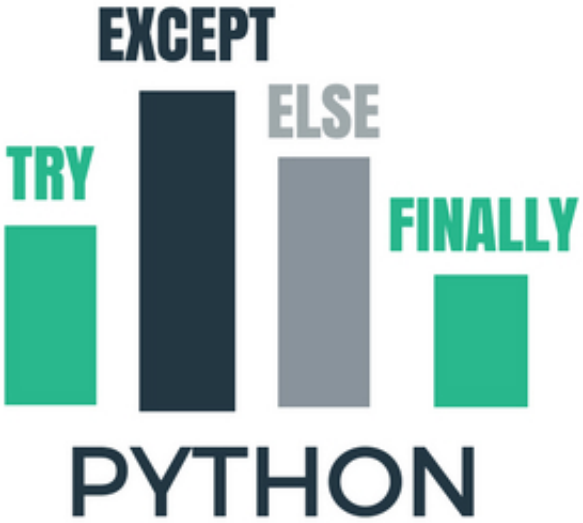
Quiz 5

Subject: Handling Exceptions: Try Not to Go BOOM

TAs:

Due Date: 12.12.2021 23:00



Problem Definition

Given the values *div*, *nondiv*, *from*, and *to*, you are expected to write a Python program which finds the numbers between *from* and *to* (both included) that are divisible by *div* but are not a multiple of *nondiv*, and then compares the obtained result with a list of numbers given for comparison.

Example:
For $div = 2$, $nondiv = 4$, $from = 20$, $to = 32$, and $comparison\_data = 22\ 26\ 30$
the result should be: 22 26 30 with a perfect match to the *comparison_data*.

Simple enough, right? Or is it?

Ask yourself: what could possibly go wrong with your program? As a matter of fact, lots of things could go wrong (e.g. what if *div* or *nondiv* is 0? What if operands are given as something other than a number?), so let's make the problem definition a little bit more specific: **Write the Python program described above so that your program won't go KABOOM no matter what happens with the input.**

Expected Exception Handling

Your program should operate on two input text files:

- **sys.argv[1]** - will provide you the input operands in the following (expected!) format: $div < space > nondiv < space > from < space > to$
  Here, each line of the file should be processed separately, (as each line will contain a set of different operands).
- **sys.argv[2]** - will provide you the data for comparison (each line of this file should be matched to the results of processing the corresponding input line in the first input file).

Hence, your program will expect to be run using the following command (assuming the input file names are **operands.txt** and **comparison_data.txt** respectively):

python3 quiz5.py operands.txt comparison_data.txt

IndexError - For the Missing Expected Command-Line Arguments

The first problem that can occur is if the number of command-line arguments is less then expected. Assume that your code is executed using the following command:

python3 quiz5.py operands.txt

Clearly, one of the expected arguments is missing, and trying to access sys.argv[2] will raise an IndexError exception. Your code must handle the problem of insufficient command-line arguments by printing the error message "**IndexError: number of input files less than expected.**".

IOError

Another problem can occur in case the input files are not reachable at the specified file path (if files do not exist, or if their names are misspelled, etc.). Assume that your code is executed using the following command (and that somefile.txt does not exist):

python3 quiz5.py operands.txt somefile.txt

Your code must handle the resulting IOError exception by printing the error message: "**IOError: cannot open somefile.txt**". Note that the file name which caused the exception must be stated in the error message.

Exception Handling Upon Successful Opening of the Input Files

If the input files are opened successfully, your code should proceed to perform the necessary calculations. The steps you should take are as follows:

- Read a line with the given operands from the first input file.
- Calculate the resulting list of numbers.
- Compare your result to the given comparison data in the second input file (by reading it from the corresponding line; i.e. the results obtained using the operands from the first line in the first input file should be compared to the data given in the first line of the second input file, etc.).

- Repeat these steps for each line of input.

In order to make your program robust against any possible errors that may arise during performing these steps, you are expected to incorporate the following exception handling:

ValueError

If any of the operands are given as non-numeric values, trying to convert them to integers will raise a ValueError exception. Your code must handle this problem by printing the error message "**ValueError: only numeric input is accepted.**", and the given input which caused the trouble (see I/O samples for the expected output format).

IndexError

If the number of operands is less than expected, trying to access them from the list of operands will raise an IndexError exception. Your code must handle this problem by printing the error message "**IndexError: number of operands less than expected.**", and the given input which caused the trouble (see I/O samples for the expected output format).

ZeroDivisionError

If any one of *div* or *nondiv* operands is given as 0, a ZeroDivisionError exception will be raised while performing the calculations. Your code must handle this problem by printing the error message "**ZeroDivisionError: You can't divide by 0.**", and the given input which caused the trouble (see I/O samples for the expected output format).

AssertionError

If everything goes well and you get a result for a set of operands, you will need to **assert** that your result matches the given comparison data. In case they do not match, your code should raise an AssertionError exception and handle it by printing your result, the given comparison data, and the error message "**Assertion Error: results don't match.**" (see I/O samples for the expected output format).

Any Other Exception

Just in case, you should be prepared for any other possible problem that can occur when your code is run, as we cannot always anticipate all future problems. In case of any other error not mentioned above, your code should print the error message "**kaBOOM: run for your life!**".

What If All Goes Well?

If all goes well and your result matches the given comparison data, your program should print your result, the given comparison data, and the following message: "**Goool!!!**" (see I/O samples for the expected output format).

Terminating Your Program

In order to prove that your code didn't blow up on any given input, your program should print the message: "˜**Game Over** ˜" just before it exits. This should be handled in a **finally** block.

Notes

- Do not miss the submission deadline.
- Save all your work until the quiz is graded.
- You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza.
- You must submit your work with the file hierarchy as stated below:

→ <quiz5.py>

Check down for I/O examples

Sample Inputs and Expected Outputs

Consider the following input files:

```
operands.txt:          comparison_data.txt
----------------       ----------------------
2 4 20 32              22 26 30
2 4 20 32              22 26
4 20 32                thisgonnagoboom
0 4 20 32              22 26 30 thisgonnablow
2 tuzak 20 pusu        thisgonnablowtoo
3 6 20 32              21 27
3.2 6 20 32            21 27
3 6 25.5 32            21 27
----------------       ----------------------
```

Run 1:

python3 quiz5.py operands.txt comparison_data.txt

The expected output:

```
------------
My results:              22 26 30
Results to compare:      22 26 30
Goool!!!
------------
My results:              22 26 30
Results to compare:      22 26
AssertionError: results don't match.
------------
IndexError: number of operands less than expected.
Given input: 4 20 32
------------
ZeroDivisionError: You can't divide by 0.
Given input: 0 4 20 32
------------
ValueError: only numeric input is accepted.
Given input: 2 tuzak 20 pusu
------------
My results:              21 27
Results to compare:      21 27
Goool!!!
------------
My results:              21 27
Results to compare:      21 27
Goool!!!
------------
My results:              27
Results to compare:      21 27
```

```
AssertionError: results don't match.

~ Game Over ~
```

Run 2:

python3 quiz5.py operands.txt

The expected output:

```
IndexError: number of input files less than expected.

~ Game Over ~
```

Run 3:

python3 quiz5.py operands.txt nonexistentfile.txt

The expected output:

```
IOError: cannot open nonexistentfile.txt

~ Game Over ~
```