Functions :-

- A function is a block of reusable code that performs specific tasks.
- Function always represented by ( )
- Function make our program more organised, short, readable and reduce repetition.

Types of functions :-

1. Built in function
   already available in python.
   eg :- input( ), print( ), len( ) .... etc

2. user defined functions
   functions created by the user by using def keyword.

Syntax :-

```
def fun_name (parameters):
    Statement
    return Value.
```

4 types :-

1) function without input and without return.
2) function with input and without return.
3) function without input and with return.
4) function with input and with return.

3). lamda function :-

The anonymous (nameless) nameless function written in single line using the lamda keyword.

24/09/2025 Assignment :-

①

1) function as parameter.
   * function can be passed as an argument to another function.
   * This is useful when we want to reuse logic.

ex:

```
def square(x):
    return x * x
def apply_function(func, value):
    return func(value)
# passing function as a parameter
result = apply_function(square, 5)
print(result)  # 25
```

Explanation :-

square ( ) is a function.
we pass square into apply_function.
inside it calls square (5) → 25.

⑤ Recursion function :-

It is a technique where a function calls itself in order to solve a problem until met a condition.

Key component :-

Base case : This is the condition that stops Recursion.
when base case is met the function returns a value.
without making further recursive calls.
- preventing infinite Recursion.

Recursive case :-

This is where function calls itself with modified input, usually moves closer to base case.

example:-

```
def factorial(n):
    if (n==0 or n==1): # base case
    return 1
    else:
        return n * factorial(n-1) # Recursive call
print (factorial(5))  # 120.
```

factorial (5) → 5 * factorial (4) = 20.
factorial (4) → 4 * factorial (3) =
factorial (3) → 3 * factorial (2)
factorial (2) → 2 * factorial (1) =
factorial (1) → returns 1

Nested functions:-

- A function written inside of another function.
  - The inner function works only inside of the outer one.

ex:-

```
def -outer-function (name):
    def -inner function():
        return f"Hello, {name}!"
    return inner function()
print (outer-function(Soundharya)
```

o/p:- Hello, soundharya.

1) outer function Soundharya is called.
2) inside inner function is defined.
3) inner function uses "Soundharya"
4). Return = "Hello, Soundharya!"

I. function without input and without return.

```
def fun_name():
    statements.
```

① program:- Calculate the simple interest:-

```
def si1():
    p = float (input ("enter p value:"))
    t = float (input ("enter t value:"))
    r = float (input ("enter r value:"))
    si = (p*t*r)/100
    print (f" simple interest is {si} for principle amount pa=
          {p}, time ={t}, rate of interest = {r}")
```

call function:- si1()
o/p:- enter p value = 30,000
enter t value = 2
enter r value = 2.5
the simple interest is 1500.0 for pa = 30000.0 time = 2.0, roi = 2.5

II. Function with input and without return.

syntax:- def fun_name (p1, p2, ... pn):
statements

program:- Calculate simple interest.

```
def si2(p,t,r):
    si = (p*t*r)/100
    print (f"simple interest {si} for pa = {p}, time = {t}, roi = {r}")
```

(o/p:) call function ⇒ si2 (25000, 3, 4) # pa = 25000, time = 3 years

simple interest 3000 for pa = 25000, time = 3, roi = 4 % interest.
roi = 4.

③ function without input and with return.

```
def si3():
    p = float(input("Enter p value:"))
    t = float(input("Enter t value:"))
    r = float(input("Enter r value:"))
    si = (p*t*r)/100
    return p,t,r,si
```

Even u call directly si3() ⇒ 3750.0 ⟵ (25000, 3, 5) ⟵ p t r

o/p:- Var = si3() # function is stored in variable.

↳ for future use we need to store return values in Var

if u need print format,

print(f"simple interest {var}:", Var)

simple interest:[25000.0, 3.0, 2.6, 1950.0) // tuple format we got all value

o/p ⇒ simple interest is statement format.

store ⇒ a,b,c,d = si3() ⇒ print(a,b,c,d)

print(f"the simple interest is {d} for pa = {a}, time = {b}, ri = {c}")

the simple interest is 3600.0 for pa = 25000.0, time = 4.0, ri = 3.6

we have to pass variables as many as u taken else it pass error.

4. function with input and with return:

```
def fun_name(p1, p2, p3 ... pn):
    statements
    return value
```

syntax:-

program using with input and with return

```
def si4():
    si = (p*t*r)/100
    return si
```

o/p:-

```
si4()
```

si4(40000, 2, 1) = 800.0

need is statement format means use this statement.

```
print(f"the simple interest value is:")
```

store is variable then u print

```
siv = si4()
print(f"the simple interest value is:", siv)
```

o/p ⇒ The simple interest value is: 800.0)

1. write a program for prime number using (for with input and with return)

```
def prime(num):
    for i in range(2, num, 1):
        if(num%i==0):
            print(f"{num} is not prime number")
            break
    else:
        print(f"{num} is prime number)
```

prime(9)

9 is not prime number

prime(11)

91 is prime number.

2. print prime numbers from given range (with i/p and without return)

```
def prime(r1, r2):
    for i in range(r1, r2+1, 1):
        for j in range(2, i, 1):
            if(i%j==0):
                break
        else:
            print(i)
```

output :-

prime (2,10) # call function then we get o/p.

o/p ⇒ 2, 3, 5, 7

prime (2, 20)

o/p ⇒ 2, 3, 5, 7, 11, 13, 17, 19

3. prime num range (with input and with return)

we need the following method for getting return value, bcz once the i value enter the break statement, the complete function will stop. so we use append method and return will give on end.

```
def prime(r1, r2):
    l1 = []
    for i in range(r1, r2+1, 1):
        for j in range(2, i, 1):
            if (i%j==0):
                break
            else:
                l1.append(i) ⇒ insert the value
    return l1 ⇒ finally return after the completion of loop
```

call function :- prime (2,10).

o/p = [2, 3, 5, 7] → list format.

4. Create a function to find largest among 3 number by with input and without return.

```
def large(x, y, z):
    if (x>y and x>z):
        print return f"{x} is largest"
    elif (y>z and y>x):
        return f"{y} is largest"
    else:
        return f"{z} is largest"
```

call function :- large (4, 9, 7)

o/p : 9 is largest.

Lambda function :-

- It is small anonymous function
- defined by using lambda keyword instead of def
- can take any num of arguments but must contain only one expression
- expression is automatically returned.
- (no need to use return.

Syntax :-

lambda arguments : expression

ex :- ①

```
s = lambda a, b : a+b
```

call function s(a,b) ⇒ s(4,5)

o/p : 9 # s(4,5) = 4+5 = 9

cube = a^3

②

```
cube = lambda a : a**3 (a*a*3)
```

call function cube(9) = 729 # 9×9×9 = 729