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

ANS:

User-Defined Exceptions in Python

* class JustException(Exception): def \_\_init\_\_(self, message): print(message)
* raise JustException("Raise an Exception")

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

ANS:

Class-based exceptions are either raised by the ABAP statement RAISE EXCEPTION or by the ABAP runtime environment. If a class-based exception occurs, the system interrupts the normal program flow and tries to find a suitable handler. If it does not find a handler, a runtime error occurs.

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

ANS:

* It can be used to gain additional information about the error encountered.
* As contents of an Argument can vary depending upon different types of Exceptions in Python, Variables can be supplied to the Exceptions to capture the essence of the encountered errors. Same error can occur of different causes, Arguments helps us identify the specific cause for an error using the except clause.
* It can also be used to trap multiple exceptions, by using a variable to follow the tuple of Exceptions.

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

ANS:

## User-defined Exceptions

Programs may name their own exceptions by creating a new exception class (see [Classes](https://docs.python.org/3/tutorial/classes.html#tut-classes) for more about Python classes). Exceptions should typically be derived from the [Exception](https://docs.python.org/3/library/exceptions.html#Exception) class, either directly or indirectly.

Exception classes can be defined which do anything any other class can do, but are usually kept simple, often only offering a number of attributes that allow information about the error to be extracted by handlers for the exception.

## Handling Exceptions

It is possible to write programs that handle selected exceptions. Look at the following example, which asks the user for input until a valid integer has been entered, but allows the user to interrupt the program (using Control-C or whatever the operating system supports); note that a user-generated interruption is signalled by raising the [KeyboardInterrupt](https://docs.python.org/3/library/exceptions.html" \l "KeyboardInterrupt" \o "KeyboardInterrupt) exception.

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

ANS:

Because it is possible (although unlikely) that this change broke existing code, the Python interpreter can be invoked the command line option -X to disable this feature, and use string exceptions like before. This option is a temporary measure - eventually the string-based standard exceptions will be removed from the language altogether. It hasn't been decided whether user-defined string exceptions will be allowed in Python 2.0.