CSE 8A/6R Fall 2025 Homework 4

Due: Thursday, November 13, 2025 before 11:59pm

Full Name: _____ PID: _____

Homework 4 Topics Covered:

1. Dictionaries
2. References and Objects
3. How Functions Work, Variable Scope

5 core competencies:
1. Code Tracing, 2. Code Comprehension, 3. Code Writing, 4. Debugging, and 5. Testing

Submission

Questions 1 - 4:

- **Must be Handwritten** in the given boxes only.
- Scanned/uploaded to Gradescope **Homework 4 Questions 1 - 4** assignment.
- *If any of the questions 1 - 4 are typed*, *then your work will be considered* ***ungradable****.*

Question 5 (process journal):

- The **process journal** can be either handwritten or **typed (preferred)** into a document.
- **In addition, starting from this week you'll need to submit the Python file (.py) to the problem. See submission instructions below.**
- Uploaded to Gradescope **Homework 4 Question 5 Process Journal** assignment.

When you submit your work, **please make sure to assign the correct pages to the questions**. Otherwise, your work will be considered **ungradable**.

Point breakdown for Homework 4:

- **Exemplary (3 homework points):** All problems receive a "Pass."
- **Satisfactory (2 homework points):** At least 4 out of 5 questions receive a "Pass."
- **Needs Improvement (1 homework point):** At least 1 out of 5 questions receive a "Pass."
- **Incomplete (0 homework points):** No problems receive a "Pass," or the assignment was not submitted, or the assignment is ungradable (e.g., illegible handwriting), or if the submission does not follow the instructions (e.g., homework is typed instead of handwritten, not answering in the boxes provided, not assigning pages to the question).

1. [Code Tracing, Code Comprehension]

A) The function swap_v1() tries to swap the values of two list elements, but it does not work as expected.

```
def swap_v1(item1, item2):
    temp = item1
    item1 = item2
    item2 = temp

students = ["Eden", "Elizabeth", "Alex", "Punyah"]
print("Before swap:", students) # Expected: ['Eden', 'Elizabeth', 'Alex', 'Punyah']
swap_v1(students[0], students[2])
print("After swap:", students) # Expected: ['Alex', 'Elizabeth', 'Eden', 'Punyah']
```

Actual Output:

```
Before swap: ['Eden', 'Elizabeth', 'Alex', 'Punyah']
After swap: ['Eden', 'Elizabeth', 'Alex', 'Punyah']
```

Show why swap_v1 does not swap the two list elements as expected **by drawing memory diagrams** (i.e., call stack with the stack frames containing references and objects) to explain what is happening behind the scenes in the computer's memory. See lecture 11 handout question 1 for the format of the memory diagram.

B) The function swap_v2() tries to swap the values of two list elements, and it works as expected.

```
def swap_v2(student_list, index1, index2):
    temp = student_list[index1]
    student_list[index1] = student_list[index2]
    student_list[index2] = temp

students = ["Eden", "Elizabeth", "Alex", "Punyah"]
print("Before swap:", students) # Expected: ['Eden', 'Elizabeth', 'Alex', 'Punyah']
swap_v2(students, 0, 2)
print("After swap:", students) # Expected: ['Alex', 'Elizabeth', 'Eden', 'Punyah']
```

Actual Output:

```
Before swap: ['Eden', 'Elizabeth', 'Alex', 'Punyah']
After swap: ['Alex', 'Elizabeth', 'Eden', 'Punyah']
```

Show why swap_v2 does correctly swap the two list elements as expected by **drawing memory diagrams** (i.e., call stack with the stack frames containing references and objects) to explain what is happening behind the scenes in the computer's memory. See lecture 11 handout question 1 for the format of the memory diagram.

2. [Code Tracing] **Draw memory diagrams** (i.e., call stack with the stack frames containing references and objects) to explain what is happening behind the scenes in the computer's memory. See lecture 11 handout question 1 for the format of the memory diagram. Your work should clearly show the call stack and track the variable `data`.

Code:

```
data = 29

def process_data(data):
   display_data()
   print("Process local:", data)

def display_data():
   print("Display global:", data)

process_data(42)
print("Outside scope:", data)
```

Output:

```
Display global: 29
Process local: 42
Outside scope: 29
```

Memory diagram:

3. [Code Comprehension] What should be the key, and what should be the value? In the following scenarios, design the dictionaries to store the data. You should specify what should become the key, what should become the value, and briefly explain your reasoning.

A. The UCSD registrar keeps track of the **course code** (eg. `"CSE 8A"`), and **how many students are enrolled in the course** (eg. `300`) for a specific quarter.

Key:

Value:

Reasoning:

B. You are creating a recipe book that keeps track of a **list of ingredients** (eg. `["fish", "potato"]`) for each **dish** (eg. `"fish and chips"`). Note: your answer should consider the *data type* of the data you want to store.

Key:

Value:

Reasoning:

4. [Code Tracing, Code Comprehension]

```
def enigma(city_list, pop_list):
   result = {}
   for i in range(len(city_list)):
       result[city_list[i]] = pop_list[i]
   return result

cities = ["Chennai", "Madison", "San Diego"]
populations = [12300000, 285300, 1420000]
city_population_dict = enigma(cities, populations)
print(city_population_dict)
```

A) What will be printed by the code above?

Show your work

Output:

B) Explain in plain English the high level purpose of the enigma function.

5. [Code Writing, Testing] Finding word occurrences by index in song lyrics.

For this question, you are submitting a ***process journal***. For detailed instructions, please check the instruction document: https://tinyurl.com/ProgrammingProcessJournal.

IMPORTANT: starting from this assignment, please submit your python file (.py) (in addition to your process journal) that you used to implement the program. Submission instructions are shown below.

In this question, your main goal is to create a dictionary that tracks where a word occurs in a song. To do this, you are implementing **two functions in the same file** with the following specifications. You should have **at least three tests for each function**, and one of the tests for each function must be the sample test shown below.

1. **Name of the function:** `string_to_list`

   Description: this function takes a string of words and separates the string into a list of words by space. You can assume all words are already all lowercase letters with no punctuation marks or special characters. *Hint*: you might find reading 8.1 very helpful!

   Parameter: `words_str`, the *string* representation of some text.

   Data type of return value: list, representing the lyrics as a list of words.

   This function should be called by `word_occurrence_dict`. However, you should also call and test the function directly.

   Sample test:

   ```
   print(string_to_list("imagine all the people living life in peace"))
   # Expected output:
   # ['imagine', 'all', 'the', 'people', 'living', 'life', 'in', 'peace']
   ```

2. **Name of the function:** `word_occurrence_dict`

   Description: this function should return a dictionary that tracks where a word occurs in a song. See the example below. *Hint*: you might find week 5 lab very helpful!

   Parameter: `lyrics`, the *string* representation of some text. You can assume all lyrics are already all lowercase letters with no punctuation marks or special characters. You should simply see a string of regular English words (as shown in examples above) making up characters, separated by spaces.

   Data type of return value: dict, counting the frequency of words in song lyrics.

   This function **must** call `string_to_list` within this function to split words.

Sample test:

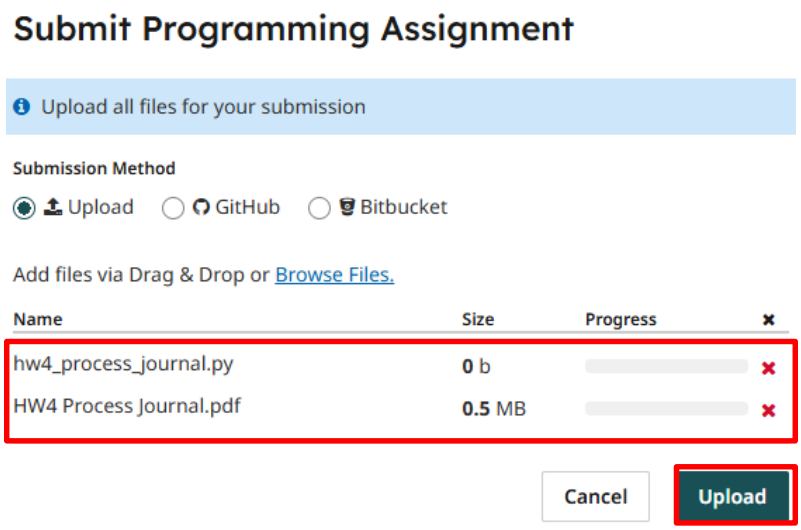
```
some_lyrics = "never gonna give you up never gonna let you down never gonna run around and desert you"
print(word_occurrence_dict(some_lyrics))
# Expected output:
# {'never': [0, 5, 10], 'gonna': [1, 6, 11], 'give': [2], 'you': [3, 8, 16], 'up': [4], 'let': [7], 'down': [9], 'run': [12], 'around': [13], 'and': [14], 'desert': [15]}
```

Submission Instructions for Process Journal - including Python file

Starting from this assignment, you must submit both the **Python file (.py file)** containing your final implementation, and the **pdf of your process journal** to this question.

- You don't need to assign pages to the pdf (you will not see the option).
- For submission, using a laptop to submit is highly recommended.
- For submission, typing the process journal is highly recommended, as you can easily save it as a pdf and include screenshots of your progress.

Your submission page should look like this (the file names can be different):



You are done! Once you submit, you might see a message "This assignment does not have an autograder configured" under "Results". This is expected. You can preview the submission under the "code" tab to make sure you see both the python file and the pdf.

