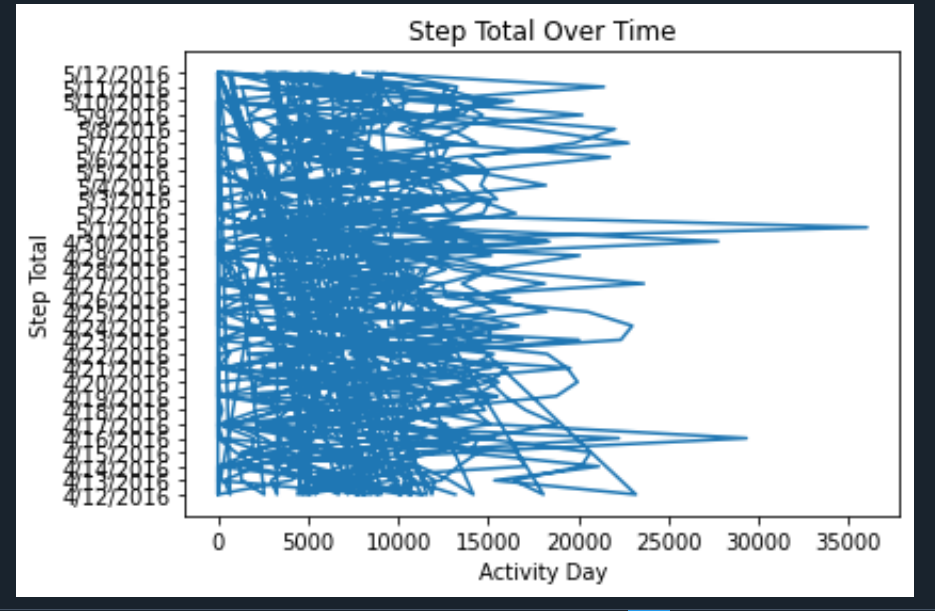
**M Kamran 2022-CS-53**

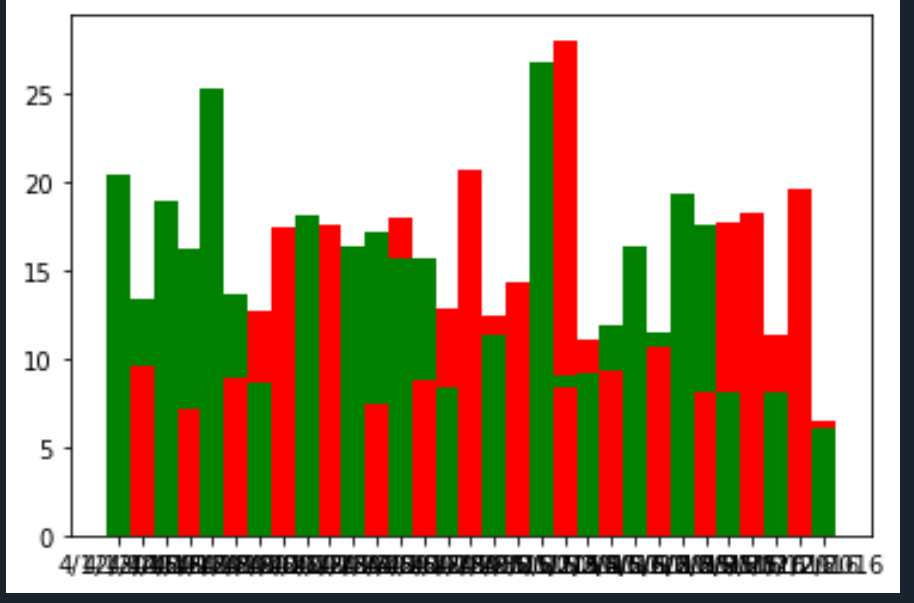
**Part 1: Data plot using graphs**

To plot the data using graphs we install matplotlib which have features to plot line, bar, scatter and pie chart etc. We also installed another extension named pandas to extract the required data from website or specific platform. Before representation of graph, the selected data is stored into two lists of python. X and Y coordinates are required in correspondence to show data.

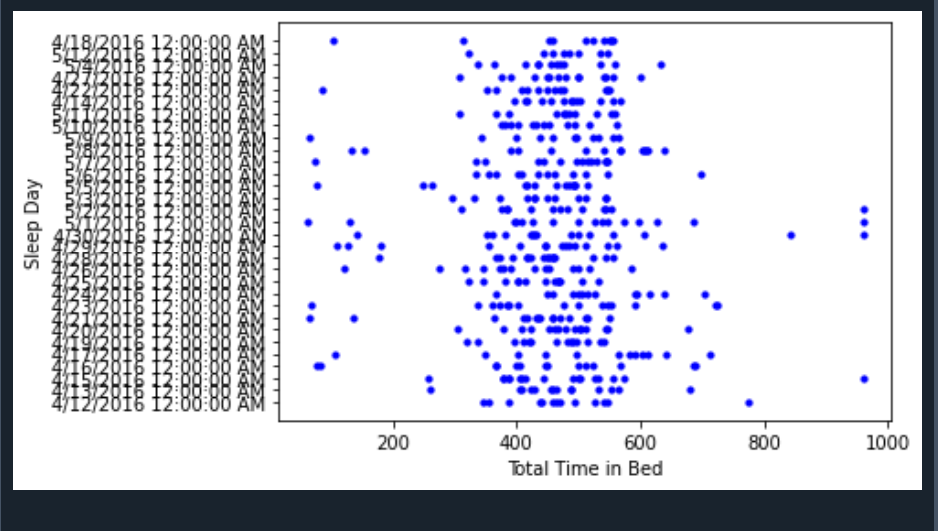
The line chart for the total number of steps on daily basis, the data is extracted from Kaggle.com



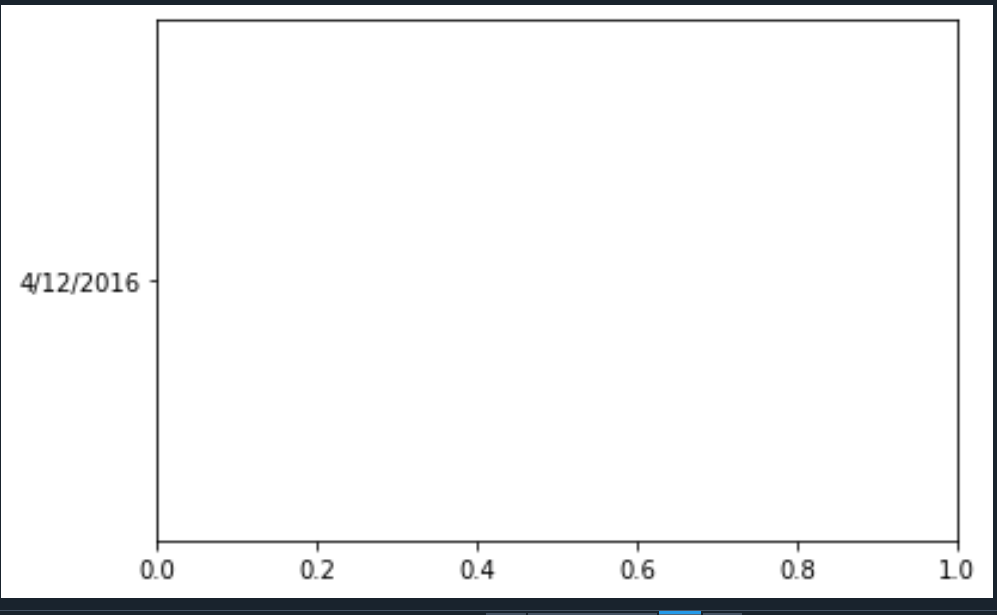
The bar chart for the daily distance covered as the data is extracted from df = pd.read\_csv('dailyActivity\_merged.csv' ).



The scatter chart for the total time in the bed. The X-coordinate represent the date and time and Y-coordinate represent the time in bed.



The Pie chart for the hourly steps on the 12th April 2016 is represented as follows



**Part2: Differentiating the Disease Category**

The given data of some individuals who have one of four diseases. The data is collected using the various symptoms of the patient. The names of four diseases are as follow: • ALLERGY • COVID • COLD • FLU. We use list to store all the disease names. The below represented the clear syntax. To represent every disease correspondence to numbers, time function is used for this purpose.

import time

import math

import matplotlib.pyplot as plt

import pandas as pd

def Euclidean\_distance(x,y):

return math.sqrt(sum((xi-yi)\*\*2 for xi,yi in zip(x,y)))

df = pd.read\_csv('Train.csv' )

symptoms=['COUGH','MUSCLE\_ACHES','TIREDNESS','SORE\_THROAT', 'RUNNY\_NOSE',

'STUFFY\_NOSE', 'FEVER', 'NAUSEA', 'VOMITING', 'DIARRHEA',

'SHORTNESS\_OF\_BREATH', 'DIFFICULTY\_BREATHING', 'LOSS\_OF\_TASTE',

'LOSS\_OF\_SMELL', 'ITCHY\_NOSE', 'ITCHY\_EYES', 'ITCHY\_MOUTH',

'ITCHY\_INNER\_EAR', 'SNEEZING', 'PINK\_EYE']

for symptom in symptoms:

plt.figure(figsize=(8,4))

plt.scatter(df['TYPE'],df[symptom],alpha=0.5)

plt.xlabel('Disease Type')

plt.ylabel(symptom)

plt.title(f'Scatter Plot of {symptom} vs. Disease Type')

plt.show()

time.sleep(1.5)

To measure the distance using Euclidean method we used the given formula to complete

def Euclidean\_distance(x,y):

return math.sqrt(sum((xi-yi)\*\*2 for xi,yi in zip(x,y)))

After calculation of distance, we assign the label to test data entry with the same label as in main data.

**Part 3: Learn to Manipulate Matrices**

Help with Muddasar

* Print 2D matrix is the first function in which an array is passed and starting index is assigned. The array is represented from specific row, column index.

def printMatrix(A,starting\_index,rows,columns):

x,y=starting\_index

for i in range(x,x+rows):

for j in range(y,y+columns):

print([i][j],end=' ')

print()

If we take input as follows the result will be

A = [

[3, 4, 5],

[2, 5, 7]

]

starting\_index = (0, 1)

rows = 2

columns = 3

Output: 4 5

5 7

* The AddMat function is used to add two matrix, the two matrix can be added if they have same number of rows and columns. The two matrix A and B is added then the result is store in third matrix named C. Two loops are executed to perform addition operation.

# ADD two matrix

def MatAdd(A,B):

C=[]

for i in range(len(A)):

row=[]

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

row.append(A[i][j]+B[i][j])

C.append(row)

return C

If we take input as follows the result will be

A = [

[3, 4, 5],

[2, 5, 7]

]

B = [

[1, 2, 3],

[4, 6, 8]

]

Output will be:

[

[4, 6, 8],

[6, 11, 15]

]

* The MatAddPat function is used to add two matrices from specific indexes. The starting index is given from where the addition of two matrices will start. The value of start and end index is given, in which the matrix data is added and store in the third list named C.

def MatAddPartial(A,B,start,size):

C=[]

x,y=start

for i in range(x,x+size):

row=[]

for j in range(y,y+size):

row.append(A[i][j] + B[i][j])

C.append(row)

return C

If we take input as follows the result will be

A = [

[3, 4, 5, 2],

[2, 5, 7, 1],

[1, 2, 3, 0],

[4, 6, 8, 9]

]

B = [

[1, 2, 3, 0],

[4, 6, 8, 9],

[3, 4, 5, 2],

[2, 5, 7, 1]

]

start = (1, 1)

size = 2

Output will be:

[

[11, 15],

[6, 11]

]

* The MatMul function is used to multiply two matrices A and B. The matrices can only be multiplied if they have same number of rows and columns. The nested loop is used to perform this operation and the result is stored in third list named C.

# matrix multiplication

def MatMul(A,B):

C=[]

for i in range(len(A)):

row=[]

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

row.append(A[i][j]\*B[i][j])

C.append(row)

return C

If we take input as follows the result will be

A = [

[2, 3],

[4, 5]

]

B = [

[1, 0],

[0, 1]

]

Output will be:

[

[2, 3],

[4, 5]

]

* The MatMulRecursive function is used to recursively called the matrix multiplication function. It is based on the divide and conquer strategy. The two arrays are divided into single elements. The multiplication can only be possible if two matrices have same number of rows and columns.

# matrix recursive multiplication

def MatMulRecursive(A,B):

if len(A)==1 and len(A[0])==1 and len(B) ==1 and len(B[0]) ==1:

return [[A[0][0] \* B[0][0]]]

n=len(A)

m=n//A12

A11=[A[i][:m] for i in range(m)]

A12=[A[i][m:] for i in range(m)]

A21=[A[i][:m] for i in range(m,n)]

A22=[A[i][m:] for i in range(m,n)]

B11=[B[i][:m] for i in range(m)]

B12=[B[i][m:] for i in range(m)]

B21=[B[i][:m] for i in range(m,n)]

B22=[B[i][m:] for i in range(m,n)]

C11=MatAddPartial(MatMulRecursive(A11, B11),MatMulRecursive(A12, B21),(0,0),m)

C12=MatAddPartial(MatMulRecursive(A11, B12),MatMulRecursive(A12, B22),(0,0),m)

C21=MatAddPartial(MatMulRecursive(A21, B11),MatMulRecursive(A22, B21),(0,0),m)

C22=MatAddPartial(MatMulRecursive(A21, B12),MatMulRecursive(A22, B22),(0,0),m)

result=[]

for i in range(n):

if i<m:

result.append(C11[i]+ C12[i])

else:

result.append(C21[i-m] + C22[i-m])

return result

if the input value is follows

A = [

[2, 3],

[4, 5]

]

B = [

[1, 0],

[0, 1]

]

Output will be:

[

[2, 3],

[4, 5] ]

**Part 4: Scrap the data from web pages**

To scrap data we use different tools like beautifulsoup, selenium, web driver and pandas. Beautifulsoup is a python package used for parsing HTML and XML documents. It is used to extract the data from web pages.

Pandas is a software library for python language used for data manipulation and data analysis. Selenium is used to carry out automated test cases for web browsers or drivers. The web driver is installed for chrome with specific version.

In this part we extract data from eduko website. We extract different type of data like Scrap all the courses from eduko and save in the csv file with the following columns in this order. CourseCode, Title, Description, CLO1, CLO2, ClO3, CLO4, TextBook1, TextBook2, Instructor, Semester.