Assignment 1

CS 421: Natural Language Processing

Due: January 26, 2024 (2 p.m. CST)

1. Introduction

Welcome to CS 421! This short programming assignment is designed to ensure that you are familiar with the basics of the Python programming language, which is the required language for all assignments in this course. Python is a concise, intuitive, interpreted programming language. It uses *dynamic typing*, which means that you can assign whatever information (strings, numbers, or anything else) you'd like to variables without prior specification. Most popular NLP and machine learning libraries have moved towards Python in recent years, and many NLP research or industry positions will expect solutions to be implemented in Python.

To help you get started, this assignment will cover basic Python tools and functionalities that you'll need to complete assignment and project deliverables this semester. Since CS 421 is not a Python programming course itself, there will be limited support following this Assignment for questions regarding Python IDEs or syntax. If you already write Python programs regularly, you will likely breeze through this assignment quickly. If this is your first time writing a Python program, don't worry too much—this will help get you up to speed! Upon successful completion of this assignment, you should be able to:

- Navigate through a Python IDE
- Use standard libraries and install third-party libraries
- Use Python data structures such as lists and dictionaries
- Read and write files

2. Instructions

Each question is labeled as `Code` or `Written` and the guidelines for each type are provided below.

Code

The `Code` questions need to be completed using Python (version 3.10+). There are no **external** packages required to complete this assignment. If you want to use an external package for any reason, you are required to get approval from the course staff on Piazza prior to submission. Templates are provided for each `Code` question (.py files) as supplementary material. Do not rename/delete any functions or global variables provided in these templates and write your solution in the specified sections. Use the `main` function (provided in templates) to test your code when running it from a terminal. Avoid writing test code in the global scope; however, you should write additional functions/classes as needed in the global scope. These templates may also contain important information and/or examples in comments so please read them carefully. This part of the assignment will be graded automatically using Gradescope.

To submit your solution for `Code` questions, you need to compress the following files (after completion) in a single `zip` file. These files should be in the root of your `zip` archive for autograding to work correctly.

- ☐ `loops.py`
- ☐ `lists.py`
- ☐ `text_analysis.py`

Submit this `zip` file on Gradescope under `Assignment 1 - Code`. All specified files need to be submitted to receive full credit.

Written

You are required to submit all `Written` questions in a single PDF file. You may create this PDF using Microsoft Word, scans of your handwritten solution, LaTeX or any other method you prefer.

To submit your solution for `Written` questions, you need to provide answers to the following questions in a single PDF.

- ☐ Q6

Before submission, ensure that all pages of your solution are present and in order. Submit this PDF on Gradescope under `Assignment 1 - Written`. Please match all questions to their respective solutions (pages) on Gradescope. Questions not associated with any pages will be considered blank or missing and all questions need to be completed to receive full credit.

3. Questions

3.1. Getting Started

Before you complete any questions in this assignment, follow the steps detailed here to set up your programming environment.

Install the PyCharm IDE.

Although you're welcome to use other IDEs when you complete your assignment or project deliverables, PyCharm will be the only *supported* IDE. The course staff will be unable to answer questions that are specific to alternative IDEs (e.g., Spyder, Eclipse, etc.). We recommend using PyCharm especially if you are unfamiliar with Python, in case you end up having questions about managing the packages and debugging the code. To install PyCharm:

1. Visit the website here, and follow the steps to download the appropriate PyCharm installer for your computer's architecture and operating system: `https://www.jetbrains.com/pycharm-edu/`. PyCharm is supported on Windows, Linux, and macOS operating systems.
2. Install PyCharm by following the instructions from your installer. If you are prompted to enter an educational license at any point, you can sign up for one for free at the link here: `https://www.jetbrains.com/community/education/#students`.
3. Make sure that your base interpreter is configured to Python 3.10 or higher. The instructions for doing this will vary slightly depending on your operating system. Visit the website here and follow the instructions for your operating system: `https://www.jetbrains.com/help/pycharm/project-interpreter.html`. If the current interpreter is listed as Python 3.10 or higher, there is no need to change anything.

Set up your project.

If you are using PyCharm to complete this assignment, perform the following steps to set up your project:

1. Create a new project using the **New Project** button on the welcome screen. Update your project location as desired; you can also update your base interpreter at this time if needed. When your settings are configured as you would like, click the **Create** button. More details about creating a new project can be found here: `https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html`.
2. Unzip the supplementary material `a1supplement.zip` in the same project folder you just created.

You are now ready to complete Assignment 1!

3.2. Code (50 points)

Q1 (5): Sum of Even Numbers

Implement the function `sum_even` (in `loops.py`) which takes as input a number N and returns the sum of the first N `even` numbers. You must use a loop to compute the sum.

Note: We encourage you to submit the assignment on Gradescope (once it's available) using the process explained in Section 2. You may submit your solution as many times as you wish before the deadline without any penalties.
Supplementary material: `loops.py`

Q2 (5): Fibonacci Sum

Fibonacci numbers are the sequence of numbers in which each number is the sum of the two preceding numbers. The first and second Fibonacci numbers are 0 and 1 respectively. The sequence is then given as:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, ...$$

Implement the function `sum_fib` (in `loops.py`) which calculates the sum of the first N Fibonacci numbers. For example, the sum of the first five Fibonacci numbers should be:

$$0 + 1 + 1 + 2 + 3 = 7$$

You are required to create an iterative solution for this problem using loops.
Supplementary material: `loops.py`

Q3 (5): List - N

Implement the function `list_minus` (in `lists.py`) which takes as input a list of integer values, and an integer N, and returns a list with every element of the input list decreased by N. For example, given the list $[2, 3, 6]$ and 5 as input, the output should be $[-3, -2, 1]$. You are not allowed to use a list comprehension in your solution for this function.
Supplementary material: `lists.py`

Q4 (10): Process List

Implement the function `process_list` (in `lists.py`) which takes as input a list of numerical values and an integer, and returns a list in which each element of the input list is decreased by N, and then raised to the third power. For example, given $[2, 3, 6]$ and 5 as input, it should return $[-27, -8, 1]$ as output. You must use the `list_minus` function from Q3 to process the list, and then implement the `cube_list` function (in `lists.py`). You are not allowed to use a list comprehension in your solution for this function.
Supplementary material: `lists.py`

Q5 (25): Text Analysis

For this multi-part question, you will analyze 9 of the stories from "The Adventures of Sherlock Holmes" by Sir Arthur Conan Doyle, as collected in the file `adventures_sh.txt`, which is also part of `a1supplement.zip` (these books can be freely used since the copyright has expired; this file was downloaded from `https://sherlock-holm.es/`). Specifically, you will perform word frequency

analysis, which is often used as a preliminary pre-processing step for NLP tasks such as stylometry[1] and text classification.[2]

(a) Before you start your analysis, you need to read the stories from the file. Implement the `read_file` function (in `text_analysis.py`) which takes as input a path to a file and returns its contents as a string. You can learn more about file handling in Python here: `https://www.w3schools.com/python/python_file_handling.asp`.

(b) Now, you can split this string into words by using the in-built Python `split`[3] function (use default arguments). Implement the `convert_to_words` function (in `text_analysis.py`) which takes as input a string and returns a list of lowercase words. You can use the in-built `lower`[4] function to convert a string to lowercase.

(c) Now that you have the words in a list, you can finally calculate word frequencies. Implement the `get_wc` function (in `text_analysis.py`) which takes as input a list of words and returns a Python dictionary. The dictionary should contain words as keys and their frequencies as values. Hint: Loop through each word and check to see whether it already exists as a key in the `word_counts` dictionary (initialized in the skeleton code). If it does, increment the value associated with that key by 1. If it doesn't, add the key to the dictionary with a value of 1.

(d) Sometimes, we need to save our results for later use. This is useful especially if the processing takes a long time and you need the results later. We can save our word counts in a JSON[5]–formatted file using Python's standard JSON library.[6] This library is already imported in the skeleton code. Implement the `to_json` function in `text_analysis.py` which takes as input a Python dictionary and a string indicating the output filepath. Create a new output file at the specified path and write the dictionary as a JSON object to that file. You can convert your dictionary to a JSON object string using the JSON library's `dumps()` function.[7] Then, open a new file named `word_counts.json` in *write* ("w") mode. Write this newly created JSON string to file.

Supplementary material: `text_analysis.py`

[1] `https://en.wikipedia.org/wiki/Stylometry`

[2] `https://sebastianraschka.com/Articles/2014_naive_bayes_1.html`

[3] `https://docs.python.org/3/library/stdtypes.html#str.split`

[4] `https://docs.python.org/3/library/stdtypes.html#str.lower`

[5] `https://www.w3schools.com/js/js_json_intro.asp`

[6] Although this package should come preinstalled with your Python installation, you can also install third-party Python packages by following the PyCharm instructions here: `https://www.jetbrains.com/help/pycharm/installing-uninstalling-and-upgrading-packages.html` or by typing `pip3 install <external_library>` in your terminal window.

[7] You can read more about this function here: `https://docs.python.org/3/library/json.html`.

3.3. Written (10 points)

Q6 (10): Briefly describe some interesting facts about the word frequencies from *The adventures of Sherlock Holmes***.**

Answer the following questions:

(i) List 4 of the most frequent *functional* words from this book, with their frequencies (articles, prepositions, determiners - see section 8.1 in textbook if uncertain).

(ii) List 4 of the most frequent nouns or verbs (excluding proper nouns like Sherlock, Holmes, London, Baker, Scotland, etc), i.e. content-bearing words, and their frequencies.

(iii) (a) In general, are functional words or content words more frequent? (b) Why do you think that is the case? (c) What would the 4 most frequent nouns/verbs tell us about the book itself, if anything; assume you don't know who Sherlock Holmes is and what he does? Answer in 4 sentences, 1 sentence each for (a) and (b), and 2 sentences for (c).

4. Rubric

This assignment will be graded according to the rubric below. Partial points may be awarded for rubric items at the discretion of the course staff.

Q1 (5 points possible)	
Autograded	+5
Q2 (5 points possible)	
Autograded	+5
Q3 (5 points possible)	
Autograded	+5
Q4 (10 points possible)	
Autograded	+10
Q5 (25 points possible)	
Autograded	+25
Q6 (10 points possible)	
Student lists 4 words of the required type (8 words total)	+3
Student clearly answers subquestions a-c (4 sentences)	+7