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

**ANS:-**

1. **raise statement - This is used to raise an inbuilt exception explicitly.**
2. **assert statement - when condition in assert statement is true, program is executed normally. When condition is false, program gives an AssertionError.**

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

**ANS:-**

**When exceptions are raised, then the exception is handled by the same class mentioned in except statement or its base classes. "Exception" is the base class(Superclass) for all other exception classes.**

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

**ANS:-**

1. **By using logging.exception() method, we can log log messages with level ERROR on the logger. I will store the information like type of error, line number, file name and path.**
2. **By using raise keyword. It will also display Traceback of the recent exception raised.**

**import logging**

**try:**

**4/0**

**except Exception as e:**

**logging.exception(e)**

**raise**

**ERROR:root:division by zero**

**Traceback (most recent call last):**

**File "<ipython-input-1-ecac2bd4b516>", line 4, in <module>**

**4/0**

**ZeroDivisionError: division by zero**

**--------------------------------------------------------------**

**ZeroDivisionError Traceback (most recent call last)**

**<ipython-input-1-ecac2bd4b516> in <module>**

**2**

**3 try:**

**----> 4 4/0**

**5 except Exception as e:**

**6 logging.exception(e)**

**ZeroDivisionError: division by zero**

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

**ANS:-**

1. **We can print the text of an exception, by using the object of an Exception class. We can access different attributes like messgae, args, traceback from that object.**
2. **By using logging.error() method**

**try:**

**a = 'abc'**

**b = 2**

**c = a+b**

**except TypeError as e:**

**print(e)**

**can only concatenate str (not "int") to str**

**try:**

**file = open('one.txt','r')**

**except Exception as e:**

**logging.error(e)**

**ERROR:root:[Errno 2] No such file or directory: 'one.txt'**

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

**ANS:-**

**In Python versions 1.5 and earlier, Exceptions were strings and not classes. Now all exceptions are classes, derived from the superclass BaseException. With classes it is more efficiently handled as classes have many useful features.**