9/7

Day-6

Functions in Python.

→ a block of reusable code that performs a specific task.

Purpose: Avoid code repetition.
improve modularity and readability

Syntax.

```
def function_name(parameters):
    # Function body
    return value
```

function with parameters.

```
def add(a, b):
    return a+b
```

Pass Statement

```
def future_function():
    pass #placeholder will add logic later
```

Return Statement

→ sends back a value from a function.

```
def Square(n):
    return n*n
result = square(4)
print(result)
```

Global and local Variables

Local: inside function, accessible only there.

Global: Outside function, accessible throughout

```
x=10  # Global.
def my_function():
    y=5  # local.
    Print(y)
```

Recursion :- a function calling itself.

```
def factorial(n)
    if n==1:
        return 1
    return n * factorial(n-1)
```

*args → Allows passing multiple arguments as a tuple.

```
def add_numbers(*args):
    return sum(args)
Print(add_numbers(1,2,3,4))
```

**Kwargs → allows passing multiple Keywords arguments as a dictionary

First Class functions : functions can be Assigned to variables, Passed as arguments, Returned from other functions. def greet(func): Print(func("hello"))

Lambda functions - asynchronous fn - defined using lambda keyword:

```
square = lambda x: x*x
```

Map() - Apply function to all elements.

```
Map(lambda x: x*x, [1,2,3])
```

filter() - Select elements based on condition

```
filter(lambda x: x%2==0, [1,2,3])
```

reduce() - Combine elements to single value.

```
reduce(lambda x, y: x*y, [1,2,3])
```