**MODULE – 4**

**Q1. What is File function in python? What is keywords to create and write file.**

**Ans.** In Python, the **file** function has been replaced by the **open** function for creating, reading, and writing files. The **open** function is used to open a file and returns a file object that can be used to perform various operations on the file.

“x” keyword is used for create a new file, but raises an error if the file already exists.

“w” keyword is used for write file, if file doesn’t exists it’ll create a new file then write in that file.

**Q2. Explain Exception handling? What is an Error in Python?**

**Ans.** An error in Python is an unexpected event that disrupts the normal flow of the program. Errors can be caused by a variety of factors, such as:

**Syntax errors:** These errors occur when the code is not written correctly. For example, a syntax error would occur if you forgot to put a colon after a function declaration.

**Runtime errors:** These errors occur when the code is syntactically correct, but it tries to do something that is impossible or illegal. For example, a runtime error would occur if you tried to divide a number by zero.

**Logical errors:** These errors occur when the code is syntactically correct and there are no runtime errors, but the code produces incorrect results. For example, a logical error would occur if you wrote a function that was supposed to calculate the factorial of a number, but it actually calculated the square of the number.

Exception handling is a way to deal with errors that occur during the execution of a Python program. When an error occurs, the Python interpreter will stop the program and print an error message.

Exception handling allows you to gracefully handle errors so that your program doesn't crash. You can use exception handling to:

The try and except statements are used to implement exception handling in Python. The try block contains the code that you want to execute. If an error occurs in the try block, the except block will be executed. The except block can contain code that will handle the error.

**Q3 . How many except statements can a try-except block have? Name Some**

**built-in exception classes:**

**Ans.** A try-except block can have at least one except statement. There is no limit to the number of except statements that a try-except block can have. However, it is important to note that each except statement can only handle a single exception type.

Here are some built-in exception classes in Python:

**Exception:** The base class for all exceptions.

**ArithmeticError:** Raised for errors that occur during arithmetic operations.

**AssertionError:** Raised when an assertion fails.

**AttributeError:** Raised when an attribute is not found on an object.

**EOFError:** Raised when the end of a file is reached.

**IOError:** Raised when an input/output operation fails.

**IndexError:** Raised when an index is out of bounds.

**KeyError:** Raised when a key is not found in a dictionary.

**NameError:** Raised when a variable is not found in the current scope.

**TypeError:** Raised when an operation is performed on an object of the wrong type.

**ValueError:** Raised when a value is not of the correct type or is out of range.

**Q4 . When will the else part of try-except-else be executed?**

**Ans.** The else part of a try-except-else block will be executed only if no exception is raised in the try block.

The else part of a try-except-else block can be used to execute code that should only be executed if no exception is raised. This can be useful for logging messages or performing cleanup tasks.

**Q5. Can one block of except statements handle multiple exception?**

**Ans.** Yes, one block of except statements can handle multiple exceptions. In Python, you can specify multiple exceptions in a single except statement by separating them with a pipe (|).

You can also use a except statement with **Exception** to handle all exceptions. This is not recommended, as it can make your code difficult to debug. However, it can be useful in some situations, such as when you want to catch all exceptions and log them.

**Q6. When is the finally block executed?**

**Ans.** The finally block in Python is executed no matter what. This means that the finally block will be executed even if an exception is raised in the try block. The finally block is useful for performing cleanup tasks, such as closing files or releasing resources.

The finally block is a versatile tool that can be used in a variety of situations. It is a good practice to use the finally block whenever you are performing cleanup tasks.

**Q7. What happens when 1 == 1 is executed ?**

**Ans.** When 1 == 1 is executed, it will evaluate to **True**. This is because the two values on either side of the equal sign are equal. The == operator in Python is used to compare two values and see if they are equal. If the two values are equal, the == operator will return True. If the two values are not equal, the == operator will return False.

In this case, the two values on either side of the equal sign are both the integer 1, The integer 1 is equal to itself, so the == operator will return **True.**

**Q.8 How Do You Handle Exceptions With Try/Except/Finally In Python?**

**Explain with coding snippets.**

**Ans.** In Python, you can handle exceptions using the try/except/finally block.

The try block is used to enclose the code that might raise an exception, while the except block is used to handle specific exceptions that may occur. The finally block is optional and is used to define code that should be executed regardless of whether an exception was raised or not. Here are coding snippets that demonstrate the usage of try/except/finally:

try:

    # Code that may raise an exception

    file = open("example.txt", "r")

    # Perform operations with the file

except :

    # Handling the FileNotFoundError

    print("Error: File not found.")

finally:

    # Code to be executed regardless of exceptions

    print("finally block executed")

**Q9. What are oops concepts? Is multiple inheritance supported in java**

**Ans.** Object-Oriented Programming (OOP) is a programming paradigm that focuses on organizing code into reusable and modular units called objects. OOP concepts provide a way to structure code based on real-world entities and their relationships. The main concepts of OOP are:

* **Encapsulation:**

Encapsulation refers to the bundling of data and methods (functions) that operate on that data within a single unit called a class. It provides data hiding and abstraction, allowing the class to control access to its internal data.

* **Inheritance:**

Inheritance enables the creation of new classes (derived classes) based on existing classes (base classes). The derived class inherits the properties (variables) and behaviors (methods) of the base class, allowing code reuse and promoting the "is-a" relationship between classes.

* **Polymorphism:**

Polymorphism allows objects of different classes to be treated as objects of a common superclass. It refers to the ability to use a single interface (e.g., a method or class) to represent different forms or behaviors. Polymorphism is achieved through method overriding and method overloading.

* **Abstraction:**

Abstraction focuses on providing simplified and essential representations of complex real-world entities. It involves hiding unnecessary details and exposing only the relevant information and behaviors. Abstraction is achieved through abstract classes and interfaces.

* **Association:**

Association represents a relationship between two or more classes. It defines how objects interact and communicate with each other. Associations can be one-to-one, one-to-many, or many-to-many, and they can be represented as instance variables or method parameters.

regarding multiple inheritance in Java, **Java does not support multiple inheritance for classes.** It means that a class cannot inherit from multiple classes simultaneously. This was a design choice made by Java's creators to avoid complexities and conflicts that can arise from multiple inheritance.

However, **Java does support multiple inheritance for interfaces.** A class can implement multiple interfaces, which allows it to inherit method signatures from multiple interfaces and provide implementations for them. This enables achieving similar benefits of code reuse and supporting multiple types/interfaces in Java.

**Q10. How to Define a Class in Python? What Is Self? Give An Example Of**

**A Python Class**

**Ans.** In Python, you can define a class using the class keyword. The class serves as a blueprint for creating objects with specific attributes (variables) and behaviors (methods).

Here's the general syntax for defining a class:

class ClassName:

# Class variables and methods go here

Pass

In the class definition, you can include attributes (variables) and methods (functions) that define the behavior of objects created from the class. These attributes and methods are accessed using the dot notation with instances of the class.

The self parameter is a reference to the current instance of the class. It is used to access the instance's attributes and methods within the class definition. By convention, self is the preferred name for this parameter, but you can use any name as long as it is the first parameter in the method definition.

Example of Python class:

class student:

    def getname(self,nname):

        print("Hello! ", nname)

a = student()

a.getname("Rohit")

in this example student class have a function which gets a name and print with a message, for using class we created an object to access class properties.

**Q 11. Explain Inheritance in Python with an example? What is init? Or What**

**Is A Constructor In Python?**

**Ans. Skip**

**Q 12. What is Instantiation in terms of OOP terminology?**

**Ans. Skip**

**Q 13. What is used to check whether an object o is an instance of class A?**

**Ans. Skip**

**Q 14. What relationship is appropriate for Course and Faculty?**

**Ans. Skip**

**Q 15. What relationship is appropriate for Student and Person?**

**Ans. Skip**