Assignment

1. What is the role of try and exception block?

- The role of a try-except block, also known as a try-catch block in some programming languages, is to handle and manage exceptions or errors that may occur during the execution of a program.

2. What is the syntax for a basic try-except block?

```
try:
    # Code that may cause an exception
    # ...
except ExceptionType:
    # Exception handling code
    # ...
```

3. What happens if an exception occurs inside a try block and there is no matching except block?

- If an exception occurs inside a try block and there is no matching except block to handle that specific exception, the program will terminate with an unhandled exception error. This error message will display information about the exception that occurred, such as the type of exception and a traceback indicating the line of code where the exception was raised.

4. What is the difference between using a bare except block and specifying a specific exception type?

bare except block: This means that the except block will handle any exception that occurs, regardless of the type.

Using a bare except block can be useful when you want to provide a generic error message or perform some common cleanup tasks for any unexpected exception.

specifying a specific: This allows you to provide tailored exception handling for that specific exception type. You can write code that specifically addresses the error condition associated with that exception.

Handling specific exception types helps you differentiate between different error scenarios and apply appropriate error handling strategies accordingly.

5. Can you have nested try-except blocks in Python? If yes, then give an example.

- Yes, you can have nested try-except blocks in Python. This means that you can have a try block within another try block, and each try block can have its own corresponding except block(s). This nesting allows for more granular exception handling and allows you to handle exceptions at different levels of your code.

6. Can we use multiple exception blocks, if yes then give an example.

Yes, you can use multiple except blocks to handle different types of exceptions. This allows you to specify separate exception handling code for each specific exception type that you anticipate in your code.

7.Write the reason due to which following errors are raised:

a. EOFError: This error is raised when the end of a file or input stream is reached unexpectedly. It typically occurs when a function or method is trying to read data from a file or input stream, but there is no more data to read.

b. FloatingPointError: This error occurs when a floating-point operation, such as division by zero or an invalid mathematical calculation, produces an undefined or non-representable result. It usually indicates an issue with floating-point arithmetic.

c. IndexError: This error is raised when trying to access an index of a sequence (such as a list, tuple, or string) that is out of range or doesn't exist. It occurs when attempting to access an element at an invalid index position.

d. MemoryError: This error occurs when an operation fails due to insufficient memory available to perform the requested action. It typically happens when a program tries to allocate more memory than the system can provide.

e. OverflowError: This error is raised when a mathematical operation exceeds the maximum representable value for a numeric type. It occurs when the result of an arithmetic calculation is too large to be stored within the available memory or exceeds the predefined limits of the numeric type.

f. TabError: This error is raised when there are inconsistencies in the usage of tabs and spaces for indentation in Python code. It typically occurs when mixing tabs and spaces in the indentation of a block, which leads to indentation errors.

g. ValueError: This error is raised when a function or operation receives an argument of the correct type but an inappropriate value. It occurs when the input provided to a function or method is not valid or doesn't conform to the expected format or range.

8. Write code for the following given scenario and add try-exception block to it.

a. Program to divide two numbers::

```
try:
    dividend = int(input("Enter the dividend: "))
    divisor = int(input("Enter the divisor: "))
    result = dividend / divisor
    print("Result: ", result)
```

```
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
```

b. Program to convert a string to an integer

```
try:
    num_str = input("Enter a number: ")
    num = int(num_str)
    print("Number:", num)
except ValueError:
    print("Error: Invalid input. Please enter a valid number.")
```

c. Program to access an element in a list

```
try:
    my_list = [1, 2, 3, 4, 5]
    index = int(input("Enter the index: "))
    element = my_list[index]
    print("Element:", element)
except IndexError:
    print("Error: Index out of range.")
```

d. Program to handle a specific exception

```
try:
```

```
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)
except ValueError:
    print("Error: Invalid input. Please enter numbers only.")
```

e. Program to handle any exception

```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)
except Exception as e:
    print("An error occurred:", str(e))
```