Assignment 1

INFSCI 0201 - Intermediate Programming with Python. Fall 2023

Due Date: October 8, 2023, before 11:59 PM

This assignment has two tasks: Writing Python programs for decomposing money and implementing a program that could 'unscramble words'.

Topics Covered

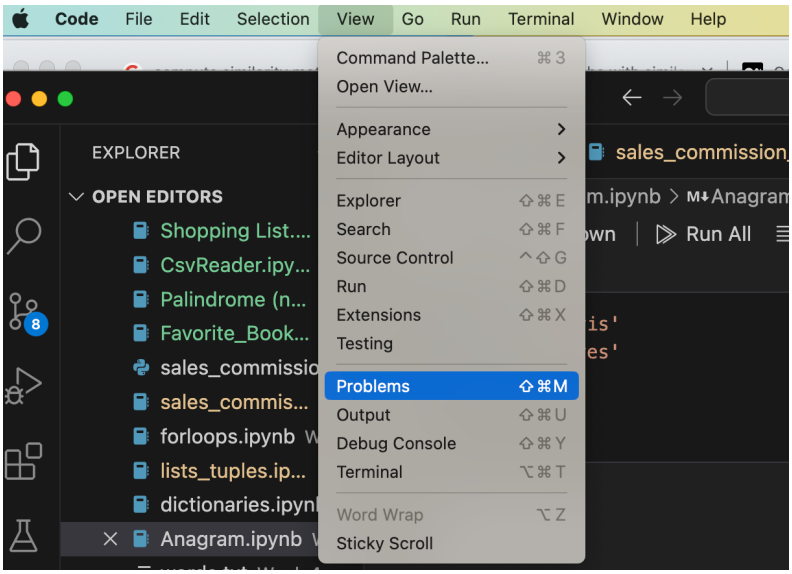
1. Variables
2. Data Types
3. Handling user input
4. Working with strings
5. Conditional statements
6. Working with lists
7. Loops
8. Reading text files
9. Git and GitHub use
10. Python code styling

Tasks

General Guidelines

- Imagine this assignment is a task assigned to you in a software company. **Implement your program as a Jupyter Notebook** so that you can present it to your colleagues during the next team meeting. Create a new Jupyter Notebook file representing each program (e.g., decomposing_money.ipynb).
- You know that your team likes to see readable code. As a junior developer, you want to show that you pay attention to details while programming. For this reason, while completing your assignment, please check **Python's style guide** here that would give you ideas on how you can write 'readable' code.

  We are humans and it is okay to make mistakes. But we should not ignore our mistakes and put extra effort to address them. That is why I suggest you use **Visual Code**. This way you can minimize/eliminate style errors and possible syntax errors much easier. To do so, open your **Problem** panel as shown below and address the issues listed on that panel. To promote putting a bit extra effort in your submissions, submissions that do not have any styling issues will get **5% extra credit** on this assignment.



- Explain your steps by dividing your code into smaller chunks and using separate Notebook cells as we see in the class.
- Make sure to comment your solution as needed. Make sure to provide comments for logic decisions as well.
- You can implement these programs using more complex programming concepts but make sure that you are only using concepts we have covered in the class. That is why please follow the instructions provided for each task carefully.
- You may get help from your friends and/or AI supported tools. But make sure that you properly cite any help you got (the name of your friend, the tool you have used, instructions you have used for the AI tool, any tutorial beyond class materials, etc.). Please be respectful to your friends and to me and make sure that you are submitting your own work with citing your help resources.

Task #1: Decomposing Money

- Ask the user to enter a number representing an amount of money from 1 dollar to 9999 dollars (*integer*). Make sure that you check if the user entered an integer value between 1 and 9999. If not, prompt user to enter a number again (until the user successfully enter one). Hint: You should use one of the loop structures we covered in the class.
- Output the number of "bucks" (single dollar), "sawbucks" (10s), "Benjamins" (100) and "grands" (1000) corresponding to the amount entered by the user. For example if the user enters 7528, the program should output:
    - 7 grands
    - 5 Benjamins
    - 2 sawbucks
    - 8 bucks
- HINT: you can decompose the total amount by using integer division and modulus (%) operator. For example dividing by 1000 will give you the number of "grands", and then using the remainder of this division to get the rest. You do not need to use loops.

- Make sure to test your code with multiple user inputs.

Task #2: Unscramble Words

In this task, you will extend the anagram example reviewed in class to create a new Python program that could “unscramble” words. Word unscramblers are commonly used in word-based games such as Wordscapes, Scrabble, Wordle, and Words With Friends. You can review an online implementation of a word unscrambler at https://unscramblex.com/

Using code examples from class (e.g. the Anagram solver), create your own version of a word unscrambler. You can either use the input file provided in class, or you can find your own file that contains all the words in the English language.

Unscrambled words refer to the process of rearranging the letters of a scrambled or jumbled word to form a meaningful word. Unlike anagrams, unscrambled words don't necessarily need to use all the original letters; the goal is to rearrange the letters to create a valid word. The resulting word may or may not be related to the original word. For example, given the scrambled word "trca," unscrambling it can yield "cart" or "car," both of which are valid words but not necessarily anagrams of each other.

Your program must do the following:

1. Read data from the words.txt and store it as a list of individual words.
    a. Make sure that all the words are in lower case
    b. Remove preceding and trailing spaces. Hint: there is a specific string function that does this for you.
    c. Remove the new line character ‘\n’
    d. Remove duplicates. Hint: Convert your list to a set and back to a list. Google this to find out how you can create a set from a list and vice versa.
2. Ask user for an input word.
    a. Assume that the user entered a single word.
    b. Make sure that the user entered an input has at least 3 characters. If the user entered less than 3 characters, ask them to re-enter (until they provide a valid input). If the input is longer than 6 characters, just take the first 6 characters (Hint: use string slicing)
    c. You can assume that the user will enter only alphabetic characters. So you do not need to check if the input contains numbers, etc.
3. Iterative over the list of words you have read from the file to find all words that could be created using the letters of the user input.
    a. Remember that we are not looking for anagrams but words that can be created by using all or some of the letters in the user input.
4. Using that list, print the words based on the following groups:
    a. All 6-letter words
    b. All 5-letter words
    c. All 4-letter words

    d. All 3-letter words
    e. Hint: You can sort a list of strings using its length. Google this to find out how!
5. Use the word “cardiothoracic” to test your program.
    a. Also test your program with inputs that has less than 3 characters and longer than 6 characters.
6. Allow user to enter multiple inputs unless they type -1 as a sentinel value (a value to terminate the program).

Grading Policy

Task	**Points**
Decomposing Money (25 points)	
Prompt the user to enter a number	4
Check user input validity and print an error message	4
Prompt user until they enter a valid input	7
Print out the number of grands first	3
Print out the number of Benjamins second	3
Print out the number of sawbucks third	2
Print out the number of bucks as a final output	2
Unscramble Words (65 points)	
Read data from the words.txt and store it as a list of individual words	10
Ask user for an input word and check its validity.	15
Successfully finding all the unscrambled words	25
Properly printing out the words based on their lengths	10
Allow user to enter another input	5
Submitting the solution through GitHub	10
Extra	
Submitting a readable code (read general guidelines)	5
Total:	105

Submission Guidelines

You need to submit your homework to a GitHub repository and share the link of that repository with me through Canvas.

- Watch Git/GitHub crash course if you haven't done so already https://www.youtube.com/watch?v=RGOj5yH7evk
- Create a GitHub account, if you do not have one
- Create a course repository for your submissions. You will submit the following assignments and the project to the same repository. Follow this tutorial: https://docs.github.com/en/get-started/quickstart/create-a-repo
    - Do not forget to add a README file as described in the tutorial.
    - Remember to create your repository as a private repository. After you create your repository, you need to include me as your collaborator. Follow these steps to add me as a collaborator to your repository: https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository
    - My GitHub handle is cskamil. Search with that to find me. Do this as early as possible so that I can help you if you have any issues.
- Clone this repository to your local machine using the one of the methods we discussed in the class (through Visual Code or GitHub desktop or if you know through the command line)
- Inside the cloned repository folder, create a folder called assignment1 (lowercase).
- For task #1, write your solution in a file called decomposing_money.ipynb. Make sure to create this file in the repository folder.
- For task #2, write your solution in a file called word_unscrambler.ipynb. Make sure to create this file in the repository folder.
- While implementing your solutions, remember to commit your changes frequently to your local repository. Again, you can use the GitHub desktop app for this but you can also use Visual Studio or command line for achieving the same result.
- When you finish your assignment, make sure to commit all your changes first locally and then push those commits to the origin repository (the actual GitHub repository that stores your code remotely).
    - Check the following tutorial pages to see how you can use GitHub desktop
    - Making commits: https://docs.github.com/en/desktop/making-changes-in-a-branch/committing-and-reviewing-changes-to-your-project-in-github-desktop
    - Pushing changes to the origin: https://docs.github.com/en/desktop/making-changes-in-a-branch/pushing-changes-to-github-from-github-desktop
- Your assignment should have the following folder/file structure at the end:

    → your GitHub Repo

        → *assignment1*

→ *word_unscrambler.ipynb*

→ *decomposing_money.ipynb*

- Finally, submit your repository link to Canvas. You will see the Assignment #1 under assignments to submit your link. This way I can map your GitHub accounts with your course accounts.

Late Submissions

All students get +2 days late submission credit. For instance, if you submit your homework 20 hours later than the deadline, you will use your 1-day late submission credit automatically and do not need to provide me any valid reason. Without any proper excuse (e.g., medical issue, family concerns, etc.) or credit use, you will lose 25% of your assignment grade after each day you submitted late.