Code Test

Problem Statement

You are tasked with writing a program that reads data from a CSV file, processes the data based on predefined conditions, and then performs several number-related calculations. The results should be written to a JSON file, and the input and output should be validated using JSON schema. Additionally, you need to write test cases and provide documentation for your code.

Input

The input CSV file will have the following format: Feel free to create more of such files with more number of records.

```
Name,Number
John,153
Alice,28
Bob,6
```

Predefined Conditions

1. **Extract/Split Values**: Read the CSV data and split it into two lists - one for names and one for numbers.
2. **Armstrong Number**: Determine whether a number is an Armstrong number. An Armstrong number of n digits is an integer such that the sum of its own digits raised to the n-th power is equal to the number itself. For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.
3. **Strong Number**: Determine whether a number is a Strong number. A Strong number is a number such that the sum of the factorial of its digits is equal to the number itself. For example, 145 is a Strong number because 1! + 4! + 5! = 145.
4. **Perfect Number**: Determine whether a number is a Perfect number. A Perfect number is a positive integer that is equal to the sum of its proper divisors (excluding itself). For example, 28 is a Perfect number because its divisors are 1, 2, 4, 7, and 14, and 1 + 2 + 4 + 7 + 14 = 28.

Output

The output JSON file should have the following format:

```
[
  {
    "Name": "John",
    "Number": 153,
    "IsArmstrong": true,
    "IsStrong": false,
    "IsPerfect": false
  },
  {
    "Name": "Alice",
    "Number": 28,
    "IsArmstrong": false,
    "IsStrong": true,
    "IsPerfect": true
  },
  {
    "Name": "Bob",
    "Number": 6,
    "IsArmstrong": false,
    "IsStrong": false,
    "IsPerfect": true
  },
  ...
]
```


Requirements

1. Implement a function or a class that performs the required calculations and writes the results to a JSON file.
2. Use JSON schema to validate the input CSV file and the output JSON file.
3. Write test cases to verify the correctness of your code.
4. Provide clear and well-structured documentation on how to run the code, including any necessary dependencies, and explain any additional features or optimizations you've implemented.
5. The candidate can choose to use Python for this assignment.
6. The candidate should submit their code on GitHub.

Additional Features/Extended Implementations

While not required, candidates can earn extra credit by adding more features or extended implementations. Some ideas include:

- Error handling: Handle cases where the input data is invalid or the file operations fail.

- Command-line interface (CLI): Allow the program to be run from the command line with options to specify input and output file paths.
- Unit tests: Write unit tests to ensure the correctness of your functions.
- Performance optimizations: Optimize your code for efficiency, especially if dealing with large datasets.
- Additional number properties: Check for other number properties like prime numbers, abundant numbers, etc.
- GUI interface: Create a simple graphical user interface to interact with the program.

By designing this coding assessment with a clear problem statement and expectations, you can effectively evaluate a candidate's skills in data manipulation, algorithm design, coding, testing, and documentation.

Resources you can use:

- What is CSV: https://en.wikipedia.org/wiki/Comma-separated_values#:~:text=Comma%2Dseparated%20values%20(CSV),uses%20commas%20to%20separate%20values.
- Working with CSVs in python https://www.geeksforgeeks.org/working-csv-files-python/
- Working with JSONs in python: https://realpython.com/python-json/
- Handling JSONs in python: https://www.geeksforgeeks.org/read-write-and-parse-json-using-python/
- Test cases in python: https://realpython.com/python-testing/
- How to Write Unit Tests for Python Functions: https://www.freecodecamp.org/news/how-to-write-unit-tests-for-python-functions/
- Git and Github for beginners: https://www.freecodecamp.org/news/git-and-github-for-beginners/
- Beginner guide to markdown: https://medium.com/@itsjzt/beginner-guide-to-markdown-229adce30074
- 6 Python best practices: https://www.datacamp.com/blog/python-best-practices-for-better-code
-