Project 5: Multitasking

Due date: Midnight of Wednesday Mar 16, 2022

Project objective:

- Understand and practice multiprocessing and multithreading in Python.
- Be able to apply multitasking solutions based on problem profiles.
- Make programs run faster!

Project overview:

In this assignment, you will revisit a familiar project from CS231. We will write a program to process eight years of Reddit comments to find the most common words overall and in each year. This assignment is intentionally vague to give you room to be creative in your solutions. You will be graded based on:

- Your choice of data structure for each part of the project (you can use any Python library).
- How you apply multiprocessing and multithreading to speed up your code.
- The improvement, in run time, over running the code serially.

Finally, you will write a **report** to showcase your work with text and figures.

The *serial_code.py* file:

In the *serial_code.py* file you will implement a base version of the code that does **not** use any multitasking concepts to speed up the code (appropriate data structure is expected), this code will be used as a base-line for comparison. In this file you are expected to do the following:

- Read in the Reddit comments files
- Count each word
- Print the 10 most **common words** in each file
- Print the frequency of a given word in each year to observe **word trends** (frequency = word_count / number_of_words)
- Time your “common word” and “word trend” code reliably for comparison

The *multitasking_code.py* file:

In the *multitasking_code.py* file you will apply your creative multitasking skills to speed up the code. You have the same expectations as the previous file. Make sure to document the number of processes/threads created and the number of processors on your computer.

Visualization:

Using Matplotlib or a similar library in Python, plot the times of the same operations in the sequential code and the multitasking code. These operations could be (as an example):

- Reading the Reddit files
- Counting words
- Finding the 10 most common words in each file
- Finding the frequency of a given word over the eight years (word trend)

These operations could be different based on how you designed your code.

Report:

Organize your report as follows, maintaining a coherent document that includes screenshots and text to communicate the objective of your project:

1. Abstract: A brief summary of the project, in your own words. This should be no more than 150 words. Give the reader context and summarize the results of your assignment.

2. Results: A section that goes over the code you implemented and the times of each operation. Make sure to document the number of processes/threads created and the number of processors on your computer.

3. Discussion: A section that interprets and describes the significance of your findings focussing on the results.

4. Extensions: Describe any extensions you undertook, including text output, graphs, tables, or images demonstrating those extensions.

5. References/Acknowledgements.

Extensions:

You can be creative here and come up with your own extensions. I will suggest the following ideas:

- Implement a ***visualization*** of how adding processes/threads affects IO-bound and CPU-bound tasks.
- Show a situation where using serial code would be faster than multiprocessing/multithreading.
- Apply the multitasking concepts learned in class to a previous code/project that you have implemented and highlight the improvement in runtime.

Project submission:

- Add all your files (except *.ipynb_checkpoints* and *__pycache__*) to your ***project_5*** directory on Google Drive.
- Copy the rubric from Moodle to the project directory after you review it.