MINOR PROJECT

Artificial Intellegence with Python

PART-I

Python Assessment

1. Take list of elements from the user and find the square root of each number in the list and store in it another list and print that list.

In [7]:

```
list = []
x = int(input("Enter size of the list"))
for i in range(0,x):
    x = int(input())
    list.append(x)
    list[i]=list[i]**2


print(list)
```

```
Enter size of the list5
3
4
6
8
2
[9, 16, 36, 64, 4]
```

2. Write a function which prints all the numbers divisible by 3 and 5

In [6]:

```
lower = int(input("Enter lower range limit:"))
upper = int(input("Enter upper range limit:"))
for i in range(lower, upper+1):
   if((i%3==0) & (i%5==0)):
      print(i)
```

```
Enter lower range limit:0
Enter upper range limit:99
0
15
30
45
60
75
90
```

3. Write a program to check whether a given letter is vowel or consonant

In [8]:

```
l = input("Input a letter of the alphabet: ")

if l in ('a', 'e', 'i', 'o', 'u'):
        print("%s is a vowel." % l)
```

```
else:
        print("%s is a consonant." % l)
```

```
Input a letter of the alphabet: R
R is a consonant.
```

5. Write a function which returns the number of vowels present in the given string

In [9]:

```
ip_str = input("Enter a string: ")


ip_str = ip_str.casefold()


count = {x:sum([1 for char in ip_str if char == x]) for x in 'aeiou'}


print(count)
```

```
Enter a string: Hi, This is sahithi from Sathayabhamma college
{'a': 6, 'e': 2, 'i': 5, 'o': 2, 'u': 0}
```

6. Print all the alphabets by using loop and ascii code

In [10]:

```
for ch in range(97, 123):
    print("ASCII value: " + str(ch) + ", Character: ", chr(ch))
```

```
ASCII value: 97, Character:  a
ASCII value: 98, Character:  b
ASCII value: 99, Character:  c
ASCII value: 100, Character:  d
ASCII value: 101, Character:  e
ASCII value: 102, Character:  f
ASCII value: 103, Character:  g
ASCII value: 104, Character:  h
ASCII value: 105, Character:  i
ASCII value: 106, Character:  j
ASCII value: 107, Character:  k
ASCII value: 108, Character:  l
ASCII value: 109, Character:  m
ASCII value: 110, Character:  n
ASCII value: 111, Character:  o
ASCII value: 112, Character:  p
ASCII value: 113, Character:  q
ASCII value: 114, Character:  r
ASCII value: 115, Character:  s
ASCII value: 116, Character:  t
ASCII value: 117, Character:  u
ASCII value: 118, Character:  v
ASCII value: 119, Character:  w
ASCII value: 120, Character:  x
ASCII value: 121, Character:  y
ASCII value: 122, Character:  z
```

7. write a program find the sum of all the even numbers of the list

In [11]:

```
nums = []
print("Enter the size of list: ", end="")
tot = int(input())
print("Enter", tot, "Elements for the list: ", end="")
for i in range(tot):
    nums.append(int(input()))
```

```python
sum = 0
count = 0
for i in range(tot):
    if nums[i]%2 == 0:
        sum = sum + nums[i]
        count = count+1

if count==0:
    print("\nEven number is not found in this list!")
else:
    print("\nSum of Even Numbers =", sum)
```

```
Enter the size of list: 8
Enter 8 Elements for the list: 2
6
8
56
98
34
28
58

Sum of Even Numbers = 290
```

9. Take 2 strings from user and then replace all the A’s with a’s and then concatenate the 2 strings and print

In [12]:

```python
s1=input("Enter string:")
s2=input("Enter string:")
s1=s1.replace('A','a')
s2=s2.replace('A','a')
print("Modified s1,s2:")
print('Concatenated String =', s1 + s2)
```

```
Enter string:SAHITHI
Enter string:CHANDHANA
Modified s1,s2:
Concatenated String = SaHITHICHaNDHaNa
```

10. write a program to get a list of odd number from the list of numbers given by user (use list comprehension)

In [20]:

```python
lst = []

# number of elements as input
n = int(input("Enter number of elements : "))

# iterating till the range
for i in range(0, n):
    ele = int(input())

    lst.append(ele) # adding the element
```

```
only_odd = [num for num in lst if num % 2 == 1]

print(only_odd)
```

```
Enter number of elements : 5
67
98
35
45
77
[67, 35, 45, 77]
```

11. write a program to print lower when you have upper letter in string and vice versa

In [16]:

```
str1="saHiThi CHAnDHAna";
newStr = "";

for i in range(0, len(str1)):
    #Checks for lower case character
    if str1[i].islower():
        #Convert it into upper case using upper () function
        newStr += str1[i].upper();
    #Checks for upper case character
    elif str1[i].isupper():
        #Convert it into lower case using lower () function
        newStr += str1[i].lower();

    else:
        newStr += str1[i];
print("String after case conversion : " +  newStr);
```

```
String after case conversion : SAhItHI chaNdhaNA
```

PART-II

1. Implement Iris classifier project

2. Get the data from local system not from web

Load data

In [21]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn import datasets
%matplotlib inline
```

In [23]:

```
import os
```

In [24]:

```
os.getcwd()
```

Out[24]:

```
'C:\\Users\\hp\\Documents\\Mainor project'
```

In [25]:

```
os.chdir('C:\\Users\\hp\\OneDrive\\Desktop\\dataset\\archive')
```

In [26]:

```
os.getcwd()
```

Out[26]:

```
'C:\\Users\\hp\\OneDrive\\Desktop\\dataset\\archive'
```

In [27]:

```
columns = ['Sepal length', 'Sepal width', 'Petal length', 'Petal width',
'Class_labels']
# Load the data
df = pd.read_csv('iris.csv', names=columns)
df.head()
```

Out[27]:

	Sepal length	Sepal width	Petal length	Petal width	Class_labels
Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa

In [51]:

```
iris = datasets.load_iris()
```

Analyze dataset

In [52]:

```
iris = pd.DataFrame(
    data= np.c_[iris['data'], iris['target']],
    columns= iris['feature_names'] + ['target']
    )
```

In [54]:

```
species = []

for i in range(len(iris['target'])):
    if iris['target'][i] == 0:
        species.append("setosa")
    elif iris['target'][i] == 1:
        species.append('versicolor')
    else:
        species.append('virginica')
```

```
iris['species'] = species
```

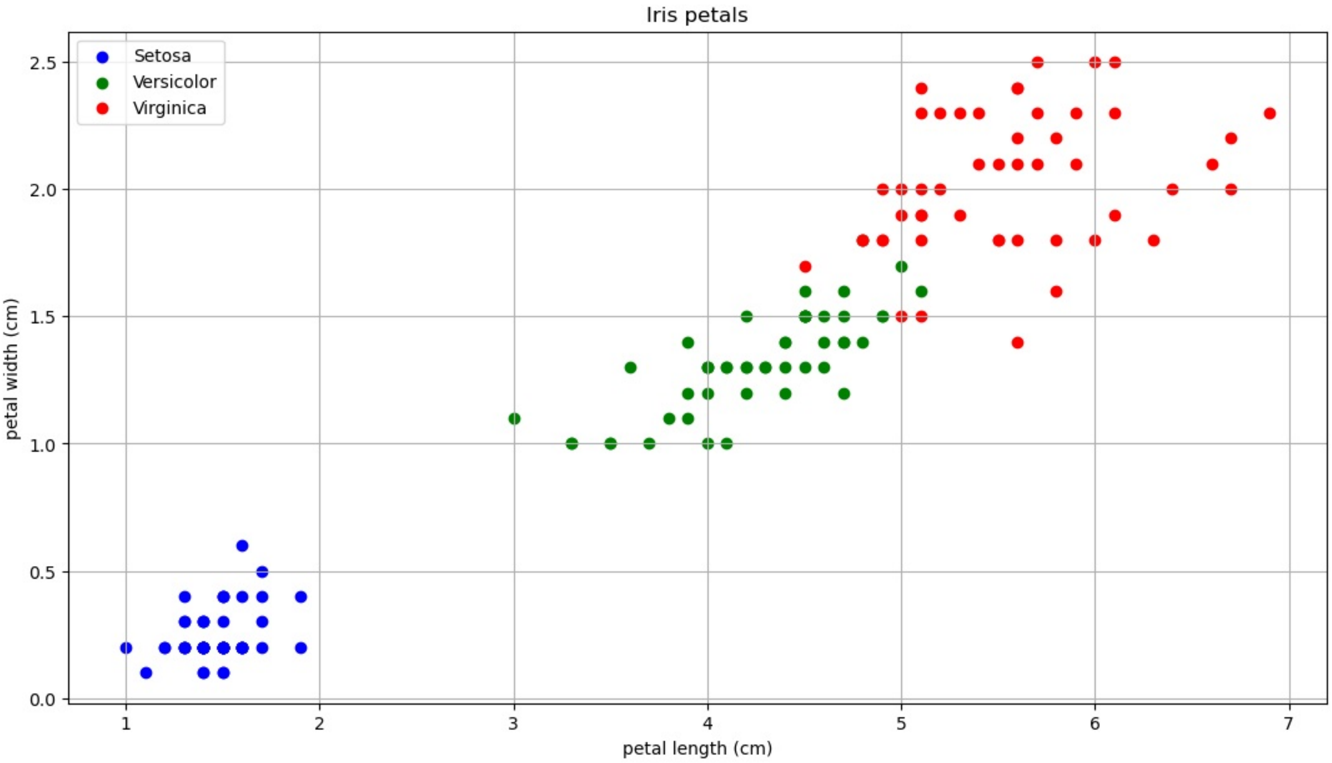
plotting dataset

In [55]:

```
import matplotlib.pyplot as plt

setosa = iris[iris.species == "setosa"]
versicolor = iris[iris.species=='versicolor']
virginica = iris[iris.species=='virginica']

fig, ax = plt.subplots()
fig.set_size_inches(13, 7) # adjusting the length and width of plot

# lables and scatter points
ax.scatter(setosa['petal length (cm)'], setosa['petal width (cm)'],
label="Setosa", facecolor="blue")
ax.scatter(versicolor['petal length (cm)'], versicolor['petal width (cm)'],
label="Versicolor", facecolor="green")
ax.scatter(virginica['petal length (cm)'], virginica['petal width (cm)'],
label="Virginica", facecolor="red")



ax.set_xlabel("petal length (cm)")
ax.set_ylabel("petal width (cm)")
ax.grid()
ax.set_title("Iris petals")
ax.legend()
```

Out[55]:

```
<matplotlib.legend.Legend at 0x17dd8c81cd0>
```



3. Try to evaluate the performance of the model by changing various parameters like split ratio etc.

performing classification

In [56]:

```
from sklearn.model_selection import train_test_split

# Droping the target and species since we only need the measurements
X = iris.drop(['target','species'], axis=1)

# converting into numpy array and assigning petal length and petal width
X = X.to_numpy()[:, (2,3)]
y = iris['target']
```

```
# Splitting into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5,
random_state=42)
```

In [57]:

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
```

Out[57]:

```
LogisticRegression()
```

Training Predictions

In [58]:

```
training_prediction = log_reg.predict(X_train)
training_prediction
```

Out[58]:

```
array([1., 2., 1., 0., 1., 2., 0., 0., 1., 2., 0., 2., 0., 0., 2., 1., 2.,
       2., 2., 2., 1., 0., 0., 1., 2., 0., 0., 0., 1., 2., 0., 2., 2., 0.,
       1., 1., 2., 1., 2., 0., 2., 1., 2., 1., 1., 1., 0., 1., 1., 0., 1.,
       2., 2., 0., 1., 2., 2., 0., 2., 0., 1., 2., 2., 1., 2., 1., 1., 2.,
       2., 0., 1., 1., 0., 1., 2.])
```

Test Prediction

In [62]:

```
test_prediction = log_reg.predict(X_test)
test_prediction
```

Out[62]:

```
array([1., 0., 2., 1., 1., 0., 1., 2., 1., 1., 2., 0., 0., 0., 0., 1., 2.,
       1., 1., 2., 0., 2., 0., 2., 2., 2., 2., 2., 0., 0., 0., 0., 1., 0.,
       0., 2., 1., 0., 0., 0., 2., 1., 1., 0., 0., 1., 2., 2., 1., 2., 1.,
       2., 1., 0., 2., 1., 0., 0., 0., 1., 2., 0., 0., 0., 1., 0., 1., 2.,
       0., 1., 2., 0., 2., 2., 1.])
```

4. . Use other algorithms and evaluate the performance of the algorithm in this dataset.

Performance in Training

In [60]:

```
from sklearn import metrics

print("Precision, Recall, Confusion matrix, in training\n")

# Precision Recall scores
print(metrics.classification_report(y_train, training_prediction, digits=3))

# Confusion matrix
print(metrics.confusion_matrix(y_train, training_prediction))
```

```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

         0.0      1.000     1.000     1.000        21
         1.0      0.923     0.889     0.906        27
         2.0      0.893     0.926     0.909        27

    accuracy                          0.933        75
   macro avg      0.939     0.938     0.938        75
weighted avg      0.934     0.933     0.933        75

[[21  0  0]
 [ 0 24  3]
 [ 0  2 25]]
```

Performance in Testing

In [61]:

```
print("Precision, Recall, Confusion matrix, in testing\n")

# Precision Recall scores
print(metrics.classification_report(y_test, test_prediction, digits=3))

# Confusion matrix
print(metrics.confusion_matrix(y_test, test_prediction))
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

         0.0      1.000     1.000     1.000        29
         1.0      1.000     1.000     1.000        23
         2.0      1.000     1.000     1.000        23

    accuracy                          1.000        75
   macro avg      1.000     1.000     1.000        75
weighted avg      1.000     1.000     1.000        75

[[29  0  0]
 [ 0 23  0]
 [ 0  0 23]]
```

PART-III

1. Study about haarcascade algorithm.

2. Try to import haarcascade algorithm for face detection in ide (.xml).

In [63]:

```
import cv2
```

In [ ]:

```
img = cv2.imread('C:\\Users\\hp\\OneDrive\\Pictures\\sahithi digital paint 1.jpg')

# Converting image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Loading the required haar-cascade xml classifier file
haar_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.



cv2.imshow('Detected faces', img)


cv2.waitKey(0)
```

3. Prepare a model which will detect the face and boundary it using green color box.

In [66]:

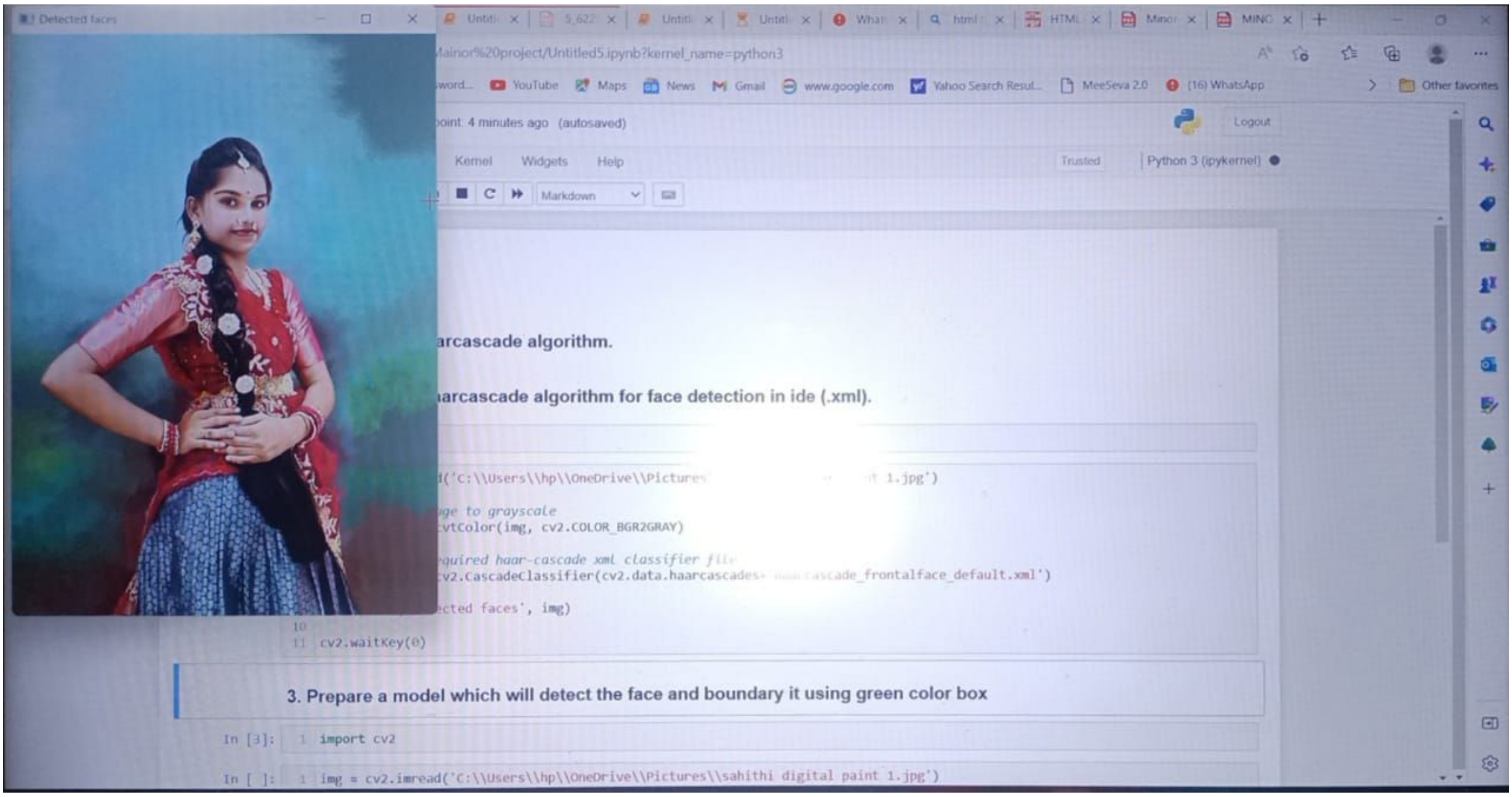
```
import cv2
```

In [ ]:

```
img = cv2.imread('C:\\Users\\hp\\OneDrive\\Pictures\\sahithi digital paint
1.jpg')

# Converting image to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Loading the required haar-cascade xml classifier file
haar_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.



# Applying the face detection method on the grayscale image
faces_rect = haar_cascade.detectMultiScale(gray_img, 1.1, 9)

# Iterating through rectangles of detected faces
for (x, y, w, h) in faces_rect:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('Detected faces', img)

cv2.waitKey(0)
```

In [ ]:

Loading [MathJax]/extensions/Safe.js



arcascade algorithm.

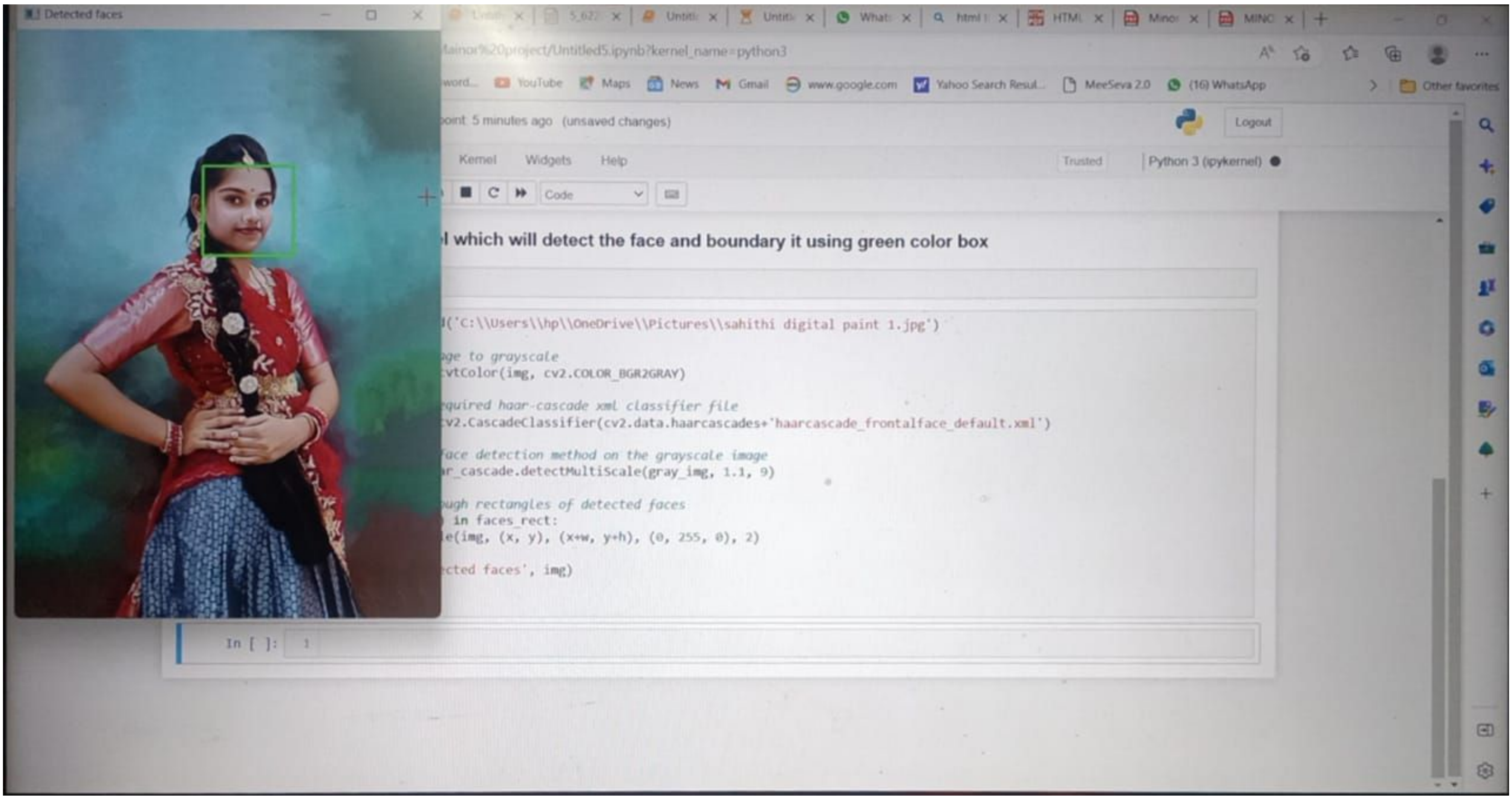
iarcascade algorithm for face detection in ide (.xml).

```
d('C:\\Users\\hp\\OneDrive\\Pictures                 t 1.jpg')

ge to grayscale
vtColor(img, cv2.COLOR_BGR2GRAY)

quired haar-cascade xml classifier file
v2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')

cted faces', img)

cv2.waitKey(0)
```

3. Prepare a model which will detect the face and boundary it using green color box

```
In [3]:  import cv2
```

```
In [ ]:  img = cv2.imread('C:\\Users\\hp\\OneDrive\\Pictures\\sahithi digital paint 1.jpg')
```



point 5 minutes ago (unsaved changes)

l which will detect the face and boundary it using green color box

```
d('C:\\Users\\hp\\OneDrive\\Pictures\\sahithi digital paint 1.jpg')

ge to grayscale
cvtColor(img, cv2.COLOR_BGR2GRAY)

equired haar-cascade xml classifier file
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')

face detection method on the grayscale image
r_cascade.detectMultiScale(gray_img, 1.1, 9)

ugh rectangles of detected faces
) in faces_rect:
e(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cted faces', img)
```