

DAY-15

SEPTEMBER-24,2025

Functions:

Function is a block of reusable code that performs a specific task.

Functions makes our program more organized , readable and reduce repetition.

Types of functions:

1. Built-in functions: Already available in python

ex: print(),type(),input(),len()

2. User-defined Functions: Functions created by the user using def keyword.

Syntax:

```
def fun_name(parameters):
```

```
    statements
```

```
    return value
```

User defined functions can be created in four ways:

1. Function without input and without return

2. Function with input and without return

3. function without input and without return

4. function with input and with return

3.Lambda function: Anonymous(nameless) functions written in a single line using the lambda keyword.

User-defined functions:

1.Function without input and without return

Syntax:

```
def fun_name():
```

statements

Example:

```
def add1():
```

```
    x = int(input("enter x value: "))
```

```
    y = int(input("enter y value: "))
```

```
    s = x + y
```

```
    print(f"the sum of {x} and {y} is {s}")
```

calling the function:

```
add1()
```

output:

```
enter x value: 5
```

```
enter y value: 6
```

```
the sum of 5 and 6 is 11
```

2. Function with input and without return

Syntax:

```
def fun_name(p1,p2...pn):
```

```
    statements
```

Example:

```
def add2(x,y):
```

```
    s = x + y
```

```
    print(f"the sum of {x} and {y} is {s}")
```

call the function:

```
add2(5,6)
```

Output:

the sum of 5 and 6 is 11

Types of function calling:

- We can directly call the function and pass the parameters like above example
- We have to declare the variables we want to use in function and then call the function to use those variables in the function.

```
a = int(input("Enter a number: "))
```

```
b = int(input("Enter a number: "))
```

output:

```
Enter a number: 5
```

```
Enter a number: 6
```

```
print(a,b)
```

output:

```
5,6
```

Calling:

```
add2(a,b)
```

output:

```
the sum of 5 and 6 is 11
```

- We can use both defined and the other number parallely.

Ex:

```
add2(12,b)
```

output:

```
the sum of 12 and 6 is 18
```

3. Function without input and with return :

Syntax:

```
def fun_name():
```

```
    statements
```

```
    return value
```

Example:

```
def add3():
```

```
    x = int(input("enter x value: "))
```

```
y = int(input("enter y value: "))
```

```
s = x + y
```

```
return s
```

call:

```
add3()
```

output:

```
enter x value: 5
```

```
enter y value: 6
```

```
11
```

- To return multiple variables

```
def add3():
```

```
    x = int(input("enter x value: "))
```

```
    y = int(input("enter y value: "))
```

```
    s = x + y
```

```
    return x,y,s
```

call:

```
ts=add3()
```

output:

```
enter x value: 5
```

```
enter y value: 6
```

```
print(f"the sum of {ts[0]} and {ts[1]} is {ts[2]}")
```

output:

```
the sum of 5 and 6 is 11
```

```
call: a,b,c = add3()
```

output:

enter x value: 5

enter y value: 6

```
print(f"the sum of {a} and {b} is {c}")
```

output:

the sum of 5 and 6 is 11

4. Function with input and with return

Syntax:

```
def fun_name(p1,p2...pn):
```

```
    statements
```

```
    return value
```

Example:

```
def add4(a,b,c):
```

```
    s = a + b + c
```

```
    return s,a,b,c
```

```
call; s,a,b,c = add4(5,6,2)
```

```
print(f"The sum of {a},{b} and {c} is {s}")
```

output:

The sum of 5,6 and 2 is 13

1. Create a function to check the given number is prime or not using with input and without return method.

Program code:

```
def prime(n):
```

```
    for i in range(2,n,1):
```

```
        if(n%i==0):
```

```
        print(n,"is not a prime number")
        break
    else:
        print(n,"is a prime number")
```

function call:

```
prime(5)
```

output: 5 is a prime number

```
prime(6)
```

output: 6 is not a prime number