**Lab 3**

**1. Write a function called check-season, it takes a month parameter and returns the season:**

**Autumn, Winter, Spring or Summer.**

**Code:**

def CheckSeason(month) :

if month in ('Dec','Jan','Feb','December','January','February'):

return 'Winter'

if month in ('Jun','Jul','Aug','June','July','August'):

return 'Summer'

if month in ('Mar','Apr','May','April','March'):

return 'Spring'

if month in ('Sep','Oct','Nov','September','October','November'):

return 'Autumm'

month = input("Enter the Month to know the Season : ")

season = CheckSeason(month)

print("The season is : ",season) 

**2. Write a function called calculate\_slope which return the slope of a linear equation.**

**Code:**

def calculate\_slope(x1,x2,y1,y2):

    if x2>x1:

        return (y2-y1)/(x2-x1)

    else :

        print("Wrong value entered...x1 must be less than x2")

        return None

x1,x2,y1,y2 = map(float,input("Enter the values of x1,x2,y1,y2 : ").split())

slope = calculate\_slope(x1,x2,y1,y2)

if slope is not None :

    print("The slope is : ",slope)



**3. Quadratic equation is calculated as follows: ax² + bx + c = 0. Write a function which**

**calculates solution set of a quadratic equation, \_solve\_quadratic\_eqn\_.**

**Code:**

import math

def solve\_quadratic\_eqn(a, b, c):

discriminant = b\*\*2 - 4\*a\*c

if discriminant > 0:

# Two distinct real roots

root1 = (-b + math.sqrt(discriminant)) / (2\*a)

root2 = (-b - math.sqrt(discriminant)) / (2\*a)

return (root1, root2)

elif discriminant == 0:

# One real root (repeated)

root = -b / (2\*a)

return root

else:

# Complex roots

real\_part = -b / (2\*a)

imaginary\_part = math.sqrt(-discriminant) / (2\*a)

root1 = (real\_part, imaginary\_part)

root2 = (real\_part, -imaginary\_part)

return (root1, root2)

a, b, c = map(float, input("Enter the coefficients a, b, c: ").split())

solutions = solve\_quadratic\_eqn(a, b, c)

print("The solutions are:", solutions)



**4. Declare a function named print\_list. It takes a list as a parameter and it prints out each**

**element of the list.**

**Code:**

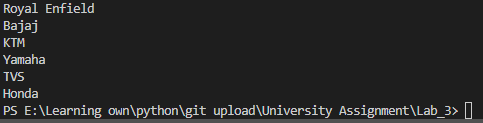
def print\_list(bikeList) :

for ele in bikeList :

print(ele)

bikeList = ['Royal Enfield','Bajaj','KTM','Yamaha','TVS','Honda']

print\_list(bikeList)

****

**5. Declare a function named reverse\_list. It takes an array as a parameter and it returns the**

**reverse of the array (use loops).**

**Code:**

def reverse\_list(arr) :

rev\_arr = []

for i in range (len(arr)-1,0):

rev\_arr.append(arr[i])

return rev\_arr

arr = [4,5,6,7,8,9]

rev\_arr = reverse\_list(arr)

print("Original : ",arr)

print("Reversed : ",rev\_arr) 

**6. Compute the sum up to n terms in the series**

**1 - 1/2 + 1/3 - 1/4 + 1/5 -... 1/n where n is a positive integer and input by user.**

**Code:**

def compute\_series\_sum(n):

total\_sum = 0.0

for i in range(1, n + 1):

term = (-1) \*\* (i + 1) / i

total\_sum += term

return total\_sum

n = int(input("Enter the number of terms (positive integer): "))

if n > 0:

result = compute\_series\_sum(n)

print(f"The sum of the series up to {n} terms is: {result:.6f}")

else:

print("Invalid input. The number of terms must be a positive integer.")



**7. Write a program to compute sin x for given x. The user should supply x and a positive integer**

**n. We compute the sine of x using the series and the computation should use all terms in the**

**series up through the term involving xn**

**sin x = x - x3/3! + x5/5! - x7/7! + x9/9! ........**

**Code:**

def factorial(n) :

if n==0 or n==1 :

return 1

else :

fact = 1

for i in range(2,n+1) :

fact = fact \* i

return fact

def sin\_calculate(x,n) :

sin\_x = 0

for i in range (n) :

sign = (-1) \*\* i

term = sign \* (x\*\*(2\*i+1))/factorial(2\*i+1)

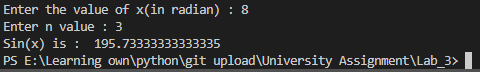
sin\_x = sin\_x + term

return sin\_x

x = float(input("Enter the value of x(in radian) : "))

n = int(input("Enter n value : "))

sinX = sin\_calculate(x,n)

print("Sin(x) is : ",sinX)

**8. Write a program to compute cosine of x. The user should supply x and a positive integer n.**

**We compute the cosine of x using the series and the computation should use all terms in the**

**series up through the term involving xn**

**cos x = 1 - x2/2! + x4/4! - x6/6! ....**

**Code:**

def factorial(n) :

if n==0 or n==1 :

return 1

else :

fact = 1

for i in range(2,n+1) :

fact = fact \* i

return fact

def cos\_calculate(x,n) :

cos\_x = 0

for i in range (n) :

sign = (-1) \*\* i

term = sign \* (x\*\*(2\*i))/factorial(2\*i)

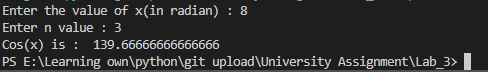
cos\_x = cos\_x + term

return cos\_x

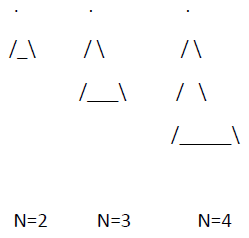
x = float(input("Enter the value of x(in radian) : "))

n = int(input("Enter n value : "))

cosX = cos\_calculate(x,n)

print("Cos(x) is : ",cosX) 

**9. Print the pattern upto N Lines:**

****

**Code:**

def printPattern(n) :

gap = n-1

for i in range (0,n):

#1st line

if(i==0):

for k in range (gap) :

print(' ', end=' ')

print(".")

gap = gap -1

continue

#last line

if(i==n-1):

for k in range (gap):

print(' ', end=' ')

gap = gap-1

print("/", end=' ')

for j in range(0,2\*i-1):

print("\_", end=' ')

print("\\", end=' ')

print()

continue

#middle lines

if(n>i):

for k in range (gap):

print(' ', end=' ')

gap = gap-1

print("/", end=' ')

for j in range(0,2\*i-1):

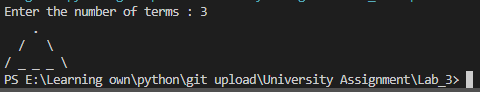
print(" ", end=' ')

print("\\", end=' ')

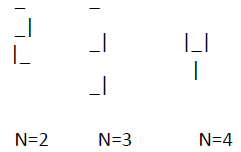
print()

continue

n =int(input("Enter the number of terms : "))

printPattern(n) 

**10. Print a number as a 8 segment display N Lines:**

****

**Code:**

segment = {

  '0' :[' \_','| |','|\_|'],

  '1' :[' ',' |',' |'],

  '2' :[' \_',' \_|','|\_'],

  '3' :['\_','\_|','\_|'],

  '4' :['','|\_|','  |'],

  '5' :[' \_','|\_',' \_|'],

  '6' :[' \_','|\_','|\_|'],

  '7' :['\_',' |',' |'],

  '8' :[' \_','|\_|','|\_|'],

  '9' :[' \_','|\_|',' \_|']

}

def print\_8\_seg(num) :

    rows = ['','','']

    for digit in str(num) :

        for j in range(3):

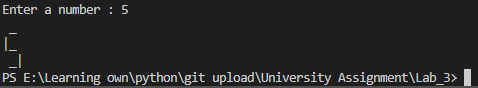
            rows[j] = segment[digit][j]

    for row in rows:

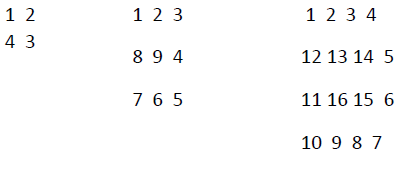
        print(row)

N = int(input("Enter a number : "))

print\_8\_seg(N)



**11. Print the pattern upto N lines:**

****

**Code:**

def spiral\_matrix(n):

    matrix = [[0] \* n for \_ in range(n)]

    num = 1

    top, bottom, left, right = 0, n - 1, 0, n - 1

    while top <= bottom and left <= right:

        # left to right

        for i in range(left, right + 1):

            matrix[top][i] = num

            num += 1

        top += 1

        # top to bottom

        for i in range(top, bottom + 1):

            matrix[i][right] = num

            num += 1

        right -= 1

        # right to left

        for i in range(right, left - 1, -1):

            matrix[bottom][i] = num

            num += 1

        bottom -= 1

        # bottom to top

        for i in range(bottom, top - 1, -1):

            matrix[i][left] = num

            num += 1

        left += 1

    return matrix

def print\_matrix(matrix):

    for row in matrix:

        print(" ".join(map(str, row)))

N = int(input("Enter the number of lines N: "))

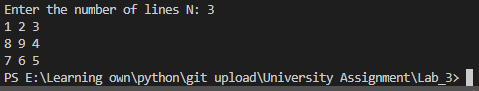
if N > 0:

    spiral\_matrix = spiral\_matrix(N)

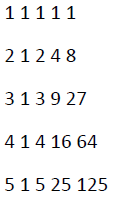
    print\_matrix(spiral\_matrix)

else:

    print("N should be a positive integer.")



**12. Write a python script that displays the following table**

****

**Code:**

def print\_matrix(matrix):

for row in matrix:

print(" ".join(map(str, row)))

n = 5

matrix = [[0] \* n for \_ in range(n)]

#1st column

for i in range (0,n) :

matrix[i][0] =i+1

#2nd column

for i in range (0,n) :

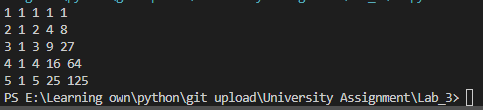
matrix[i][1] = 1

#other column

for i in range (0,n):

for j in range(2,n):

matrix[i][j] = matrix[i][0] \* matrix[i][j-1]

print\_matrix(matrix) 

**Java TO Python**

**Week 2**

**20.** **Write a Java program to read two integer values m and n and to decide and print whether m is multiple of n.**

**Code:**

m = int(input("Enter the first integer (m): "))

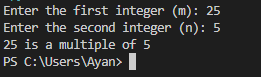
n = int(input("Enter the second integer (n): "))

if n != 0 and m % n == 0:

print(f"{m} is a multiple of {n}")

else:

print(f"{m} is not a multiple of {n}")



**21.** **Write a Java program to display prime numbers between a given interval.**

**Code:**

def is\_prime(num):

    if num <= 1:

        return False

    for i in range(2, int(num\*\*0.5) + 1):

        if num % i == 0:

            return False

    return True

def display\_primes(lower, upper):

    print(f"Prime numbers between {lower} and {upper} are:")

    for num in range(lower, upper + 1):

        if is\_prime(num):

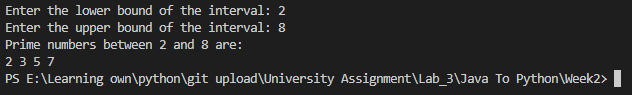
            print(num, end=" ")

    print()

lower = int(input("Enter the lower bound of the interval: "))

upper = int(input("Enter the upper bound of the interval: "))

display\_primes(lower, upper)



**22.** **Write a Java program to check whether a given number is Armstrong Number or not.**

**Code:**

def is\_armstrong(number):

digits = str(number)

num\_digits = len(digits)

armstrong\_sum = sum(int(digit) \*\* num\_digits for digit in digits)

return armstrong\_sum == number

number = int(input("Enter a number: "))

if is\_armstrong(number):

print(f"{number} is an Armstrong number.")

else:

print(f"{number} is not an Armstrong number.")



**Write Java programs for the patterns given bellow: (23-25)**

**23.** ****

**Code:**

def generate\_pattern(rows):

current\_number = 1

for i in range(rows):

num\_elements = 2 \* i + 1

for \_ in range(num\_elements):

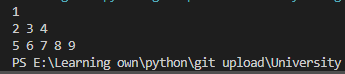
print(current\_number, end=" ")

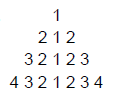
current\_number += 1

print()

num\_rows = 3

generate\_pattern(num\_rows)



**24.** ****

**Code:**

def generate\_pattern(n):

for i in range(1, n + 1):

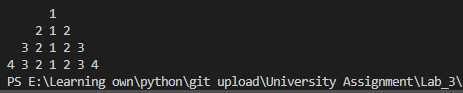
row = list(range(i, 0, -1)) + list(range(2, i + 1))

row\_str = " ".join(map(str, row))

leading\_spaces = " " \* (2 \* (n - i))

print(leading\_spaces + row\_str)

num\_rows = 4

generate\_pattern(num\_rows) 

**25.** ****

**Code:**

def generate\_pattern(n):

for i in range(1, n + 1):

row = f"{i}"

leading\_spaces = " " \* (i - 1)

internal\_spaces = " " \* (2 \* (n - i) - 1)

if i < n:

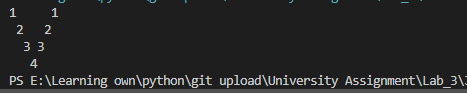
row = leading\_spaces + row + internal\_spaces + row

else:

row = leading\_spaces + row

print(row)

num\_rows = 4

generate\_pattern(num\_rows) 

**Week 3**

1. **Write a Java program to calculate Sum & Average of an integer array.**

**Code:**

def calculate\_sum\_and\_average(arr):

total\_sum = sum(arr)

average = total\_sum / len(arr) if len(arr) > 0 else 0

return total\_sum, average

n = int(input("Enter the number of elements in the array: "))

arr = []

for i in range(n):

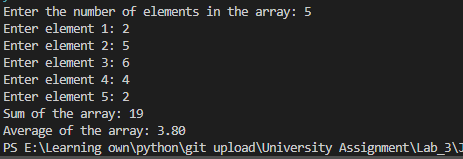
element = int(input(f"Enter element {i+1}: "))

arr.append(element)

total\_sum, average = calculate\_sum\_and\_average(arr)

print(f"Sum of the array: {total\_sum}")

print(f"Average of the array: {average:.2f}")

****

1. **Write a Java program to implement stack using array.**

**Code :**

**Code:**

stack = []

def push(element):

stack.append(element)

print(f"{element} is pushed to the stack.")

def pop():

if len(stack) == 0:

print("Stack is empty. Cannot pop.")

else:

element = stack.pop()

print(f"{element} is popped from the stack.")

def peek():

if len(stack) == 0:

print("Stack is empty. Nothing to peek.")

return None

return stack[-1]

def display():

if len(stack) == 0:

print("Stack is empty!")

else:

print("Displaying stack:")

for i in stack:

print(i, end=" ")

print()

while True:

print("\n-----------")

print("Stack Menu:")

print("-----------")

print("1. Push")

print("2. Pop")

print("3. Peek")

print("4. Display")

print("5. Exit")

try:

choice = int(input("Enter your choice: "))

except ValueError:

print("Invalid input. Please enter a number.")

continue

if choice == 1:

element = int(input("Enter the element: "))

push(element)

elif choice == 2:

pop()

elif choice == 3:

result = peek()

if result is not None:

print("Peek is:", result)

elif choice == 4:

display()

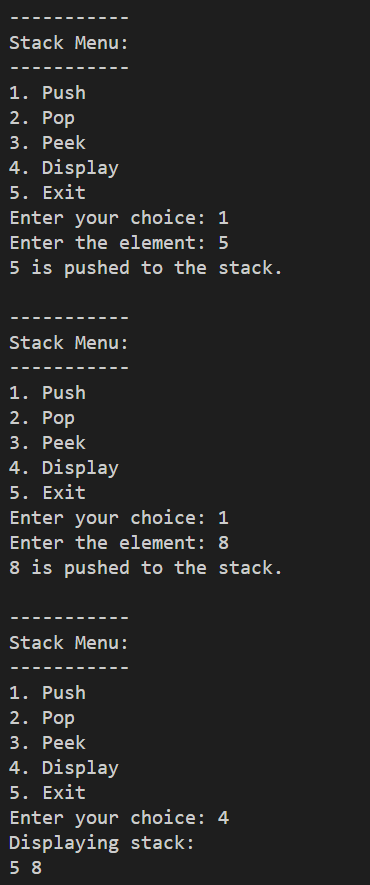
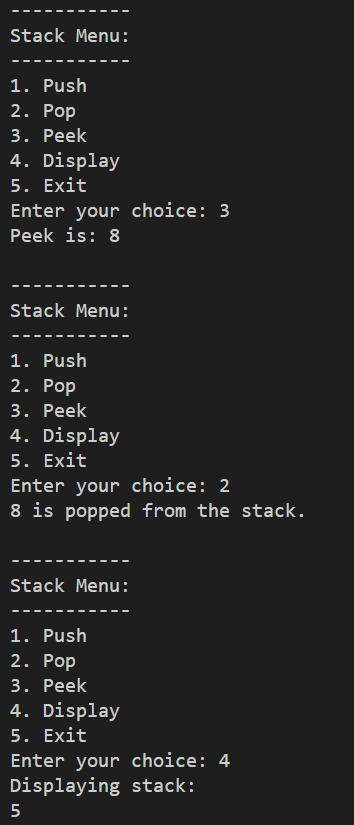
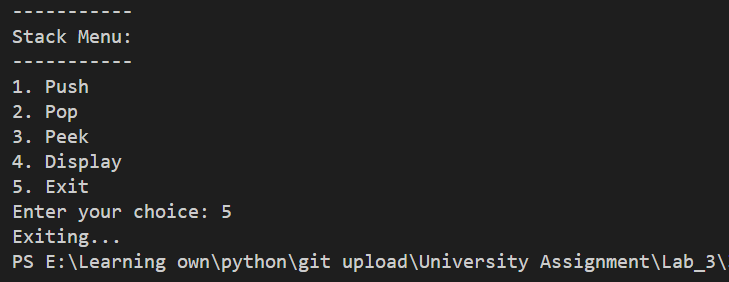
elif choice == 5:

print("Exiting...")

break

else:

print("Invalid choice. Please enter a number between 1 and 5.")

**** **** 

1. **Write a Java program to implement Queue using array.**

**Code:**

queue = []

def enqueue(element):

queue.append(element)

print(f"{element} has been enqueued to the queue.")

def dequeue():

if len(queue) == 0:

print("Queue is empty. Cannot dequeue.")

else:

element = queue.pop(0)

print(f"{element} has been dequeued from the queue.")

def peek():

if len(queue) == 0:

print("Queue is empty. Nothing to peek.")

return None

return queue[0]

def display():

if len(queue) == 0:

print("Queue is empty!")

else:

print("Displaying queue:")

for i in queue:

print(i, end=" ")

print()

while True:

print("\n-----------")

print("Queue Menu:")

print("-----------")

print("1. Enqueue")

print("2. Dequeue")

print("3. Peek")

print("4. Display")

print("5. Exit")

try:

choice = int(input("Enter your choice: "))

except ValueError:

print("Invalid input. Please enter a valid number.")

continue

if choice == 1:

element = int(input("Enter the element: "))

enqueue(element)

elif choice == 2:

dequeue()

elif choice == 3:

result = peek()

if result is not None:

print("Peek is:", result)

elif choice == 4:

display()

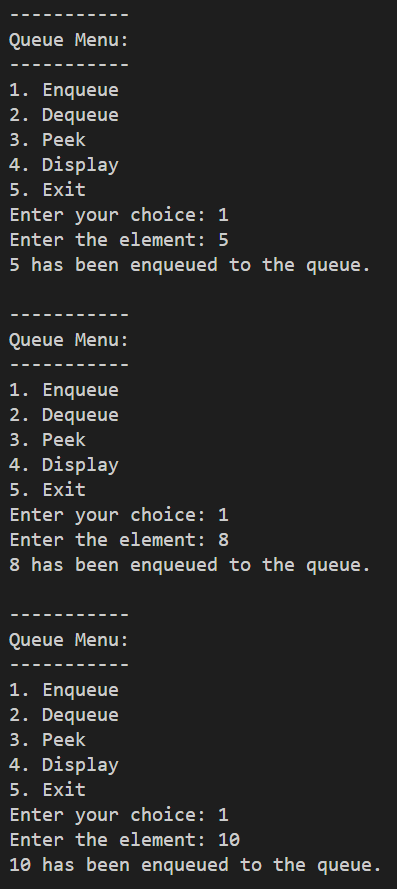
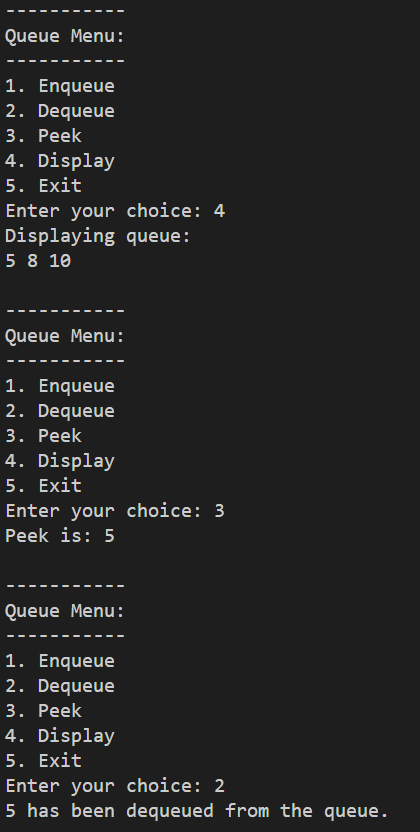
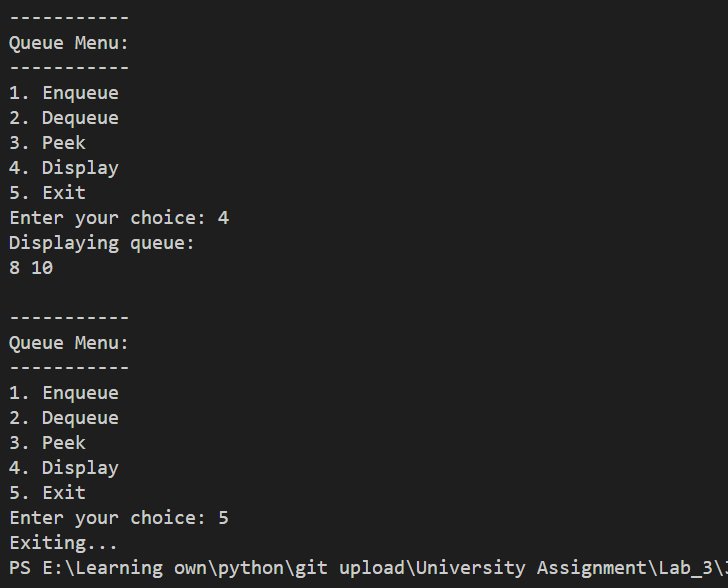
elif choice == 5:

print("Exiting...")

break

else:

print("Invalid choice. Please enter a number between 1 and 5.")

**** **** 

**Lab 4**

1. **Write a program to enter a string. Calculate the length of the string. Find the substring country.Count the occurences of each word in the given sentence.**

**If the String as input is India is my motherland. I love my country. Capital of India is New Delhi.**

**Code :**

strr = input("Enter the string : ")

lenght = len(strr)

print(f"lenght is {lenght}")

subString = "country"

index = strr.lower().find(subString)

print(f"Position of {subString} is {index}")

import string

translator = str.maketrans('', '', string.punctuation)

cleaned\_string = strr.lower().translate(translator)

words = cleaned\_string.split()

word\_count = {}

for word in words:

if word in word\_count:

word\_count[word] += 1

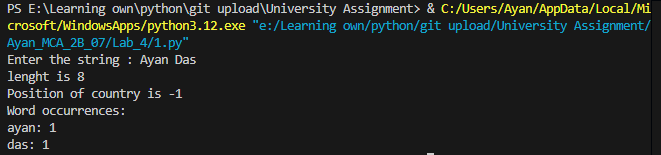
else:

word\_count[word] = 1

print("Word occurrences:")

for word, count in word\_count.items():

print(f"{word}: {count}")



1. **Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically. Suppose the following input is supplied to the program:**

**without,hello,bag,world**

**Then, the output should be:**

**bag,hello,without,world**

**Code :**

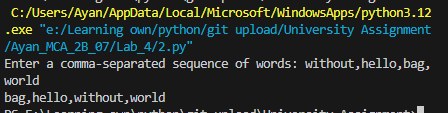
input\_string = input("Enter a comma-separated sequence of words: ")

words = input\_string.split(',')

sorted\_words = sorted(words)

output\_string = ','.join(sorted\_words)

print(output\_string)



1. **Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized. Suppose the following input is supplied to the program:**

**Hello world**

**Practice makes perfect**

**Then, the output should be:**

**HELLO WORLD**

**PRACTICE MAKES PERFECT**

**Code :**

input\_lines = []

print("Enter the lines (Press Enter twice to stop):")

while True:

line = input()

if line:

input\_lines.append(line)

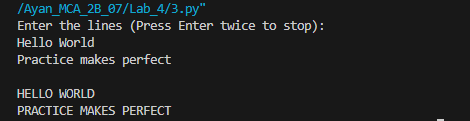
else:

break

output\_lines = [line.upper() for line in input\_lines]

for line in output\_lines:

print(line)

****

1. **Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically. Suppose the following input is supplied to the program:**

**hello world and practice makes perfect and hello world again**

**Then, the output should be:**

**again and hello makes perfect practice world**

**Code :**

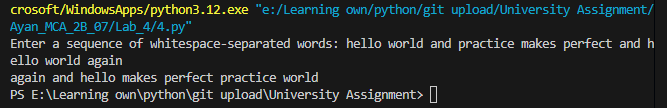
input\_string = input("Enter a sequence of whitespace-separated words: ")

words = input\_string.split()

unique\_sorted\_words = sorted(set(words))

output\_string = ' '.join(unique\_sorted\_words)

print(output\_string)

****

1. **Write a program that accepts a sentence and calculate the number of letters and digits. Suppose the following input is supplied to the program:**

**hello world! 123**

**Then, the output should be:**

**LETTERS 10**

**DIGITS 3**

**Code :**

input\_string = input("Enter a sentence: ")

letters = 0

digits = 0

for char in input\_string:

    if char.isalpha():

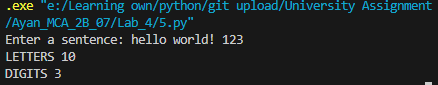
        letters += 1

    elif char.isdigit():

        digits += 1

print(f"LETTERS {letters}")

print(f"DIGITS {digits}")

****

1. **Write a program which accepts a string as input to print &quot;Yes&quot; if the string is &quot;yes&quot; or &quot;YES&quot; or &quot;Yes&quot;, otherwise print &quot;No&quot;.**

**Code :**

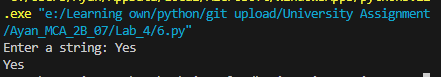
input\_string = input("Enter a string: ")

if input\_string.lower() == "yes":

print("Yes")

else:

print("No"))

****

1. **Write a program which accepts a sequence of words separated by whitespace as input to print the words composed of digits only.**

**Example:**

**If the following words is given as input to the program:**

**2 cats and 3 dogs.**

**Then, the output of the program should be:**

**[‘2’,’3’]**

**In case of input data being supplied to the question, it should be assumed to be a console input.**

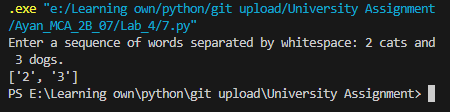
**Code :**

input\_string = input("Enter a sequence of words separated by whitespace: ")

words = input\_string.split()

digit\_words = [word for word in words if word.isdigit()]

print(digit\_words)

****

1. **Please write a program which count and print the numbers of each character in a string input by console.**

**Example:**

**If the following string is given as input to the program:**

**abcdefgabc**

**Then, the output of the program should be:**

**a,2**

**c,2**

**b,2**

**e,1**

**d,1**

**g,1**

**f,1**

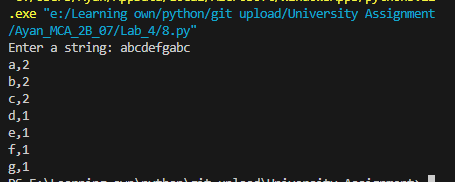
**Code :**

from collections import Counter

input\_string = input("Enter a string: ")

char\_count = Counter(input\_string)

for char in sorted(char\_count.keys()): print(f"{char},{char\_count[char]}")

****

1. **Write a program that accepts a string**
2. **reverses it.**
3. **checks whether it is a palindrome.**
4. **checks whether it ends with a specific substring.**
5. **capitalize the first letter of each word in a string**
6. **check if a string is anagram of another string**
7. **remove vowels from string**
8. **find length of the longest word in a sentence**

**Code :**

def reverse\_string(s):

# Reverse the given string.

return s[::-1]

def is\_palindrome(s):

# Check if the given string is a palindrome.

return s == s[::-1]

def ends\_with\_substring(s, substring):

# Check if the string ends with the given substring.

return s.endswith(substring)

def capitalize\_words(s):

# Capitalize the first letter of each word in the string.

return ' '.join(word.capitalize() for word in s.split())

def is\_anagram(s1, s2):

# Check if s1 is an anagram of s2.

return sorted(s1) == sorted(s2)

def remove\_vowels(s):

# Remove vowels from the given string.

vowels = 'aeiouAEIOU'

return ''.join(char for char in s if char not in vowels)

def longest\_word\_length(sentence):

# Find the length of the longest word in a sentence.

words = sentence.split()

return max(len(word) for word in words) if words else 0

if \_\_name\_\_ == "\_\_main\_\_":

input\_string = input("Enter a string: ")

reversed\_string = reverse\_string(input\_string)

print("Reversed string:", reversed\_string)

palindrome\_status = is\_palindrome(input\_string)

print("Is palindrome:", palindrome\_status)

substring = input("Enter a substring to check if the string ends with it: ")

ends\_with\_status = ends\_with\_substring(input\_string, substring)

print("Ends with substring:", ends\_with\_status)

capitalized\_string = capitalize\_words(input\_string)

print("Capitalized words:", capitalized\_string)

another\_string = input("Enter another string to check if it's an anagram: ")

anagram\_status = is\_anagram(input\_string, another\_string)

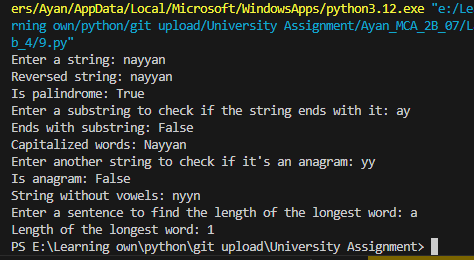
print("Is anagram:", anagram\_status)

string\_without\_vowels = remove\_vowels(input\_string)

print("String without vowels:", string\_without\_vowels)

sentence = input("Enter a sentence to find the length of the longest word: ")

longest\_word\_len = longest\_word\_length(sentence)

print("Length of the longest word:", longest\_word\_len)****