Today's Topics

- what are functions
- function arguments
- local and global variables
- return statements
- global keyword

What are **Functions** in Python?

In **Python**, a **function** is a block of reusable code that performs a specific task. Functions help you break your program into smaller, manageable parts and avoid repeating the same code multiple times.

Why Use Functions?

- **Code Reusability**: Write once, use many times.
- **Modularity**: Break down large problems into smaller parts.
- **Readability**: Makes your code cleaner and easier to understand.
- **Maintainability**: Easier to fix bugs or make changes.

Types of Functions

1. **Built-in Functions** – e.g., `print()`, `len()`, `sum()`
2. **User-defined Functions** – Functions you create using `def`

Points to Remember

- Use `def` keyword to define a function
- You can pass **arguments** (inputs)
- You can use `return` to send back a result
- Functions can have **default parameters**
- Functions can return **multiple values** (in tuples)

In [16]:
```
l=10
b=20

print('area of deewar is',l*b)
result = l*b
```

```
area of deewar is 200
```

In [17]:
```
result
```

Out[17]: 200

In [18]:
```
#l=length,b=breath
# functions
# area functions

# def functions_name(parameters):
# return statements

def area(l,b): #built
    print('area of wall is',l*b)
```

In [19]:
```
area(10,20)
area(100,20)
area(100000,20000)
area(1065,9088)
```

```
area of wall is 200
area of wall is 2000
area of wall is 2000000000
area of wall is 9678720
```

In [3]:
```
# built in functions
print('hello world')
# id('a')
# len('vipul')
sorted([13,15,4598,456,24])
# int()
# str()
# list()
#sets()
# float()
# dict()
#bool()
```

```
hello world
```

Out[3]: [13, 15, 24, 456, 4598]

In [4]:
```
# abs
age=-18
abs(age)
```

Out[4]: 18

In [5]:
```
# min,max
# mean([13,2341,5,584,545,145])
```

In [6]:
```
# function arguments
#sum of 2 numbers
def sum(x,y):
```

```
    print('sum of x,y is',x+y)
```

In [7]:
```
sum(103,234)
```

```
sum of x,y is 337
```

In [8]:
```
# vipul(120,13)
```

In [9]:
```
# functions with 3- or more arguments
def mean_of_4(a,b,c,d):
    print((a+b+c+d)/4)
```

In [10]:
```
mean_of_4(12,23,45,63)
```

```
35.75
```

In [11]:
```
# 1 arguement
def cube(x):
    print(x**3)
cube(3)
```

```
27
```

In [12]:
```
# without any arguments
def greet():
    print('hello world')
```

In [13]:
```
# parameter--> function build
# agruments--> call function
```

In [14]:
```
greet()
```

```
hello world
```

In [15]:
```
# print staements vs return statements
```

In [16]:
```
# print statements--> degbugging
def sum(x,y):
    print(x+y)
    print('vipul')
    print('ram')
```

In [17]:
```
sum(10,20)
```

```
30
vipul
ram
```

In [18]:
```
def sum(x,y):
    print(x+y)
    return x+y
    print('vipul')
    print('ram')
```

In [19]:
```
sum(10,20)
```

```
30
```

Out[19]: 30

In [20]:
```
def mean(x,y):
    print((x+y)/2)
```

In [21]:
```
result=mean(10,20)
```

```
15.0
```

In [22]:
```
result
```

In [23]:
```
# return statement
def mean(x,y):
    return (x+y)/2
```

In [24]:
```
result=mean(102,24)
```

In [25]:
```
result
```

Out[25]: 63.0

In [26]:
```
x = 10 # global
def funct():
    x=20 # local
    print(x)
    return x
```

In [27]:
```
x
```

Out[27]: 10

In [28]:
```
funct()
```

```
20
```

Out[28]: 20

In [29]:
```
x
```

Out[29]: 10

In [30]:
```
x = 100 #global
def func1():
    y = 200 # local
    return x
```

In [31]:
```
func1()
```

Out[31]: 100

In [32]:
```
# whf
# local--> variable global
# global keyword
x = 10

def func(): # think
    global x
    x = 200
```

In [33]:
```
x
```

Out[33]:
```
10
```

In [34]:
```
# call --> done
```

In [35]:
```
func()
```

In [36]:
```
x
```

Out[36]:
```
200
```

In [37]:
```
# Built in function
```

In [38]:
```
input('enter your name')
```

Out[38]:
```
'vipul'
```

In [ ]: