The Final Assignment

Python Online Bootcamp Spring 2022

As part of the final assessment for the Python Bootcamp course you are required to complete the following final assignment which will challenge your understanding of the learning material. This assignment will assess your ability to create Python programs and you will be awarded a certificate of completion if your submission is deemed satisfactory.

Deadline

The deadline for final assignment is **Friday, 24 June 2022, 23:59 PM (Norway time)**.

Marking scheme

The marking scheme for this assignment will follow the same structure as the daily task marking scheme, however there will be a weighting applied to the grade of the requirement.

	Weighting
Does the code run?	20%
Does it meet the performance criteria?	60%
Commenting and Documentation?	20%

Does the code run?

- When your source code is opened it will be executed. **The code must run unassisted**.
- If any runtime exceptions occur marks will be taken from this category.
- If your code does not execute then I will attempt to debug it.
    - If it was due to a trivial error you may still get some marks.
    - If it is a catastrophic error you get zero and you now run the risk of me having to read your code line by line to see if it meets the performance criteria

Does it meet the performance criteria?

- In layman's terms: "Does your code do what it is supposed to do?"
- You will get a mark based on how close your code meets the performance criteria.
- Pay close attention to the required output of the task.
    - If you need to pretty print output, write your own print statements
    - If you need to output to a file make sure it saved in the correct folder
    - ...etc

Commenting and Documentation?

- Classes and methods should have a doc string explaining what it does
- Complicated processes should have an inline comment
- Use identifiers that are “Pythonic”, especially iterators and items
    - I.e. “`for book in books`” ...etc
- If you have code with a known error but you could not fix it in time then tag the problem with `# BUG`: with a comment. If you give me an explanation why you think the code isn't working then I may be more forgiving with the other marking criteria. However you must give me some indication you are aware of the bug.

Guidelines for the doing the assignment

1. **This is your program**. Code it to the practice that best suits you. The performance criteria will tell what the input & output of each objective must be, but the manner in which you do the processing is up to you. If you prefer to use lists: use lists. Want to save your file to .txt? Save it to a .txt. As long as your code does what it is asked to do you will succeed.
2. **If a task needs clarity: ask**. The outcomes for each task should be crystal clear to you. If a task is ambiguous or the language doesn't make sense it is your responsibility to message me on Slack and ask for clarity.
3. **Code consistently**. Try to identify coding patterns when going from task to task on how to solve particular sections. For example if you have an amazing pretty print function that you implemented in one task, copy the code and use it again. OR make a generic function and reuse where needed!
4. **Use what you know**. You are welcome to open your previous daily assignments or the course notes to remind yourself how to do certain tasks again.
5. **When in doubt: comment**. If you are unsure if the outcome is perfect or your code has a small bug in it and you’re unsure how to fix it, use comments to tell me what's going on! This way it helps me understand what you’re thinking and will guide me on what mark to give you.
6. **Communicate with your peers**. You are welcome to talk to your peers and collaborate in understanding the problems and creating a solution. But…
7. **Code your own work**! Do NOT copy code from someone else’s final assignment. This ties back to point number 4: use what you personally know to solve the problem. If your peer is using a dict to do a task but you’re more familiar with a list then use a list.
8. **Always use persistent data where possible.** If an existing file with data is on your workstation, load its content into memory rather than simply overriding it.

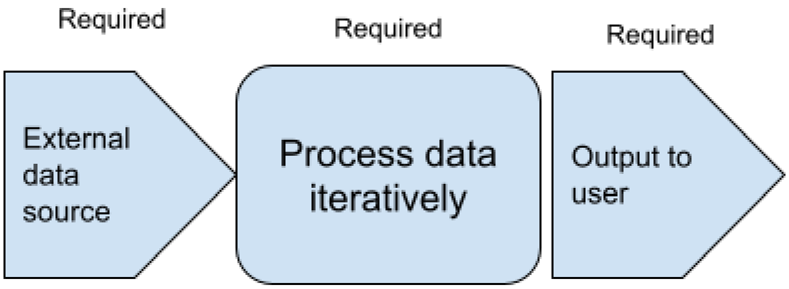
Below is the start of the assignment. Read carefully and good luck!

Final Assignment: A script with a goal

Your task is to create a Python microscript that helps automate a single task you have to do daily. Create a single .py file that can run by simply double clicking an icon on a desktop computer. Your script may be as complicated or simple as you desire. You must demonstrate that your program does the following:

1. Read in data from at least one external source. This can be a file on your computer's harddrive or data scraped from a website.
2. Perform data processing in an iterative manner, using a loop or other control structure.
   a. You **must** have at **minimum one loop or data aggregation** to fulfill this requirement
3. Export some processed information to the user of your program. This can be an output file, command line print, or otherwise.

This can be visualized as the following below:



You are welcome to use any data source you use on a daily basis, as long as it does not infringe on any data privacy policies (see last section of this document). Here are some helpful hints to help you guide your decision making process:

- We have learned many ways to perform file input throughout the course: file.readlines(), pd.read_excel(), json.load() are good places to start.
- DataFrame and Series objects use aggregations to perform iterative changes to the entire dataset, but if all else fails a for loop or a list comprehension allows for the ultimate control of changing values in a collection.
- Exporting information can also be done in several ways: print(), file.write(), etc. Make sure that your script notifies your user that it has successfully completed its task!
- Test your final program a few times before submitting! Double click the .py file, or run every cell in the notebook. Test, test, test!
- If you are using any modules that require additional installation please provide instructions as part of the submission. A simple text file with explanation is sufficient.

What to hand in

Your assignment must be zipped up and then submitted to the Final Assignment task near the bottom of the course moodle page. Inside your zip file you must have the following:

- Your complete .py or a .ipynb file that can be converted into one
- Any data files need to be read in, such as excel spreadsheets, json files, etc.
- Any additional documentation, such as a readme or how to use instructions.

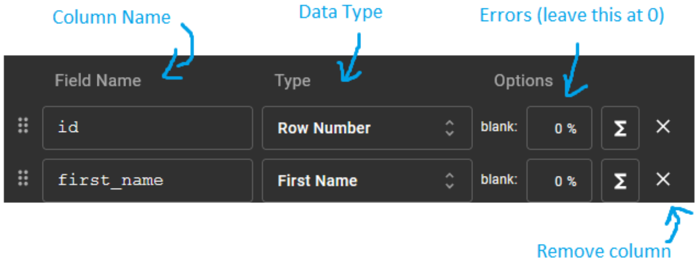
You are welcome to edit your submission before the cutoff date of **24 June 2022: 23:59**.

Data privacy

Although this assignment aims to solve a real world problem you have in your daily life, please DO NOT submit private data that may infringe on the GDPR competency of your workplace or the data privacy of people in your care. An acceptable alternative to using your real data is proxy data, created from an online source. For this I recommend using this website:

- https://www.mockaroo.com/

Mockaroo allows you to create mock real world data of up to 1000 rows in a table. Use this to create proxy data with column headers similar to your real world data. Below is a small example of how to use the website.



Column Name

Data Type

Errors (leave this at 0)

Field Name	Type	Options		
id	Row Number	blank: 0 %	Σ	×
first_name	First Name	blank: 0 %	Σ	×

Remove column

Once you've modeled your mock data, simply click on download at the bottom and use that file instead of your real data. Please include this file into your final submission. If you need help using Mockaroo, feel free to contact me on Slack.