We are going to get more practice with what we've learned about strings, lists, loops, conditionals and variables. This time, we are practicing on questions that could be asked in an entry-level interview and will be similar to questions on the final exam.

On exams, you won't see a video of the final product, so be sure to carefully read the descriptions. Also on exams, you'll be hand-writing code. By coding directly in the Google Doc (or writing it down and transcribing it later), you can get practice with that. Feel free to check your work before submission.

Please make a copy of this Google Doc (File > Make a Copy) and fill in the answers below. Then, you can go to File > Download > pdf when you are done and **upload your answers to Canvas.**

Note: If you get stuck on this (or any) lab assignment, it is completely acceptable to 1) talk to a classmate and discuss what they think and 2) ask the instructor or CS tutor for help. See the syllabus for the appropriate email addresses if you are not doing this lab during the Thursday lab period.

1) [20 points] During professional sports, a lot of data is collected[1] which is used for analysis and commentary. When watching professional soccer, commentators will mention how many kilometers a player has run or the ball has traveled throughout the game - players will routinely run 10-15 km during the 90 minute game.

How do they calculate these stats? They use overhead cameras and some fancy software to record the locations of players and the ball repeatedly throughout the game. For example, they might record the X Y coordinates every second and store those to CSV files, resulting in about 6000 rows:

```
ball_liverpool_vs_manchester_united_jan_14_2023.csv
0, 60.0, 40.0
1, 61.5, 38.0
2, 62.0, 37.2
3, 65.2, 40.6
...
5982, 8.3, 90.3
```

The first column is the timestamp - that is, how many seconds have elapsed since the beginning of the half. The units for the second and third column are meters, as measured from one of the corners of the pitch.

To calculate the total distance traveled, we need to find the distance traveled between each set of points (using the Pythagorean theorem) and add them all up.

The distance traveled between second 0 and second 1 is $\sqrt{(61.5 - 60.0)^2 + (38.0 - 40.0)^2}$ **= 2.233**

The distance traveled between second 1 and second 2 is $\sqrt{(62.0 - 61.5)^2 + (37.2 - 38.0)^2}$ **= 0.943**

In general, one subtracts the previous y coordinate from the current y coordinate, subtracts the previous x coordinate from the current x coordinate, squares both quantities, adds them up, and takes the square root of that sum.

[1] https://www.youtube.com/watch?v=GyN-qpVfOWA is a great look into collecting data for professional soccer games.

1a) [optional] This space is for you to plan your algorithm and to identify which components you need to bring together to solve this. You may get partial credit for your algorithm/plan here.

1.

1b) [5 points] Write a function that takes in 4 parameters: two pairs of x,y coordinates and returns how much distance is between them.

```
# Your code should go here

import math

def distance(x1, y1, x2, y2):
      part1 = (x2 - x1) ** 2
      part2 = (y2 - y1) ** 2
      answer = math-sqrt(part1 + part2)
      return answer

print distance
```

1c) [15 points] Write a function that takes in 1 parameter: the name of a csv file. It should read in that file, which is in the format above, and return the total distance traveled according to the data. This function should call your function from 1b. Even if your function in 1b was incomplete, pretend it is fully functional. No external libraries are required (nor permitted) for this question.

```
# Your code should go here
      #open and load file
#pull out data using "readlines" from file
      #close file
      create a gatherer list
      #use a loop split and  int()
      use append to add to list
      #find the distance and add them all up
#return total distance gatherer
import math
def total_distance(file_name):
      f = open(file_name, "r")
      lines = f.readlines()
      f.close()
      data = []
      for line in lines:
            parts = line.split(",") #list of string
            x = float(parts[1])
            y = float(parts[2])
            data.append([x,y])
      total_distance = []
```

```
    for i in range(1, len(data)):
        x1, y1 = data[i - 1]
        x2, y2 = data[i]
        distance = math.sqrt((x2-x1)**2 + (y2 -y1)**2)
        total_distance += distance
    return data
```

2) [20 points] In the video game Professor Layton and the Curious Village, there is a puzzle that needs solving called 33333! The goal is to find a 5 digit number that when a 4 digit number is subtracted from it, the answer is 33333. The key restriction is that you can only use the digits 1 through 9 once.

Write some Python code that uses brute force to find both solutions. Your code should try many possibilities to find and print the correct answers. No external libraries are required (nor permitted) for this question (e.g. do not `import random`). As per the linked page, there are two solutions - your code should print those out and nothing more.

I don't care what the actual answers are to the puzzle - they are on the linked wiki page. What I do care about is your ability to write code that could find the answers *if* we didn't know them.

2a) [optional] This space is for you to plan your algorithm and to identify which components you need to bring together to solve this. You may get partial credit for your algorithm/plan here.

1. def a function
2. 2.create a for range loop for 5 digits
3.

2b) Write the code to implement the algorithm from question 2.

```
# Your code should go here
def all_used_once(num1, num2):
    num1 = str(num1)
    num2 = str(num2)

    together = num1 + num2


    for i in range(1, 10):
        if together.count(str(i)) > 1:
            return False
    return True

def solve_puzzle():

    num1 = 36251
```

```
    num2 = 2918

    # Check the condition
    answer = num1 - num2
    if answer == 3333 and all_used_once(num1, num2):
        print("WE GOT IT!")
        print("5-digit number: " + str(num1))
        print("4-digit number: " + str(num2))
        print("No solution found.")


solve_puzzle()
```

2c) [optional, 5 bonus points] Modify your code to see if answers exist for other numbers. Do this by wrapping your code in a function (if it is not already) and take in "the magic number" as a parameter. In 2b, the "magic number" was 33333. For example, if the result is 111111 or 77777, can you find an answer by changing the function call?

3) [20 points] Suppose there are a bunch of playing cards on the ground. A "royal court" is a Jack, Queen, and King of the same suit.

For example, from the list `["8H", "QD", "JD", "KS", "JS", "QD", "KD", "QS"]` **one could construct 2 royal courts. JD, QD, KD (Jack, Queen, King of Diamonds) and JS, QS, KS (Jack, Queen, King of Spades).**

3a) [3 points] Create a different set of example data. It should be a list of strings. How many royal courts can be created from your example? Identify those royal courts using words or highlighting.
cards = ["10H", "JH", "QH", "KH", "3D", "JD", "KD", "QD", 2S", "KS", "JS", "QS", "8C", "JC", "QC"]
number of royal court: 3

3b) [optional] This space is for you to plan your algorithm and to identify which components you need to bring together to solve this. You may get partial credit for your algorithm/plan here.
Hint: We talked about this problem in class on Tuesday.(I missed tuesday (puts a sad face))
def function
create a dictionary of list
for each loop
index dictionary, create 2 variables to store data after indexing
if statement
create temporary holder
for each loop
if statement
return

3c) [17 points] Write a function that takes 1 parameter: a list of cards on the ground. It should return how many royal courts can be constructed.

```
# Your code should go here
def count_royal_courts(cards):

    royal_court_counts = {'H': {'J': 0, 'Q': 0, 'K': 0},
                          'D': {'J': 0, 'Q': 0, 'K': 0},
                          'S': {'J': 0, 'Q': 0, 'K': 0},
                          'C': {'J': 0, 'Q': 0, 'K': 0}}
      for card in cards:

        rank = card[:-1]
        suit = card[-1]


        if rank == 'J':
            royal_court_counts[suit]['J'] += 1
        elif rank == 'Q':
```

```
            royal_court_counts[suit]['Q'] += 1
        elif rank == 'K':
            royal_court_counts[suit]['K'] += 1


    royal_courts = 0
    for suit in royal_court_counts:

        if royal_court_counts[suit]['J'] > 0 and royal_court_counts[suit]['Q'] >
0 and royal_court_counts[suit]['K'] > 0:
            royal_courts += 1

    return royal_courts


cards = ["8H", "QD", "JD", "KS", "JS", "QD", "KD", "QS"]
print(count_royal_courts(cards))
```