CSE 1321L: Programming and Problem Solving I Lab

Fall 2025

Lab 12

Module 6

What students will learn

- Handle Exception by using Try and Except blocks.

Content

Overview

Through this lab, you will practice using exception handling to make your programs more reliable, by manually addressing potential issues. Exception handling will allow you to handle errors that otherwise would have abruptly stopped or crashed your program.

Final note: *Do not cheat*

If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write code in your future job interviews, so learn it now to secure a high-paying job later.

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statement, etc. As always, user input is shown in **red** and **bold**.

Lab 12A: My Integer Collection

You are going to build a simple program that will prompt the user for 10 numbers and store them in a list. Additionally, the program should make sure the user only enters whole numbers (integers) by handling the ValueError exception.

Requirements:

- Create a function called **input_int()**. You are going to be using this function to take input from the user.
- The **input_int()** function will have one parameter which you will use as the input prompt.
- The function should call the **input()** and convert the input value into an int.
- The function should also use a try block and catch ValueError type exception.
- If the program catches a ValueError exception, meaning that the value the program tried to convert into an integer type failed, the program should print this as an error message:
    - "Please enter a whole number (no decimals), try again."
- The function should keep prompting the user until the user enters a valid whole number value.
- **Main** Program:
    - In your main program, start by creating a list.
    - Prompt the user for 10 numbers and remember to use the **input_int()** function instead of the regular **input()** function.
    - After the user is done inputting 10 numbers, print them.

Sample Output #1:

```
[My Integer Collection]
Enter an integer: 10
Enter an integer: 20
Enter an integer: 30.1
Please enter a whole number (no decimals), try again.
Enter an integer: 0.0
Please enter a whole number (no decimals), try again.
Enter an integer: 30
Enter an integer: 40
Enter an integer: 50
Enter an integer: 60
Enter an integer: 70
Enter an integer: 80
Enter an integer: 90
Enter an integer: 100

These are the numbers you entered:
0. 10
1. 20
2. 30
3. 40
4. 50
5. 60
```

```
6. 70
7. 80
8. 90
9. 100
```

Sample Output #2:

```
[My Integer Collection]
Enter an integer: One
Please enter a whole number (no decimals), try again.
Enter an integer: Two
Please enter a whole number (no decimals), try again.
Enter an integer: 1
Enter an integer: 2
Enter an integer: 3
Enter an integer: 4
Enter an integer: 5
Enter an integer: 6
Enter an integer: 7
Enter an integer: 8
Enter an integer: 9
Enter an integer: .10
Please enter a whole number (no decimals), try again.
Enter an integer: 10

These are the numbers you entered:
0. 1
1. 2
2. 3
3. 4
4. 5
5. 6
6. 7
7. 8
8. 9
9. 10
```

Lab 12B: Division

Create a calculator program that will handle just division operations. This program should make sure the user only enters numerical values and handles whenever the user attempts to perform a division by 0.

Requirements:

- The program should prompt the user to input two numbers and convert these values into a float.
- The program then should perform a division with those two numbers and print the result.
- Lastly, the program should ask the user if they want to perform another division operation.
- The program make use of exception handling to handle two types of exceptions:
  - ValueError to handle errors with the number input type.
  - ZeroDivisionError to handle division by zero errors.
- For each type of exception, the program instead of abruptly stopping, should not stop. Instead, it should print these errors and either continue with the program or re-prompt the user to enter another value:
  - "Please enter numerical values."
  - "We cannot divide by zero."

Sample Output #1:

```
[Division]
Enter first number: 4
Enter second number: 0
We cannot divide by zero.

Do you want to perform another division (Y/N)?: Y

Enter first number: 0
Enter second number: 4
0.0/4.0 = 0.0
Do you want to perform another division (Y/N)?: Y

Enter first number: Four
Please enter numerical values.

Do you want to perform another division (Y/N)?: Y

Enter first number: 4
Enter second number: Zero
Please enter numerical values.

Do you want to perform another division (Y/N)?: Y

Enter first number: 40
Enter second number: 3.15
40.0 / 3.15 = 12.70
Do you want to perform another division (Y/N)?: N
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Lab12A.py
  - Lab12B.py
- Programs (source code files) must be uploaded to Gradescope by the due date.