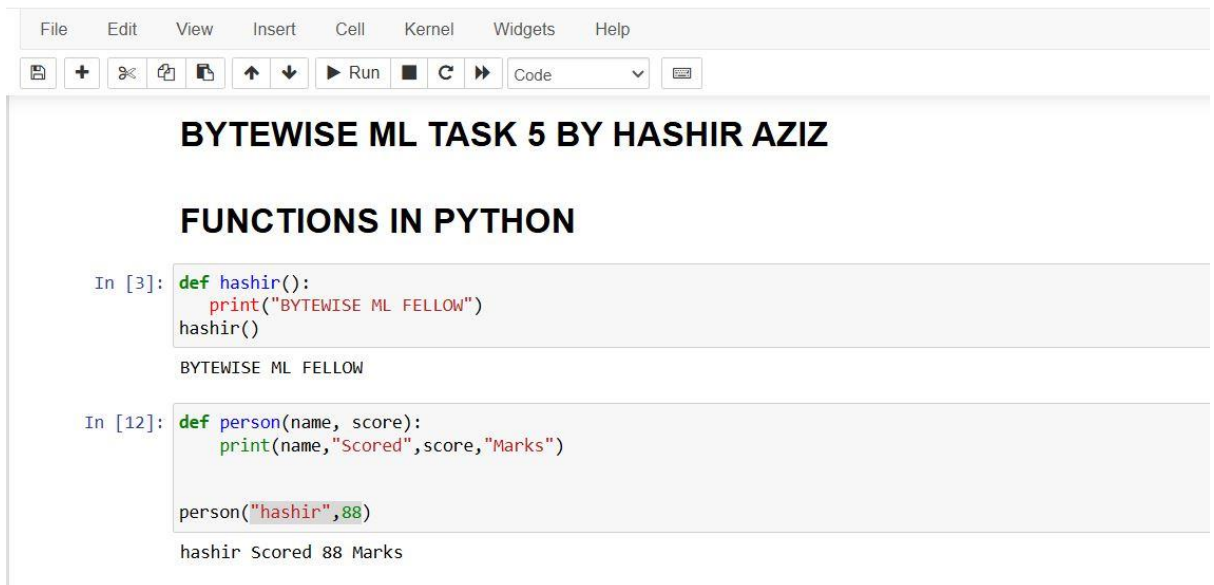


# HASHIR AZIZ MACHINE LEARNING (TASK 5)

## FUNCTION IN PYTHON

- **How to define a Function:** We use the `def` keyword, followed by the function name, parentheses `()`, and a **colon** `:` and Inside the parentheses, we can specify parameters that the function can accept.

 jupyter BYTEWISE ML TASK 5 BY HASHIR AZIZ Last Checkpoint: 4 hours ago (unsaved changes)



The Jupyter Notebook interface displays the following code and output:

```
File Edit View Insert Cell Kernel Widgets Help
[Icons] [Run] [Code]

BYTEWISE ML TASK 5 BY HASHIR AZIZ

FUNCTIONS IN PYTHON

In [3]: def hashir():
        print("BYTEWISE ML FELLOW")
        hashir()

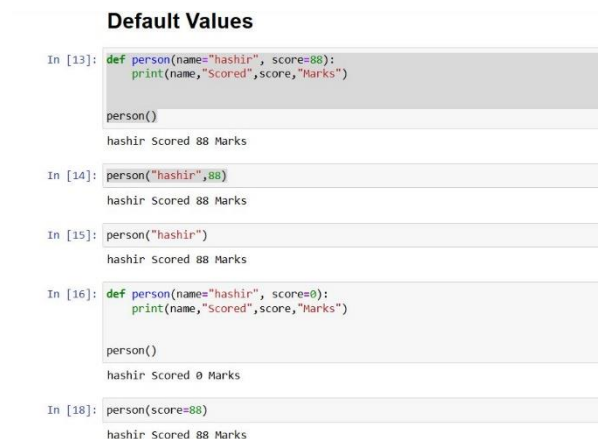
        BYTEWISE ML FELLOW

In [12]: def person(name, score):
          print(name, "Scored", score, "Marks")

          person("hashir", 88)

          hashir Scored 88 Marks
```

- Default parameter passing:



The Jupyter Notebook interface displays the following code and output under the heading "Default Values":

```
Default Values

In [13]: def person(name="hashir", score=88):
          print(name, "Scored", score, "Marks")

          person()

          hashir Scored 88 Marks

In [14]: person("hashir", 88)

          hashir Scored 88 Marks

In [15]: person("hashir")

          hashir Scored 88 Marks

In [16]: def person(name="hashir", score=0):
          print(name, "Scored", score, "Marks")

          person()

          hashir Scored 0 Marks

In [18]: person(score=88)

          hashir Scored 88 Marks
```

In above code the method of overriding also lies as it overrides the marks values.

**PASSING MULTIPLE PARAMETERS:** We can pass multiple parameters in our function like in below code I have passed my exam score of 5<sup>th</sup> semester of Artificial Intelligence, and I have used the (\*) so that I can pass multiple values in the score and the result comes in the form of **Tuple**.

## MULTIPLE PARAMETERS

```
In [22]: def person(name,*score):  
         print(name,"scored",score)  
         #print(score)  
  
         person("Hashir",88,75,54,66,73,70)
```

Hashir scored (88, 75, 54, 66, 73, 70)

### TUPLE:

In tuple we cannot modify the values as tuple is just like a list but we can only read the tuple.

### Normal code in python with error:

#### Normal code in python with error

```
In [25]: num1=int(input("Enter the first number:\n"))  
         num2=int(input("Enter the second number:\n"))  
         print(num1,num2)
```

Enter the first number:  
5  
Enter the second number:  
hashir

```
-----  
ValueError                                Traceback (most recent call last)  
Cell In[25], line 2  
      1 num1=int(input("Enter the first number:\n"))  
----> 2 num2=int(input("Enter the second number:\n"))  
      3 print(num1,num2)
```

**ValueError:** invalid literal for int() with base 10: 'hashir'

In this code I take integer input from the user first and then print the user input, user entered the first integer input, and the second input is in string format, so error occurred here. Now I will deal this with Try and exception handling.

## Try and exception handling in python

```
In [28]: def Hashir_score(sub1, sub2):
        try:
            result = sub1/sub2
        except ZeroDivisionError:
            print("Error: Cannot divide by zero.")
            return None
        else:
            print("Division successful.")
            return result
        finally:
            print("Execution complete.")

result = Hashir_score(10, 2)
print(f"Result: {result}")

result = Hashir_score(10, 0)
print(f"Result: {result}")

Division successful.
Execution complete.
Result: 5.0
Error: Cannot divide by zero.
Execution complete.
Result: None
```

### In this above code:

The try block attempts to divide sub1 by sub2. If sub2 is zero, a Zero\_Division\_Error occurs, and the except block handles it. If no error occurs, the else block runs, indicating success.

The finally block runs in both cases, ensuring the "Execution complete." message is printed.

**Local Variable:** It is defined within the function. Here is the example of location var:

## LOCAL VARIABLE

```
In [33]: def HASHIR():
        y = 5
        print(y)

HASHIR()

5
```

```
In [34]: print(y) #CALLING GLOBAL VARIABLE OUTSIDE THE FUNCTION AND IT CAUSES ERROR

-----
NameError                                Traceback (most recent call last)
Cell In[34], line 1
----> 1 print(y)

NameError: name 'y' is not defined
```

**Global Variable:** Global variables are defined outside of any function and can be accessed anywhere in the code. Here is the below code:

## GLOBAL VARIABLE

```
In [36]: x = 10
def HASHIR():
    print(x)
HASHIR()
```

10

```
In [37]: x
```

Out[37]: 10

```
In [38]: print(x) #CALLING GLOBAL VARIABLE OUTSIDE THE FUNCTION AND IT CAUSES NO ERROR
```

10

---

**GLOBAL KEYWORD:** This tells Python that you are referring to the global variable, not creating a new local variable.

## CALLING GLOBAL FUNCTION INSIDE A FUNCTION

```
In [41]: x = 10
def HASHIR():
    global x
    x = 5
    print(x)
HASHIR()
```

5

```
In [40]: print(x)
```

5

---

## LAMDA FUNCTION:

A lambda function in Python is a small, anonymous function defined with the lambda keyword. Lambda functions can have any number of arguments but only one expression. They are often used for short, simple operations that are needed temporarily, such as when sorting or filtering data. They are useful for short operations and are often used in functions like filter(), map(), and sorted().

## LAMDA FUNCTION IN PYTHON

```
In [29]: MARKS = lambda x, y: x + y  
print(MARKS(2, 3))
```

5

## WITH ONE ARG

```
In [30]: square = lambda x: x * x  
print(square(4))
```

16

In this code first we define the function with lamda keyword, and then comes the expression. Here I have tried with one expression (x).