PYTHON SESSIONAL
ASSIGNMENT 3

1. The Basic Try-Except

Write a program that asks the user for two numbers and divides the first by the second. Use a try-except block to catch and handle any ZeroDivisionError, printing a friendly error message like "Error: You cannot divide by zero!" instead of letting the program crash.

#CODE START

```
# Program to divide two numbers with error handling

try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    # Division
    result = num1 / num2
    print("Result:", result)

except ZeroDivisionError:
    print("Error: You cannot divide by zero!")
```

OUTPUT

```
Enter the first number: 60
Enter the second number: 12
Result: 5.0
```

2. Handling Multiple Exceptions

Modify the program from question 1. Now, also handle a ValueError that occurs if the user enters something that isn't a number (e.g., a letter). Print an appropriate message for this error as well (e.g., "Error: Please enter only numbers.").

#CODE START

```
# Program to divide two numbers with error handling

try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
```

```
    result = num1 / num2
    print(f"The result of division is: {result}")

except ZeroDivisionError:
    print("Error: You cannot divide by zero!")

except ValueError:
    print("Error: Please enter only numbers.")
```

OUTPUT

Enter the first number: 23

Enter the second number: 0

Error: You cannot divide by zero!

3. The Else Clause
Extend the division program further. Use a try-except-else block. If the division is successful (no exception is raised), use the else clause to print the result with a message like "The result is: [result]".

#CODE START

```
# program to divide two numbers with try-except-else

try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    result = num1 / num2

except ZeroDivisionError:
    print("Error: You cannot divide by zero!")
```

```
except ValueError:
    print("Error: Please enter only numbers.")

else:
    print(f"The result is: {result}")
```

OUTPUT

Case 1

Enter the first number: 157

Enter the second number: 14

The result is: 11.214285714285714

Case 2

Enter the first number: 12

Enter the second number: 0

Error: You cannot divide by zero!

4. The Finally Clause

You are reading a number from a file called data.txt.

- Inside a try block, open the file, read the contents, and convert it to an integer.
- Handle a FileNotFoundError if the file doesn't exist.
- Handle a ValueError if the file's content can't be converted to an integer.
- Use a finally block to print a message saying "File operation attempted." This message should print **whether an exception occurred or not**.

#CODE START

Case 1

Enter a number to save in the file: 12

Number 12 saved to 'data.txt' successfully.

The number read from file is: 12

File operation attempted.

PYTHON SESSIONAL
ASSIGNMENT 3

5. Getting Exception Information

Write a program that tries to access the fifth element of the list my_list = [10, 20, 30]. Catch the resulting exception (likely an IndexError). In your except block, print both a custom error message and the standard error message from the exception object itself. (Hint: Use as to assign the exception to a variable).

#CODE START

```
# Program to access the fifth element of a list and handle exceptions


my_list = [10, 20, 30]


try:
    # Try to access the fifth element (index 4)
    element = my_list[4]
    print(f"The fifth element is: {element}")


except IndexError as e:
    # Print custom message and the standard exception message
    print("Error: Tried to access an element that does not exist.")
    print(f"Standard error message: {e}")
```

OUTPUT

```
Error: Tried to access an element that does not exist.
Standard error message: list index out of range
```

6. Raising an Exception

Create a function called check_age(age) that takes an age. If the age is negative, raise a ValueError with a custom message like "Age cannot be negative!". Otherwise, print "Age is valid." Write code to call this function and handle the potential ValueError.

#CODE START

```
# Function to check age
def check_age(age):
    if age < 0:
        # Raise a ValueError if age is negative
        raise ValueError("Age cannot be negative!")
```

```
    else:
        print("Age is valid.")


# Code to call the function and handle the exception
try:
    user_input = int(input("Enter your age: "))
    check_age(user_input)


except ValueError as e:
    print(f"Error: {e}")
```

OUTPUT

CASE 1

Enter your age: 34

Age is valid.

CASE 2

Enter your age: -6

Error: Age cannot be negative!