Name: DHWANI GUPTA

Enrollment Number: EBEON0223761985

Exception Handling Theory Questions

1.What is an exception?

- An exception is an event, which occurs during the execution of a program which disturbs the normal flow of the program's instructions
- Basically, it is a python object which represents an error.
- When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, program will crash.

2. Can you explain the difference between errors and exceptions?

The difference between errors and exceptions are listed below in the table: -

Parameters	Errors	Exception
Definition	An error is a problem that prevents the program from running correctly.	An exception is a problem that occurs during program execution but can be handled by the program.
Program Crashing	Errors can be caused by syntax mistakes, logical mistakes, or external factors such as hardware or network failures. They often result in the program crashing or not functioning as intended, and they need to be fixed by the programmer before the program can run correctly.	Exceptions are events that occur during program execution that disrupt the normal flow of the program. They can be caused by unexpected inputs, system errors, or other runtime conditions. Exceptions can be handled by the program using try-except statements, which allow the program to catch the exception and take appropriate action instead of crashing.
Summary	Errors are problems that prevent a program from running correctly.	Exceptions are problems that occur during program execution but can be handled by the program. Handling exceptions allows programs to gracefully handle unexpected conditions and continue running without crashing.

3. How can you ensure that a certain code block runs no matter whether there's an exception or not?

- We can ensure it with "**finally**" clause. A finally clause is always, whether an exception has occurred or not.

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

Some examples of built-in exceptions are listed below: -

1) **TypeError**: It is a built-in exception in Python that is raised when an operation or function is applied to an object of inappropriate type. For example, if you try to concatenate a string and an integer using the '+' operator, a TypeError will be raised because the two objects are not of the same type and cannot be concatenated.

2) **ValueError**: It is a built-in exception in Python that is raised when a function or operation receives an argument that has the right type but an inappropriate value.For example, if you try to convert a string to an integer using the int() function, but the string is not a valid integer, a ValueError will be raised.

3) **NameError**: It is a built-in exception in Python that is raised when a name is not found in the local or global namespace. This usually occurs when you try to reference a variable or function that has not been defined or declared. For example, if you try to print the value of a variable that has not been defined, a NameError will be raised.

4) **ZeroDivisionError**: It is a built-in exception in Python that is raised when an attempt is made to divide a number by zero. For example, if you try to divide a number by zero using the '/' operator, a ZeroDivisionError will be raised

5) **IndexError**: It is a built-in exception in Python that is raised when an index is out of range for a given sequence or list. For example, if you try to access an element in a list using an index that is greater than or equal to the length of the list, an IndexError will be raised.

6) **KeyError**: It is a built-in exception in Python that is raised when you try to access a dictionary key that does not exist in the dictionary. For example, if you try to access a value in a dictionary using a key that is not present in the dictionary, a KeyError will be raised.

7) **AttributeError**: It is a built-in exception in Python that is raised when an attribute or method is not found for a given object. For example, if you try to access an attribute or method of an object that does not exist, an AttributeError will be raised.

8) **IOError**: It is a built-in exception in Python that is raised when an input/output operation fails, such as when a file cannot be opened or written to. For example, if you try to open a file that does not exist, an IOError will be raised.

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

In Python, you can use a try/except/finally block to handle exceptions that might occur in your code.Here is basic syntax explaining try/except/else/ finally: -

1. try: → Run the code. Checks whether there is an exception or not in the code.
2. except: → Execute the code when there is an exception.
3. else: → Runs when No exception.
4. finally: → Always run, irrespective of exception occurred or not.

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 keyword** is used to raise an exception. An exception is a type of error which occurs during the execution of a program, and the raise keyword is used to signal that an error has occurred and to specify the type of error.

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

- Else block is an optional block along with the try and except block, which runs when there is no exception in the program. It tells the difference between the normal flow and the exceptional flow of the program.

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

The syntax for try and except block is as following: -

```
try:
    # Code that might raise an exception
    # ...
except ExceptionType1:
    # Code to handle ExceptionType1
    # ...
except ExceptionType2:
    # Code to handle ExceptionType2
    # ...
else:
    # Optional: Code to execute if no exception is raised
    # ...
finally:
    # Optional: Code that always runs, regardless of exceptions
    # ...
```

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?

- Multiple except block in try-except block example: -

```
import sys
import os
try:
    x = int(input('Enter x.: '))
    y = int(input('Enter x.: '))
    print(x/y)
    os.remove('abc.txt')

except ZeroDivisionError:
    print(sys.exc_info())
except:
    print(sys.exc_info())
else:
    print('Exception not occurred')
finally:
    print('Program is completed')
```

```
Enter x.: 45
Enter x.: 5
9.0
(<class 'FileNotFoundError'>, FileNotFoundError(2, 'The system cannot find the file specified'), <traceback object at 0x0000021
822989600>)
Program is completed
```

Explanation: In this Program, In the beginning, user input is asked. In addition to that, by using divide (/) operator, have divided the x and y value. Furthermore, a file is removed. First except block is for **ZeroDivisionError**, Second Except block is for **FileNotFoundError**. In addition to that, else and finally blocks are also added. Now if there is **ZeroDivisionError** then first except block is executed and if **FileNotFoundError** is there is the block as shown in the figure up there, the second except block is executed. And if no exception is there in the code, else block is executed. Finally block is always executed, irrespective exception is there or not. It runs hierarchal. **Default error** should be written in the last, else the code will give error.