24/09/20

Task:9 Implement Exceptions and Expectation handling in python

Student marks Validation

Aim:

To write a python program that accepts marks from the user and raises an exception if the entered marks are negative or greater than 100. The program should handle the exception and display an appropriate error message.

Algorithm:

1. Start the program
2. Accept marks as input from the user
3. Convert the input to an integer
4. Check if the marks are less than 0 or greater than 100
   - if true raise a ValueError with a suitable message
5. if valid, display the marks
6. Use a try-except block to handle any ValueError raised
7. End the program

Program:

```
try:
    marks = int(input("Enter student marks : "))
    if marks < 0 or marks > 100:
        raise ValueError("marks should be bet 0 and 100")
    print("marks entered successfully:", marks)
except ValueError as e:
    print("Error :", e)
```

Result:

The program successfully validates the student mark and handles exceptions for invalid or out-of-range inputs

Sample output:

Case 1: valid input

Enter student marks: 85

marks entered successfully: 85

Case 2: invalid input

Enter student marks: 120

Error: marks should be between 0 and 100

Case 3: negative marks

Enter Student marks: -5

Error: marks should be between 0 and 100

b. Division calculator with Exception

Aim:

To write a python program that accepts two numbers from the user and perform division while handling exception such as division by zero and invalid input

Algorithm:

Start the program
Accept two numbers from the user numerator and denominator
use try-except blocks to handle possible errors
Convert both inputs to float
Perform devision and display the result
Handle the following exception:
zero Devision ~~Error~~ Error if the denominator is zero
value Error if the ipu is not a number
End the program

```
try:
    num1 = float(input("Enter numerator: "))
    num2 = float(input("Enter denominator: "))
    result = num1/num2
    print("Result of division:", result)
Except ZeroDivisionError:
    print("Error: Division by zero")
Exept ValueError:
    print("Error: Please enter a numeric value")
```

Result:

The program correctly performs division and handles ZeroDivisionError and ValueErrors Exceptions effectively, ensuring the program does not crash for invalid inputs

Sample output:

Case 1: valid input:

Enter numerator: 10

Enter denominator: 2

Result of division: 5.0

Case 2: Division by zero

Enter numerator: 8

Enter denominator: 0

Error: Division by zero is not allowed

Case 3: Invalid input:

Enter numerator: ten

Enter denominator: 5

Error: Please enter valid numeric value

Sample output:

Case 1: valid input:

Enter numerator: 10
Enter denominator: 2
Result of division: 5.0

Case 2: Division by zero

Enter numerator: 8
Enter denominator: 0
Error: Division by zero is not allowed

Case 3: Invalid Input:

Enter numerator: ten
Enter denominator: 5
Error: Please enter valid numeric value