*In this workshop, you'll gain practice with nested loops, strings and string methods, and other aspects of Python that we have seen so far. Your workshop leader will guide you through the process. Please **do not** work ahead.*

1. Consider the following code:

```
def printName(name):
    for i in range(5):
        print(i * " ")
        for j in range(3):
            print(name, end=" ")
        print()

name = input("What is your name? ")
printName(name)
```

   (a) How many `for` loops are in the above code?

   (b) Is one `for` loop completely executed before the next loop begins? Briefly explain.

   (c) Label the *inner* loop and the *outer* loop.

   (d) Given an input of `AJ`, what does the program print?

2. Create a Python program that prompts the user for an integer and then prints an inverted right triangle containing that many rows similar to the output below:

```
Enter the number of rows: 5
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

3. Write a program that prompts the user for information about students in a class and their quiz scores. For each student, ask for the student's name, then ask for exactly three quiz grades. To indicate that there are no more students' grades to enter, the user should type 'done' for the student name. After the scores for each student have been entered, the program should print the average score for that student.

   Use a nested loop structure: the outer loop should be a `while` loop (i.e., while there are more students to get scores for) and the inner loop should obtain the three quiz grades for a given student.

   At the end of collecting information for *all* students, print the average score for all quizzes and all students. The average should be rounded to the nearest two decimal places. You can assume that scores for at least one student are entered.

4. Consider the code below:

   ```
   userInput = input("Enter a string that contains only letters: ")
   if onlyLetters(userInput):
       print("Your string is valid")
   else:
       print("Your string is invalid")
   ```

   Write the function `onlyLetters`, used in the code above, which should return `True` if a string only contains letters and `False` if it contains any non-letters. In your solution, loop over the characters of the string *by index*.

5. Write a function named `twoChars` that accepts two parameters: a string, which can be of any length, and a string composed of a single character. The function should determine whether there are two distinct occurrences of the given character in the string and return a boolean. *Use a nested for loop to write this function.*

   For example, `twoChars("!abra!cadabra!", "!")` should return `True`, as should `twoChars("Pineapple", "p")`.