# Week-3 and Week-4

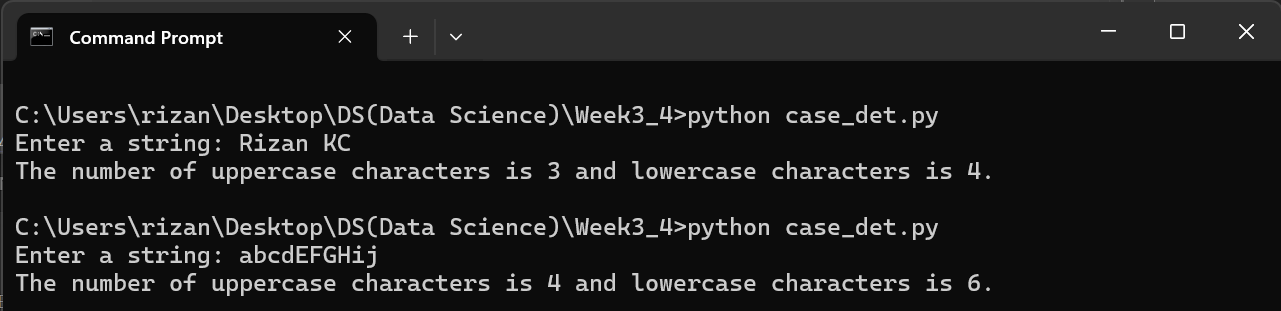
1. File name: case\_det.py

**Description:**

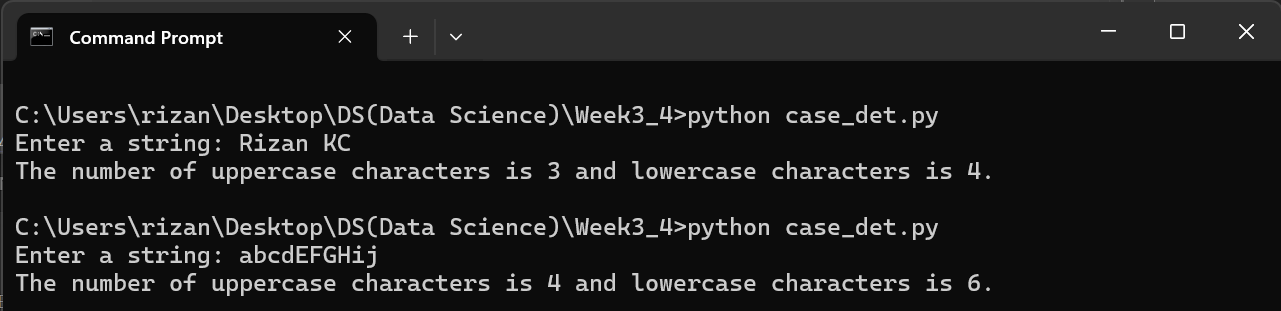
The program counts the number of lowercase and uppercase letters in a string.

The program has function which takes a string as an argument which checks if the letter in the word is uppercase or lowercase. While checking, a counter for both uppercase and lowercase letter is calculated by raising the counter by 1 time based on the check. If a letter is lowercase, then counter for lowercase is raised by 1 and similarly for uppercase. After all the check, the function returns counter values for both.

**Test 1 (Inputs: Rizan KC (uppercase - 3, lowercase - 4), abcdEFGHij(uppercase – 4, lowercase – 6)**



**Test 2 (Inputs: Rizan12345 (uppercase – 1, lowercase – 4)**

****

Based on the test, the program works properly.

1. File name: prime\_det.py

**Description:**

The program checks if the given number is a prime or not.

Prime number are the number that are divisible by only 1 and the number itself. Example: 2,3,5,7,11,13,17…..

The program has a function which takes a number as a parameter which is to be checked. As prime number are the numbers which are divisible by only two numbers. So, we check the number by dividing it numbers ranging from 1 to itself. When the number of divisors becomes two, the function returns the message prime number otherwise not a prime number.

**Test 1 (Inputs: 2,7, 13 are prime numbers)**:

A computer screen with white text

AI-generated content may be incorrect.

**Test 2 (Inputs: 4, 6, 20 are not a prime numbers):**

A computer screen with white text

AI-generated content may be incorrect.

Based on the test, the program works properly.

1. File name: armstrong\_check.py

**Description:**

The program checks if the given number is an Armstrong or not.

An Armstrong number is a number that is equal to sum of it’s own digit where each digits are raise to the power of number of digits in the number.

Example: 153, 370, 1634

The number of digits is 3.

153 = 1^3 + 5^3 + 3^3

153 = 1 + 125 +27

153 = 153

The program has the function which takes the number as a parameter, and analyze the number if it is Armstrong or not. The function returns “Armstrong” if the number is prime or not.

Test is carried out.

**Test 1 (inputs: 153, 370 and 1634):**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Test 2 (inputs: 121 (not an Armstrong number)):**

**A screen shot of a computer

AI-generated content may be incorrect.**

Based on the test, it is found that the program works properly.

1. File name: sort\_names.py

**Description:**

The program sorts a list of names in an alphabetical order.

The program has a function which takes the list of name as an argument and returns the sorted list in an alphabetical order. The function uses .sort() method to sort the names and returns them.

Test is carried out:

**Test:**

**A black screen with white text

AI-generated content may be incorrect.**

Based on the test, the program works properly.

1. File name: calculator.py

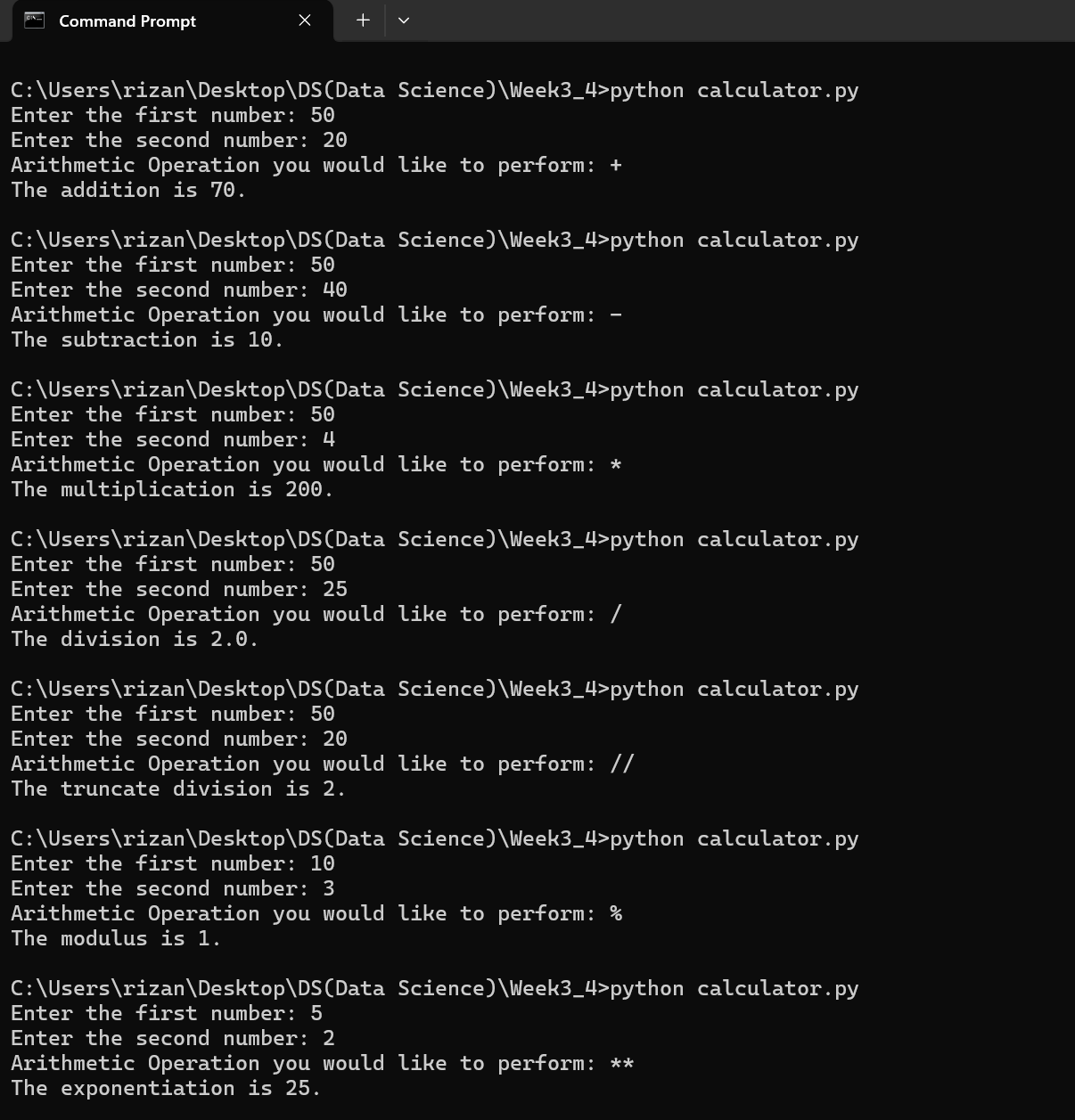
**Description:**

The program calculates the different arithmetic operation from addition, subtraction, division, modulus, multiplication, exponentiation and truncate division on any two given inputs.

The program contains the seven functions for the respective arithmetic operations. Each function takes two argument and performs the operation based on user’s preference. For division, modulus and truncate division, if the divisor is 0 then the function returns invalid message. Otherwise, each of the function returns respective result at function call.

Test is carried out.

**Test 1:**

****

**Test 2:**

**A computer screen with white text

AI-generated content may be incorrect.**

Based on the test, the program works properly.

1. File name: list\_int.py

**Description:**

The program prompts the user for a series of integers and stores in a list where the values should be in between 1 to 100 and displays the resulting list.

The program takes multiple input until the user enters ‘n’. The inputs which are in range of 0 to 100 are only appended on an empty list. After input process is completed, then the final appended list is displayed.

**Test:**

**A computer screen with white text

AI-generated content may be incorrect.**

Based on the text, the program filters out the number which is not in the range of 0 to 100 and then displays the resulting list.

1. File name: count\_a.py

**Description**:

The program prompts the user to enter a list of names and stores the values in a list. Then, the program calculates the number of times the character ‘a’ appears within the list.

The program counts the number of a using loop where we iterate over the list which is typecasted as string and compare each character of the string with ‘a’. With this, the counter of ‘a’ increases and returns the final count of ‘a’ present in the list.

Test:

A screenshot of a computer program

AI-generated content may be incorrect.

Based on the test, the program works properly.

1. File name: list\_manipulation.py

**Description:**

The program prompts the user to enter integer values to populate two lists and print the messages:

1. Whether the lists are of the same length.
2. Whether the elements in each list sum to the same value.
3. Whether there are any values that occur in both lists.

The length of lists are checked using len() function where we compare them using len(list1) == len(list2). If true, the program displays “Both lists are of same length”.

For the sum, we iterate a variable over both the list, and compare the sum of both the lists. For comparing the values which are present in both list, we iterate over both the list in nested loop structure and compare the values. As soon as, it finds the common values, the program prints the message.

**Test 1 (Different length and common value):**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Test 2(same length, all common values and sums up to same values):**

A screenshot of a computer program

AI-generated content may be incorrect.

Based on the test input, the program works fine.

1. File name: record\_avgtemp.py

**Description:**

The program adds the average temperature to the dictionary, if that day is not present in the dictionary or updates the value if the day existed in the dictionary.

The program has function named add\_daily\_temp() function which takes three arguments one is an empty dictionary, second is day of the week while third argument is the value corresponding to day of the week. These corresponding key value pairs are added on the empty dictionary if these pair does not exist or else it will update the value.

**Test:**

**A black screen with white text

AI-generated content may be incorrect.**Based on the test, the program works properly.

1. File name: temperature\_dic.py

**Description:**

The program asks the users to input the average temperature for each day of the week.

The program which has functions get\_daily\_temps() which takes a dictionary and asks user for the temperature. The days as key and temperature as value are added to the dictionary. Then, the function returns the dictionary and displays it.

**Test:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Based on the test, the program works properly.

1. File name: concatenate\_dict.py

**Description:**

The program creates three dictionaries, concatenates all three in the dictionary named ‘nums’. Also, adds a new key value pair (7:70), updates the value of 80 to key 3, removes the third element, sums and multiplies up all the values present in the dictionary ‘nums’ as well as displays the maximum and minimum values in ‘nums’.

The program uses dictionary methods to perform certain operations.

**Test:**

**A computer screen with white text

AI-generated content may be incorrect.**Based on the test, the program works properly.

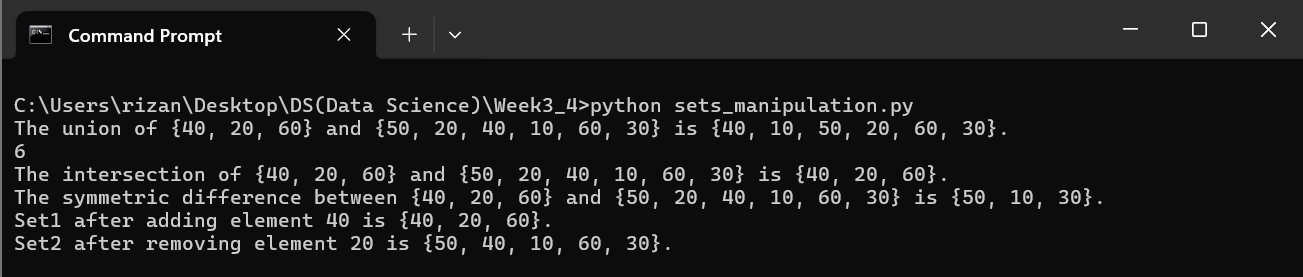
1. File name: sets\_manipulation.py

**Description**:

The program creates two sets, performs different set operations like union, intersection, symmetric difference, adding and removing an element from respective sets.

The program uses sets method to accomplish the required objectives.

**Test:**

****

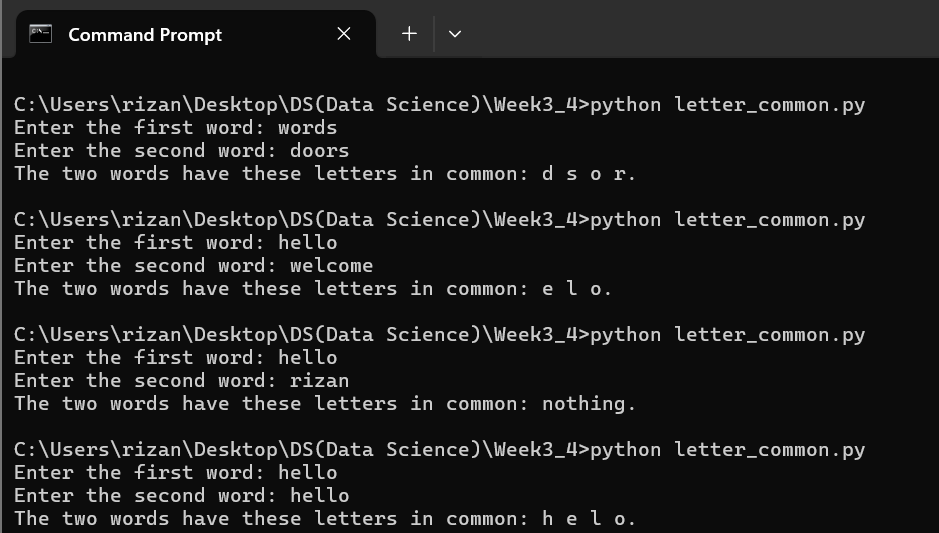
Based on the test, the program works properly.

1. File name: letter\_common.py

**Description:**

The program asks user to input two words and then displays the letters which are common among the two words. It uses sets to find the common element and returns those element and displays it. The function word\_intersection takes two words as an argument and returns the common characters separated by space.

**Test:**

****

Based on the test, the program works properly.