LAB 02

1. Write a python code that will output all numbers included between 2000 and 4000 (both inclusive) and that are, divisible by 9 but are not a multiple of 2.

   The output should be a sequence of comma-separated numbers.

2. Write a python code that implements the following formula and outputs the value of R.

   $$R = \sqrt{\frac{(3 \cdot X \cdot Y)}{H}}$$

   X and H are constant values: X is 70. H is 25.

   The values of Y should be entered by the user, as a sequence of comma-separated numbers.

   Here is an example. If you code takes as input the following sequence of numbers:

   112, 250, 340

   The corresponding output should be:

   31, 46, 53

3. In a computer game, each player is represented by a tuple (name, age, score). Write a python code that sorts a set of tuples in ascending order. Your code should input the tuples as a sequence of three comma-separated values *string, number, number*. Your code should be able to sort the tuples by name, then by age, then by score, according to the following priority *name > age > score*.

   The following is an example input and its corresponding output. Please note that you don't need to input from the user just include the following in the code.

   Anna,18,70

   John,20,80

   Jony,19,95

   Json,23,120

   Then, the output of the program should be:

   [('Anna', 18, 70), ('John', 20, 80), ('Jony', 19, 95), ('Json', 23, 120)]

   Hint: consider using python *itemgetter*.

4. Consider the following two lists:

listA = [5, 23, 7, 89,7, 90]
listB = [89, 25, 89, 7, 14, 76, 5, 11, 5]

Write a python code that takes as input two lists and returns a list that contains only the common elements between listA and listB excluding duplicates. Make sure your code works on two lists of different sizes.

5. In python strings are considered lists. Therefore, any operation you can perform on lists can be applied to strings. For example, it is possible to iterate over the characters of a given string:

```
my_string = "box"
for c in my_string:
  print("letter: " + c)
```

Will give the result:
letter: b
letter: o
letter: x

Write a program that asks the user for a string and print out whether this string is a palindrome or not.
Note: A palindrome is a string that reads the same forwards and backwards. For example, the strings 'deleveled' and 'madam' are palindromes.

6. Given the following:

```
words = [
   'London', 'Oslo', 'Paris', 'Rome', 'Paris', 'Geneva', 'Paris', 'Milano',
   'Geneva', 'Paris', 'Granada', 'Rome', 'Rome', 'London', 'London', 'Geneva',
   'Geneva', "Oslo", 'Rome', 'Oslo', 'Oslo', 'Rome', 'Oslo', 'Rome',
   'Geneva', 'Granada', "Granada", 'London']
```

Write a Python program to count the most common words in a list of words. A sample output should be as follow:

```
[('Rome', 6), ('Geneva', 5), ('Paris', 4), ('Granada', 3)]
```

Hint: Consider using *Counter* from python *collections*. Check documentation here: https://docs.python.org/3.11/library/collections.html#collections.Counter

7. Write a tic-tac-toe game that allows a user [0] to play against the computer [x]. It should produce a similar output as in figure 1.

```
Player is [O] and computer is [X]
  |   |
---------
  |   |
---------
  |   |
---------
# Make your move ! [1-9] : 1
O |   |
---------
  |   |
---------
X |   |
---------
# Make your move ! [1-9] : 2
O | O | X
---------
  |   |
---------
X |   |
---------
# Make your move ! [1-9] : 3
 >> Invalid number ! Try again !
O | O | X
---------
  |   |
---------
X |   |
---------
# Make your move ! [1-9] : 5
O | O | X
---------
  | O |
---------
X | X |
---------
# Make your move ! [1-9] : 9
O | O | X
---------
  | O |
---------
X | X | O
---------
*** Congratulations ! You won ! ***
```

Figure 1: Tic-Tac-Toe

Please note that the numbering of the grid begins from the top left square '1' and increasing by moving to the right. The last bottom right square is labeled '9' (see code output in figure 1).
When it is the computer's turn to play, you can make it occupy a free square randomly.

End.