

# **21th April 2021**

Previous Day:-

- Nested loop
- Continue
- Break
- Pass
- MCQs

## **Python Function**

Lecture Flow:-

- Functions
  - programs

## **Topics & Explanation**

### **1. Functions**

[click here for reference](#)

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

#### **Program 1:-**

The below program is divided into 3 parts but actually it is a single program.

```
50 def hotel_menu():  
51     age = input("Enter your age: ")  
52     age = int(age)  
53     print("Hotel Menu: ")  
54     print("Idli")  
55     print("Vada")  
56     print("Dosa Sambar")  
57     print("Biryani")  
58     print("Kulche")
```

Here we are defining a function with `def function_name`. So we have a function called `hotel_menu()` and inside function we are taking input age and printing hotels menu.

```
59 if age > 18:  
60     print("Bar Menu:")  
61     print("Corona")  
62     print("Smirnoff")  
63     print("Beluga")  
64     print("Bombay Sapphire")
```

This is the main program where we are putting the condition that if the age(input) is greater than 18 then print the bar menu and below line 68 will also execute which call the function `hotel_menu()`. If the age(input) is less than 18 than it will only call the function.

```
68 hotel_menu()
```

### Output 1:-

```
Enter your age: 21
Hotel Menu:
Idli
Vada
Dosa Sambar
Biryani
Kulche
Bar Menu:
Corona
Smirnoff
Beluga
Bombay Sapphire
~/RichBiodegradableLicenses$
```

**Note:-** The function should be defined first then the function should be called. If you call the function and then define it. it won't work.

### Program 2:-

```
77 def tables_good(n):
78     for i in range(1, 11):
79         print(n, " X ", i, " = ", n*i)
80
81 n = input("Enter a number: ")
82 n = int(n)
83 tables_good(n)
84
85 m = 100
86 tables_good(m)
87 tables_good(12)
88
```

- The main program is from line 81 to line 87.
- The function is from line 77 to line 79.
- The function is to print the table of **n(input)** number.
- In the main program we took input and typecasted it.
- We call the function 3 times in line 83(with input as n), in line 86(with input as m), and line 87(with input as 12).

- When the function is called it goes to line 77, in line 77 the input is passed into the argument `n` and that `n` is taken as input to print the table of given input.

### Output 2:-

```
Enter a number: 33
33 X 1 = 33
33 X 2 = 66
33 X 3 = 99
33 X 4 = 132
33 X 5 = 165
33 X 6 = 198
33 X 7 = 231
33 X 8 = 264
33 X 9 = 297
33 X 10 = 330
100 X 1 = 100
100 X 2 = 200
100 X 3 = 300
100 X 4 = 400
100 X 5 = 500
100 X 6 = 600
100 X 7 = 700
100 X 8 = 800
100 X 9 = 900
100 X 10 = 1000
12 X 1 = 12
12 X 2 = 24
12 X 3 = 36
12 X 4 = 48
12 X 5 = 60
12 X 6 = 72
12 X 7 = 84
12 X 8 = 96
12 X 9 = 108
12 X 10 = 120
~/RichBiodegradableLicenses$
```

### Program:-

```
77 def tables_good(n=1):
78     for i in range(1, 11):
79         print(n, " X ", i, " = ", n*i)
91
92 tables_good()
```

- In this above program, in the main program line 92 the function is called with no value given as input.
- When the function is called it goes to the line 77, but in this case we have marked the default value as `n=1`. This will only execute when the input is not given .

- The function will then print the table of **one**.

Output:-

```
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10
```

Program:-

We can use the function to return the value and it can be stored into any variable for any further use.

```
140 def factorial_even_better(n):
141     fact = 1
142     for num in range(1, n+1):
143         fact = fact * num
144     return fact
145
146 fact_26 = factorial_even_better(26)
```

- The function starts from line 140 to line 144.
- In line 144 we are returning the value of fact
- In the main function line 146 the function returning the value is stored in the main **fact\_26**.

### Program:-

```
146 def alternate_universe_factorial(n, m):
147     fact_n = 1
148     fact_m = 1
149     for num in range(1, n+1):
150         fact_n = fact_n*num
151     for num in range(1, m+1):
152         fact_m = fact_m*num
153     print(fact_n+fact_m)
154
155 alternate_universe_factorial(26, 35)
```

In the above program we have given two inputs in the argument while calling the function.

```
166 print("35!-26!+5! = ", fact_35-fact_26+factorial_even_better(5))
```

In this above line as an example, we can even call the function and can directly add it also we can store the value of the function and store in the variable and add it altogether.

**Note:-** If the function does not return anything, then the variable will store the value **None**. **None** is the default value of the non returning function.

## MCQs

What is the output of:

```
def factorial_even_better(n=5):  
    fact = 1  
    for num in range(1, n+1):  
        fact = fact * num  
    return fact  
  
factorial_even_better()
```

Attempted - 37 (68.52%)

EASY



- |   |        |
|---|--------|
| <input checked="" type="checkbox"/> 120 | 86.49% |
| <input type="checkbox"/> 1              | 8.11%  |
| <input type="checkbox"/> error          | 5.41%  |
| <input type="checkbox"/> 0              |        |

What is the benefit(s) of functions?

Attempted - 39 (72.22%)

EASY



- |  |        |
|--|--------|
| <input checked="" type="checkbox"/> Reduce code length             | 84.62% |
| <input checked="" type="checkbox"/> Reduces code repetition        | 84.62% |
| <input type="checkbox"/> Reduces code speed                        | 7.69%  |
| <input checked="" type="checkbox"/> Increases code customizability | 79.49% |

What is the output of:

```
def factorial_even_better(n=5):  
    fact = 1  
    for num in range(1, n+1):  
        fact = fact * num  
    return fact
```

Attempted - 39 (72.22%)

EASY



☒ no output

74.36%

☐ error

2.56%

☐ None

10.26%

☐ 120

12.82%

Predict the output:

```
def alternate_universe_factorial(n, m):  
    fact_n = 1  
    fact_m = 1  
    for num in range(1, n+1):  
        fact_n = fact_n*num  
    for num in range(1, m+1):  
        fact_m = fact_m*num  
  
alt = alternate_universe_factorial(26, 35)  
print(alt)
```

Attempted - 38 (70.37%)

EASY



☒ None

78.95%

☐ 26!

5.26%

☐ 26!+35!

10.53%

☐ error

10.53%