**Module 8) Advance Python Programming**

* **Printing on Screen**

1. Introduction to the print() function in Python

* The print() function outputs data to the standard output device (typically the console). It can accept multiple arguments, which are separated by spaces by default.
* print("Hello", "World") # Outputs: Hello World

1. Formatting outputs using f-strings and format().

* **f-strings (formatted string literals):** Introduced in Python 3.6, f-strings provide a concise and readable way to embed expressions inside string literals.

name = "Alice"

age = 30

print("My name is {name} and I am {age} years old.")

* **format() method:** Allows insertion of variables into strings using placeholders.

name = "Alice"

age = 30

print("My name is {} and I am {} years old.".format(name, age))

* Reading Data from Keyboard

1. Using the input() function to read user input from the keyboard

* The input() function reads a line from input, returning it as a string.
* user\_input = input("Enter something: ")

2. Converting user input into different data types (e.g., int, float, etc.).

* User input is read as a string; to use it as a number, it must be converted.

age = int(input("Enter your age: "))

height = float(input("Enter your height in meters: "))

* Opening and Closing Files

1. Opening files in different modes ('r', 'w', 'a', 'r+', 'w+').

* 'r': Read mode (default).
* 'w': Write mode; creates a new file or truncates an existing one.
* 'a': Append mode; writes data at the end of the file.
* 'r+': Read and write mode; does not truncate the file.
* 'w+': Write and read mode; truncates the file.

2. Using the open() function to create and access files.

* file = open("example.txt", "w")

3. Closing files using close().

* It's important to close a file after its operations are complete to free system resources.
* file.close()
* Reading and Writing Files

1. Reading from a file using read(), readline(), readlines().

* read(): Reads the entire file.
* readline(): Reads one line at a time.
* readlines(): Reads all lines into a list.

1. Writing to a file using write() and writelines().

* write(): Writes a string to the file.
* writelines(): Writes a list of strings to the file.
* Exception Handling

1. Introduction to exceptions and how to handle them using try, except, and finally.

* Exceptions are errors detected during execution. In Python, exceptions can be handled using the try and except blocks. The finally block can be used to execute code regardless of whether an exception occurred or not.

try:

# Code that might raise an exception

except SomeException:

# Code that runs if the exception occurs

finally:

# Code that runs no matter what

2. Understanding multiple exceptions and custom exceptions.

* Multiple exceptions can be handled by specifying them as a tuple in an except block. Custom exceptions can be created by inheriting from the Exception class.

try:

# Code that might raise multiple exceptions

except (TypeError, ValueError) as e:

# Handle both exceptions

* Class and Object (OOP Concepts)

1. Understanding the concepts of classes, objects, attributes, and methods in Python.

* A class is a blueprint for creating objects. It defines attributes (variables) and methods (functions).
* An object is an instance of a class.

class Car:

def \_\_init\_\_(self, brand, model):

self.brand = brand

self.model = model

mycar = Car("Mahindra", "Thar")

print(mycar.brand, mycar.model)

1. Difference between local and global variables.

* Local variables exist within a function and are deleted once the function exits.
* Global variables exist throughout the program.
* Inheritance

1. Single, Multilevel, Multiple, Hierarchical, and Hybrid inheritance in Python

* **Single Inheritance:** A child class inherits from a single parent.
* **Multilevel Inheritance:** A child inherits from another child.
* **Multiple Inheritance:** A child inherits from multiple parents.
* **Hierarchical Inheritance:** Multiple child classes inherit from one parent.
* **Hybrid Inheritance:** Combination of the above types.

2. Using the super() function to access properties of the parent class

* The super() function is used in a child class to **call methods or constructors of its parent class**.  
  It is useful when **overriding** methods and still needing parent class functionality.
* Method Overloading and Overriding

1. Method overloading: defining multiple methods with the same name but different parameters.

* **Method Overloading** means having **multiple methods with the same name** but **different arguments**.
* Python does not support traditional overloading like Java or C++.  
  Instead, we can use **default arguments** or **\*args** to simulate overloading.

1. Method overriding: redefining a parent class method in the child class

* **Method Overriding** is when a **child class defines a method with the same name** as a method in the **parent class**.
* The child class version **replaces** the parent’s method when called from the child object.  
  It is useful in **polymorphism** and customizing inherited behavior.
* SQLite3 and PyMySQL (Database Connectors)

1. Introduction to SQLite3 and PyMySQL for database connectivity.

* SQLite3 is a built-in Python module that allows database operations without installing anything externally. It stores data in a single .db file and is suitable for lightweight applications. PyMySQL is a third-party library that connects Python to MySQL databases. It requires installation using pip install pymysql. Both are used to perform database operations like storing, retrieving, and managing data directly from Python programs.

1. Creating and executing SQL queries from Python using these connectors.

* To create and execute SQL queries in Python, first connect to the database using sqlite3.connect() or pymysql.connect(). Then, create a cursor object with .cursor(). You can run SQL statements like CREATE TABLE, INSERT INTO, or SELECT using cursor.execute(). Finally, commit changes using .commit() and close the connection with .close(). This allows full control of database actions directly through Python code.
* Search and Match Functions

1. Using re.search() and re.match() functions in Python’s re module for pattern matching.

* Python’s re module provides functions for pattern matching using regular expressions.  
  re.search() scans the entire string for a match of the pattern.  
  re.match() only checks for a match **at the beginning** of the string.  
  Both return a match object if found, else return None.

1. Difference between search and match.

* re.match() checks if the pattern is **only at the beginning** of the string.
* re.search() checks if the pattern is **anywhere in the string**.
* re.match() fails if the match is not at position 0, while re.search() succeeds if the pattern is found at any position.