GRA4157 mid-term

3 October 2025

Exercise 1 — Lists and Dictionaries (4 points)

Consider the list of integers:

```
nums = [4, -2, 15, 0, 8, 23, -7, 9, 5, -3]
```

(a) Write a Python function `count_positive(xs)` that takes such a list as input and returns the number of positive integers in the list. Do not use list comprehensions or the built-in `sum` with conditions; use a loop.

(b) Without running code, determine the outputs of the following calls:

```
print(nums[2:5])
print(nums[-4] + nums[1])
print(nums[::3])
print(nums[::-2])
```

(c) Consider the dictionary

```
grades = {"Alice": 87, "Bob": 92, "Carla": 75, "David": 92, "Eva": 68}
```

Write a function `top_students(d)` that takes such a dictionary and returns a list of all names (keys) that have the maximum grade.

(d) You are given two dictionaries of population numbers (in millions):

```
pop_2020 = {"Norway": 5.4, "Sweden": 10.3, "Denmark": 5.8, "Iceland": 0.36}
pop_2025 = {"Norway": 5.6, "Sweden": 10.6, "Denmark": 6.0, "Finland": 5.6}
```

Write a function `growth(pop20, pop25)` that returns a new dictionary where each key is a country present in both dictionaries, and the value is the difference `pop_2025 - pop_2020`.

Exercise 2 — Reading and Writing Files (3 points)

Consider the input file `data.txt`:

```
Header information
Some text that should be ignored
Matrix starts below:
3 4 5
6 7 8
9 10 11
```

(a) Read the file using pure Python. For each row of numbers, put all values into a list of integers. Collect these lists into a larger list called `matrix`. The final result should look like `[[3,4,5],[6,7,8],[9,10,11]]`.

(b) Write a Python function `sum_rows(mat)` that takes such a matrix (list of lists) as input and returns a new list where each element is the sum of the corresponding row. For the example above, the output should be `[12, 21, 30]`.

(c) Write a program that writes the row sums from part (b) to a file called `sums.csv`. The file should have the header `row,sum`, followed by one row per line, e.g.:

```
row,sum
1,12
2,21
3,30
```

Exercise 3 — Vectorized Computations and Pandas (5 points)

You have been hired by HydroAnalytics AS to analyze sensor data from a hydropower plant. The dataset is stored in a Pandas DataFrame named `data` with the following columns:

```
   timestamp       flow_rate   head   power_output
0  00:00:00        120.0       50.0   5300.0
1  00:00:10        122.0       49.8   5280.0
2  00:00:20        125.0       50.2   5400.0
3  00:00:30        130.0       50.1   5500.0
...
```

Here: - `flow_rate` is measured in m^3/s, - `head` is the water head in meters, - `power_output` is the actual generated power in kW.

(a) According to physics, the theoretical power output is given by

$$P_{theoretical} = \rho \cdot g \cdot Q \cdot H \cdot \eta,$$

where $\rho = 1000$ kg/m^3, $g = 9.81$ m/s^2, Q is flow_rate, H is head, and $\eta = 0.9$ is a constant efficiency factor. Using NumPy vectorized computations, add a new column `p_theoretical` with the theoretical power output.

(b) (2 pts) Compute a new column `efficiency` defined as

$$\text{efficiency} = \frac{\texttt{power\_output}}{\texttt{p\_theoretical}}$$

for each timestamp. Then, using Pandas, identify all time intervals (start and end timestamps) where the efficiency was consistently below 0.85 for at least 30 seconds.

(c) (2 pts) The engineers claim that some rows in the dataset are redundant: if three consecutive rows lie perfectly on the same straight line in the *power_output vs. time* curve, then the middle row can be removed without changing the curve. Write a function that removes all such redundant rows using vectorized computations (no explicit Python for-loops). The resulting DataFrame should be as small as possible while preserving the exact piecewise linear shape of the original curve.

Exercise 4 — Web Scraping and Pandas (5 points)

You will now work with web scraping and Pandas to analyze population data of U.S. states and territories.

(a) Use the `requests` library with an appropriate header (for example `User-Agent: PythonRequests`) to fetch the HTML source from the following page: https://en.wikipedia.org/wiki/List_of_U.S._states_and_territories_by_population

(b) (2 pts) Use `BeautifulSoup` to locate the first `<table>` element in the page. Recall that an HTML table is typically organized with:

- a header row (`<tr>`) containing column names inside `<th>` cells,
- subsequent rows containing data values inside `<td>` cells.

Use the image provided in Figure 1 (file `List_of_US_States.png`) to interpret the table structure. Extract the headers from the first row of the table, and then loop over the remaining rows. For each row, build a dictionary where the keys are the column names (from the header row) and the values are the corresponding entries. Collect all dictionaries into a list.

(c) Convert the list of dictionaries into a Pandas DataFrame. Make sure to clean the data so that numerical columns are stored as numeric types (e.g. integers or floats) rather than strings.

(d) Using the cleaned DataFrame, answer the following:

- Which entry has the largest value in the population column?
- What was the US population in 2020 and 2024, according to the data in the table?
- What is the combined population of all entries that individually account for less than 1% of the U.S. population?

State and territory rankings [edit]

Text

U.S. states and territories by population

Small

	State or territory	Census population[8][9][a]		Change, 2010–2020[9][a]		House seats[b]		Pop. per elec. vote (2020)[c]	Pop. per seat (2020) [a]	% US (2020)	% EC (2020)
		July 1, 2024 (est.)	April 1, 2020	%	Abs.	Seats	%				
1	Alabama	5,157,699	5,024,279	5.12%	244,543	7	1.61%	558,253	717,754	1.499%	1.67%
2	Alaska	740,133	733,391	3.26%	23,160	1	0.23%	244,464	733,391	0.219%	0.56%
3	American Samoa[13]	N/A	49,710	−10.46%	−5,809	1*	—	—	—	0.015%	—
4	Arizona	7,582,384	7,151,502	11.88%	759,485	9	2.07%	650,137	794,611	2.134%	2.04%
5	Arkansas	3,088,354	3,011,524	3.28%	95,606	4	0.92%	501,921	752,881	0.899%	1.12%
6	California	39,431,263	39,538,223	6.13%	2,284,267	52	11.95%	732,189	760,350	11.800%	10.04%
7	Colorado	5,957,493	5,773,714	14.80%	744,518	8	1.84%	577,371	721,714	1.723%	1.86%
8	Connecticut	3,675,069	3,605,944	0.89%	31,847	5	1.15%	515,135	721,189	1.076%	1.30%
9	Delaware	1,051,917	989,948	10.25%	92,014	1	0.23%	329,983	989,948	0.295%	0.56%
10	District of Columbia	702,250	689,545	14.60%	87,822	1*	—	229,848	—	0.206%	0.56%
11	Florida	23,372,215	21,538,187	14.56%	2,736,877	28	6.44%	717,940	769,221	6.428%	5.58%
12	Georgia	11,180,878	10,711,908	10.57%	1,024,255	14	3.22%	669,494	765,136	3.197%	2.97%

Figure 1: Excerpt of the Wikipedia table with U.S. states and territories by population.

Good luck!