CP5805 Assignment 2
Challenge Tasks

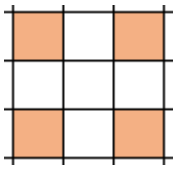
Create a single Python file with the functions requested below.
Submit your challenge solution as a separate file – Lastname_Firstname_A2_challenge.py.

Task 1: Quality control [2 marks]

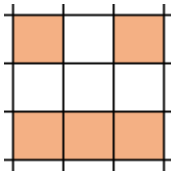
Plastyco manufactures sheets of plastic and has a process that scans for defects in 100×100 grid locations. Sheets with defects can still be salvaged so long as the following conditions are met:

- There are no more than 8 defects overall
- There are no more than 4 defects in any 3×3 square

The image below shows 4 defects in a 3×3 square. A sheet with this set of defects could pass quality control.



The next image shows 5 defects in a 3×3 square; this sheet would be rejected.



Write a function **is_sheet_usable(sheet)**, that takes a numpy array. In the array, if a grid location has a defect, the array will contain 1, otherwise it will contain 0. The function should return True if the pane can be used, otherwise False.

Error checking

If **sheet** is not a numpy ndarray, **raise TypeError**.

If **sheet** has a shape other than 100×100, **raise ValueError**.

If **sheet** contains values other than 0 or 1, **raise ValueError**.

Task 2: Combining CSVs [4 marks]

Suppose we have a folder containing a large number of CSV files. Each CSV should have a column called **key** that should be used as the index. If the file does not have a **key** column, it should be skipped.

Most of the files will have a similar set of columns, with some variations. We need to combine all the data from each file into a single DataFrame.

Write a function **combine_csv_files(directory, output_filename)**.

- **directory** is the folder location where the files can be found
- **output_filename** is where the combined DataFrame should be saved

The function should check through **directory** for all files with a **.csv** extension. Each of these files should be loaded as a DataFrame with **key** as its index. If there are .csv files that cannot be loaded as a DataFrame, these should be skipped. Do not allow the function to crash because there is a .csv file with an invalid format. If a .csv file does not contain a **key** column, it should also be skipped.

The DataFrames should then be **concat**enated vertically into a single DataFrame. E.g., if we had a 10-row frame and a 20-row frame these would produce a 30-row frame.

Reorder the columns in alphabetical order. If any column has more than 50% nulls, it should be dropped. In the case of unmatched columns (in some CSVs but not others), leave values with nulls until/unless these columns get dropped after all the frames have been concatenated. Reorder the rows on their index, then save the DataFrame as a CSV file to the given **output_filename** – include the header, and the index (**key**).

Note: assume both file paths, **directory** and **output_filename**, are relative to the working directory of the code. Do **not** use **os.chdir** or any other method to change the current working directory. Your code does not have to handle issues with the **directory** or **output_filename** – if they are not valid, it is fine for the function to crash.

You have been provided with sample input in the form of a folder of CSV files (zipped for convenience), **csvdir.zip**, and **task2_out.csv** which is the sample output your code should produce from these input files. As part of the challenge of this task, you will need to consider create your own test cases to make sure your function meets all of the requirements above.

Error checking

If a file with a **csv** extension cannot be loaded as a DataFrame, skip it. Do not allow the function to crash.

If **directory** or **output_filename** are invalid, it is acceptable for the function to crash.

Task 3: Creating simple HTML pages [4 marks]

Suppose we want to create some very basic static HTML pages from data in CSV files.

Assume the CSV files are in a folder called **htmlplots**. In this folder is a file called **index.txt** that lists the CSV files that should be turned into HTML pages.

Write a function called **make_html_files()**

For each of these files create an HTML file (.html) that contains:

- A header with the name of the dataset (CSV filename with .csv removed)
- A link to the previous HTML file (unless this is the first)
- A link to the next HTML file (unless this is the last)
- The data in table format (pandas has a function that will help)
- A PNG image of a line plot of the data (matplotlib can save plots to a file)

All the HTML files and corresponding PNG images should be saved in the **htmlplots** folder.

The pages should be linked in the order the CSV files appear in **index.txt**.

An HTML file is just a text file with particular content. Use the HTML files in the sample output as a starting point, but you may 'upgrade' your HTML to something more advanced as long as it provides the same required functionality.

See **htmlplots.zip** for example input and output.

The basic structure of your HTML files should be:

<html><body>
<h1>*name of dataset*</h1>

links for next and previous

HTML code for the table of data
image tag
</body></html>

Error checking

Not required.