Lab 4

Program structure

In this lab, you will add error handling and plotting functionality to an existing program. You will begin from the relatively short program `batch_means.py` (see the folder "Code for labs" on the course site), which could use some improvement. You will find test data in the files `sampleX.csv`, for $X \in \{1, 2, 3, 4\}$, in the same folder.

4.1 Learning goals

- You should be able to follow recommended practices for writing readable code.
- You should be able to rewrite existing code to follow recommended practices.
- You should be able to use exceptions for error handling.
- You should be able to use a library to plot data.

4.2 Submission

The lab should be submitted as usual via PeerGrade.

Submit your solution in a file with the final program. Include comments explaining the changes you have made for each task.

Obs: in this lab, you will use the library `matplotlib`, and you can also use the library `numpy` if desired.

4.3 The initial program

The goal of this program is to read in a data file with (invented) data from several batches of measurements, taken from different points on the plane, and for each batch calculate the average of the measurements taken inside the unit circle. A point (x, y) in the plane is inside the unit circle if $x^2 + y^2 \leq 1$.

Measurements taken outside the unit circle should be ignored. The data file has four columns separated with commas: it is a so-called `csv` file (where "csv" stands for *comma-separated values*). The first number records which batch a measurement belongs to, while the second and third record the x- and y-coordinates where the measurement was taken, and the fourth number is the measurement itself.

For example, a data file could contain the following lines:

```
1, 0.1, 0.2, 73
1, 0.11, 0.1, 101
2, 0.23, -0.01, 17
2, 0.9, 0.82, 23
```

Here we have measurements from two batches, 1 and 2, and so the program will calculate two averages. Note that the last measurement is outside the unit circle.

4.4 Tasks

4.4.1 A: Better structure (3 points)

Your task is to use what you've learned from the document "Basic principles of programming" on the course website and the compendium chapter "Rules of thumb for program structure and good code" to make this code more readable, easier to reuse in other projects, and easier to debug. You should be able to break up the one function that is given into several smaller, more reusable functions.

Test your code with the file `sample1.csv` as well as your own test files.

4.4.1.1 Requirements

- Your version of the program should not remove any functionality from the given program (don't worry about any bugs yet).
- Your functions should be documented with docstrings.
- Identifiers should be descriptive.

4.4.1.2 Tips

- It doesn't matter if your code is longer than the original file.
- Make changes step by step, verifying that the program still works after every change. It is harder to make a large change without affecting functionality.
- Write in a comment how you have improved the program.

4.4.2 B: Error handling (2 points)

If you test the program on the test files `sample2.csv`, `sample3.csv`, and `sample4.csv`, you will discover that there are some issues with `batch_means.py`. For this task, you will add error handling (to your improved program from Task A), using `try-except` so that crashes are avoided and the averages are calculated as well as possible despite issues in the input file. You should not correct the test files, but rather change the program so that it can deal with bad input data.

4.4.2.1 Requirements

- It should be possible to analyze all the given test files without issues.
- If it is impossible to open a filename that the user has entered, the program should point this out to them with a human-readable message rather than by raising an exception.
- If it is not possible to interpret a line of input data, the program should print a warning and ignore the line.
- Note in comments which errors you corrected and how you did it.

4.4.3 C: Sorted batch-data (1 point)

As `batch_means.py` is currently implemented, it will print the averages for each batch in an order determined by the order of lines in the input file. Change the program so that the printed output is sorted, i.e., so that the average for batch 1 is printed before batch 2 and so on. In other words, instead of an execution of the program looking like:

```
Which csv file should be analyzed? sample3.csv

Batch    Average
3        2.0
1        87.0
2        17.0
```

it should look like:

```
Which csv file should be analyzed? sample3.csv

Batch    Average
1        87.0
2        17.0
3        2.0
```

Tip: you probably want to use the function `sorted`. Read more in Python's documentation. Alternatively, you can use your own sorting algorithm from lab 3 (but remember to include it in your submission if you do).

4.4.4 D: Plotting the values (4 points)

Read up on the module `matplotlib`: https://matplotlib.org/stable/tutorials/introductory/pyplot.html.

You can import the library as follows:

```
import matplotlib.pyplot as plt
```

Obs: for this to work, you may first need to install `matplotlib`. You can do this in Anaconda or use the Python package installer `pip`. For installation instructions, see https://matplotlib.org/stable/users/installing.

Any function from `matplotlib.pyplot` can now be accessed by writing `plt.` before its name. For example, to use the `plot(...)` function, you would write `plt.plot(...)`.

Add now the following function:

```python
def plot_data(data,f):
    # here be code
```

This function should plot the data stored in the argument `data` using `matplotlib`. You should not filter out points outside the unit circle, but you should plot the circle itself along with the points. You do not need to plot any averages. The plotted data should then be saved as a PDF in a file `f.pdf` (where `f` is the second parameter to `plot_data`).

The function `plot_data` should then be called in the main program after the averages have been printed, so that an execution of the program could look like so:

```
Which csv file should be analyzed? sample4.csv
```

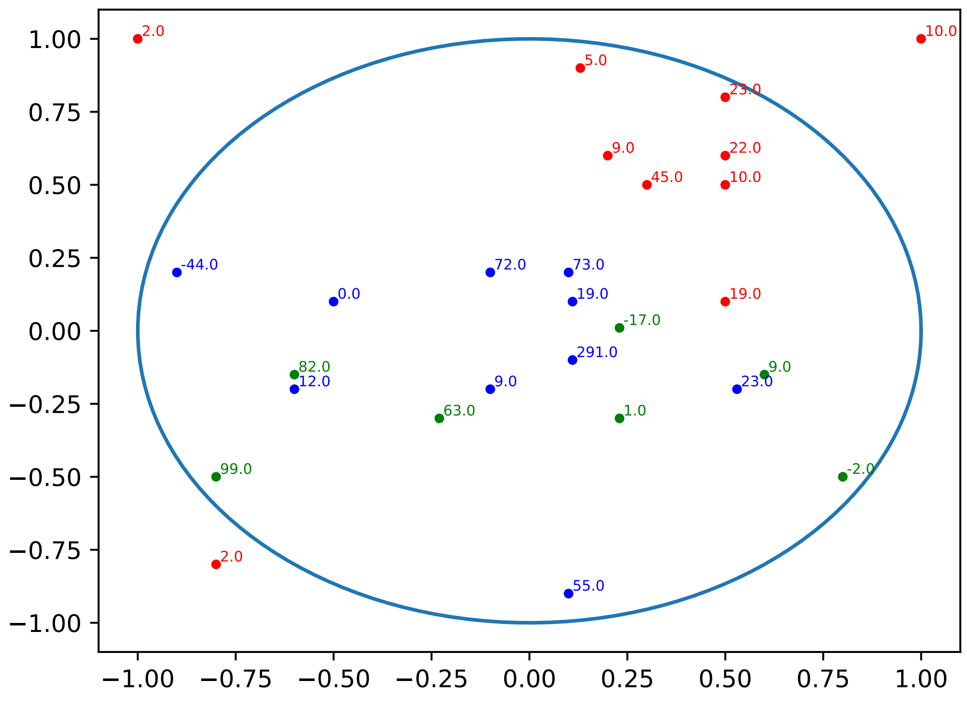
```
Warning: wrong input format for entry: 2  -0.93  -0.01  77

Warning: wrong input format for entry: 2  0.93  -0.01  53

Batch    Average
1        51.0
2        33.57142857142857
3        19.0

A plot of the data can be found in sample4.pdf
```

The file `sample4.pdf` would then contain the following plot:



Note that the unit circle looks more like an oval. This is due to the fact that the *y*-axis is smaller than the *x*-axis. It is possible to fix this, but you are not required to do so.

4.4.4.1 Requirements

- It should be possible to plot all the given test files without issues.
- All points, including those outside the unit circle, should be plotted. You do not need to plot any averages, only the measurements from the input file.

- All measurements in the same batch should be plotted with the same color, and each batch should have its own color. It is OK if you only support a finite number of differently-colored batches; the important thing is that it works for all the test files.
- You should draw the unit circle in the plot. There are many different ways to do so. The `numpy` library may be helpful.

4.4.4.2 Tips

- The type of the `data` parameter to your `plot_data` function will depend on how you solved the previous tasks. You should document its type in the function's docstring.
- For this task, you will find it useful to read documentation and look at examples from the internet.
- In this task, it is completely OK to use code you find online, but you should include a reference in a comment to the place you found the code.
- There are many different ways to draw a circle with `matplotlib`. Search the web for ideas! **Obs:** you are not required to use `numpy` to do this, but you can if you want to.