Exercises

Reading from and writing to files is a straightforward, but powerful ability. It helps us automate tasks, generate reports, and run analyses on existing data. Let's practice:

**Exercise 1**

Using this list:

quizzes = [

    {"name": "Guillermo", "quiz1": 80, "quiz2": 85, "quiz3": 82},

    {"name": "Jamie", "quiz1": 78, "quiz2": 72, "quiz3": 80},

    {"name": "Otto", "quiz1": 92, "quiz2": 89, "quiz3": 96},

    {"name": "Christina", "quiz1": 91, "quiz2": 85, "quiz3": 94},

    {"name": "Ceasar", "quiz1": 62, "quiz2": 65, "quiz3": 73},

    {"name": "Barbara", "quiz1": 78, "quiz2": 68, "quiz3": 78},

    {"name": "Rosan", "quiz1": 84, "quiz2": 85, "quiz3": 81},

    {"name": "Marco", "quiz1": 79, "quiz2": 72, "quiz3": 87},

]

Create a new .txt file for each student. The name of the file should be in the format <name>\_final\_report.txt, and should look like this:

Quiz 1: <score>

Quiz 2: <score>

Quiz 3: <score>

----

Average: <calculate-average>

**Exercise 2**

Convert this dict into a JSON file: **Do not** modify the original dict.

COPY

[

    {"name": "Afghanistan", "code": "AF"},

    {"name": "Åland Islands", "code": "AX"},

    {"name": "Albania", "code": "AL"},

    {"name": "Algeria", "code": "DZ"},

    {"name": "American Samoa", "code": "AS"},

    {"name": "Western Sahara", "code": "EH"},

    {"name": "Yemen", "code": "YE"},

    {"name": "Zambia", "code": "ZM"},

    {"name": "Zimbabwe", "code": "ZW"}

]

**Exercise 3**

Read ikea.txt file from [repo](https://github.com/holdbracha/File-IO-Adva-ex), and save its data into both a JSON and a CSV file.

The CSV should look like this:

item, price

closet, 500

hammer, 18

bed, 300

fridge, 800

...

And the JSON should look like this:

[

    {"closet": 500},

    {"hammer": 18},

    {"bed": 300},

    {"fridge": 800},

    ...

]

**Hint**: You might want to use the [split command](https://www.w3schools.com/python/ref_string_split.asp) to "split" the text into two parts: the name and the price. Because all the rows in the .txt file have a - in between the name and price, you can use that as your indicator for where to split.

**Exercise 4**

Use [this CSV](https://github.com/holdbracha/File-IO-Adva-ex/blob/main/test_reports.csv) and [this JSON](https://github.com/holdbracha/File-IO-Adva-ex/blob/main/test_reports.json), and combine them into one .txt file with the following format:

COPY

Report: <report\_name>

Number: <report-count>﻿

-----

Time: <time>

Owner: <owner>

Category: <category> - <sub-category>

FAIL/PASS <status-dependant>

======================================

======================================﻿

Report: <report\_name>

etc...

**Notes**:

* You'll have to save the CSV and JSON data into your own files
* Do not modify either the CSV or the JSON files
* There is no report-count data in the CSV/JSON - you have to generate the count as you go
* You should display *either* FAIL or PASS as the last field; depending on the value of status
* For any empty values, display a \*\*\*

**Extension**

Write a program that will generate csv files with nutrition values according to specific diets.

The grogram will get the data from a json file.

Here is an example json file: [JSON](https://github.com/noizwaves/nutrition/blob/master/data/food.json). **﻿Do not** modify the JSON file.

The program should know how to analyse the data in the JSON file, and save it into a CSV file, saving only the relevant nutrition data.

For example, maybe for a sports diet we would like to focus on protein, and carbohydrate,

so our csv would look like this:

name, protein, carbohydrate

Muesli (Almond), 12.3, 51.7

Wholegrain Rolled Oats, 13.3, 60.3

Almond Milk, 0.5, 4.6

...

For weight loss we might want to focus on fat and sugars,

so our csv would look like this:

name, fat, sugars

Muesli (Almond), 9.9,19.7

Wholegrain Rolled Oats, 9.8, 1.2

Almond Milk,1.2, 4.4

...

Try to write generic code, that will allow easy and safe changes when needed.

**Note**: there is more data in the JSON than you need to save in the CSV, **but** not all of the fields exist for all the data - so make sure you're validating accordingly.