**📝 Day 12 – Home Assignments: Exception Handling – try, except, finally**

**🔹 Part A: Core Try-Except Practice**

1. **Basic Input Validation**
   * Ask the user to enter a float number.
   * Catch ValueError and print "Invalid input. Please enter a decimal number."
2. **String to Integer Conversion**
   * Take a string input and try converting it to an integer.
   * Catch exceptions and print the type of error using type(e).
3. **List Index Validation**

colors = ["red", "blue", "green"]

* + Ask user to input an index and print the color at that index.
  + Catch IndexError.

**🔹 Part B: Working with Multiple Errors**

1. **Math and File Operations Combined**
   * Ask user to input a number and a file name.
   * Perform division 100 / number and open the given file.
   * Catch ZeroDivisionError and FileNotFoundError in separate except blocks.
2. **Nested Try Blocks**
   * In outer try: convert user input to integer.
   * In inner try: divide a fixed number by the input.
   * Catch errors in appropriate places and explain with print statements.

**🔹 Part C: Finally Block Use**

1. **Simulate Locking/Unlocking**

lock = False

* + In try: set lock = True and simulate an error (e.g., 1 / 0)
  + In finally: set lock = False and print "Resource released".

1. **File Always Closes**
   * Open a file (use open("sample.txt", "w")) in try.
   * Write to it, raise an error manually using raise
   * In finally, close the file and print "File closed".

**🔹 Part D: Real-World Inspired Tasks**

1. **Student Marks Entry**
   * Ask the user to enter a student’s mark (0–100).
   * If out of range, raise ValueError.
   * Catch the error and print "Invalid mark. Must be between 0 and 100."
2. **User Registration**
   * Ask for username and password.
   * If username is empty, raise ValueError.
   * If password is less than 6 characters, raise ValueError.
   * Catch and display both errors.
3. **Simple File Reader with Fallback**
   * Ask user for a file name.
   * If file doesn’t exist, catch FileNotFoundError and create a new file with default content.
   * Print "New file created."
4. **Custom Calculator**

* Take two numbers and an operator as input.
* Perform the calculation inside try.
* Handle:
  + ZeroDivisionError
  + ValueError for bad input
  + Exception for invalid operators

1. **ATM Simulator**

* Initial balance = 5000
* Ask user to withdraw an amount.
* If amount > balance, raise an exception with message "Insufficient balance".
* Always print "Transaction ended" in finally.

1. **Loop Until Success**

* Ask the user to enter an integer.
* If invalid, catch the exception and repeat the question until the input is valid.