Practical Lab 7
The tasks set out below surround the topics introducing Lists and Tuples. You should attempt *all* tasks over the next week.

In a number of cases you are directed to programs on Blackboard. In each case you should save a copy of the code to your own folder before editing.

<u>Lists and Tuples</u>

Task 1
Design a program that requests a user enter rainfall for the 12 months of the year, which is then entered into a list. The program should process the list to calculate and display the total rainfall for the year, the average monthly rainfall, and finally display the month number (1-12) with the highest and lowest rainfall.

Task 2
You will find a file named `charge_accounts.txt` within *practical lab* folder on BBL. This file contains a list of a company's valid bank account numbers. Each account number is a seven-digit number, e.g. 5658845. Write a program that reads the contents of the file into a list. The program should request the user to enter a charge account number before checking whether the number is valid by searching for it in the list. If the number is in the list, the program should display a message indicating the number is valid. If the number is not in the list, the program should display a message indicating the number is invalid and offer the user a further attempt to enter a correct number.

Task 3
When analysing data it is often desirable to remove the most extreme values prior to performing other calculations. Write a function that accepts a list of values and a non-negative integer, *n*, as its parameters. The function should create a new copy of the list with the *n* largest elements and *n* smallest elements removed. Then it should return the new copy of the list as the function result. Test your program by declaring and list and a value for n, which you pass to the function. *NOTE: The order of the elements in the returned list does not need to match the order of the original list.*

Optional Task: Extend your program to demonstrate your function, reading in a list of values, and a 'n' value from the keyboard. Display the original list and the list with the *n* elements removed from. You program should return an error message if the value entered for n is less than 2.

Task 4
When writing out a list of items in English, one normally separates the items with commas. In addition, the word 'and' is normally included before the last item, unless the list only contains one item. Consider the following examples:

- Car
- Car and boat
- Car, boat and bikes
- Car, boat, bike and plane

Write a function that takes a list of strings as its only parameter. Your function should return a string that contains all of the items in the list formatted in the manner described previously as its only result. Include a main program that reads several items from the user, formats them by calling your function, and then displays the result returned by the function.

Task 5
Refactor your implementations of Tasks 1-4 so that all of the code resides within a single program. Extend the program so that it presents a command line interface menu to users, requesting them to choose between menu options 1-4. A valid menu selection will trigger execution of the relevant Task above and then returns the user to the menu. The user should be able to enter 5 to exit the program, however, if they enter an invalid option they should remain within the program menu.

Task 6
Access the *Week 7 practical folder* and implement both programs discussed in the 'Employee_Prog_Walkthrough' and 'Actor_grades_prog_Walkthrough' videos.

Extend the first program to calculate the average gross pay by:
# Q1. Tally pay amounts
# Q2. Calculate average (Tally / number of employee)
# Q3. Print formatted average salary amount

Extend the second program to as follows:
1. Alter the data file, separating name into first name and surname, and adding in 'year of study' as an additional element.
2. Update the print function to accommodate the changes in step 1.
3. Add a method to prompt the user for input of a 'surname' prior to searching and returning relevant list items should a match be located within the list.