L09 Tasks

L09-T1: “None” in Python

L09-T2: Exception handling with file read & write

L09-T3: Handling exceptions with user input

L09-T4: Handling multiple exceptions

L09-T5: Matrix

Submission:

- Submit **all** the exercises to Moodle via CodeGrade.

Note:

- If you see some "Code Structure Tests" hidden in CodeGrade but they are not mentioned in the task description, you don't need to worry about them. They are just there to make sure the code is ok.

- Be especially careful with spaces so that your output follows the sample output. Note that also each input-string in the program is ending with '\n'. The reason for this is that it makes the output in CodeGrade more readable.

L09-T1: “None” in Python

In this task, you will work with None type (see material of Week 5).

Write a function named `input_integer()` that prompts the user to enter an integer. Use a `try-except` block to catch the `ValueError` exception that occurs when the user enters a non-integer value. The function should return the entered integer or `None` in case of an exception.

Then write a main program that calculates the sum of a given integers. First, it should ask the user how many integers he/she wants to enter by using `input_integer()`. Then, it uses `input_integer()` to get the required number of integers, and finally it prints their sum.

Below is the structure that you have to use in the function `input_integer()`:

```
def input_integer():
     try:
          …
          return …
     except ValueError:
          …
          return None
```

Please ensure that your program has this function with the provided structure, “Code structure test” will be used to check for this structure.

Example run 1:

```
Enter an integer:
4
Now give 4 integers!
Enter an integer:
1
Enter an integer:
g
Invalid input. Please enter an integer.
Enter an integer:
2
Enter an integer:
3
Enter an integer:
Lahti
Invalid input. Please enter an integer.
Enter an integer:
4
The sum of the entered integers is: 10
```

Example run 2:

```
Enter an integer:
3
Now give 3 integers!
Enter an integer:
1
Enter an integer:
4
Enter an integer:
1
The sum of the entered integers is: 6
```

L09-T2: Exception handling with file read & write

Write a Python program that:

- Reads integers from a text file (one per line) into a list in a subroutine.
- Then, writes the list to another text file in a separate subroutine.
- Handles exceptions related to file processing in each subroutine, with all operations inside the handler. So that possible error situations can be caught.

In the main program:

- Prompt for input and output file names.
- Stop the program if file processing fails. Then, the program should print this error message:

  `"Error processing file 'x.txt'."`

Test files `T2_file_in1.txt` and `T2_file_in2.txt` are on Moodle. They are already in CodeGrade, so please do not upload them.

Note: "Code structure tests" will be used to make sure that exception handling is used correctly, and that other parts of the code work well.

Example run 1 (successful):

```
Enter the name of the file to be read:
T2_file_in1.txt
File 'T2_file_in1.txt' read successfully, 7 lines.
Enter the name of the file to be written:
T2_file_out.txt
File 'T2_file_out.txt' was successfully written.
```

Example run 2 (problem with reading a file):

```
Enter the name of the file to be read:
T2_file_in2.txt
Error processing file 'T2_file_in2.txt'.
```

Example run 3 (problem with writing a file):

```
Enter the name of the file to be read:
T2_file_in1.txt
File 'T2_file_in1.txt' read successfully, 7 lines.
Enter the name of the file to be written:
/var/cannot_write.txt
Error processing file '/var/cannot_write.txt'.
```

L09-T3: Handling exceptions with user input

Write a program to test for various exceptions on user input, especially `ValueError, IndexError, ZeroDivisionError, and TypeError`. Implement your program as a menu-based program. You should make your own subroutine to test every error type.

In the error-testing functions, you should use a structure where you first try to perform the desired operation with typically some lines (about 2 - 4) of code, and if it is not successful, you go to the exception handler. The exception handler should only consider the exception to which this subroutine is devoted to.

- `def valueError():ValueError` is an exception that occurs when a function receives an argument of the correct data type but an inappropriate value. For example, a negative integer is passed to math.sqrt(). Show the results in a format that has 2 decimals.

- `def indexError(): IndexError` typically occurs when referring to a list with an index that does not exist. Therefore, in the subroutine, test with a list of elements [11, 22, 33, 44, 55].

- `def zeroDivisionError():` The number 4 is divided by the given divider. In division calculation, show the results in a format that has 2 decimals.

- `def typeError(): TypeError` occurs, for example, when trying to multiply two strings together.

Note: you **should not** use the if-condition in those functions, **only `try-except` is allowed**. "Code structure tests" will be used to make sure that the code is ok, so please make sure that your program has all the required functions with exact names.

Your program should not crash if something else than an integer is given as a menu entry, see the example run below.

Example run 1:

```
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
1
Give a non-negative integer:
-2
ValueError happened. Non-negative number expected for square
root.
```

```
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
1
Give a non-negative integer:
2
Square root of 2 is 1.41.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
2
Input index 0-4:
6
Got an IndexError with index 6.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
2
Input index 0-4:
2
List value is 33 at index 2.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
3
Enter divider:
0
ZeroDivisionError occurred, divider 0.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
```

```
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
3
Enter divider:
2
4/2 = 2.0.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
4
Enter number:
car
Got TypeError. Two strings cannot be multiplied together.
What do you want to do:
1) Test for ValueError
2) Test for IndexError
3) Test for ZeroDivisionError
4) Test for TypeError
0) Stop
Your choice:
something
ValueError happened. Enter the selection as an integer.
Your choice:
0
See you again!
```

L09-T4: Handling multiple exceptions

Write a program that prompts the user to enter two floating-point numbers. The program should perform division on these numbers and use `try-except` blocks to handle both the division by zero exception and the possibility of the user entering non-numeric input. Round the result to 8 decimals and print it to the screen according to the examples below

Example run 1.

```
Enter the first number:
t
Enter the second number:
3
You must enter valid numbers
```

Example run 2.

```
Enter the first number:
4
Enter the second number:
six
You must enter valid numbers
```

Example run 3.

```
Enter the first number:
4
Enter the second number:
0
You cannot divide by zero
```

Example run 4.

```
Enter the first number:
4.1
Enter the second number:
189.4
The result of 4.1 / 189.4 is 0.02164731
```

L09-T5: Matrix

Write a program that begins by creating the following matrix:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

Your program prompts the user to enter both row and column indices. Let us agree that indexing starts from zero. The program attempts to access the value at the specified position in the matrix. It uses a `try-except` block to handle the `IndexError` in case the user enters row or column indices that are out of bounds. It also includes a `ValueError` handling in case the user doesn't enter valid integers for the indices.

Example run 1.

```
Enter the row index:
1
Enter the column index:
2
Value at position (1, 2): 6
```

Example run 2.

```
Enter the row index:
3
Enter the column index:
1
Error: Index out of bounds. Please enter valid row and column
indices.
```