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

Some standard exceptions which are found are include ArithmeticError, AssertionError, AttributeError, ImportError, etc.

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

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.

There might arise a situation where there is a need for additional information from an exception raised by Python.   
Python has two types of exceptions namely, [Built-In Exceptions](https://www.geeksforgeeks.org/built-exceptions-python/) and [User-Defined Exceptions](https://www.geeksforgeeks.org/user-defined-exceptions-python-examples/).

Using arguments for Exceptions in Python is useful for the following reasons:

* 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.

try:

pass

except Exception as e:

if hasattr(e, 'message'):

print(e.message)

else:

print(e)

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