MODULE – 8

## ( Advance Python Programming)

# Printing on Screen

1. • Introduction to the print() function in Python.

In Python, the print() function is a built-in tool used to display output on the screen, essentially allowing you to "print" any data or message you want to see in the console; it takes in one or more objects as input, converts them to strings, and displays them with a space between each object by default, adding a new line at the end of the output.

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

The format() function takes a value and a format specifier as arguments. Then, it applies the specifier to the value to return a formatted value. The format specifier must follow the rules of the string formatting mini-language.

# Reading Data from Keyboard

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

By fetching input and assigning it to variables, your code can react to adjustable conditions rather than just executing static logic flows. This personalizes programs to individual users. The input() function is the simplest way to get keyboard data from the user in Python.

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

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the built-in functions like int(), float(), str(), etc to perform explicit type conversion. This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

# Opening and Closing Files

1. • Opening files in different modes ('r', 'w', 'a', 'r+', 'w+').
2. Read Mode: 'r' The read mode ('r') assumes the role of the default mode for file opening in Python. ...
3. Write Mode: 'w' The write mode ('w') opens the gateway to writing data into a file. ...
4. Binary Mode: 'b' ...
5. Append Mode: 'a' ...
6. Read and Write Mode: 'r+'

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

The open() python is used to open a file in either the write mode or the read mode. We use the open() function with two arguments, the file name and the mode, whether to read or write, to return a file object.

3>• Closing files using close().

The close() method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done. Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

# Reading and Writing Files

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

To use the readlines function in Python, open a file in read mode, call readlines() on the file object, and it will return a list of lines from the file, each line as a separate list item.

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

The write() method in Python is used to write data to a file. It takes a string argument and appends it to the end of the file's content. If the file doesn't exist, it creates a new file.

The writelines() function is a method of Python's file objects, used to write a list or sequence of strings to a file. Unlike the write() method, which writes a single string, writelines() enables the writing of multiple strings in one go.

# Exception Handling

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

**The typical structure is try-except-finally , where:**

1. The try block contains the code that may raise an exception.
2. The except block handles the exception if one occurs.
3. The finally block contains code that should always be executed, whether an exception occurred or not.
4. • Understanding multiple exceptions and custom exceptions.

Key points about multiple exceptions:

* **Handling different error scenarios:**

When you might encounter various potential errors in a section of code, you can use multiple catch blocks to handle each exception type uniquely, providing more granular error management.

* **Exception hierarchy:**

Most programming languages have an exception hierarchy where more specific exceptions inherit from broader ones, allowing you to catch a general exception type and then further refine the handling based on the specific exception caught.

* **Catch-all clause:**

Sometimes, you might include a final catch block without specifying an exception type to handle any unexpected errors that might occur.

# Class and Object (OOP Concepts)

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

n Python, a class acts as a blueprint for creating objects, defining the data (attributes) and actions (methods) that an object can have; an object is an instance of a class, essentially a tangible entity with its own set of attributes and the ability to perform the methods defined in the class; attributes are variables within a class that store data specific to an object, while methods are functions defined within a class that operate on the object's data, allowing it to perform specific actions.

1. • Difference between local and global variables.
2. The main difference between local and global variables is their scope, or the part of a program where they can be accessed.

|  |  |  |
| --- | --- | --- |
|  | **Local variable** | **Global variable** |
| Scope | Can only be accessed within a function or block | Can be accessed throughout the entire program |
| Lifetime | Exist only during the function's execution | Remain in memory for the duration of the program |
| Use cases | Useful for test variables in loops | Useful for passing information between sections of code |

# Inheritance

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

**In Python inheritance, "Single" means a class inherits from only one parent class, "Multilevel" refers to a class inheriting from another derived class (creating a chain), "Multiple" allows a class to inherit from multiple parent classes at once, "Hierarchical" means multiple classes inherit from a single parent class, and "Hybrid" combines two or more of these inheritance types within a single class structure.**

* **Single Inheritance:**

**A simple case where a child class only inherits properties and methods from one parent class.**

* **Multilevel Inheritance:**

**A class inherits from another class which itself is derived from a different class, creating a chain of inheritance.**

* **Multiple Inheritance:**

**A class inherits features from more than one parent class simultaneously.**

* **Hierarchical Inheritance:**

**Multiple classes are derived from a single base class, forming a tree-like structure.**

* **Hybrid Inheritance:**

**A combination of different inheritance types, like using both multiple and hierarchical inheritance within a class structure.**

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

he `Child` constructor uses `super(parentValue)` to call the constructor of the parent class and initializes the `parentVariable`. The `displayValues` method in the child class uses `super. parentVariable` to access the `parentVariable` from the parent class.

# Method Overloading and Overriding

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

Method overloading in programming allows defining multiple methods with the same name but different parameters (either in number, type, or order).

* **Same Name, Different Parameters:**

The core principle of method overloading is having multiple methods within the same class that share the same name but have distinct parameter lists.

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

Method overriding is an OOP feature that allows a child class to provide its own implementation for a method already defined in its parent class. The child class will inherit all the methods from the parent class. However, it can override specific inherited methods by redefining them using the same method signature.

# SQLite3 and PyMySQL (Database Connectors)

1. Introduction to SQLite3 and PyMySQL for database connectivity.

SQLite is a server-less database and is self-contained. This is also referred to as an embedded database which means the DB engine runs as a part of the app. On the other hand, MySQL requires a server to run. MySQL will require a client and server architecture to interact over a network.

1. • Creating and executing SQL queries from Python using these connectors.
2. Create a new file named app.py.
3. Add a module docstring. """ Connects to a SQL database using pyodbc """
4. Import the pyodbc package. import pyodbc.
5. Create variables for your connection credentials. ...
6. Create a connection string variable using string interpolation. ...
7. Use the pyodb

# Search and Match Functions

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