Assignment 8 (100 points)

Write-up Question 0:

Collaborated with Logan Sherwin

https://www.w3schools.com/python/python_try_except.asp
https://projectgurukul.org/python-website-monitoring/
https://github.com/cms-dev/cms/issues/928
https://stackoverflow.com/questions/1949318/checking-if-a-website-is-up-via-python
https://stackoverflow.com/questions/1949318/checking-if-a-website-is-up-via-python
https://stackoverflow.com/questions/7047790/how-can-i-input-data-into-a-webpage-to-scrape-the-resulting-output-using-python
https://realpython.com/python-web-scraping-practical-introduction/
https://www.crummy.com/software/BeautifulSoup/bs4/doc/
https://www.edureka.co/community/51644/python-unicodedecodeerror-codec-decode-position-invalid
https://stackoverflow.com/questions/51992254/python-get-html-source-code-of-a-web-page
https://www.pythontutorial.net/python-basics/python-write-text-file/#:~:text=First%2C%20open%20the%20text%20file,using%20the%20close()%20method
https://itsmycode.com/python-write-text-file/
https://www.computerhope.com/issues/ch001877.htm

Write-up Question 1 (5 points): What was the final, successful nonce you got?

The final, successful nonce is 70753223.

Write-up Question 2 (5 points): Report the average number of hashes it took for this approach to find a nonce that solves the hash puzzle. How does this compare to the incremental approach?

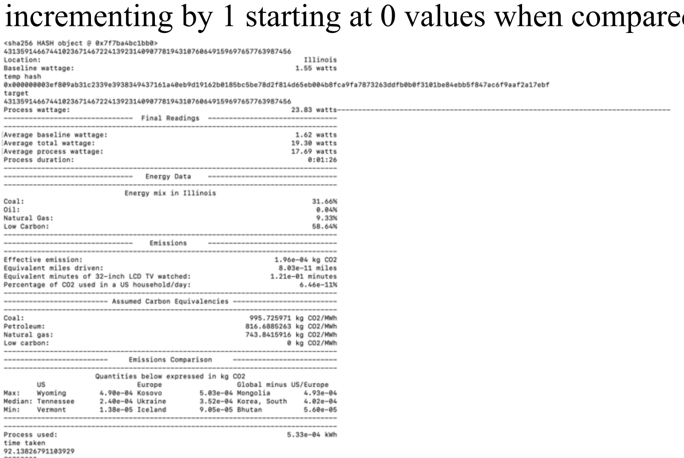
<u>1-2.1:</u>
counter
197625751
nonce
95313520826551582641559185019047944110318025350063715685675343331112874912458
<u>1-2.2:</u>
counter
289109351
nonce
48555054469402444622566957409607370206803895721969329192959042098402375509532
<u>1-2.3:</u>
counter
257035964
nonce
82878981237660283652784866250727768805905783097484484846294664497405562423746
<u>1-2.4:</u>
counter
565589309

nonce
90230703767863719167193982351547149315584568063274049355815222296450177342920
<u>1-2.5:</u>
counter
7362212
nonce
32877000086473309617092979351867821388405842968959121982906720843164545476082

The average number of hashes the program took was 263344517.4. This means that the randomization approach took, on average, 192591294.4 more times to guess the right value when compared to the nonce of the nonrandomized incrementing values (we can find the difference here because the nonce for the incremented value from 1-1 represents the number of times a value was checked as we are incrementing in a non-randomized += 1 way).

Write-up Question 3 (5 points): Would it be more advantageous to a) guess the nonce values randomly, b) guess incrementally with initial_nonce being 1, or c) guess incrementally with initial_nonce being the nonce of the previous block? Make sure to explain why you came to this conclusion. To answer this question, you should consider what you found in the previous questions, the sources we provided, and the fact that each miner (or mining pool) on the blockchain network has their own wallet address (where the bitcoins would go if they receive any).

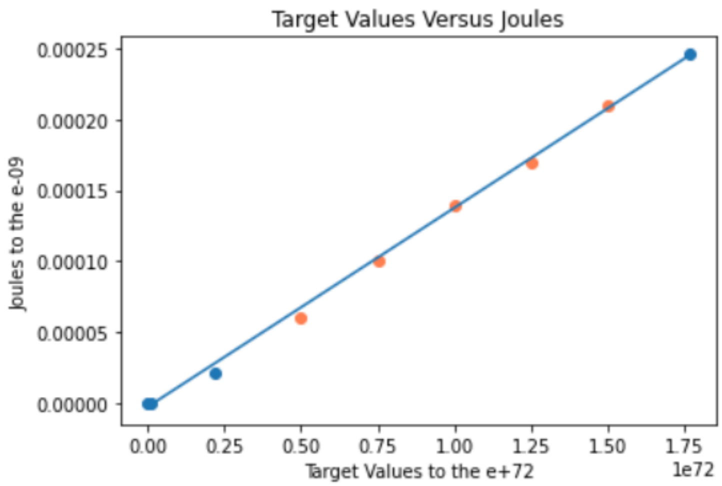
It would be best for a bitcoin miner to guess incrementally with initial_nonce being the nonce of the previous block, assuming that this "previous block" is one where the miner did find bitcoins. This would eliminate the resetting of the incrementing which means that, with a new "beginning" value that is not just set at 0, there will not be rechecking of values that would occur if the value was reset to 0 every time. Additionally, we see how the average number of times that the function loop had to be run through is, on average, smaller when considering the incrementing by 1 starting at 0 values when compared to the random guessing approach.


```
<sha256 HASH object @ 0x7f7ba4bc1bb0>
431359146674410236714672241392314090778194310760649159697657763987456
Location:                                                    Illinois
Baseline wattage:                                          1.55 watts
temp hash
0x000000003ef809ab31c2339e3938349437161a40eb9d19162b0185bc5be78d2f814d65eb004b8fca9fa7873263ddfb0b0f3101be84ebb5f847ac6f9aaf2a17ebf
target
431359146674410236714672241392314090778194310760649159697657763987456
Process wattage:                                          23.83 watts
-----------------------------  Final Readings  -----------------------------
-----------------------------------------------------------------------------
Average baseline wattage:                                  1.62 watts
Average total wattage:                                    19.30 watts
Average process wattage:                                  17.69 watts
Process duration:                                             0:01:26
-----------------------------------------------------------------------------
-----------------------------    Energy Data    -----------------------------
-----------------------------------------------------------------------------
                           Energy mix in Illinois
Coal:                                                          31.66%
Oil:                                                            0.04%
Natural Gas:                                                    9.33%
Low Carbon:                                                    58.64%
-----------------------------------------------------------------------------
-----------------------------     Emissions     -----------------------------
-----------------------------------------------------------------------------
Effective emission:                                    1.96e-04 kg CO2
Equivalent miles driven:                                 8.03e-11 miles
Equivalent minutes of 32-inch LCD TV watched:          1.21e-01 minutes
Percentage of CO2 used in a US household/day:                6.46e-11%
-----------------------------------------------------------------------------
----------------------- Assumed Carbon Equivalencies -----------------------
-----------------------------------------------------------------------------
Coal:                                          995.725971 kg CO2/MWh
Petroleum:                                    816.6885263 kg CO2/MWh
Natural gas:                                  743.8415916 kg CO2/MWh
Low carbon:                                             0 kg CO2/MWh
-----------------------------------------------------------------------------
-----------------------    Emissions Comparison    -----------------------
-----------------------------------------------------------------------------
                    Quantities below expressed in kg CO2
        US                     Europe                  Global minus US/Europe
Max:    Wyoming        4.90e-04 Kosovo         5.03e-04 Mongolia        4.93e-04
Median: Tennessee      2.40e-04 Ukraine        3.52e-04 Korea, South    4.02e-04
Min:    Vermont        1.38e-05 Iceland        9.05e-05 Bhutan          5.60e-05
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Process used:                                               5.33e-04 kWh
time taken
92.13826791103929
70753223
```

Write-up Question 4 (5 points): Report your results in Joules, with scientific notation rounded to three decimal places (e.g. 1.234 x 10^{-2}). How fast was your machine (i.e., how many hashes per second)? If you used your own machine, also briefly describe its specs (e.g., "IBM Thinkpad laptop running Ubuntu 20.04 with an Intel CORE i7 processor"). If you used the machines we set up, say so.

With an average process wattage, we see that 17.69 watts were used in this process with a repetition of 70753223. This means that the wattage is 2.50023946e-7 per hash on average. This means that, when calculating Joules, given the time taken of 92.13826791103929 seconds, the Joules per hash are 0.000023036773321 watts/second. For these calculations and energy usage, I used the Linux.cs set up provided from the instructions on abigailbarnes@cs25910a.

Write-up Question 5 (5 points): In your write-up, include your graph and indicate which points were computed in your experiment and which are estimates.



In the graph, the light blue points are calculated from running 1-5.py in the Linux terminal, whereas the coral points are the estimates (found according to the graphed line of best fit).

Assume every node has your same computational power (from Part 1-4) and the overall Bitcoin network is performing over 220 million trillion (2.2 x 10^{20}) hashes per second according to this graph.

Write-up Question 6 (5 points): How many nodes would the Bitcoin network need to have if making the assumption that everyone is running a machine with equivalent computational power to your computer? How much energy total would be consumed (in watts)?

This would require 286494637006553 (286494637006552.3, to be specific) to carry ou t the required transactions. In total, the energy consumed would be 71630519.65221582 watts

Write-up Question 7 (5 points): Assuming every node is an Antminer S19 (95 Terahashes per second, 3250 Watts), and the same overall rate of hashes per second from the previous question (170 million Terahashes) applies here, how many nodes would we expect the Bitcoin network to have? How much energy total would be consumed (in Watts)?

Considering these new values, the number of required nodes would be 2315790 (23157 89.473684211, to be exact) and the total energy that is consumed in watts would be 7526315789. 473685 watts.

Write-up Question 8 (8 points): The file tranco_L6X4-1m.csv.zip contains a zipped version of the Tranco list of the top million websites. Scraping a million websites would take too long; take what you believe to be an appropriate sample of this list. As your first task, prepare this scrape. You might decide to download all of these pages locally. You might instead decide to complete the rest of the sub-tasks and then run your scrape in real time. Both are valid approaches. In your write-up, briefly describe your approach to sampling the top million webpages.

In sampling the top 1 million web pages, I randomly select approximately 100 to test my code in general. From those 100, my computer kept crashing (probably because I was still running my random nonce functions…), and I decided to take 50 of the urls from that list of 100. Of these 100, I also checked for various errors such as the website being available scraping and also ensuring that the website was online and not throwing 404 errors. I then take these scraped, beautiful-souped web pages and wrote them to text files for storage purposes while also keeping a list to refer to these files in an easy, loop-able way. Finally, I begin my analysis for later questions by looping through each file and pulling the data as necessary.

Write-up Question 9 (8 points): The W3C internationalization guide notes two acceptable ways to indicate that UTF-8 is being used. What fraction of webpages you scraped (i) used the first, shorter method; (ii) used the second, longer method; (iii) indicated some different character encoding; (iv) didn't indicate the character encoding?

In the code, of the 50 pulled websites (and further narrowing down after weeding out errors), I an undeniably working with less data that I would like. From this small sample size of randomized URLs, the proportion of websites that use the shorter, first method is 0.633 and the second, longer method is used with a proportion of 0.367. Finally, the category of neither is seen to be resting at a proportion value of 0.0. Additionally, after analyzing the files, I don't believe any of the files had a different character encoding of the ones I pulled. This would also fall into the category of "none" values.

```
Proportion of Short:
0.6333333333333333
Proportion of Long
0.3666666666666664
Propotion of None
0.0
```

Write-up Question 10 (8 points): The Mozilla Develop Network guide gives a preferred and a discouraged way for denoting the language of a page. What fraction of webpages you scraped (i) used the preferred method; (ii) used the discouraged method; (iii) didn't indicate the language of the page?

Of the 50 selected and further refined URLs according to scrape-ability of the website, the Proportion of the preferred method of denoting language is 0.71, the proportion of the not preferred method of denoting language is 0.00, and the proportion of websites that did not indicate a language in either way was 0.29.

```
Proportion of Prefered:
0.7058823529411765
Proportion of Not Prefered
0.0
Propotion of Not Indicated
0.29411764705882354
```

Write-up Question 11 (8 points): Create a table indicating the fraction of pages in your scrape in different languages, from the most to the least prevalent. Use the actual name of the languages, in addition to the two-character code and its frequency, in your table. Ignore pages that did not specify the language, but be sure to handle any cases where multiple languages were specified for a page. Include this table in your write-up.

```
{'en': 10,
 'no language specified': 10,
 'en-US': 4,
 'ja': 1,
 '': 1,
 'es-ES': 1,
 'fr': 1,
 'ru': 1,
 'it-IT': 1,
 'tr': 1,
 'fa-IR': 1,
 'de': 1,
 'EN-US': 1}
```

Write-up Question 12 (8 points): What were the most common UTF-8 characters themselves used on the pages you visited? Create a table indicating the most common UTF-8 characters in your scrape, from the most to the least prevalent. Pick a sensible cut-off fraction for "most common" since the full list would get really long. Use the unicodedata function to include the official "name" of each character, in addition to the character itself and its frequency, in your table. Include this table in your write-up.

	character name	character sign	occurances
0	SPACE		203721
1	LATIN SMALL LETTER E	e	141207
2	LATIN SMALL LETTER T	t	125984
3	LATIN SMALL LETTER A	a	108175
4	LATIN SMALL LETTER I	i	104272
..	...	...	...
107	CYRILLIC SMALL LETTER EM	м	691
108	LATIN SMALL LETTER A WITH RING ABOVE	å	658
109	CYRILLIC SMALL LETTER U	у	550
110	LATIN SMALL LETTER A WITH TILDE	ã	546
111	LATIN SMALL LETTER I WITH ACUTE	í	528

[112 rows x 3 columns]