09/07/2025

Day 6

Functions in Python:

- block of reusable codes that performs a specific task.
  - reusable codes.
  - use multiple times.
  - avoid redudancy.
  - readability.

Defining a function:

functions in Python are defined using the def keyword.

Syntax:

```
def function name (parameters):
    # function body
    return value.
```

- function name
- parameters
- return

```
def greet():
    Print ("hai, welcome to Python")
# function_name () calling,
greet ()
```

hai, welcome to Python.

```
def sum (num1, num2):
    Print ("addition") of
    Print (num1 + num2)
    return (num1 + num2)
result = sum (10, 20)
Print (result)
# 30.
```

Global variable & local variable:

outside function: global variable

inside function: local variable.

Recursion in Python:

function calls itself.

ex- factorial:

```
def factorial (num):
    if num == 1:
        return 1
    return n * factorial (n-1)
Print (factorial (5))
# output: 120.
```

* args :

multiple arguments.

Pass as tuple.

** kwargs :

- allows passing multiple keyword arguments as a dictionary.

First class function :

function as an argument for another function.

```
def shout (text):
    return text.upper()
def greet (func):
    print (func ("hello "))
greet (shout)
# output : HELLO
```

Lambda function :

- small anonymous function.
- defined using the lambda keyword.

function_name = lambda parameter : logics.

```
square = lamba x : x * x
print (square (5))      # output : 25
```

1. map () function:

- applies a function to all items in an iterable.

```
numbers = [1, 2, 3, 4]
squared = list (map (lambda x : x * x, numbers))
Print (squared)
# output : [1, 4, 9, 16].
```

2. filter () function:

- filter elements based on a condition

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list (filter (lambda x : x % 2 == 0, numbers))
print even_numbers
# output : [2, 4, 6]
```

3. reduce () function:

reduces a list to a single value.

```
from functools import reduce numbers = [1, 2, 3
Product = reduce (lambda x, y : x * y, numbers)
Print (Product)
# output : 24
```

Inner Function:

A function defined inside another function.

Eg:

```
def outer - function ():
    def inner - function ():
    print ("I am an inner function.")
    inner - function () outer function ()
```

Output:

I am an inner function.