed during the function call. The default value is assigned by using the assignment(=) operator of the form keyword name=value.

EXAMPLE:

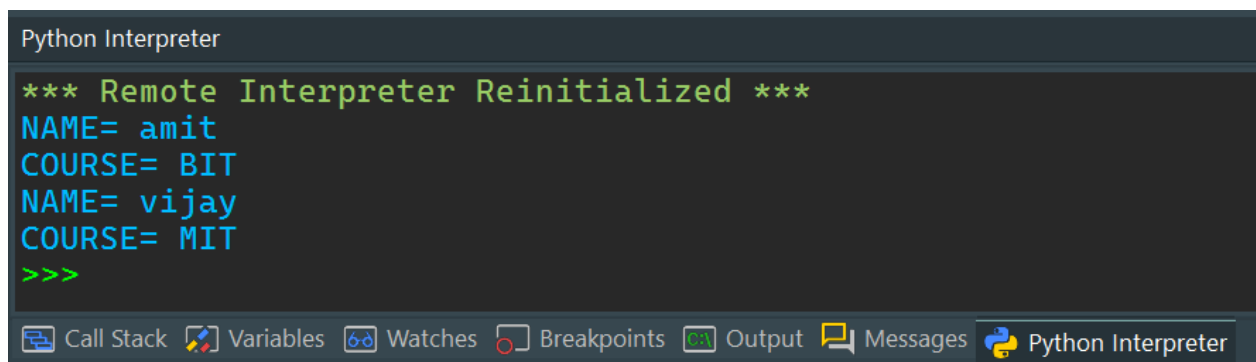


The screenshot shows the PyScripter IDE with a file named 'default_function.py'. The code defines a function 'details' with two parameters: 'name' and 'course' (with a default value of 'BIT'). The function prints the name and course. Below the function definition, the function is called twice: 'details("amit")' and 'details("vijay", "MIT")'.

```
def details(name, course="BIT"):
    print("NAME=", name)
    print("COURSE=", course)

details("amit")
details("vijay", "MIT")
```

OUTPUT:



The screenshot shows the Python Interpreter output window. It displays the output of the function calls: 'NAME= amit' and 'COURSE= BIT' for the first call, and 'NAME= vijay' and 'COURSE= MIT' for the second call. The output is preceded by a green separator line and the text '*** Remote Interpreter Reinitialized ***'.

```
*** Remote Interpreter Reinitialized ***
NAME= amit
COURSE= BIT
NAME= vijay
COURSE= MIT
>>>
```

RECURSIVE FUNCTION IN PYTHON

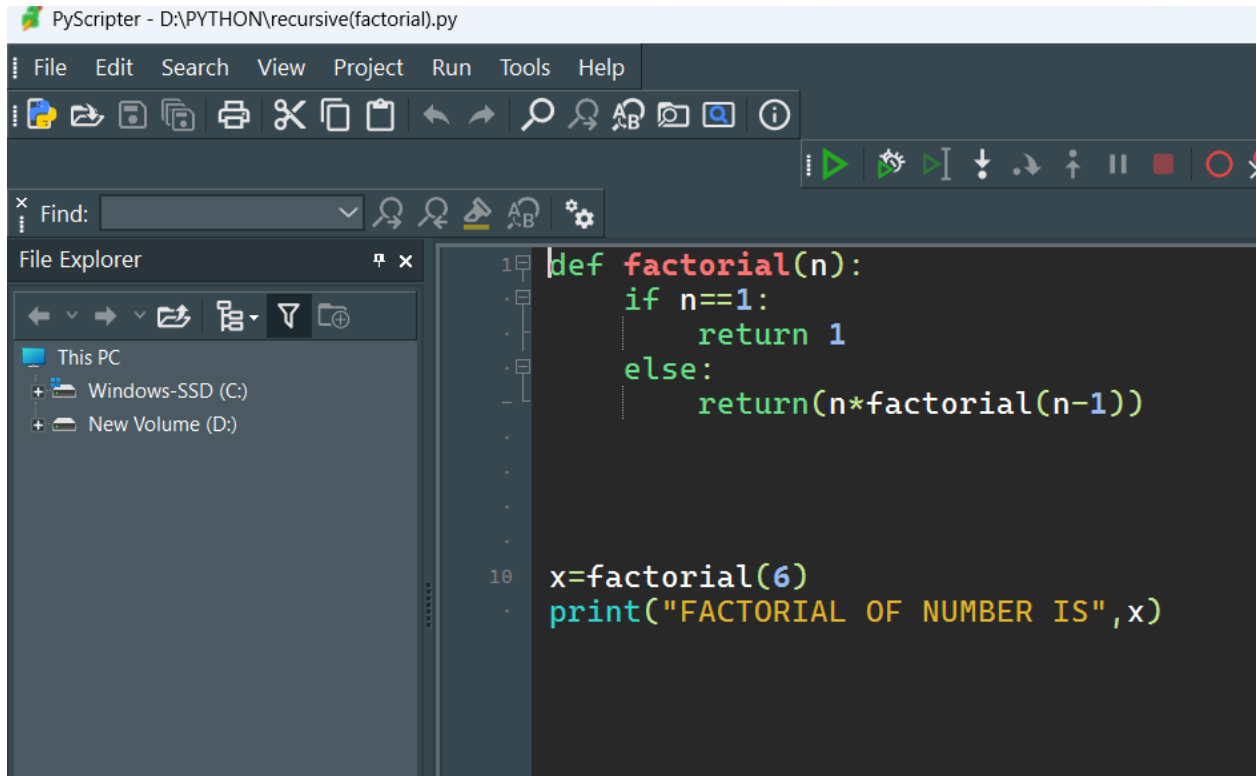
The term Recursion can be defined as the process of defining something in terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

Advantages of using recursion

- A complicated function can be split down into smaller sub-problems utilizing recursion.
- Sequence creation is simpler through recursion than utilizing any nested iteration.
- Recursive functions render the code look simple and effective.

Let us solve a problem of finding a factorial of a number using recursion.

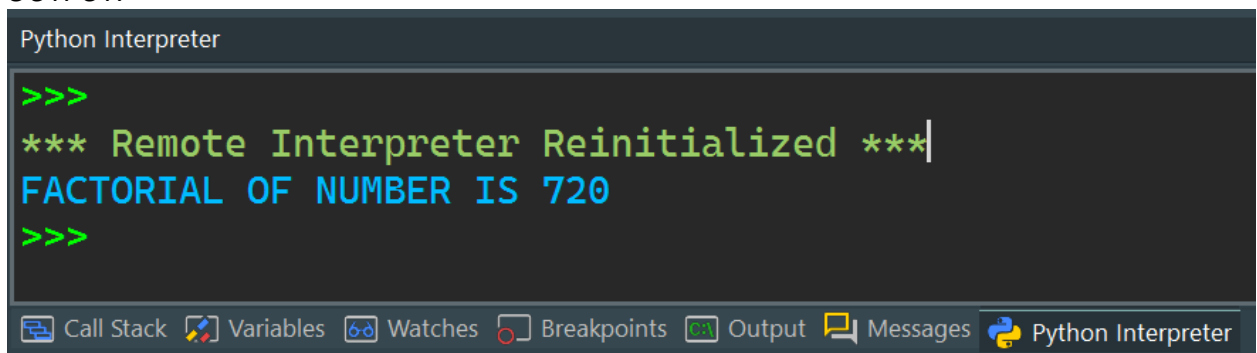
EXAMPLE:



The screenshot shows the PyScripter IDE with a file named 'recursive(factorial).py'. The code defines a recursive function 'factorial(n)' that returns 1 for n==1 and n*factorial(n-1) otherwise. It then calls 'x=factorial(6)' and prints the result. The File Explorer on the left shows the project structure.

```
def factorial(n):  
    if n==1:  
        return 1  
    else:  
        return(n*factorial(n-1))  
  
x=factorial(6)  
print("FACTORIAL OF NUMBER IS",x)
```

OUTPUT:



The screenshot shows the Python Interpreter window with the output of the program. It displays a message about the remote interpreter being reinitialized, followed by the printed result 'FACTORIAL OF NUMBER IS 720'.

```
>>>  
*** Remote Interpreter Reinitialized ***  
FACTORIAL OF NUMBER IS 720  
>>>
```