CS 171
Week 5 Lab

Overview

This lab consists of a collection of problems that you may solve in any order. To earn full credit, you must submit solutions to all problems at one time.

Important:

- You must be present in the lab session to earn credit
    - If you are recorded as absent but still complete the work, you will not earn credit. If you arrived after the CA/TA took attendance, be sure to check in so you are recorded as present/late.
    - If you miss the lab meeting, contact your instructor via email right away to request a make-up session.
- You may collaborate with at most one lab partner; Feel free to switch partners next week if you wish
- Each student must submit their own work to Gradescope for credit, even if you worked with a partner
- At most 100 points are possible. To get a combined score, submit all of your solutions at once.
- Please only use features of the language that have been covered in reading and lectures
    - Solutions which make use of language features not yet covered will not receive credit.
    - Students whose submissions repeatedly make use of additional skills not covered in the course so far may be suspected of violating academic integrity.
    - *Even if you know the language already, please work within the subset of the language covered up to this point in the course.*
- You may not use artificial intelligence tools to solve these problems.
- Your lab work is due at the end of the lab period. Late work will be accepted with a 30-point penalty if it is submitted within 12 hours of the end of your lab.

Helpful to Know:

- The last score earned will be recorded for your lab score
- To get a combined lab score, you must submit multiple files at the same time
- Lab assistants are here for your support. Ask them questions if you are stuck!

Materials to Use:

- Lecture slides for Weeks 1 - 5
- Think Python Chapters 1 - 9

Problem A: Largest Small Values

Submission File: `lab05a.py`

Part 1

Write a function, `largestValueBelowThreshold(values, threshold)` which receives a list of integer values and a single integer threshold. The function should return the largest value in the list that is less than the threshold. If there are no values in the list less than the threshold, return None.

Part 2

Write a main program which allows the user to enter a list of integer values and a threshold, as demonstrated below. Since this is a timed activity, you may assume that all inputs will be integers (no need to validate).

```
Enter threshold value: 5
How many values in the list? 7
Enter value 1: 8
Enter value 2: 6
Enter value 3: 7
Enter value 4: 5
Enter value 5: 3
Enter value 6: 0
Enter value 7: 9
The largest value in the list [8, 6, 7, 5, 3, 0, 9] less than 5
is 3.
```

Remember that the auto-grader needs your main program to be written below the "if __name__...." line in order to properly test your function.

```
# Define your function(s) here


# Implement your main below
if __name__ == "__main__":
    # Your main code goes here, indented
```

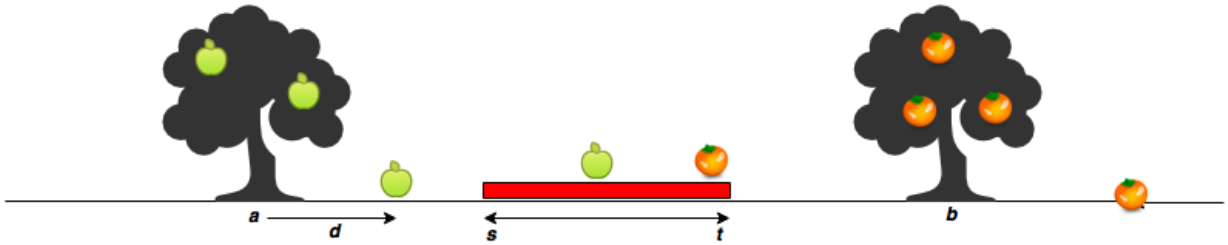
Problem B: Apples & Oranges

Submission File: `lab05b.py`

Sam's house has an apple tree and an orange tree that yield an abundance of fruit. Using the information given below, determine the number of apples and oranges that land on Sam's house.

In the diagram below

- The red region denotes the house, where *s* is the start point, and *t* is the endpoint. The apple tree is to the left of the house, and the orange tree is to its right.
- Assume the trees are located on a single point, where the apple tree is at point *a*, and the orange tree is at point *b*.
- When a fruit falls from its tree, it lands *d* units of distance from its tree of origin along the *x*-axis.
    - A negative value of *d* means the fruit fell *d* units to the tree's left, and a positive value of *d* means it falls *d* units to the tree's right.



Given the value of *d* for *m* apples and *n* oranges, determine how many apples and oranges will fall on Sam's house (i.e., in the inclusive range).

Example

Sam's house is between *s = 7* and *t = 10*. The apple tree is located at *a = 4* and the orange at *b = 12*. There are *3* apples and *3* oranges. Apples are thrown *apples = [2, 3, -4]* units distance from *a*, and *oranges = [3, -2, -4]* units distance. Adding each apple distance to the position of the tree, they land at *[4+2, 4+3, 4+-4] = [6, 7, 0]*. Oranges land at *[12 + 3, 12 + -2, 12 + -4] = [15, 10, 8]*. One apple and two oranges land in the inclusive range *7-10*, so we return the list *[1, 2]*.

Function Description

Complete the `countApplesAndOranges` function. It should return a list containing the number of apples in position 0 and the number of oranges in position 1 that land on Sam's house.

`countApplesAndOranges` has the following parameter(s):

- *s*: integer, starting point of Sam's house location.
- *t*: integer, ending location of Sam's house location.
- *a*: integer, location of the Apple tree.
- *b*: integer, location of the Orange tree.
- *apples*: integer list, distances at which each apple falls from the tree.
- *oranges*: integer list, distances at which each orange falls from the tree.

Example

`countApplesAndOranges` (7, 11, 5, 15, [-2, 2, 1], [5, -6]) should return [1, 1]

The first apple falls at position 5 – 2 = 3.
The second apple falls at position 5 + 2 = 7.
The third apple falls at position 5 + 1 = 6.
The first orange falls at position 15 + 5 = 20.
The second orange falls at position 15 – 6 = 9.

Only one fruit (the second apple) falls within the region between 7 and 11, so 1 is the first element in the result list.
Only the second orange falls within the region between 7 and 11, so 1 is the last element in the result list.

Source: This problem was adapted from the HackerRank problem Apple and Orange.

Problem C: Double Letter Words

Submission File: `lab05c.py`

Part 1

Write the function, `doubleLetterWords(words)`, which receives a list of strings and returns a list containing only the entries within the input list which are double letter words. These are words which have repeated letters back-to-back.

Examples of double letter words:

- Progra**mm**ing
- C**oo**peration
- Pe**pp**er
- **Aa**rdvark *(note: case does not matter)*

Your function should return the list of double letter words in the same order that they occurred in the input list.

Ex:

- Input: ["A", "rolling", "stone", "gathers", "no", "moss"]
- Output: ["rolling", "moss"]

If there are no double letter words in the list provided, return an empty list.

Part 2

Prompt the user to enter a phrase. Display a list of double letter words contained in the phrase (or a message indicating that the phrase does not contain any double letters).

Ex:

```
Enter phrase: I heard that the moon is made of cheese
Double Letter Words:
moon
cheese
```

Ex:

```
Enter phrase: I heard it through the grapevine
No double letter words found.
```