24-09-2025, Wednesday

FUNCTIONS: It is a block of reusable code that performs a specific task.

* Fns makes our programs more organized, readable & reduce repetition.

Types of fns:

1. Built in fn: Already available in python

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

2. User-defined fns: Fns are created by the user using "def" keyword.

def → define Syntax: def fun_name(parameters):
    statements
    return value

Parameters — input to pass for fns.

Create fns using 4 ways:-

i) fn without input & without return value

ii) fn with input & without return value

iii) fn without input & with return value

iv) fn with input & with return value

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

---

1. Users - Defined fns :

i) fn without input & without return value

```
def fun-name():
    statements.
```

→ Addition of 2 no.s :

```
def add():
    x = int(input("Enter x value"))      } # variable declaration.
    y = int(input("Enter y value"))
    s = x+y
    print(f"The sum of {x} and {y} is {s}")
add()   # calling the fn
```

x & y are local variable, so, we cannot use outside the fn. only inside the fn.

```
add()
print(x,y)   # Error due to local variable.
```

* We're not passing input as parameter, so we are declaring inside the fn.

(ii) fn ~~without~~ with input & without return.

```
def fun-name(p1, p2, ... pn):
    statements
```

Ex

```
def add1(x,y):
    s = x+y
    print(s)
add1(5,2)      o/p 7   # fn calling
```

How many parameters we have passed during the fn calling the same no. of input should pass.

```
a = int(input("Enter a value:"))
b = ---
add1(a,b)      # fn calling
  (or)
add1(a,53)     # fn calling
```

(iii) fn without input & with return value.

```
def fun-name():
    statements
    return value
```

```
def add():
    x = int(input("Enter x value: "))
    y = ——————
    s = x+y
    return s
add2()
```

→ o/p Enter x value: 5
——————: 8
13

⇒ return x, y, s
add2()
o/p (5, 8, 13) in tuple format

```
⇒ sum = add2()    # storing fn in a variable
print(f" The sum of 2 no is : ", sum)
```

Enter x value : 5
——————: 8
The sum of 2 no. is : 13

```
⇒ a, b, c = add2()    # no. of return value = 3, so, we are storing
print(f" The sum of {a} and {b} is {c} ")    fn using 3 variables.
```

o/p Enter x value: 5
——————: 4
The sum of 5 and 4 is 9

```
return x, y, s
⇒ sum = add2()
print(f" The sum of {sum[0]} and {sum[1]} is {sum[2]} ")
```

(iv) fn with input & with return value

```
def fun-name(p1, p2 .... pn):
    statements
    return value.
```

Note: If we use return, don't need of print statement

Ex
```
def add(x, y):
    s = x+y
    return x, y, s
add(2, 5)        o/p  2, 5, 7
```

or
```
x, y, s = add(2, 5)
print(f" The sum of {x} and {y} is {s} ")
```

o/p The sum of 5 and 2 is 7

⇒ Create a fn to check the given no. is prime or not using with input & without return method.

```
def prime(n):
    for i in range(2, n, 1):
        if(n % i == 0):
            print(n, "Prime no.")
            break
    else:
        print(n, "Not Prime")
prime(5):
```

o/p 5 Prime no.

Nested fn

⇒ Create a fn to print prime no. from the given range, Using with input & without using return method

```
def prime(a, b):
    print(f" Prime no. b/w {a} and {b} are: ", end=" ")
    for num in range(a, b+1, 1):
        if (num > 1):
            for i in range(2, num):
                if (num % i == 0):
                    break
            else:
                print(num, end=" ")
prime(2,9)
```

o/p Prime no b/w 2 and 9 are: 2 3 5 7

a = 2 b = 9 num = 2, 3, 4, 5, 6, 7, 8, 9

⇒ num = 2
if 2 > 1, i = (2, 2) this inner loop don't run bcz 2,2 range. don't have any value
i = 2
No divisor found, hence go to else block go to else block of for loop & print 2

⇒ num = 3 i = (2, 3) = 2,
if (3 % 2 == 0) False # No divisor found
So, else: print 3

⇒ num = 4 i = (2, 4) = 2, 3
i = 2 if (4 % 2 == 0) True break # Divisor found
~~i = 3, 4 % 3 == 0~~

⇒ num = 5 i = (2, 5) = 2, 3, 4
i = 2 5 % 2 == 0 False
i = 3 5 % 3 == 0 False
i = 4 5 % 4 == 0 False No divisor found
Hence go to else block.
o/p 2, 3, 5

⇒ num = 6 i = (2, 6) = 2, 3, 4, 5
i = 2 6 % 2 == 0 True Divisor is found So, Break

⇒ num = 7 i = (2, 7) = 2, 3, 4, 5, 6
i = 2 7 % 2 == 0 False
7 % 3
7 % 4 == 0 False o/p 2, 3, 5, 7
7 % 5
7 % 6

$\Rightarrow$ num = 8 $\quad$ i° = (2, 8) = 2, 3, 4, 5, 6, 7, 8

i° = 2 $\quad$ 8 % 2 == 0 $\quad$ True $\quad$ break

$\Rightarrow$ num = 9 $\quad$ i° = (2, 9) = 2, 3, 4, 5, 6, 7, 8

i° = 2 $\quad$ 9 % 2 == 0 $\quad$ False

i° = 3 $\quad$ 9 % 3 == 0 $\quad$ True $\quad$ break $\quad$ o/p 2, 3, 5, 7