Certification Program in Data Analytics

Topic: Exception Handling Theory Questions

1. What is an exception?

Ans.:- an exception is an error that occurs during the execution of a program. When an exception occurs, the program stops running and Python raises an exception object that contains information about the error.
Exceptions can occur for many reasons, such as:

- *Trying to access an undefined variable*
- *Attempting to divide by zero*
- *Trying to open a file that doesn't exist*
- *Passing invalid arguments to a function*
- *and many more.*

2. Can you explain the difference between errors and exceptions?
Ans.:-
An error is a problem that occurs during the execution of a program that prevents the program from running as intended. This can be due to a syntax error, a semantic error, a runtime error, or any other issue that prevents the program from working correctly.

An exception, on the other hand, is a specific type of error that occurs during the execution of a program due to exceptional circumstances, such as invalid input or a resource that is not available. Exceptions can be raised explicitly by the programmer, or they can be raised implicitly by the Python interpreter.

3. How can you ensure that a certain code block runs no matter whether there's an exception or not?
Ans.In Python, you can use a **finally** block to ensure that a certain code block runs no matter whether there's an exception or not. The **finally** block is executed after the **try** block and any corresponding **except** blocks have finished executing, whether or not an exception was raised.

4. Can you give me some examples of built-in exceptions?

1. **SyntaxError**: Raised when there is a syntax error in the code.
2. **TypeError**: Raised when an operation or function is applied to an object of inappropriate type.
3. **ValueError**: Raised when an operation or function receives an argument of the correct type but with an inappropriate value.
4. **ZeroDivisionError**: Raised when attempting to divide a number by zero.
5. **IndexError**: Raised when trying to access an index that does not exist in a list, tuple, or string.
6. **KeyError**: Raised when trying to access a dictionary key that does not exist.
7. **FileNotFoundError**: Raised when trying to open a file that does not exist.
8. **AttributeError**: Raised when trying to access an attribute of an object that does not exist.
9. **NameError**: Raised when trying to access a variable or function that does not exist.
10. **OverflowError**: Raised when a mathematical operation overflows the maximum representable value.

5. How do you handle exceptions with try/except/finally?

Ans.

1. The **try** block contains the code that you want to execute. If an exception occurs within the **try** block, Python will immediately jump to the appropriate **except** block.
2. The **except** block contains the code that will handle the exception. You can specify which type of exception to catch by including the exception type in parentheses after the **except** keyword. If you don't specify an exception type, the **except** block will catch all exceptions.
3. The **finally** block contains code that will always execute, regardless of whether an exception was raised or not. This block is often used to clean up resources, such as closing a file or a database connection.

6. "Every syntax error is an exception but every exception cannot be a syntax error". Justify the Statement.

- Syntax errors are detected when we have not followed the rules of the particular programming language while writing a program and that may cause of arising an exception,
- While exception can occur even without any syntax error, for example, exception like trying to open a file which does not exist, division by zero, etc. so this is how Every syntax error is an exception but every exception cannot be a syntax error.

7. What is the use of raise statement?

The raise statement is used to raise an exception. When you raise an exception, you signal to the calling code that something has gone wrong and that the current function is unable to handle the error. The raise statement is typically used in combination with the try and except blocks to handle and propagate exceptions

8. What is the use of else block in exception handling?

The **else** block in exception handling is used to specify code that should be executed if no exceptions are raised in the **try** block. The **else** block is optional, and it follows all the **except** blocks.

9. What is the syntax for try and except block?

```
try:
   # Code that might raise an exception
   # ...
except ExceptionType:
   # Code that will handle the exception of type ExceptionType
   # ...
```

10. What is the purpose of finally block?

The Purpose of finally block is to do clean up action, every time after completion of the program.

11. Can you explain the difference between try-except and try-finally block?

- Try-except is used for catching and handling exceptions.
- Try-finally is used for ensuring that the code is executed, regardless of an exception.

12. Can you give example of using multiple except block in try-except block?