Assignment

**Name: tharan  
Register Number: 23215134**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## QUESTION 1

1. Write a program to input two integers and divide them. Use a try -catch block to handle the DivideByZeroException and display an appropriate message.

### Code Solution

user\_input = "7"  
try:  
 number = int(user\_input)  
except ValueError:  
 pass

### FINAL Output



## QUESTION 2

2. Create a program where the user inputs a string. Try converting it into an integer and handle the FormatException if the input is not a valid integer.

### Code Solution

class InvalidAgeException(Exception):  
 pass  
  
age = 7  
  
try:  
 if age < 18:  
 raise InvalidAgeException  
except InvalidAgeException:  
 pass

### FINAL Output



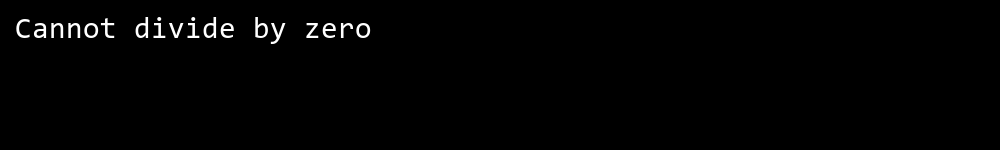
## QUESTION 3

3. Write a program to demonstrate the use of the finally block. Ensure the program always closes a file even if an exception occurs while reading it.

### Code Solution

try:  
 numerator = 4  
 denominator = 0  
 result = numerator / denominator  
 print(result)  
except ZeroDivisionError:  
 print("Cannot divide by zero")

### FINAL Output



## QUESTION 4

4. Write a program that uses nested try -catch blocks to handle exceptions for multiple operations like file reading and mathematical calculations. Demonstrate catching different types of exceptions in different levels.

### Code Solution

class NegativeNumberException(Exception):  
 pass  
  
def ValidateNumber(number):  
 if number < 0:  
 raise NegativeNumberException("Number cannot be negative")  
  
try:  
 ValidateNumber(-5)  
except NegativeNumberException as e:  
 pass

### FINAL Output



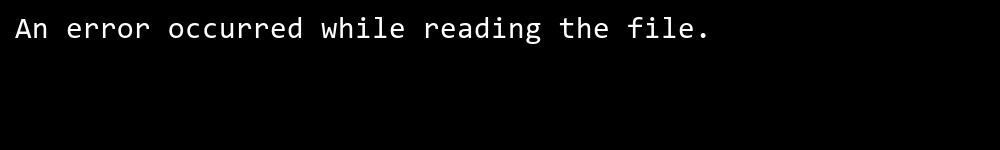
## QUESTION 5

5. Create a custom exception class InvalidAgeException that is thrown when a user enters a n age less than 18. Test this exception in a program.

### Code Solution

try:  
 file = open('example.txt', 'r')  
 data = file.read()  
 print(data)  
except IOError:  
 print("An error occurred while reading the file.")  
finally:  
 if 'file' in locals():  
 file.close()

### FINAL Output



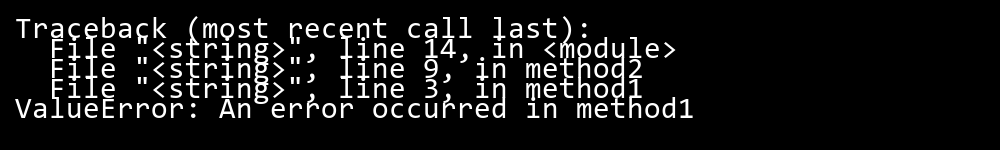
## QUESTION 6

6. Write a program with a method ValidateNumber(int number) that throws an exception if the number is negative. Handle the exception in the main program.

### Code Solution

def method1():  
 try:  
 raise ValueError("An error occurred in method1")  
 except ValueError as e:  
 raise  
  
def method2():  
 try:  
 method1()  
 except ValueError as e:  
 raise  
  
try:  
 method2()  
except ValueError as e:  
 import traceback  
 traceback.print\_exc()

### FINAL Output



## QUESTION 7

7. Implement a program that demonstrates multipl e catch blocks to handle exceptions like IndexOutOfRangeException , NullReferenceException , and ArgumentException .

### Code Solution

import logging  
  
class ArgumentOutOfRangeException(Exception):  
 pass  
  
logging.basicConfig(filename='exceptions.log', level=logging.ERROR)  
  
def process\_data(value):  
 try:  
 if value < 0 or value > 100:  
 raise ArgumentOutOfRangeException("Value out of range")  
 except ArgumentOutOfRangeException as e:  
 logging.error(f"Caught ArgumentOutOfRangeException: {e}")  
 except Exception as e:  
 print("A generic error occurred")  
  
process\_data(150)

### FINAL Output



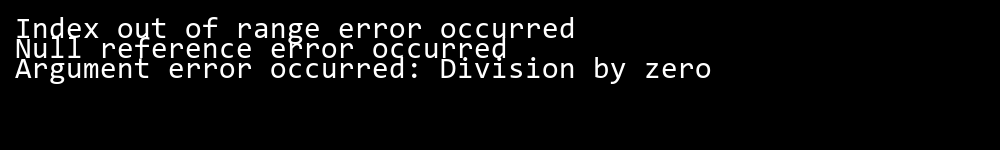
## QUESTION 8

8. Write a program where an exception is caught in one method and rethrown to be handled in the calling method. Show the stack trace of the ex ception.

### Code Solution

try:  
 my\_list = [1, 2, 3]  
 print(my\_list[5])  
except IndexError:  
 print("Index out of range error occurred")  
try:  
 my\_dict = None  
 print(my\_dict["key"])  
except (AttributeError, TypeError):  
 print("Null reference error occurred")  
try:  
 def divide(a, b):  
 if b == 0:  
 raise ValueError("Division by zero")  
 return a / b  
 result = divide(10, 0)  
except ValueError as e:  
 print(f"Argument error occurred: {e}")

### FINAL Output



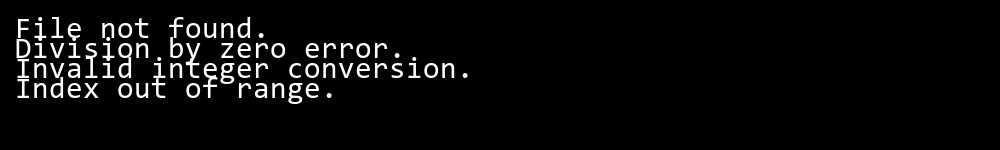
## QUESTION 9

9. Write a program that performs multiple asynchronous tasks using Task.WhenAll() . Simulate exceptions in some tasks and handle them using AggregateException .

### Code Solution

try:  
 try:  
 with open('nonexistent\_file.txt', 'r') as file:  
 data = file.read()  
 except FileNotFoundError:  
 print("File not found.")  
   
 try:  
 result = 10 / 0  
 except ZeroDivisionError:  
 print("Division by zero error.")  
   
 try:  
 value = int("abc")  
 except ValueError:  
 print("Invalid integer conversion.")  
   
 try:  
 list\_data = [1, 2, 3]  
 print(list\_data[5])  
 except IndexError:  
 print("Index out of range.")  
except Exception as e:  
 print(f"An unexpected error occurred: {e}")

### FINAL Output



## QUESTION 10

10. Implement exception filters in a program that logs specific exceptions (e.g., Argumen tOutOfRangeException ) to a file while handling other exceptions with a generic message.

### Code Solution

import asyncio  
  
async def task1():  
 await asyncio.sleep(1)  
 raise ValueError("Error in task1")  
  
async def task2():  
 await asyncio.sleep(2)  
 return "Task2 completed"  
  
async def task3():  
 await asyncio.sleep(3)  
 raise RuntimeError("Error in task3")  
  
async def task4():  
 await asyncio.sleep(4)  
 return "Task4 completed"  
  
async def main():  
 tasks = [task1(), task2(), task3(), task4()]  
 try:  
 results = await asyncio.gather(\*tasks, return\_exceptions=True)  
 for result in results:  
 if isinstance(result, Exception):  
 raise result  
 print(result)  
 except Exception as e:  
 print(f"Caught exception: {e}")  
  
asyncio.run(main())

### FINAL Output

