**Python Advanced Assignment 8**

Q1. What are the two latest user-defined exception constraints in Python 3.X?

Ans-) The two latest user-defined exception constraints in Python 3.x are that the exception must inherit from the BaseException class, and that the \_\_str\_\_ method should be used to specify the error message associated with the exception.

Q2. How are class-based exceptions that have been raised matched to handlers?

Ans-) When a class-based exception is raised, Python searches for a matching exception handler by looking for the first except block that matches the exception type or a base class of the exception type. If a matching handler is found, the code within that block is executed. If no matching handler is found, the exception is propagated up to the calling function or the interpreter.

Q3. Describe two methods for attaching context information to exception artefacts.

Ans-) Two methods for attaching context information to exception artifacts are:

Using the args attribute of the exception object to provide additional information about the error. This can be done by passing the additional information as arguments when raising the exception, and then accessing them through the args attribute in the exception handler.

Using the \_\_context\_\_ and \_\_cause\_\_ attributes of the exception object to chain exceptions together and provide more detailed information about the error. This can be done by using the raise statement with an exception object that has the \_\_context\_\_ or \_\_cause\_\_ attribute set to the original exception.

Q4. Describe two methods for specifying the text of an exception object&#39;s error message.

Ans-) Two methods for specifying the text of an exception object's error message are:

Overriding the \_\_str\_\_ method in the exception class to return a string representation of the error message.

Using the args attribute of the exception object to specify the error message as a string or tuple of strings.

Q5. Why do you no longer use string-based exceptions?

Ans-) String-based exceptions are no longer used in Python because they are less flexible and less consistent than class-based exceptions. Class-based exceptions provide a more structured way to handle errors, allowing for better organization of error types and more detailed error messages. Additionally, class-based exceptions support inheritance and polymorphism, which makes it easier to define hierarchies of error types and handle related errors in a consistent way.