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 former exam questions. Question 1 was from a previous CIS 2000 midterm.

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. I encourage you to try that out here by just typing into the Google Doc.

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] Assume there exists a variable called `dog_weights` **that is dictionary of dog breeds and their average adult mass in kilograms.**
This is an example of what data it might have, but the data can be different:

```
{
    "Affenpinschers": 4.5,
    "Afghan Hounds": 26,
    "Airedale Terriers": 29,
    "Akitas": 48,
    "Alaskan Malamutes": 33,
    "American English Coonhounds": 25,
    "American Eskimo Dogs (toy)": 3.5,
    "American Eskimo Dogs (mini)": 6,
    "American Eskimo Dogs (standard)": 11,
    # ...
    "Yorkshire Terriers": 3,
}
```

We want to count how many dog breeds are small, medium, or large. A small dog breed has an average weight of less than 10kg, a medium dog breed is between 10kg and 30kg (inclusive) and a large dog breed is greater than 30kg.
In the above example, there are 4 small dog breeds, 4 medium dog breeds, and 2 large dog breeds.

1a) [5 points] Fill out the problem solving question checklist below for question 1. Put your answers in a different color.

1. Read the problem statement. Twice.
    a. What does the input data look like? a dictionary full of strings What data type is it? dictionary What content does it have?Dog breeds and weight
    b. How does it enter our code? in variable d
    c. What are the outputs? if the dog is small, medium or large
    d. How does the output depend on the input? the weight determines what size the dog is
    e. Why is this a problem that we are solving with a computer? faster process and more accuracy
2. Find the right tools for the job.
    a. Have we solved or seen problems like this before? something similar yes
    b. What additional variables do I need beyond the outputs and inputs? for loop, gatherer, steppe,
    c. What questions are we going to have to ask using conditionals? asking if the weight is less than or greater that a set of numbers
    d. What things are we going to need to repeat with loops?
        i. For each? For range? While? For each, if statement
    e. Can I use any variable roles or dictionary patterns to structure the code? yes
    f. What existing functions or methods will be helpful? dictionary methods, list and string methods,len function
3. Plan out the algorithm with some pseudo code.
    1. assume users input is already put in dict. is dog weight
    2 .create gatherers to keep track of dogs size
    3.create a for each loop
4. create if and else statements
5. display result

1b) [15 points] Please type the code that counts how many dog breeds in the dictionary `dog_weights` are classified as small, medium, and large and prints out all three quantities. Your code should work for any string that is in `dog_weights`, not just the example.

```
# dog_weights = {
    "Affenpinschers": 4.5,
    "Afghan Hounds": 26,
    "Airedale Terriers": 29,
    "Akitas": 48,
    "Alaskan Malamutes": 33,
    "American English Coonhounds": 25,
    "American Eskimo Dogs (toy)": 3.5,
    "American Eskimo Dogs (mini)": 6,
    "American Eskimo Dogs (standard)": 11,
    # ...
    "Yorkshire Terriers": 3,
}

small_dog = 0
```

```
medium_dog = 0
large_dog = 0

for weights in dog_weights.values():
    if weights <=10:
        small_dog +=1
    elif 10<= weights <= 30:
        medium_dog +=1
    elif weights >= 30:
        large_dog +=1
print ("There are", small_dog, medium_dog, large_dog)
```

1c) [bonus; 4 points] There was another version of this question: How many dog breeds are "hound"s? That is, how many breed names contain "hound" (not caring about upper case or lower case). In the example, there are 2 hounds. Write the code that counts and prints out how many hounds there are the `dog_weights` **dictionary.**

```
# Your code should go here
dog_weights = {
    "Affenpinschers": 4.5,
    "Afghan Hounds": 26,
    "Airedale Terriers": 29,
    "Akitas": 48,
    "Alaskan Malamutes": 33,
    "American English Coonhounds": 25,
    "American Eskimo Dogs (toy)": 3.5,
    "American Eskimo Dogs (mini)": 6,
    "American Eskimo Dogs (standard)": 11,
    # ...
    "Yorkshire Terriers": 3,
}

hounds_count = 0

for breed in dog_weights.keys():
    if "Hounds" in breed:
        hounds_count += 1

print("There are {hounds_count} hound breeds in the dictionary.")
```

1d) [bonus; 6 points] There was another another version of this question: We want to find the average mass of all dog breeds that have "Terrier" in their name. In the above example, there are 2 dog breeds with "Terrier" in the name ("Airedale Terriers" and "Yorkshire Terriers"), and the average mass of 29 and 3 is (29 + 3)/2 == 16. Write the code that calculates and prints the average mass of all Terriers in the `dog_weights` **dictionary.**

```
# Your code should go here
```

2) [20 points] Suppose we are on a scavenger hunt and we find a letter with some prose on it. Some other clue informs us that we need to find the longest word in the prose that does not have an `'i'` **in it. Assume there exists a variable called** `prose` **with some text**[1] **like:**

```
"There are two parts to learning craftsmanship: knowledge and work. You must gain the knowledge of principles, patterns, practices, and heuristics that a craftsperson knows, and you must also grind that knowledge into your fingers, eyes, and gut by working hard and practicing."
```

The longest word in that example with no `'i'` **in it is** `"craftsperson"`**. Punctuation (periods, commas, exclamation points, colons, semicolons, and question marks) should be ignored and do not count towards the length of a word. If there are multiple words that tie for the longest, your code should print one of 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.Note that a string called prose already exists

1. note string already exists prose
2. Go through all punctuation marks we want to ignore
    1. Remove that character from prose
3. remove that character from prose
4. Break up prose into multiple words
5. Make a variable to keep track of the longest word so far.
    1. Initialize that variable to the empty string
6. Go through all the words from step 3
    1. Ask if the word contains an "i" or an "I"
        1. If not, skip the word
        2. Go through all characters in prose from step 1
    2. Ask if this word is longer than the longest word so far.
        1. If it is, save the word as the longest word so far.
7. Display the longest word found in prose

[1] Introduction to *Clean Code* by Robert C. Martin, page xxvi

2b) Please type the code here that finds the longest word in the existing variable `prose` that does not contain an `'i'` and prints it out. Your code should work for any string that is in `prose`, not just the example. Your code should not print anything else other than the longest word.

```
# Your code should go here
prose = "There are two parts to learning craftsmanship: knowledge and work. You must gain the knowledge of principles, patterns, practices, and heuristics that a craftsperson knows, and you must also grind that knowledge into your fingers, eyes, and gut by working hard and practicing."

prose = prose.replace(".", "")
prose = prose.replace("?", "")
prose = prose.replace(",", "")
prose = prose.replace("!", "")
prose = prose.replace(";", "")



words = prose.split()


longest_word = ""


for word in words:

    if 'i' in word.lower():
        continue


    if len(word) > len(longest_word):
        longest_word = word


print("The longest word that does not contain 'i' is:", longest_word)
```

2c) [bonus; 5 points] Augment your code for 2b to print all words that have the longest length, but do not contain an `'i'`. That is, if the prose also included the word "jackhammered", which is the same length as "craftsperson", your code should print both of those words out (and nothing

else). If the prose contained "fabrications", that word should not be printed out; despite it being the same length, it has an `'i'` in it.

```
# Your code should go here
prose = "There are two parts to learning craftsmanship: knowledge and work. You
must gain the knowledge of principles, patterns, practices, and heuristics that a
craftsperson knows, and you must also grind that knowledge into your fingers,
eyes, and gut by working hard and practicing. Jackhammered and fabrications are
important."

prose = prose.replace(".", "")
prose = prose.replace("?", "")
prose = prose.replace(",", "")
prose = prose.replace("!", "")
prose = prose.replace(";", "")


words = prose.split()


longest_words = []
max_length = 0


for word in words:

    if 'i' in word.lower():
        continue

    if len(word) > max_length:
        longest_words = [word]
        max_length = len(word)
    elif len(word) == max_length:
        longest_words.append(word)

print("The longest words that do not contain 'i' are:",
','.append(longest_words))
```

3) [20 points] Write some code that asks a user to type in a number of players. Then, it should prompt them to enter in the names of those players, one at a time. Finally, it should print out the

players in a random order. Here is an example of what the terminal might look like after running the code; The underlined words are what the user typed in:

```
Enter the number of players: 4
Enter the name of player 1: Kaylee
Enter the name of player 2: Bob
Enter the name of player 3: Emma
Enter the name of player 4: Margie
The players should play in the following order:
Bob
Emma
Kaylee
Margie
```

Your code should handle any positive integer of players to earn full credit. Additionally, your code should show the correct player number when asking the user to type in names and all the player names should be displayed on their own line in order to receive all the points (just follow the formatting above and you'll be fine).

You will need to start your code with `import random`

3a) [5 points] Create an example of the desired terminal output if the user wants to have a 6 player game. Like above, use underlined words to indicate what the user types and a `monospace font` **for all other input and output. Make up whatever names you like.**

```
Enter the number of players: 6
Enter the name of player 1: Kaylee
Enter the name of player 2: Bob
Enter the name of player 3: Emma
Enter the name of player 4: Margie
Enter the name of player 5: Chas
Enter the name of player 6: Ari
The players should play in the following order:
Chas
Emma
Bob
Kaylee
Ari
Margie
```

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: Python gives us a way to shuffle a list, but not a dictionary. Is it in your notes?

1.start code off with import random

2. create a variable and get user to type in how many players they want

3.create a gatherer
4.use a for range loop
5.mix up names the user put in
6. make a for each loop
7.display outcome

3c) [15 points] Write the code to implement the algorithm from question 3.

```
# Your code should go here
import random

num_players = int(input("Enter the number of players: "))

players = []

for i in range(1, num_players + 1):
    player_name = input(f"Enter the name of player {i}: ")
    players.append(player_name)

random.shuffle(players)

print("The players should play in the following order:")
for player in players:
    print(player)
```