**LAB CYCLE 1￼**

**1. Program to Print all non-Prime Numbers in an Interval.**

**Code:**

print("Name:Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch:2022-2024")

lower\_value=int(input("Enter the Lowest Range value:"))

upper\_value=int(input("Enter the upper Range value:"))

print("The non-prime numbers in range are:")

for number in range (lower\_value,upper\_value + 1):

if number > 1:

is\_prime = True

for i in range(2, number):

if (number % i) == 0:

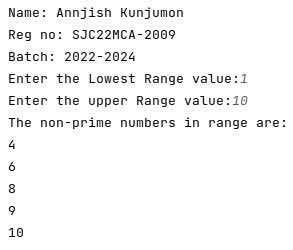
is\_prime = False

break

if not is\_prime:

print(number)

**Output:**



**2. Program to print the first N Fibonacci numbers.**

**Code:**

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

n= int(input("Limit of series:\n"))

a=0

b=1

count=0

if(n <= 0):

print("Invalid input..")

else:

print("Fibonacci Series.....")

while(count < n):

print(a)

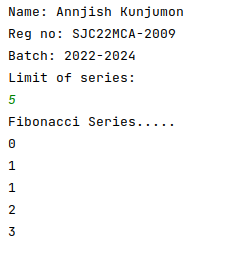
next=a+b

a=b

b=next

count=count+1

**Output:**



**3. Given sides of a triangle, write a program to check whether given triangle is an**

**isosceles, equilateral or scalene.**

**Code:**

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

print("Input length of Triangle Sides:")

x=int(input("x:"))

y=int(input("y:"))

z=int(input("z:"))

if( x == y == z):

print("Equilateral triangle")

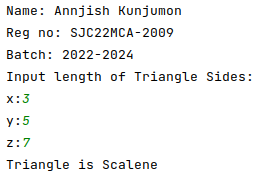
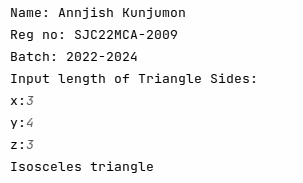
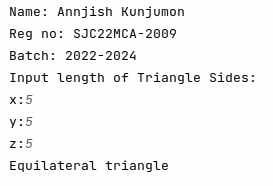
elif(x == y or y==z or z==x):

print("Isosceles triangle")

else:

print("Triangle is Scalene")

**Output:**



**4. Program to check whether given pair of number is coprime.**

**Code:**

import math

def are\_coprime(a, b):

gcd = math.gcd(a, b)

return gcd == 1

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

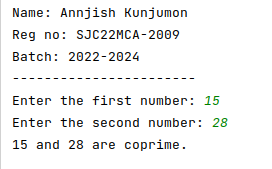
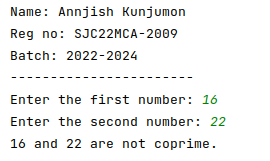
if are\_coprime(num1, num2):

print(f"{num1} and {num2} are coprime.")

else:

print(f"{num1} and {num2} are not coprime.")

**Output:**



**5. Program to find the roots of a quadratic equation(rounded to 2 decimal places)**

**Code:**

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

import math

a = float(input("Enter value of a: "))

b = float(input("Enter value of b: "))

c = float(input("Enter value of c: "))

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

if discri > 0:

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

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

print(f"Root 1: {round(root1, 2)}")

print(f"Root 2: {round(root2, 2)}")

elif discri == 0:

root = -b / (2\*a)

print(f"Root: {round(root, 2)}")

else:

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

img\_part = math.sqrt(-discri) / (2\*a)

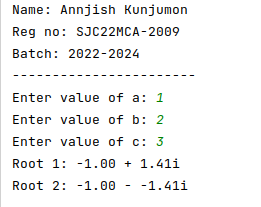
root1 = complex(real\_part, img\_part)

root2 = complex(real\_part, -img\_part)

print(f"Root 1: {root1.real:.2f} + {root1.imag:.2f}i")

print(f"Root 2: {root2.real:.2f} - {root2.imag:.2f}i")

**Output:**



**6. Program to check whether a given number is perfect number or not(sum of factors=number)**

**Code:**

def is\_perfect\_number(num):

if num <= 0:

return False

sum\_of\_divisors = 0

for i in range(1, num):

if num % i == 0:

sum\_of\_divisors += i

return sum\_of\_divisors == num

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

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

if is\_perfect\_number(num):

print(f"{num} is a perfect number.")

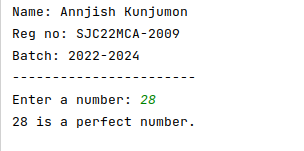
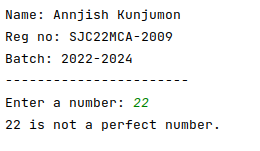
else:

print(f"{num} is not a perfect number.")

except ValueError:

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

**Output:**



**7. Program to display Armstrong numbers upto 1000.**

**Code:**

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

print("Arm strong numbers upto 1000 are:")

for i in range(1,1000):

n=i

sum=0

while(i>0):

rem= i % 10

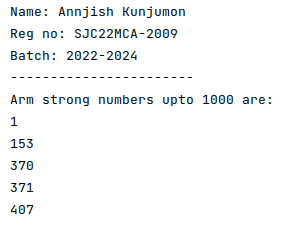
sum= sum + (rem \* rem \* rem)

i= i//10

if (sum == n):

print(n)

**Output:**



**8. Store and display the days of a week as a List, Tuple, Dictionary, Set. Also demonstrate different ways to store values in each of them. Display its type also.**

**Code:**

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

days\_list = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

print("List:", days\_list)

print("Type:", type(days\_list))

days\_tuple = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

print("Tuple:", days\_tuple)

print("Type:", type(days\_tuple))

days\_dict = {0: "Monday", 1: "Tuesday", 2: "Wednesday", 3: "Thursday", 4: "Friday", 5: "Saturday", 6: "Sunday"}

print("Dictionary:", days\_dict)

print("Type:", type(days\_dict))

days\_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}

print("Set:", days\_set)

print("Type:", type(days\_set))

**Output:**



**9. Write a program to add elements of given 2 lists.**

**Code:**

def add\_lists(list1, list2):

if len(list1) != len(list2):

return None

result = []

for i in range(len(list1)):

result.append(list1[i] + list2[i])

return result

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

list1 = input("Enter the first list of numbers separated by spaces: ").split()

list1 = [int(x) for x in list1]

list2 = input("Enter the second list of numbers separated by spaces: ").split()

list2 = [int(x) for x in list2]

result = add\_lists(list1, list2)

if result is None:

print("The lists have different lengths and cannot be added.")

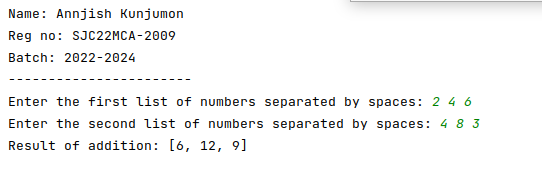
else:

print("Result of addition:", result)

except ValueError:

print("Invalid input. Please enter valid numbers separated by spaces.")

**Output:**



**10. Write a program to find the sum of 2 matrices using nested List.**

**Code:**

def add\_matrices(matrix1, matrix2):

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):

return None

result = [[0 for \_ in range(len(matrix1[0]))] for \_ in range(len(matrix1))]

for i in range(len(matrix1)):

for j in range(len(matrix1[0])):

result[i][j] = matrix1[i][j] + matrix2[i][j]

return result

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

rows = int(input("Enter the number of rows: "))

cols = int(input("Enter the number of columns: "))

print("Enter elements of the first matrix:")

matrix1 = []

for i in range(rows):

row = input(f"Enter elements of row {i + 1} separated by spaces: ").split()

matrix1.append([int(x) for x in row])

print("Enter elements of the second matrix:")

matrix2 = []

for i in range(rows):

row = input(f"Enter elements of row {i + 1} separated by spaces: ").split()

matrix2.append([int(x) for x in row])

result = add\_matrices(matrix1, matrix2)

if result is None:

print("Matrix dimensions are not compatible for addition.")

else:

print("Sum of matrices:")

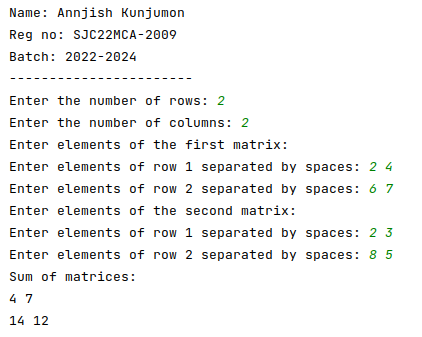
for row in result:

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

except ValueError:

print("Invalid input. Please enter valid numbers.")

**Output:**



**11. Write a program to perform bubble sort on a given set of elements.**

**Code:**

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(0, n - i - 1):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

swapped = True

if not swapped:

break

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

elements = input("Enter elements separated by spaces: ").split()

elements = [int(x) for x in elements]

bubble\_sort(elements)

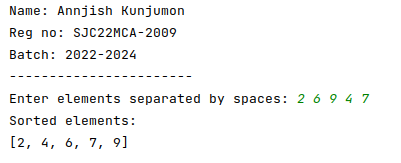
print("Sorted elements:")

print(elements)

except ValueError:

print("Invalid input. Please enter valid numbers separated by spaces.")

**Output:**



**12. Program to find the count of each vowel in a string(use dictionary).**

**Code:**

def count\_vowels(string):

vowel\_counts = {'A': 0, 'E': 0, 'I': 0, 'O': 0, 'U': 0}

string = string.upper()

for char in string:

if char in vowel\_counts:

vowel\_counts[char] += 1

return vowel\_counts

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

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

vowel\_counts = count\_vowels(input\_string)

print("Vowel counts:")

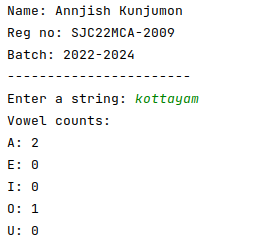
for vowel, count in vowel\_counts.items():

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

except ValueError:

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

**Output:**



**13. Write a Python program that accept a positive number and subtract from this**

**number the sum of its digits and so on. Continues this operation until the number is positive**

**(eg: 256-&gt;2+5+6=13**

**256-13=243**

**243-9=232……..)**

**Code:**

def sum\_of\_digits(n):

digit\_sum = 0

while n > 0:

digit\_sum += n % 10

n //= 10

return digit\_sum

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

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

if num <= 0:

print("Please enter a positive number.")

else:

while num > 0:

digit\_sum = sum\_of\_digits(num)

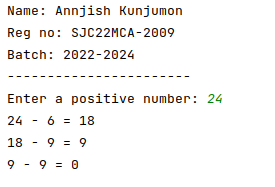
print(f"{num} - {digit\_sum} = {num - digit\_sum}")

num -= digit\_sum

except ValueError:

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

**Output:**



**14. Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number.**

**Code:**

def find\_absent\_digits(mobile\_number):

all\_digits = set("0123456789")

mobile\_digits = set(mobile\_number)

absent\_digits = all\_digits - mobile\_digits

return sorted(list(absent\_digits))

try:

print("Name: Annjish Kunjumon")

print("Reg no: SJC22MCA-2009")

print("Batch: 2022-2024")

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

mobile\_number = input("Enter a 10-digit mobile number: ")

if len(mobile\_number) == 10 and mobile\_number.isdigit():

absent\_digits = find\_absent\_digits(mobile\_number)

if absent\_digits:

print("Absent digits in the mobile number:", ', '.join(absent\_digits))

else:

print("The mobile number contains all digits from 0 to 9.")

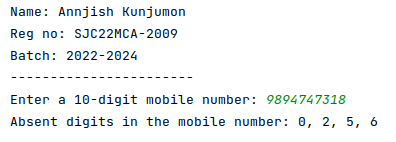
else:

print("Invalid input. Please enter a valid 10-digit mobile number.")

except ValueError:

print("Invalid input. Please enter a valid 10-digit mobile number.")

**Output:**



**LAB CYCLE 2**

**1. Create a three dimensional array specifying float data type and print it.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

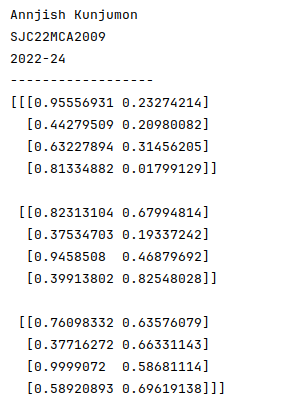
print("------------------")

x, y, z = 3, 4, 2

array\_3d = np.random.rand(x, y, z).astype(float)

print(array\_3d)

**Output:**

****

**2. Create a 2 dimensional array (2X3) with elements belonging to complex data type**

**and print it. Also display**

**a. the no: of rows and columns**

**b. dimension of an array**

**c. reshape the same array to 3X2**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

complex\_array = np.array([[1+2j, 2+3j, 3+4j],

[4+5j, 5+6j, 6+ 7j]])

print("Complex array:")

print(complex\_array)

num\_rows, num\_cols = complex\_array.shape

print(f"Number of rows: {num\_rows}")

print(f"Number of columns: {num\_cols}")

dimensions = complex\_array.shape

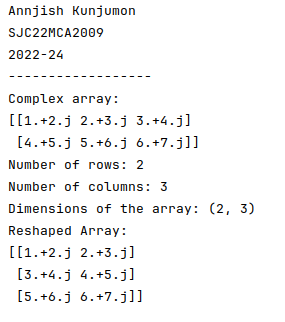
print(f"Dimensions of the array: {dimensions}")

reshaped\_array = complex\_array.reshape(3,2)

print("Reshaped Array:")

print(reshaped\_array)

**Output:**

****

**3. Familiarize with the functions to create**

**a) an uninitialized array**

**b) array with all elements as 1,**

**c) all elements as 0**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

uninitialized\_array = np.empty((3, 4))

print("Uninitialized Array:")

print(uninitialized\_array)

ones\_array = np.ones((2, 3))

print("\nArray with All Elements as 1:")

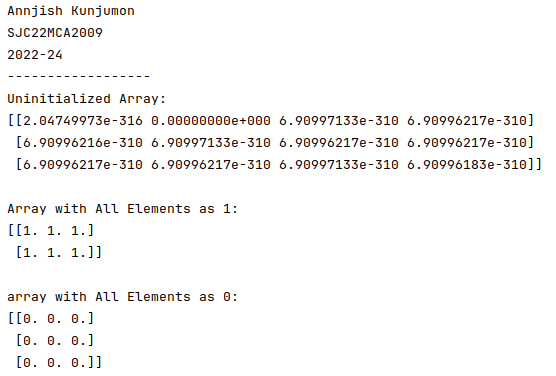
print(ones\_array)

zeros\_array = np.zeros((3, 3))

print("\narray with All Elements as 0:")

print(zeros\_array)

**Output:**

****

**4. Create an one dimensional array using an arrange function containing 10 elements.**

**Display**

**a. First 4 elements**

**b. Last 6 elements**

**c. Elements from index 2 to 7**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

arr = np.arange(10)

first\_4\_elements = arr[:4]

print("first 4 elements:", first\_4\_elements)

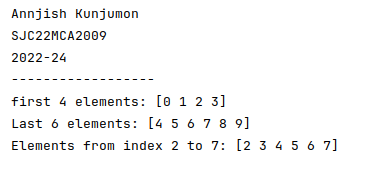
last\_6\_elements = arr[-6:]

print("Last 6 elements:", last\_6\_elements)

elements\_2\_to\_7 = arr[2:8]

print("Elements from index 2 to 7:", elements\_2\_to\_7)

**Output:**

****

**5. Create an 1D array with arrange containing first 15 even numbers as elements**

**a. Elements from index 2 to 8 with step 2(also demonstrate the same using**

**slice function)**

**b. Last 3 elements of the array using negative index**

**c. Alternate elements of the array**

**d. Display the last 3 alternate elements**

**6. Create a 2 Dimensional array with 4 rows and 4 columns.**

**a. Display all elements excluding the first row**

**b. Display all elements excluding the last column**

**c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row**

**d. Display the elements of 2 nd and 3 rd column**

**e. Display 2 nd and 3 rd element of 1 st row**

**f. Display the elements from indices 4 to 10 in descending order(use**

**–values)**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

arr = np.arange(2, 32, 2)

slice\_a = arr[2:9:2]

last\_3\_elements = arr[-3:]

alternate\_elements = arr[::2]

last\_3\_alternate\_elements = alternate\_elements[-3:]

print("Original Array:", arr)

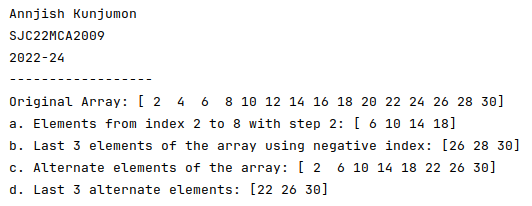
print("a. Elements from index 2 to 8 with step 2:", slice\_a)

print("b. Last 3 elements of the array using negative index:", last\_3\_elements)

print("c. Alternate elements of the array:", alternate\_elements)

print("d. Last 3 alternate elements:", last\_3\_alternate\_elements)

**Output:**

****

**6. Create a 2 Dimensional array with 4 rows and 4 columns.**

**a. Display all elements excluding the first row**

**b. Display all elements excluding the last column**

**c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row**

**d. Display the elements of 2 nd and 3 rd column**

**e. Display 2 nd and 3 rd element of 1 st row**

**f. Display the elements from indices 4 to 10 in descending order(use**

**–values)**

**Code:**

import numpy as np

array\_2d = np.array([[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]])

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

print("All elements excluding the first row:")

print(array\_2d[1:, :])

print("\nAll elements excluding the last column:")

print(array\_2d[:, :-1])

print("\nElements of the 1st and 2nd column in the 2nd and 3rd row:")

print(array\_2d[1:3, 0:2])

print("\nElements of the 2nd and 3rd column:")

print(array\_2d[:, 1:3])

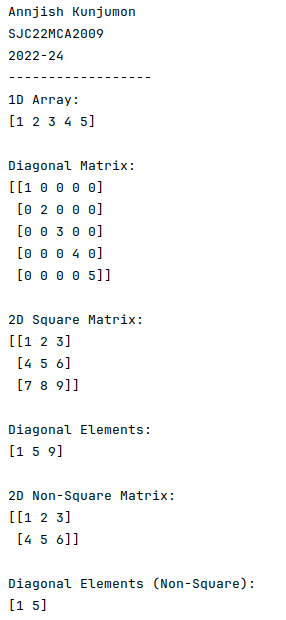
print("\n2nd and 3rd element of the 1st row:")

print(array\_2d[0, 1:3])

print("\nElements from indices 4 to 10 in descending order:")

print(array\_2d.flatten()[10:3:-1])

**Output:**

****

**7. Create two 2D arrays using array object and**

**a. Add the 2 matrices and print it**

**b. Subtract 2 matrices**

**c. Multiply the individual elements of matrix**

**d. Divide the elements of the matrices**

**e. Perform matrix multiplication**

**f. Display transpose of the matrix**

**g. Sum of diagonal elements of a matrix**

**Code:**

import numpy as np

matrix1 = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

matrix2 = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

addition\_result = matrix1 + matrix2

print("Matrix addition:")

print(addition\_result)

subtraction\_result = matrix1 - matrix2

print("\nMatrix Subtraction:")

print(subtraction\_result)

multiplication\_result = matrix1 \* matrix2

print("\nMatrix Element-wise Multiplication:")

print(multiplication\_result)

division\_result = matrix1 / matrix2

print("\nMatrix Element-wise Division:")

print(division\_result)

matrix\_multiplication\_result = np.dot(matrix1, matrix2)

print("\nMatrix Multiplication:")

print(matrix\_multiplication\_result)

matrix1\_transpose = np.transpose(matrix1)

print("\nTranspose of Matrix1:")

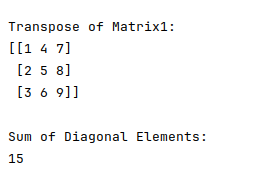
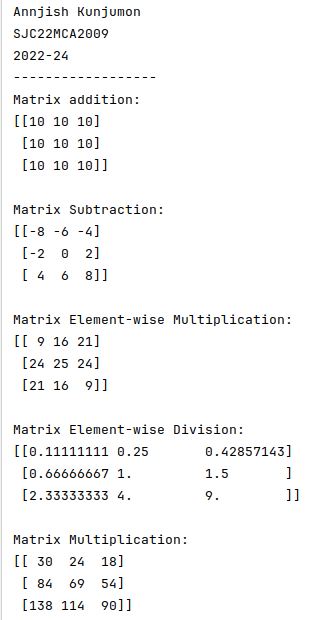
print(matrix1\_transpose)

diagonal\_sum = np.trace(matrix1)

print("\nSum of Diagonal Elements:")

print(diagonal\_sum)

**Output:**

****

**8. Demonstrate the use of insert() function in 1D and 2D array.**

**Code:**

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

my\_list = [1, 2, 3, 4, 5]

my\_list.insert(2, 10)

print(my\_list)

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

my\_2d\_array = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

element\_to\_insert = 10

row\_index = 1

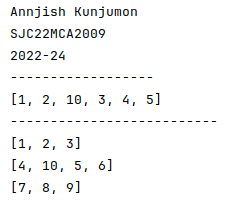
column\_index = 1

my\_2d\_array[row\_index].insert(column\_index, element\_to\_insert)

for row in my\_2d\_array:

print(row)

**Output:**

****

**9. Demonstrate the use of diag() function in 1D and 2D array.(use both square matrix and matrix with different dimensions).**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

arr\_id = np.array([1, 2, 3, 4, 5])

diagonal\_matrix = np.diag(arr\_id)

print("1D Array:")

print(arr\_id)

print("\nDiagonal Matrix:")

print(diagonal\_matrix)

arr\_2d\_square = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

diagonal\_elements = np.diag(arr\_2d\_square)

print("\n2D Square Matrix:")

print(arr\_2d\_square)

print("\nDiagonal Elements:")

print(diagonal\_elements)

arr\_2d\_non\_square = np.array([[1, 2, 3],

[4, 5, 6]])

diagonal\_elements\_non\_square = np.diag(arr\_2d\_non\_square)

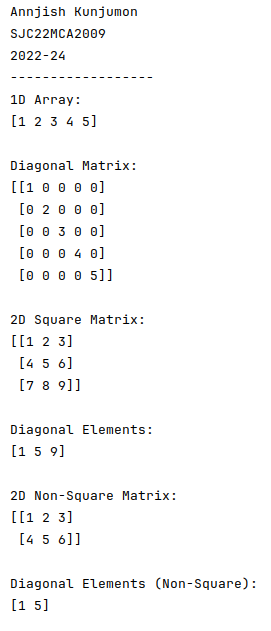
print("\n2D Non-Square Matrix:")

print(arr\_2d\_non\_square)

print("\nDiagonal Elements (Non-Square):")

print(diagonal\_elements\_non\_square)

**Output:**

****

**10. Create a square matrix with random integer values(use randint()) and use**

**appropriate functions to find:**

**i) inverse**

**ii) rank of matrix**

**iii) Determinant**

**iv) transform matrix into 1D array**

**v) eigenvalues and vectors**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

matrix\_size = 3

matrix = np.random.randint(1, 10, size=(matrix\_size, matrix\_size))

print("Original Matrix:")

print(matrix)

if np.linalg.matrix\_rank(matrix) == matrix\_size:

inverse\_matrix = np.linalg.inv(matrix)

print("\nInverse Matrix:")

print(inverse\_matrix)

else:

print("\nThe matrix is not invertible (its rank is less than the size).")

rank = np.linalg.matrix\_rank(matrix)

print("\nRank of the Matrix:", rank)

determinant = np.linalg.det(matrix)

print("\nDeterminant of the Matrix:", determinant)

matrix\_1d = matrix.flatten()

print("\nMatrix as 1D Array:")

print(matrix\_1d)

eigenvalues, eigenvectors = np.linalg.eig(matrix)

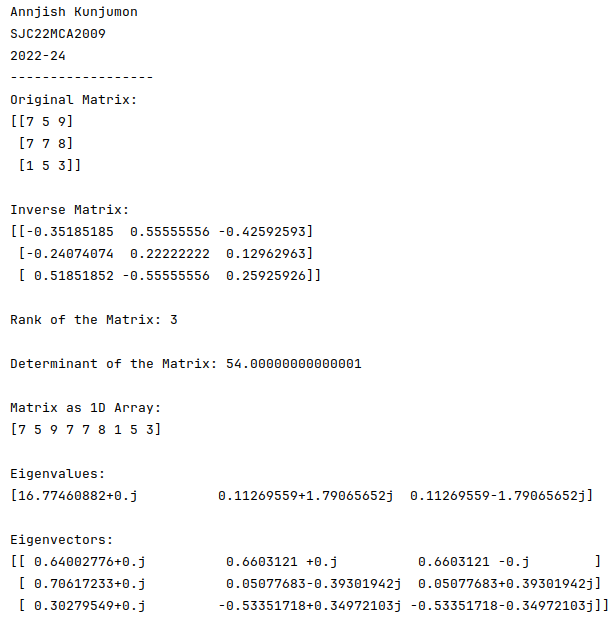
print("\nEigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)

**Output:**

****

**11. Create a matrix X with suitable rows and columns**

**i) Display the cube of each element of the matrix using different**

**methods(use multiply(), \*, power(),\*\*)**

**ii) Display identity matrix of the given square matrix.**

**iii) Display each element of the matrix to different powers.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

X = np.array([[1, 2,3 ],

[4, 5, 6],

[7, 8, 9]])

X\_cube\_multiply = np.multiply(X, np.multiply(X, X))

X\_cube\_operator = X \* X \* X

X\_cube\_power = np.power(X, 3)

X\_cube\_double\_star = X \*\* 3

identity\_matrix = np.identity(X.shape[0])

X\_power\_2 = np.power(X, 2)

X\_power\_3 = np.power(X, 3)

X\_power\_4 = np.power(X, 4)

print("Original Matrix X:")

print(X)

print("\nCubed Matrix (Method 1 - multiply()):")

print(X\_cube\_multiply)

print("\nCubed Matrix (Method 2 - \* operator):")

print(X\_cube\_operator)

print("\nCubed Matrix (Method 3 - power()):")

print(X\_cube\_power)

print("\nCubed Matrix (Method 4 - \*\* operator):")

print(X\_cube\_double\_star)

print("\nIdentity Matrix:")

print(identity\_matrix)

print("\nMatrix to Different Powers:")

print("X^2:")

print(X\_power\_2)

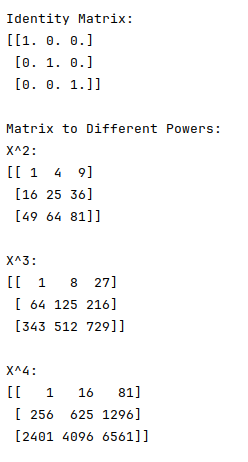
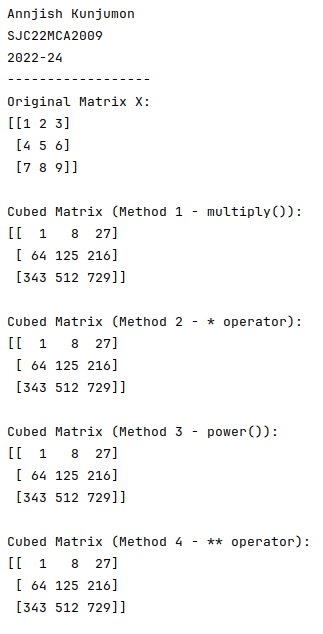
print("\nX^3:")

print(X\_power\_3)

print("\nX^4:")

print(X\_power\_4)

**Output:**

****

**11b. Create a matrix Y with the same dimension as X and perform the operation X 2 +2Y.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

X = np.array([[1, 2],

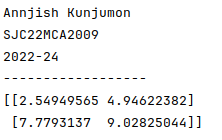
[3, 4]])

Y = np.random.rand(\*X.shape)

result = X \* 2 + 2 \* Y

print(result)

**Output:**

****

**12. Define matrices A with dimension 5x6 and B with dimension 3x3.**

**Extract a sub matrix of dimension 3x3 from A and multiply it with B. Replace the**

**extracted sub matrix in A with the matrix obtained after multiplication.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

print()

A = np.array([[1, 2, 3, 4, 5, 6],

[7, 8, 9, 10, 11, 12],

[13, 14, 15, 16, 17, 18],

[19, 20, 21, 22, 23, 24],

[25, 26, 27, 28, 29, 30]])

B = np.array([[2, 3, 4],

[5, 6, 7],

[8, 9, 10]])

submatrix\_A = A[:3, :3]

result = np.dot(submatrix\_A, B)

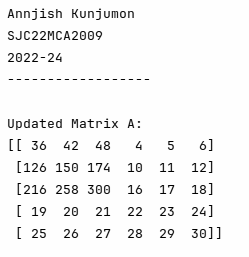
A[:3, :3] = result

*# Display the updated matrix A*

print("Updated Matrix A:")

print(A)

**Output:**

****

**13. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

A = np.array([[1, 2, 3],

[4, 5, 6]])

B = np.array([[7, 8],

[9, 10],

[11, 12]])

C = np.array([[13, 14],

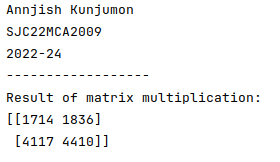
[15, 16]])

result = np.matmul(np.matmul(A, B), C)

print("Result of matrix multiplication:")

print(result)

**Output:**

****

**14. Write a program to check whether given matrix is symmetric or Skew Symmetric.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

*# Function to check if a matrix is symmetric*

def is\_symmetric(matrix):

return np.array\_equal(matrix, matrix.T)

*# Function to check if a matrix is skew-symmetric*

def is\_skew\_symmetric(matrix):

return np.array\_equal(matrix, -matrix.T)

*# Input matrix dimensions*

rows = int(input("Enter the number of rows: "))

cols = int(input("Enter the number of columns: "))

*# Initialize an empty matrix*

matrix = np.zeros((rows, cols))

print("Enter matrix elements row by row:")

for i in range(rows):

row = input().split()

for j in range(cols):

matrix[i][j] = float(row[j])

*# Check if the matrix is symmetric or skew-symmetric*

if is\_symmetric(matrix):

print("The matrix is symmetric.")

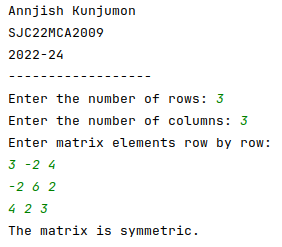
elif is\_skew\_symmetric(matrix):

print("The matrix is skew-symmetric.")

else:

print("The matrix is neither symmetric nor skew-symmetric.")

**Output:**

****

**15. Given a matrix-vector equation AX=b. Write a program to find out the value of X**

**using solve(), given A and b as below**

**X=A -1 b.**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

A = np.array([[2, 1, -2],

[3, 0, 1],

[1, 1, -1]])

b = np.array([[-3],

[5],

[-2]])

try:

X = np.linalg.solve(A, b)

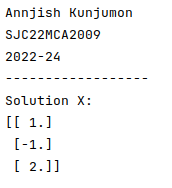
print("Solution X:")

print(X)

except np.linalg.LinAlgError:

print("Matrix A is singular. The system of equations may not have a unique solution.")

**Output:**

****

**16. Write a program to perform the SVD of a given matrix A. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.**

**Use the function: numpy.linalg.svd()**

**Singular value Decomposition**

**Matrix decomposition, also known as matrix factorization, involves describing a given matrix using its constituent elements.**

**The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. This approach is commonly used in reducing the no: of attributes in the given data set.**

**The SVD of mxn matrix A is given by the formula**

**Code:**

import numpy as np

print("Annjish Kunjumon")

print("SJC22MCA2009")

print("2022-24")

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

A = np.array([[5, 27, 32], [14, 53, 62], [67, 88, 19]])

U, S, Vt = np.linalg.svd(A)

A\_hat = U @ np.diag(S) @ Vt

print("Original Matrix A:")

print(A)

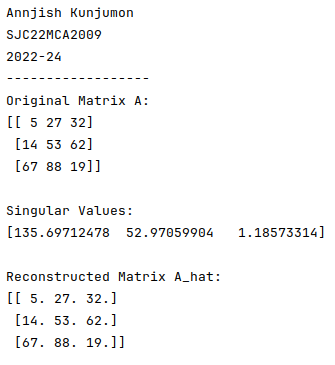
print("\nSingular Values:")

print(S)

print("\nReconstructed Matrix A\_hat:")

print(A\_hat)

**Output:**

****

**LAB CYCLE 3**

**1. Sarah bought a new car in 2001 for $24,000. The dollar value of her car changed each year as shown in the table below.**

**Value of Sarah’s Car**

**Year Value**

**2001 $24,000**

**2002 $22,500**

**2003 $19,700**

**2004 $17,500**

**2005 $14,500**

**2006 $10,000**

**2007 $ 5,800**

**Represent the following information using a line graph with following style properties**

* **X- axis - Year**

**Y –axis - Car Value**

* **title –Value Depreciation (left Aligned)**
* **Line Style dash dot and Line-color should be red**
* **point using \* symbol with green color and size 20**

**Subplot() provides multiple plots in one figure.**

**Code:**

import matplotlib.pyplot as plt

*# Data*

years = [2001, 2002, 2003, 2004, 2005, 2006, 2007]

car\_values = [24000, 22500, 19700, 17500, 14500, 10000, 5800]

*# Create a new figure*

plt.figure(figsize=(10, 6))

*# Create a subplot with the specified properties*

plt.subplot(111) *# Only one subplot in this example*

plt.plot(years, car\_values, linestyle='-.', color='red', marker='\*', markersize=20, markerfacecolor='green')

plt.title("ANNJISH KUNJUMON\n SJC22MCA-2009 \n2022-2024", loc="right")

plt.title("Value Depreciation", loc="left")

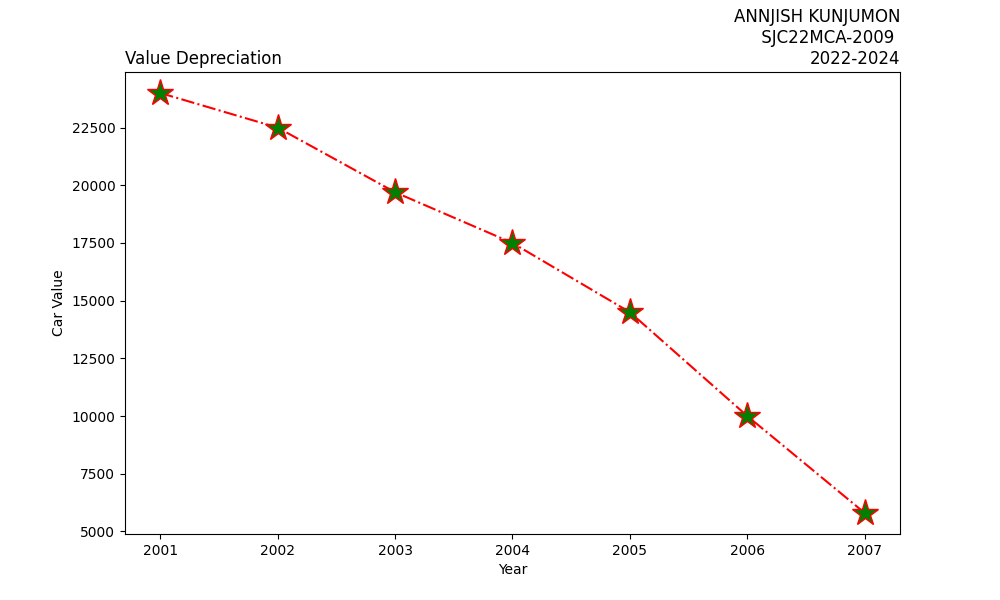
plt.xlabel("Year")

plt.ylabel("Car Value")

*# Show the plot*

plt.show()

**Output:**

****

**2. Following table gives the daily sales of the following items in a shop**

| Day | Mon | Tues | Wed | Thurs | Fri |
| --- | --- | --- | --- | --- | --- |
| Drinks | 300 | 450 | 150 | 400 | 650 |
| Food | 400 | 500 | 350 | 300 | 500 |

**Use subplot function to draw the line graphs with grids(color as blue and line style dotted) for the above information as 2 separate graphs in two rows**

1. **Properties for the Graph 1:**

* **X label- Days of week**
* **Y label-Sale of Drinks**
* **Title-Sales Data1 (right aligned)**
* **Line –dotted with cyan color**
* **Points- hexagon shape with color magenta and outline black**

1. **Properties for the Graph 2:**

* **X label- Days of Week**
* **Y label-Sale of Food**
* **Title-Sales Data2 ( center aligned)**
* **Line –dashed with yellow color**
* **Points- diamond shape with color green and outline red**

**Code:**

import matplotlib.pyplot as plt

*# Data*

days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri']

drinks\_sales = [300, 450, 150, 400, 650]

food\_sales = [400, 500, 350, 300, 500]

*# Create subplots with two rows*

fig, axs = plt.subplots(2, 1, figsize=(8, 8))

*# Graph 1: Drinks*

axs[0].plot(days, drinks\_sales, linestyle='--', color='cyan', marker='H', markersize=8, markerfacecolor='magenta', markeredgecolor='black')

axs[0].set\_xlabel('Days of Week')

axs[0].set\_ylabel('Sale of Drinks')

axs[0].set\_title('Sales Data1', loc='right')

axs[0].set\_title('ANNJISH KUNJUMON\nSJC22MCA-2009 \n2022-2024', loc='left')

axs[0].grid(True, color='blue', linestyle='dotted')

*# Graph 2: Food*

axs[1].plot(days, food\_sales, linestyle='-', color='yellow', marker='D', markersize=8, markerfacecolor='green', markeredgecolor='red')

axs[1].set\_xlabel('Days of Week')

axs[1].set\_ylabel('Sale of Food')

axs[1].set\_title('Sales Data2', loc='center')

axs[1].grid(True, color='blue', linestyle='dotted')

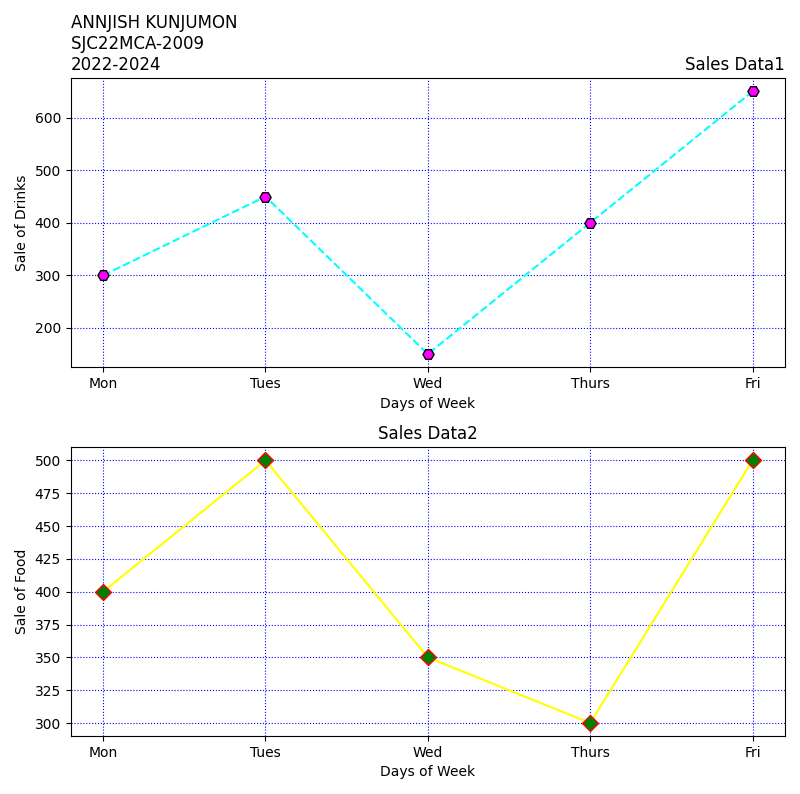
*# Adjust layout*

plt.tight\_layout()

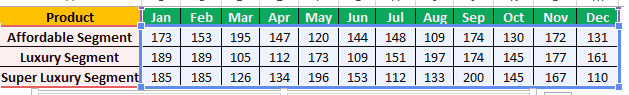
*# Display the plots*

plt.show()

**Output:**

****

**3. Create scatter plot for the below data:(use Scatter function)**



**Create scatter plot for each Segment with following properties within one graph**

* **X Label- Months of Year with font size 18**
* **Y-Label- Sales of Segments**
* **Title –Sales Data**
* **Color for Affordable segment- pink**
* **Color for Luxury Segment- Yellow**
* **Color for Super luxury segment-blue**

**Code:**

import matplotlib.pyplot as plt

import numpy as np

month = np.array(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])

AS = np.array([173,153,195,147,120,144,148,109,174,130,172,131])

LS = np.array([189,189,105,112,173,109,151,197,174,145,177,161])

SLS = np.array([185,185,126,134,196,153,112,133,200,145,167,110])

plt.xlabel('Months of Year', fontsize=18)

plt.ylabel('Sales of Segments')

plt.title('Sales Data')

plt.title('ANNJISH KUNJUMON\nSJC22MCA-2009 \n2022-2024', loc='right')

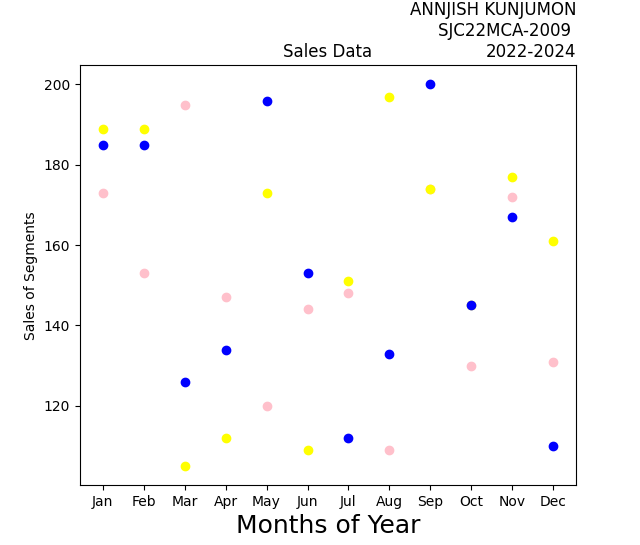
plt.scatter(month,AS, label='Affordable Segment', color='pink')

plt.scatter(month,LS, label='Luxury Segment', color='yellow')

plt.scatter(month,SLS, label='Super Luxury Segment', color='blue')

plt.show()

**Output:**

****

**4.Display the above data using multiline plot( 3 different lines in same graph)**

* **Display the description of the graph in upper right corner(use legend())**
* **Use different colors and line styles for 3 different lines**

**Code:**

import matplotlib.pyplot as plt

import numpy as np

month = np.array(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])

AS = np.array([173,153,195,147,120,144,148,109,174,130,172,131])

LS = np.array([189,189,105,112,173,109,151,197,174,145,177,161])

SLS = np.array([185,185,126,134,196,153,112,133,200,145,167,110])

plt.xlabel('Months of Year', fontsize=18)

plt.ylabel('Sales of Segments')

plt.title('Sales Data')

plt.title('ANNJISH KUNJUMON\nSJC22MCA-2009\n2022-2024', loc='right')

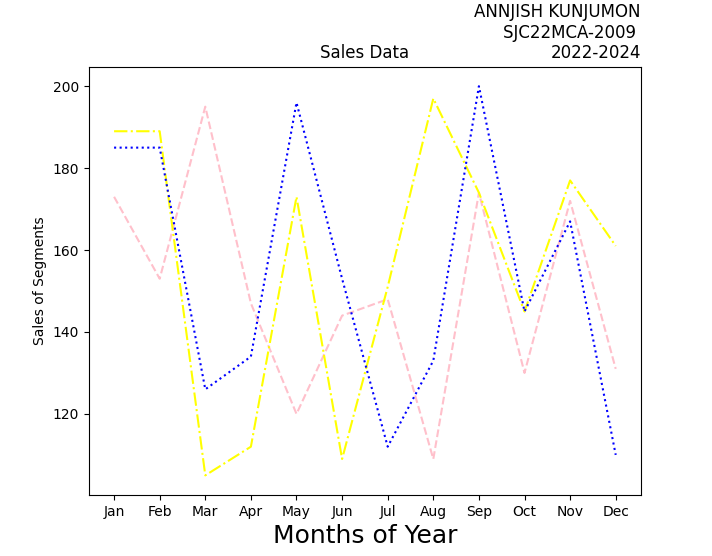
plt.scatter(month,AS, label='Affordable Segment', color='pink')

plt.scatter(month,LS, label='Luxury Segment', color='yellow')

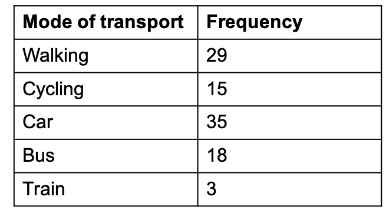
plt.scatter(month,SLS, label='Super Luxury Segment', color='blue')

plt.show()

**Output:**

****

**5.100 students were asked what their primary mode of transport for getting to school was. The results of this survey are recorded in the table below. Construct a bar graph representing this information.**



**Create a bar graph with**

* **X axis -mode of Transport and Y axis ‘frequency’**
* **Provide appropriate labels and title**
* **Width .1, color green**

**Code:**

import matplotlib.pyplot as plt

import numpy as np

mode\_transport = np.array(['Walking','Cycling','Car','Bus','Train'])

feq = np.array([29,15,35,18,3])

plt.xlabel('Mode of Transport')

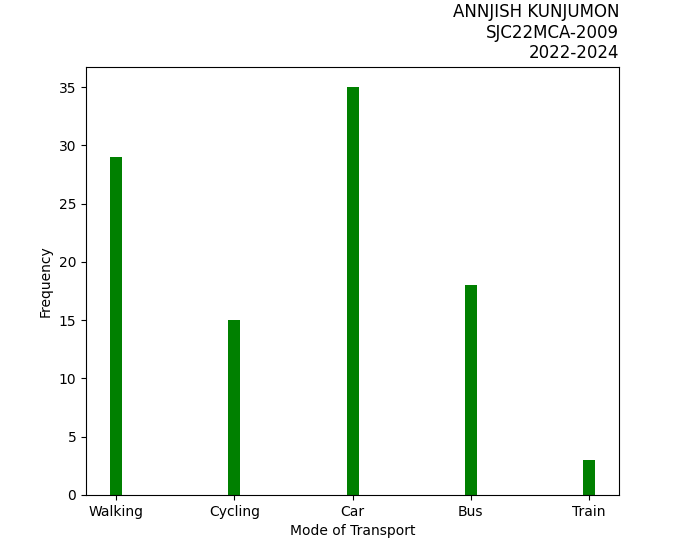
plt.ylabel('Frequency')

plt.title('ANNJISH KUNJUMON\nSJC22MCA-2009\n2022-2024', loc='right')

plt.bar(mode\_transport,feq, width=0.1, color='green')

plt.show()

**Output:**

****

**6.We are provided with the height of 30 cherry trees.**

**The height of the trees (in inches): 61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87.Create a histogram with a bin size of 5.**

**Code:**

import matplotlib.pyplot as plt

import numpy as np

x = np.random.normal([61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87])

plt.hist(x, bins=range(40,110,5), )

plt.title('Cherry tree heights',loc='left')

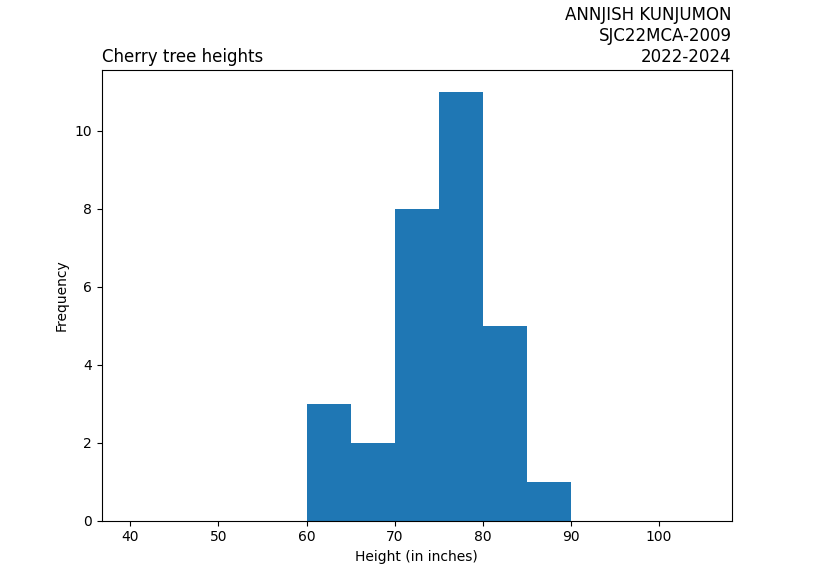
plt.title('ANNJISH KUNJUMON\nSJC22MCA-2009\n2022-2024', loc='right')

plt.xlabel('Height (in inches)')

plt.ylabel('Frequency')

plt.show()

**Output:**

****