Week 10

Write and run your programs with IDLE editor. Submit finished programs to CodeGrade. Note that some tasks have several steps (A, B, C, …) in CodeGrade.

IMPORTANT: End each input-command string with a newline symbol \n. For example:
`variable = input("Some text:\n")`

Task 1. We have not discussed the reserved word `None` in Python. `None` is a special value that represents "empty" or "no value". It is also the default return value of a function that does not have any explicit `return` statements. This means that in Python, a function *always* returns something, even if it is `None`.

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.

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.

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:
python
Invalid input. Please enter an integer.
Enter an integer:
4
The sum of the entered integers is: 10
```

Task 2. Write a program asks for the user a file name. It tries to open the file, and it deals with the `FileNotFoundError` exception if the file does not exist. If the file exists, the program prints the content to the screen. If not, it the program prints `"Error: File not found."`

Example run 1.

```
Enter the file name:
testing.txt
Error: File not found.
```

Example run 2.

```
Enter the file name:
python_poem.txt
File content:
In bytes and bits, a serpent's embrace,
Python whispers, with elegance and grace.
From loops to functions, its scales unfold,
A language story, in code, untold.
```

Task 3: 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
```

Task 4: 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.
```

Example run 3.
```
Enter the row index:
1
Enter the column index:
j
Error: Please enter valid integers for row and column indices.
```