#1(a)

str1=input ("Enter string1: ")

str2=input ("Enter string2: ")

print(len(str1))

print(str1+str2)

print(str1.lower())

print(str1.upper())

print(str2.islower())

############################################################################

#1(b)

list1=[1,2,3,4,5,6]

print(list1.pop())

list1.remove(list1[1])

print (list1)

list1.append(2)

print (list1)

list1.reverse()

print (list1)

list1.sort()

print (list1)

############################################################################

tup1 = (10,2,3,4,5,6)

print(len(tup1))

print(max(tup1))

print (min(tup1))

print(tup1[2:4])

############################################################################

dict={"fruits":("apple","banana","mango")}

dict1={"name":"xyz","name2":"john"}

print(dict1.keys())

print(dict.items())

print(dict1.values())

print(type(dict))

print(dict1)

print(dict.clear())

#############################################################################

#2a

list=[1,2,3,4,5,6,7]

print(min(list))

print("\n")

#2b

list1=[2,34,54,64,5,4]

for i in list1:

if i%2==0:

print(i)

else:pass

print("\n")

#2c

list2=[12,23,45,-5,-1,-7]

for i in list2:

if i>=0:

print(i)

else:pass

#################################################################################

factorial:

num=int(input("Enter any number"))

factorial=1

if num<0:

print("Please enter positive number")

elif num==0:

print("the factorial of 0 is 1")

else:

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

factorial=factorial\*i

print("Thr factorial of",num,"is",factorial)

##################################################################################

#3b

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

num=0

char1=0

for char in my\_string:

if char.isalpha():

char1+=1

elif char.isdigit():

num+=1

print("char: ",char1)

print("num: ",num)

##################################################################################

def func(l):

sum =0

for x in l:

sum+=x

length = len(l)

print('Average of no.s: %f'%(sum/length))

print('sum of no.s: %f'%(sum))

l1 = list(map(int,input("enter no.s: ").split()))

func(l1)

####################################################################################

def freq\_wrd(str):

str=str.split()

uniq\_wrds = set(str)

for words in uniq\_wrds:

print("freq of ",words,'is',str.count(words))

str1 = 'apple orange apple orange apple'

freq\_wrd(str1)

###################################################################################

# Write a program to receive two integers from the keyboard,

# And find their sum, product and subtraction through a user defined function

def sum(a, b):

c = a + b

return c

def product(a, b):

d = a \* b

return d

def subtract(a, b):

e = a - b

return e

def division(a,b):

f=a/b

return f

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

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

print("The sum of the numbers is: ",sum(a, b))

print("The product of the numbers is: ",product(a, b))

print("The subtraction of the numbers is: ",subtract(a, b))

print("The division of the numbers is: ",division(a, b))

#####################################################################################

# Write a program that uses functions to calculate the gross salary of a employee,

# where gross salary is defined as basic salary + DA + HRA

# DA is 80% of basic salary HRA is 20% of basic salary.

def gross\_salary(x):

DA = (80/100) \* x

HRA = (20/100) \* x

gross = x + DA + HRA

return gross

x = int(input("Input the basic salary: "))

print("The Gross salary is: ",gross\_salary(x))

########################################################################################

#importing the opencv module

import cv2

from google.colab.patches import cv2\_imshow

# using imread('path') and 0 denotes read as grayscale image

#img = cv2.imread(r'C:\Users\Michael Edinburgh\Desktop\child.jpg')

img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/child.jpg')

#This is using for display the image

cv2\_imshow (img)

###################################################################################################

#we will convert our sample image to grayscale and the display it.

import cv2

img = cv2.imread(r'/content/drive/MyDrive/Colab Notebooks/SE Python Programming/Keerti.jpg')

gray\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2\_imshow(gray\_image)

###################################################################################################

import cv2

import numpy as np

img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/Keerti.jpg',0)

cv2\_imshow(img)

cv2\_imshow( cv2.blur(img, (3,3)))

####################################################################################################

#the code for detecting edges in your images.

import cv2

import numpy as np

x=cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/Keerti.jpg',0)

edges = cv2.Canny(x,60,255)

res = np.vstack((x,edges))

cv2\_imshow(res)

####################################################################################################

import cv2

import numpy as np

from matplotlib import pyplot as plt

img=cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/Keerti.jpg',0)

plt.hist(img.ravel(),256,[0,256]);

plt.show()

#########################################################################################################

import cv2

# import Numpy

import numpy as np

# reading an image using imreadmethod

x=cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/Keerti.jpg',0)

equ = cv2.equalizeHist(x)

cv2\_imshow( res)

plt.hist(x.ravel(),bins = 256, range = [0,256])

plt.show()

plt.hist(equ.ravel(),bins = 256, range = [0,256])

plt.show()

###########################################################################################################

MORPHOLOGY

import cv2

import numpy as np

img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/f\_image.jpg',0)

kernel = np.ones((5,5),np.uint8)

dilation = cv2.dilate(img,kernel,iterations = 3)

cv2\_imshow(img)

cv2\_imshow(dilation)

import cv2

import numpy as np

img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/SE Python Programming/f\_image.jpg',0)

kernel = np.ones((5,5),np.uint8)

erode = cv2.erode(img,kernel,iterations = 3)

cv2\_imshow(img)

cv2\_imshow(erode)

################################################################################################################

LINEAR REGRESSION

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn import linear\_model

import matplotlib.pyplot as plt

features = [[2],[1],[5],[10]]

#print(features)

labels = [27, 11, 75, 155]

plt.scatter(features, labels)

clf = linear\_model.LinearRegression()

clf=clf.fit(features,labels)

predicted = clf.predict([[11]])

print(predicted)

predicted = clf.predict(features)

print(predicted)

#To retrieve the intercept:

print(clf.intercept\_)

#For retrieving the slope:

print(clf.coef\_)

#Example1

features = [[6],[2],[10],[4],[8]]

#print(features)

labels = [9, 11, 5, 8,7]

import matplotlib.pyplot as plt

plt.title('Scatter Plot')

plt.scatter(features, labels)

plt.xlabel('Independent variable')

plt.ylabel('Dependent variable')

plt.show()

clf = linear\_model.LinearRegression()

clf=clf.fit(features,labels)

predicted = clf.predict(features)

print(predicted)

predicted = clf.predict(features)

print(predicted)

#To retrieve the intercept:

print(clf.intercept\_)

#For retrieving the slope:

print(clf.coef\_)

from sklearn import metrics

ypredicted=clf.predict(features)

rmse = metrics.mean\_squared\_error(labels, ypredicted)

print(rmse)

#######################################################################################################################################

MULTIPLE LINEAR REGRESSION

import numpy as np

from sklearn.linear\_model import LinearRegression

x = [[3, 8], [4,5], [5,7], [6,3], [2,1]]

y = [-3.7,3.5,2.5,11.5,5.7]

x, y = np.array(x), np.array(y)

model = LinearRegression().fit(x, y)

print('intercept:', model.intercept\_)

print('slope:', model.coef\_)

y\_pred = model.predict(x)

print('predicted response:', y\_pred, sep='\n')

x\_new=[[3,5]]

y\_new = model.predict(x\_new)

print(y\_new)

df = pandas.DataFrame({'Actual': y,'Predicted': y\_pred})

df1 = df.head(5)

print(df1)

from sklearn import metrics

ypredicted=model.predict(x)

mse1 = metrics.mean\_squared\_error(y, y\_pred)

print(mse1)

########################################################################################################################################

LOGISTIC REGRESSION

import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, confusion\_matrix

x = np.arange(10).reshape(-1, 1)

y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

print(x)

print(y)

model = LogisticRegression(solver='liblinear')

#model = LogisticRegression(solver='liblinear', random\_state=0)

model.fit(x, y)

model = LogisticRegression(solver='liblinear', random\_state=0).fit(x, y)

model.classes\_

model.intercept\_

model.coef\_

model.predict\_proba(x)

model.predict(x)

model.score(x, y)

confusion\_matrix(y, model.predict(x))

########################################################################################################################################

SVM

X\_train=np.array([[3, 1],[3, -1],[6, 1],[6, -1],[1,0],[0,1],[0,-1],[-1,0]])

y\_train=[1, 1 ,1, 1, 0, 0, 0, 0]

plt.scatter(X\_train[:,0], X\_train[:,1])

X\_train = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])

y\_train = [0, 0, 0, 1, 1, 1]

# Building the classifier

from sklearn import svm

# Initialize SVM classifier

clf = svm.SVC(kernel='linear')

clf = clf.fit(X\_train, y\_train)

predictions = clf.predict(X\_train)

print(predictions)

predictions = clf.predict([[-4,2]])

print(predictions)

support\_vectors = clf.support\_vectors\_

print(support\_vectors)

###############################################################################################################################

**from** scipy **import** linalg

**import** numpy **as** np

*# The function takes two arrays*

a = np.array([[1, 1], [4, 9]]) *# x + y = 40 , 4x + 9y = 200*

b = np.array([40, 200])

*# Solving the linear equations*

res = linalg.solve(a, b)

print(**"no of true and false questions is %d and \n multiple choice questions are %d"**%(res[0],res[1]))

#####################################################################

**import** numpy **as** np

*# create numpy 2d-array*

m = np.array([[1, 13, 2],

[2, 7, 4],

[4, 9, 6]])

print(**"Original 2D array:\n"**,**m)**

w, v = np.linalg.eig(m)

*# printing eigen values*

print(**"Eigen values:\n"**,w)

*# printing eigen vectors*

print(**“eigenvectors:\n"**,v)

##################################################################

**from** sympy **import** fft

from sympy import ifft

**import** numpy **as** np

seq = np.array([14, 21, 13, 44])

transform = fft(seq)

inverse = ifft(transform)

print(transform)

print(inverse)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_