**Assignment - 8**

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

Ans: In Python 3.X, the two latest user-defined exception constraints are:

* The exception class must be derived from the built-in Exception class or one of its subclasses.
* The exception class typically should have an \_\_init\_\_ method to allow customization of the exception instance upon instantiation.

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

Ans: Class-based exceptions that have been raised are matched to handlers based on the inheritance hierarchy of the exception classes. Python searches for the nearest enclosing except clause with a matching exception class or one of its base classes. If no matching handler is found, the exception propagates up the call stack until it is caught or results in an unhandled exception error.

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

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

* Adding custom attributes to the exception instance within its \_\_init\_\_ method to provide additional context information.
* Using the with\_traceback() method to attach a traceback object from another exception to the current exception, preserving the original traceback information.

Q4. Describe two methods for specifying the text of an exception object's error message.

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

* Defining a custom \_\_str\_\_ method in the exception class to return a formatted error message string.
* Passing the error message string as an argument to the base class's \_\_init\_\_ method when creating an instance of the exception class.

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

Ans: String-based exceptions are no longer used because they lack the ability to provide detailed information about the error and make it difficult to handle exceptions programmatically. Class-based exceptions offer more flexibility and extensibility, allowing developers to create custom exception hierarchies with additional attributes and methods for better error handling and debugging. Additionally, class-based exceptions are more compatible with Python's exception handling mechanisms and are considered more Pythonic.