Programming Exercise 6

Concurrent Programming

Background

In this assignment, you will be downloading JPG images for flags from all the countries in the world from a public website. The files appear in the following folder:

https://www.sciencekids.co.nz/images/pictures/flags96/

The names of the files have the following form:

```
country-name.jpg
```

where **country-name** stands for the country name. For example, the JPG for the United States is *United_States.jpg*, and the full URL is *https://www.sciencekids.co.nz/images/pictures/flags96/United_States.jpg*. The country names are in the file *flags.txt* in the Canvas folder for this assignment.

Requirements

You will write three similar scripts that do the following:

- Download all flag files into a local directory named **flags**
- Report the number of bytes downloaded
- Report the elapsed time of the script (using **time.perf_counter()**)

The only difference between the scripts is that for two of them, you will be using concurrent programming. The four versions will use the following respective schemes for downloading:

1. Download sequentially (no concurrency) (G_seq.py)
2. Download concurrently using futures with processes (G_proc.py)
3. Download concurrently using futures with threads (G_thread.py)

Use the requests module HTTP requests. Inspect your file folder after each execution to verify that all the JPGs have been freshly downloaded. Upload your three scripts as .py files and the three output sets (output in .txt format file) both in a zip file. For the output example:

Elapsed time: 10.28958823
3110197 bytes downloaded

Implementation Notes

You may have security issues when sending an email from Python. If you use gmail, you can create an app password if you have 2-step verification enabled (which you should!). Here are the instructions:

Sign in with App Passwords

Tip: App Passwords aren't recommended and are unnecessary in most cases. To help keep your account secure, use "Sign in with Google" to connect apps to your Google Account.

An App Password is a 16-digit passcode that gives a less secure app or device permission to access your Google Account. App Passwords can only be used with accounts that have 2-Step Verification turned on.

When to use App Passwords

Tip: iPhones and iPads with iOS 11 or up don't require App Passwords. Instead use "Sign in with Google."

If the app doesn't offer "Sign in with Google," you can either:

- Use App Passwords
- Switch to a more secure app or device

Create & use App Passwords

If you use 2-Step-Verification and get a "password incorrect" error when you sign in, you can try to use an App Password.

1. Go to your Google Account.
2. Select Security.
3. Under "Signing in to Google," select App Passwords. You may need to sign in. If you don't have this option, it might be because:
    1. 2-Step Verification is not set up for your account.
    2. 2-Step Verification is only set up for security keys.
    3. Your account is through work, school, or other organization.
    4. You turned on Advanced Protection.
4. At the bottom, choose Select app and choose the app you using › Select device and choose the device you're using › Generate.
5. Follow the instructions to enter the App Password. The App Password is the 16-character code in the yellow bar on your device.
6. Tap Done.

Tip: Most of the time, you'll only have to enter an App Password once per app or device, so don't worry about memorizing it.