National University of Computer and Emerging Sciences

Lab Exercise 02
DL2001-Introduction to Data Science Lab

Course Instructor	Ms. Mariam Nasim
Lab Instructor(s)	Ms. Rida Amir
Section	BDS-3A
Semester	Fall 2025

Department of Data Science

FAST-NU, Lahore, Pakistan

Exercise

1. Given an integer count representing the number of eggs, return a string in the format 'Egg count is even' if the number is even, and 'Egg count is odd' if the number is odd.

2. Create a string made of the last three and the first three characters of the original string s and return the new string, so 'intelligence' creates 'cneint'. However, if the string length is less than 4, return instead the empty string.

3. Given a string s, return a new string where all vowels (a, e, i, o, u case-insensitive) are replaced with *, except the first vowel, which should remain unchanged. Assume the input string has at least one vowel.

   e.g. 'Application' creates ' Appl*cat**n'

   Hint: *s.replace(stra, strb) returns a version of string s where all instances of stra have been replaced by strb.*

4. Given two strings a and b (length 2 or more), create a new string in the form a+b and b+a separated by a space. Return the result.

   e.g.

   'abc', 'xyz' → 'abcxyz xyzabc'

   'hi', 'there' → 'hithere therehi'

5. Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with 'a' first.

   e.g., ['mix', 'xyz', 'apple', 'xanadu', 'aardvark'] creates ['aardvark', 'apple', 'mix', 'xanadu', 'xyz']

6. Given a list of non-empty tuples, return a list sorted in decreasing order by the first element in each tuple.
   e.g. [(1, 7), (1, 3), (3, 4, 5), (2, 2)] creates [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
   Hint: *use a custom key= function to extract the last element form each tuple.*

7. Given a nested list of integers, create a function to flatten the list to a single-level list.
   e.g., [[1, 2, 3], [4, 5]] will be flattened to [1, 2, 3, 4, 5].

8. Given two lists of integers, create a function to return a list that contains only the elements that are common between the two lists (without duplicates). Make sure your program works with lists of different sizes.

9. Given two strings, create a function to detect if they are **pangrammatic twins**. Two strings are considered pangrammatic twins if they contain the same set of unique letters, regardless of the number of occurrences and case.
   Ignore spaces, punctuation, and treat uppercase/lowercase letters as the same.
   e.g.,

   'Listen', 'Silent' → returns True
   'School master', 'The classroom' → returns True
   'Hello', 'Ole!' → returns True
   'World', 'Word' → returns False

10. Given a sorted list of integers, create a function to perform **binary search** on the list and return the index of the target element. If the target element is not found, return -1.

11. Given a list of integers, create a function to return the count of distinct elements in the list.
    e.g., the list [1, 2, 2, 3, 4, 2] would return 4.

12. Given two lists of integers, create a function to return the union and intersection of the sets as a tuple of two lists. The union is the set of elements that appear in either set, while the intersection is the set of elements that appear in both sets.

13. Given a dictionary, return the key(s) whose corresponding value(s) are the longest string(s). If multiple keys have values with the same maximum length, return all such keys.
    e.g., {'a': 'apple', 'b': 'banana', 'c': 'pear', 'd': 'grapefruit'} should return ['b', 'd'] since 'banana' and 'grapefruit' both have the longest length (6 and 10 respectively if you adjust the example).

14. Given two dictionaries d1 and d2, return a new dictionary that contains only the common keys from both dictionaries. The values for those keys should be stored as a tuple, where the first element is the value from d1 and the second is from d2.
    e.g.,

    d1 = {'a': 1, 'b': 2, 'c': 3}
    d2 = {'b': 20, 'c': 30, 'd': 40}

Output: {'b': (2, 20), 'c': (3, 30)}

15. Given a list of elements, return a new list containing the elements that appear only once in the original list. The result should not contain any duplicates.
    e.g.,

    Input: [1, 2, 2, 3, 4, 4, 5]
    Output: [1, 3, 5]