In [79]:
```
s1=input("Enter string : ")
j=len(s1)
s=" "
for i in range (0,j) :
    if(ord(s1[i])>=ord('a') and ord(s1[i])<=ord('z')) :
        x= chr(ord(s1[i])+ord("A")-ord('a'))
    elif(ord(s1[i])>=ord('A') and ord(s1[i])<=ord('Z')) :
        x= chr(ord(s1[i])+ord('a')-ord('A'))
    s=s+x
print(s)
```

```
Enter string : Teja
 tEJA
```

PART-3 HAAR-CASCADE

IRIS CLASSIFIER PROJECT

sub part 1&2

In [62]:
```
#downloading csv file
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
columns = ['Sepal length', 'Sepal width', 'Petal length', 'Petal width', 'targ
#Loading the data from local system
df = pd.read_csv(r'C:\Users\tejas\OneDrive\Desktop\iris.data.csv')
df.head()
```

Out[62]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

subpart 3

In [73]:
```
# Evaluating the performance
# Split the data to train and test dataset.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.6)

#found 0.6 split ratio to be better
from sklearn.svm import SVC
svn = SVC()
svn.fit(X_train, y_train)
# Predict from the test dataset
predictions = svn.predict(X_test)
# Calculate the accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

Out[73]: 0.9777777777777777

sub part 4

In [74]:
```
#decision tree
mod_dt = DecisionTreeClassifier(max_depth = 3, random_state = 1)
mod_dt.fit(X_train,y_train)
prediction=mod_dt.predict(X_test)
print("The accuracy of the Decision Tree i","{:.3f}".format(accuracy_score(y_t
```

```
The accuracy of the Decision Tree i 0.978
```

Part 3 Haar cascade

In [*]:
```
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier("C:\\Users\\tejas\\Downloads\\haarcascade
# Read the input image
img = cv2.imread(r"C:\\Users\\tejas\\OneDrive\\Pictures\\Camera Roll\\WIN_2023
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Part 4