1\_numbers\_dictionary.py

Syntax errors and Exceptions are fixed. The task is fulfilled.

2\_email\_validator.py

I like lines 20 and 23. You are showing different ways of problem solving.

Both files are according PEP8.

That’s the third homework I’m evaluating and it’s the best one.

Excellent homework!!!

ex.1\_numbers\_dictionary.py

Homework is not according to PEP8 – file name contains dot, missing empty lines…

If line 12 was placed between lines 8 and 9, “else” statement could be omitted. I suppose you have used “else” statement from training point of view only. The problem is solved.

ex.2\_email\_validator.py

Homework is not according to PEP8 – file name contains dot, missing empty lines…

If there is more than one ‘@’ line 17 will raise ValueError.

Length is misspelled in “EMAIL\_MIN\_LENGHT”.

The problem is solved.

Good homework!

01\_numbers\_dictionary.py

According to the enquiry - different try-except blocks are used to catch errors. If line 15 was placed on line 10, “else” statement could be omitted. I suppose you have used “else” statement from training point of view only. The problem is solved.

02\_email\_validator.py

Length is misspelled in “USERNAME\_MIN\_LENGHT”.

The problem is solved.

Good homework!

First Task:  
The different try-except blocks, used to catch errors, are applied correctly in various scenarios. For instance, when attempting to convert input to an integer, it raises a ValueError, and so forth. The first task is executed flawlessly!  
  
Second Task:  
I will start with the regex which validates the domain. I like this approach, ensuring it only contains lowercase letters. Instead of regex, you can define a function called, for example, is\_invalid\_domain, which checks if the domain is valid or invalid. However, the approach with regex is 100% correct. I need to give credit for the case that handles multiple '@' symbols in the email; in the real world, if you have to validate an email, this should be considered 100%.  
Overall, excellent homework!