Module – 4 (Advance python programming)

1.) What is File function in python? What is keywords to create and write file.

- In Python, file handling is accomplished using the built-in `open()` function to create, read, write, and manipulate files. The `open()` function takes two parameters: the file name and the mode. The modes include:
  - `'r'`: Read (default).
  - `'w'`: Write.
  - `'a'`: Append.
  - `'b'`: Binary mode.
  - `'x'`: Exclusive creation (fails if the file already exists).
  - `'t'`: Text mode (default).
- To write to a file in Python using a for statement, you can follow these steps: Open the file using the open() function with the appropriate mode ('w' for writing). Use the for statement to loop over the data you want to write to the file. Use the file object's write() method to write the data to the file.
- File handling is an important activity in every web app. The types of activities that you can perform on the opened file are controlled by Access Modes. These describe how the file will be used after it has been opened.
- These modes also specify where the file handle should be located within the file. Similar to a pointer, a file handle indicates where data should be read or put into the file.

- **To create a new file in Python, use the open () method, with one of the following parameters: "x" - Create - will create a file, returns an error if the file exist "a" - Append - will create a file if the specified file does not exist "w" - Write - will create a file if the specified file does not exist.**

13.) Explain Exception handling? What is an Error in python?

- Exception & Error Handling in Python:
- Errors and exceptions can lead to program failure or unexpected behavior, and Python comes with a robust set of tools to improve the stability of the code.
- Errors and exceptions can lead to unexpected behavior or even stop a program from executing. Python provides various functions and mechanisms to handle these issues and improve the robustness of the code. In this tutorial, we will learn about various error types and learn about built-in functions with examples.
- Errors are problems that occur in the program due to an illegal operation performed by the user or by the fault of a programmer, which halts the normal flow of the program. Errors are also termed bugs or faults. There are mainly two types of errors in python programming.
- Exceptions are events that are used to modify the flow of control through a program when the error occurs. Exceptions get triggered automatically on finding errors in Python.
    - try/except: catch the error and recover from exceptions hoist by programmers or Python itself.

- try/finally: Whether exception occurs or not, it automatically performs the clean-up action.

- Errors in Python can be of three types:
  1. Compile time Errors
  2. Logical Errors
  3. Runtime Errors
- Exception handling is a mechanism in Python that allows you to deal with errors or exceptional situations in a program. An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. When an exception occurs, it is said to be "raised," and if it's not handled, the program will terminate with an error message.
- In Python, exceptions are raised when there is an error in the code. Common types of exceptions include:
  - **SyntaxError**: Occurs when there is a syntax error in the code.
  - **NameError**: Raised when a local or global name is not found.
  - **TypeError**: Raised when an operation or function is applied to an object of the wrong type.
  - **ValueError**: Raised when a built-in operation or function receives an argument with the right type but an inappropriate value.
  - **FileNotFoundError**: Raised when a file or directory is requested but cannot be found.

14. How many except statements can a try-except block have? Name Some built-in exception classes:

- More than zero
- Exception handling with try, except, else, and finally
- Try: This block will test the excepted error to occur
- Except: Here you can handle the error
- Else: If there is no exception then this block will be executed
- Finally: Finally block always gets executed either exception is generated or not

15.) When will the else part of try-except-else be executed?

- In a `try-except-else` block in Python, the `else` part is executed if the code inside the `try` block executes successfully without raising any exceptions. It is a block of code that is only executed when there are no exceptions raised in the `try` block.
- The `else` block is optional, and it provides a way to specify code that should be executed only when the `try` block completes successfully. If any exception is raised within the `try` block, the control is transferred to the corresponding `except` block, and the `else` block is skipped.
- Using the `else` block in exception handling is useful when you want to distinguish between the successful execution of the `try` block and the occurrence of an exception. It allows you to separate the code that may raise exceptions from the code that should execute only if no exceptions occur.
- **The else part is executed when no exception occurs.**

16. Can one block of except statements handle multiple exception?

- Yes, like except TypeError, SyntaxError, etc.
- Each type of exception can be specified directly. There is no need to put it in a list.
- Yes, a single **except** block can handle multiple exceptions in Python. You can specify multiple exception types within a single **except** block, separated by parentheses. This allows you to handle different exceptions with the same block of code.
- Using a single **except** block for multiple exception types can make the code more concise and readable, especially when the same error handling logic applies to different exceptions.

17.) When is the finally block executed?

- Finally block is always executed after leaving the try statement. In case if some exception was not handled by except block, it is re-raised after execution of finally block. finally block is used to deallocate the system resources.
- The finally block will always be executed, no matter if the try block raises an error or not:
- The `finally` keyword is used in try...except blocks. It defines a block of code to run when the try...except...else block is final.
- The `finally` block will be executed no matter if the try block raises an error or not.
- This can be useful to close objects and clean up resources.

18.) What happens when „1"== 1 is executed?

- It simple evaluates to False and does not raise any exception.

  Because one is string and other is integer.

19.) How Do You Handle Exceptions With Try/Except/Finally In Python? Explain with coding snippets.

- Exception handling with try, except, else, and finallyFirst try clause is executed i.e. the code between try and except clause. If there is no exception, then only try clause will run, except clause will not get executed. If any exception occurs, the try clause will be skipped and except clause will run.

- In Python, you can use the `try`, `except`, and `finally` blocks to handle exceptions. Here's a coding snippet that demonstrates how to use these blocks:

```
try:
    num_str = input("Enter a number: ")
    num = int(num_str)
    result = 10 / num
    print("Result:", result)

except ValueError:
    print("Invalid input. Please enter a valid number.")

except ZeroDivisionError:
    print("Cannot divide by zero.")

else:
    print("No exceptions occurred.")

finally:
    print("This will be executed no matter try except runs.")
```

   - The `try` block contains the code that might raise an exception. It tries to get user input, convert it to an integer, and perform a division.
   - The `except` blocks handle specific exceptions (`ValueError` and `ZeroDivisionError`). If any of these exceptions occur, the corresponding block is executed.
   - The `else` block is executed if the `try` block completes without raising any exceptions.
   - The `finally` block is always executed, whether there was an exception or not. It's typically used for cleanup tasks.
- When you run this code, you can observe the different behaviors based on user input. For example, if the user enters a non-integer or zero, the corresponding exception block will be executed, and the `finally` block ensures that the cleanup code is always executed.

Explain with coding snippets.

- A code Snippet is a programming term that refers to a small portion of re-usable source code, machine code, or text. Snippets help programmers reduce the time it takes to type in repetitive information while coding. Code Snippets are a feature on most text editors, code editors, and IDEs.

21.) What are oops concepts? Is multiple inheritance supported in java

- **Major OOP (object-oriented programming) concepts in Python**

  1.) .class
  2.) .object
  3.) .inheritance
  - single

- **multiple**
- **multilevel**

4.) .encapsulation

5.) .polymorphism

- **method overloading**
- **method overriding**

6.) .access specifire/modifire (data hiding)

- **private**
- **public (default in python)**

- ➢ Java supports single inheritance, meaning a class can extend only one superclass. This design choice was made to avoid the complications and ambiguities that can arise with multiple inheritance. However, Java supports a form of multiple inheritance through interfaces.
- ➢ In Java, a class can implement multiple interfaces. An interface is a collection of abstract methods that a class can implement. By implementing multiple interfaces, a class can inherit the abstract methods and achieve a form of multiple inheritance. This approach avoids the problems associated with multiple inheritance of implementation (i.e., from classes) while allowing multiple inheritance of interface (i.e., from interfaces).

22.) How to Define a Class in Python? What Is Self? Give An Example Of A Python Class.

- ➢ Defining class in python is very easy. It is defined as
- ➢ class ‘name’:
- ➢ Object():
- ➢ Self is inbuilt feature which is the first default positional argument.
- ➢ class Studentinfo:

    - id = 101
    - name = xyz
    - print(“Id:”,id)
    - print(“name:”,name)
- st = Studentinfo()
- O/P:
    - id: 101
    - name = xyz
- A class is defined using the `class` keyword, and it typically includes attributes (variables) and methods (functions). The `self` parameter is used to refer to the instance of the class and is the first parameter of all instance methods.

25.) Explain Inheritance in Python with an example? What is init? Or What Is A Constructor In Python?

- Inheritance relationship defines the classes that inherit from other classes as derived, subclass, or sub-type classes. Base class remains to be the source from which a subclass inherits. For example, you have a Base class of “Animal,” and a “Lion” is a Derived class. The inheritance will be Lion is an Animal.
- Class a:
    - Print(“hello”)
- Class b(a):
    - Print(“I am python!!!”)
- Cl = b()
- cl.b

- O/P :

    - Hello
    - I am python!!!

- Init: The __init__ method is the Python equivalent of the C++ constructor in an objectoriented approach. The __init__ function is called every time an object is created from a class.
- Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a new class (called a derived or child class) to inherit attributes and methods from an existing class (called a base or parent class). This promotes code reuse and allows you to create a more specialized class based on an existing one.

26.) What is Instantiation in terms of OOP terminology?

- Instantiation − The creation of an instance of a class. Method − A special kind of function that is defined in a class definition. Object − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- Creating an instance of class.
- Instantiation refers to creating an object/instance for a class.
- In object-oriented programming (OOP) terminology, instantiation refers to the process of creating an instance of a class. An instance, or object, is a concrete occurrence of a class, and it represents a specific entity with its own set of attributes and behaviors.
- Here are the key concepts related to instantiation:

1. **Class:**
   - A class is a blueprint or template for creating objects. It defines the attributes (data) and behaviors (methods) that the objects of the class will have.
2. **Object:**

- An object is an instance of a class. It is a concrete realization of the class's blueprint, with its own unique data and functionality.

3. **Instantiation:**
   - Instantiation is the process of creating an instance of a class. It involves calling the class's constructor (often named `__init__` in Python) to initialize the attributes of the object.
4. **Constructor:**
   - The constructor is a special method in a class that is called during the instantiation process. It is responsible for initializing the attributes of the object.

27.) What is used to check whether an object o is an instance of class A?

- Using isinstance() function, we can test whether an object/variable is an instance of the specified type or class such as int or list. In the case of inheritance, we can checks if the specified class is the parent class of an object. For example, isinstance(x, int) to check if x is an instance of a class int .
- In Python, you can use the **`isinstance()`** function to check whether an object **`o`** is an instance of a particular class **`A`**. The **`isinstance()`** function returns **`True`** if the object is an instance of the specified class or a subclass of that class, and **`False`** otherwise.
- Here's the syntax:
  - `isinstance(object, classinfo)`
    - `object`: The object to be checked.
    - `classinfo`: A class or a tuple of classes to check against.

28.) What relationship is appropriate for Course and Faculty?

- One to one relationship and multilevel.
- The relationship between a **`Course`** and **`Faculty`** in a university or educational context is typically a many-to-many relationship. This means that multiple courses can be taught by multiple faculty members, and, conversely, a faculty member can teach multiple courses.
- In an object-oriented programming context, this relationship is often modeled using classes.

29.) What relationship is appropriate for Student and Person?

- Many to one : multiple.
- The relationship between **`Student`** and **`Person`** is often modeled as an "is-a" relationship, indicating that a student is a type of person. This relationship is known as inheritance in object-oriented programming.
- Inheritance allows the **`Student`** class to inherit attributes and behaviors from the **`Person`** class, as a student is essentially a specialized type of person with additional features related to their student status.