Problem 1. Getting starting with Gomoku

You can skip this question if you've successfully gotten started with Gomoku

In the question, you use code similar to what's given to you in `put_sequece_on_board`. This is to help you get started with the Gomoku project.

First, read the code of `put_sequece_on_board`, and make sure you understand it. Talk to a TA if necessary.

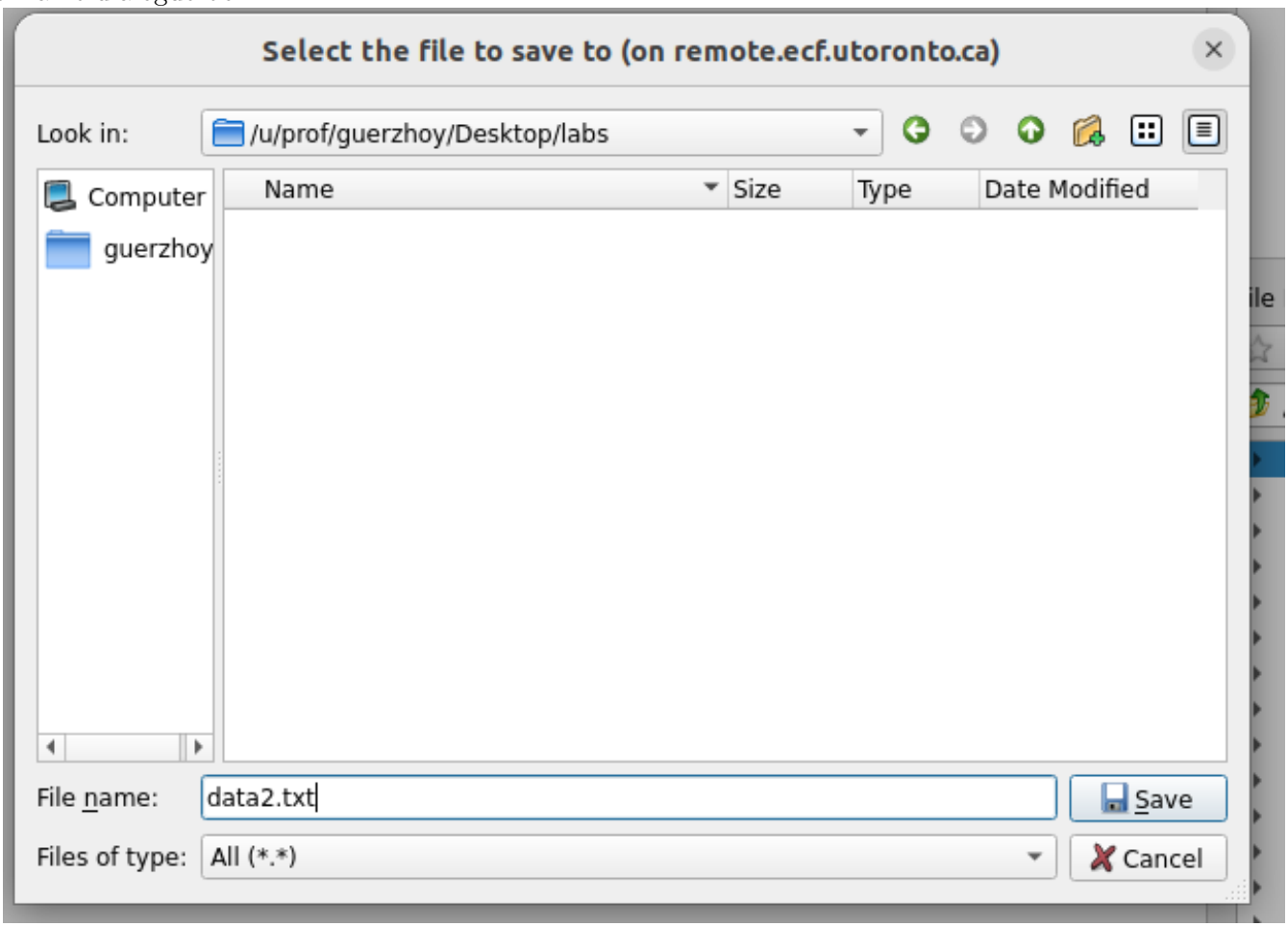
Write a function `is_sequence_complete(board, col, y_start, x_start, length, d_y, d_x)`.

The function should return `True` if there is a sequence of exactly `length` stones starting at location `y_start, x_start` of colour `col`. If there is a stone of colour `col` either immediately before or immediately after the sequence, the function should return `False`. If there is no sequence of length `length` starting at location `(d_y, d_x)`, the function should return `False`.

Problem 2. Creating a plain text file

For the next several questions, you will work with Python's built-in functions. The idea is to learn new things!

Create the plain text file `data2.txt`. To do that, open a new file in Pyzo, write some text in it, and then use File→Save As, select `All ("*.*")` in Files of type dialogue box, and enter the file name in the File Name dialogue box.



Problem 3.

Write a program that opens the file `data2.txt` and prints all the lines in it that contain the the word "lol" in any mixture of upper and lower case (note: both the line "they lolled" and the line 'lOL" are considered to contain the word "lol."). Hint: use `str`'s `find` method `https://docs.python.org/3/library/string.html`. Make sure that your output is not double-spaced. Test your function by creating a test file called `data2.txt` using Pyzo)

Here is how to read in text from a file:

```
f = open(<filename>)
# e.g., f = open("data2.txt")

text = f.read()
# text is a string that contains the contents of the entire file
```

Remember that your program and the file `data2.txt` must be in the same folder, and you should run the program in Pyzo using `Run file as script`.

Problem 4.

Complete and test the following function.

```
def dict_to_str(d):
    """Return a str containing each key and value in dict d. Keys and
    values are separated by a comma. Key-value pairs are separated
    by a newline character from each other.
    For example, dict_to_str({1:2, 5:6}) should return "1, 2\n5, 6".
    (the order of the key-value pairs doesn't matter and can be different
     every time).
    """

    pass  # replace this with your code
```

Problem 5.

Complete and test the following function. You will need to use the built-in `sorted` function:

```
   L = [4, 1, 6]
   L1 = sorted(L)  # L1 is [1, 4, 6]

def dict_to_str_sorted(d):
    """Return a str containing each key and value in dict d. Keys and
    values are separated by a comma. Key-value pairs are separated
    by a newline character from each other, and are sorted in
    ascending order by key.
    For example, dict_to_str_sorted({1:2, 0:3, 10:5}) should
    return "0, 3\n1, 2\n10, 5". The keys in the string must be sorted
```

```
    in ascending order."""

    pass  # replace this with your code
```

For this question, you will need to use the `str` method `split` https://docs.python.org/3/library/stdtypes.html#str.split.

```
s = "hello world"
L = s.split()  # L is ["hello", "world"]
```

Problem 6.

For this question, you will compute word counts in a large corpus of text. Word frequency lists (a frequency of the word is the ratio of the number of times the word appears to the total number of words in a document) are commonly used in computational linguistics. You will write a function that finds the 10 most frequently occurring words in a text file, and a program that uses this function to find the 10 most frequently-occurring words in *Pride and Prejudice*. For the purposes of this problem, you may assume, if you like, that all words are separated by spaces and that there is no punctuation in the file, and all the words are in lowercase, so that the list of words in the file text.txt is given by

```
open("text.txt", encoding="latin-1").read().split()
```

Part (a)

First, store the number of times that the word w appears in the text in `word_counts[w]`. For example, if the word "the" appears in the file 5 times, `word_counts["the"]` should be 5.

For example, if the contents of your file are the opening of *Notes from the Underground* by Fyodor Dostoyevsky, translated by Constance Garnett:

> I am a sick man. I am a spiteful man. I am an unattractive man. I believe my liver is diseased. However, I know nothing at all about my disease, and do not know for certain what ails me.

, and you do not process the text in any way other than using `read().split()`, the list of words will be:

```
["I", "am", "a", "sick", "man.", "I", "am", "a", "spiteful", "man.", "I", "am",
"an", "unattractive", "man.", "I", "believe", "my", "liver", "is", "diseased.",
"However,", "I", "know", "nothing", "at", "all", "about", "my", "disease,", "and",
"do", "not", "know", "for", "certain", "what", "ails", "me."]
```

`word_counts` should then be

```
{"sick": 1, "man.": 3, "at": 1, "what": 1, "nothing": 1, "do": 1, "is": 1, "me.": 1,
"I": 5, "ails": 1, "an": 1, "am": 3, "know": 2, "disease,": 1, "not": 1, "liver": 1,
"believe": 1, "all": 1, "my": 2, "certain": 1, "However,": 1, "and": 1, "for": 1,
"unattractive": 1, "spiteful": 1, "about": 1, "a": 2, "diseased.": 1}
```

For example, the word `"man."` (with the period at the end) appears 3 times in the text, so its entry in the dictionary `word_counts` is 3.

Create the file `test.txt` using Pyzo, and test your function using that file. Use a small file so that you can check that your function works correctly.

Part (b)

Write a function with the signature `top10(L)` that takes in a list L of 100 different integers, and returns a list of the 10 largest integers in `L`.

Part (c)

Now, obtain the top 10 most-frequent words from the dictionary `freq`. To do that, you need to sort the data by the word counts. You cannot sort dictionaries directly, but you can use the following trick:

```
inv_freq = {6: "the", 12: "a", 1:"hi"}
print(sorted(inv_freq.items()))
```

First, experiment with this code and understand what it is doing, and then apply the technique to finding the top 10 most frequent words. Test your function by creating a small text file where you can find the top n most-frequent words manually, running your code on this file, and comparing the results. Then, download the text of Pride and Prejudice from `http://www.gutenberg.org/files/1342/1342-0.txt` and obtain the top 10 most frequent words in Pride and Prejudice.

Problem 7.

Most web pages are written in HTML (HyperText Markup Language). HTML is a fairly complicated language, but here are some basics:

- Paragraphs are enclosed between `<p>` and `</p>`
- Bolded text is enclosed between `<b>` and `</b>`
- A link to `http://www.google.com`, which appears as the text "Google" would look like this:

```
<a href = "http://www.google.com/">Google</a>
```

Download `hello.html` from `http://www.cs.toronto.edu/~guerzhoy/180/labs/hello.html` . To do that, open the link in a web browser, and save the HTML file by pressing `[Cntr-s]` and specifying the location to which to save the file in the dialogue box that pops up. Now open your local copy of hello.html that you just saved with the text editor. You can use Pyzo as the text editor. For other options: n Windows, use the Notepad text editor. On Mac, use TextEdit. Use the Plain Text Mode in TextEdit `https://apple.stackexchange.com/questions/17433/can-textedit-save-as-plain-text`. On Linux, you can use gedit.

Also open the local copy of `hello.html` in a web browser. To view the source of the HTML file, click inside the page in the browser with the right mouse button/cntrl-click on a Mac, and select "View source".

Modify `hello.html` as follows.

- Make the word "Vinci" appear in bold
- Add a link to Yahoo.com's search results for "engineering science."

Show the results to a TA, in the browser and the text editor. By looking at your answer to the previous question, figure out a way to search Yahoo by modifying the text in the address text box instead of typing search terms in the search text box in the browser. Note that when searching Yahoo, the address line in the browser changes depending on the search terms.

Problem 8.

Yahoo.com displays the number of results for the search terms entered on the bottom left of the results page. Use `[Right Click]`→`[View Page Source]` to see the HTML source of the results page, and find in the HTML file the number of results is (to do that, search for the number of the search results that you see in Firefox in the HTML file). Now, write a function in Python that takes in a search term and returns the number of results for that search term on Yahoo.com. To do that, you need to automatically download the Yahoo! search results page. Files on the web can be read similarly to how you would read local files. For example, you can read and then display the source code of a file on my webpage using:

```
import urllib.request
f = urllib.request.urlopen("http://www.cs.toronto.edu/~guerzhoy/180/hi.html")
page = f.read().decode("utf-8")
f.close()
print(page)
```

Here's an excerpt from Dave Barry's Ask Mister Language Person column:

> Q. Do you take questions from attorneys? A. Yes. That will be $ 475. Q. No, seriously, I'm an attorney, and I want to know which is correct: "With regards to the aforementioned' blah blah blah." Or: "With regards to the aforementioned yak yak yak." A. That will be $ 850.
>
> `http://articles.baltimoresun.com/1991-10-13/features/1991286206_1_blah-yak-yak-transpire`

And here's an excerpt from the *New York Times* column *On Language*:

> Neale Gifford writes: **"One practice that annoys me is the use of 'one-year anniversary' or 'five year anniversary' instead of 'first' or 'fifth.' Reason? Anniversary is derived from the Latin annus meaning 'year.'"** Ed Morman writes of "n-year anniversary," **"It's not very mellifluous and it is, of course redundant. Help me bring back 'nth anniversary.'"**
>
> `http://www.nytimes.com/2010/07/18/magazine/18onlanguage-anniversary.html`

One way to settle those usage disputes is to check which variant appears on the web more often[1]

Write a function with the signature `choose_variant(variants)` which takes in a list of usage variants `variants` and returns the variant which returns the most search results on Yahoo.com. For example,

`choose_variant(['"five-year anniversary"', '"fifth anniversary"'])` should return `'"fifth anniversary"'`, since that phrase appears more often. (When searching for a phrase, we enclose it in quotes to indicate that we're searching for a phrase and not for a set of keywords.) Now call `choose_variant()` with `["top ranked school uoft", "top ranked school waterloo"]`.

Note that spaces need to be converted to `"%20"` in the search query. You can use `urllib.parse.quote(search_ter` for that:

```
>> urllib.parse.quote("a b")
"a%20b"
```

[1] While this method is used by writers as well as linguists to determine which linguistic constructs are more popular, the counts are actually not quite accurate. See, e.g., here: `http://itre.cis.upenn.edu/~myl/languagelog/archives/001834.html`

Problem 9. More web scraping

The results of the Canadian Computing Contest are here: `https://web.archive.org/web/20230314171150/https://cemc.uwaterloo.ca/contests/past_contests/CCCResults.pdf`.

We would like to know how many times the name of each high school appears in the document.

It is possible to get text from a pdf document directly, but here I suggest just copy-and-pasting the text into a text editor (e.g., Pyzo), and working with that.

Store the contents of the pdf file into a string in Python. Now, write code that displays the number of times each high school name appears in the text, highest to lowest.

You will have to do *some* manual work here, but you should try and minimize the amount of manual work, so that your technique could work for data from any year, not just 2022.