**Student Name : Varshitha chintareddy.**

# Artificial Intelligence with Python Assessment

**Part 1:** Solve all the 11 questions of python assignment and create a pdf mention the title in markdown format etc.

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.

## Code & Output:

main.py — Run — Shell

```
sqr_list = []
n=int(input("enter a number:"))
for i in range(1,n):
  sqr_list.append(i*i)
print(*sqr_list)
```

```
enter a number:10
1 4 9 16 25 36 49 64 81
>
```

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

## Code & Output:

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

main.py — Run — Shell

```
def result(N):
    for num in range(N):
            if num % 3 == 0 and num % 5 == 0:
                print(str(num) + " ", end = "")

            else:
                pass
if __name__ == "__main__":
    N = 100
    result(N)
```

```
> 65
0 15 30 45 60 75 90 65
>
```

## Code & Output:

main.py — Run — Shell

```
ch = input("Enter a character: ")
if(ch=='A' or ch=='a' or ch=='E' or ch =='e' or ch=='I'
 or ch=='i' or ch=='O' or ch=='o' or ch=='U' or ch=='u'):
    print(ch, "is a Vowel")
else:
    print(ch, "is a Consonant")
```

```
Enter a character: G
G is a Consonant
>
```

**Student Name : Varshitha chintareddy.**

4.Calculate the distance between any two characters given by user (Example distance between “a” and “d” is 3).

**Code & Output:**

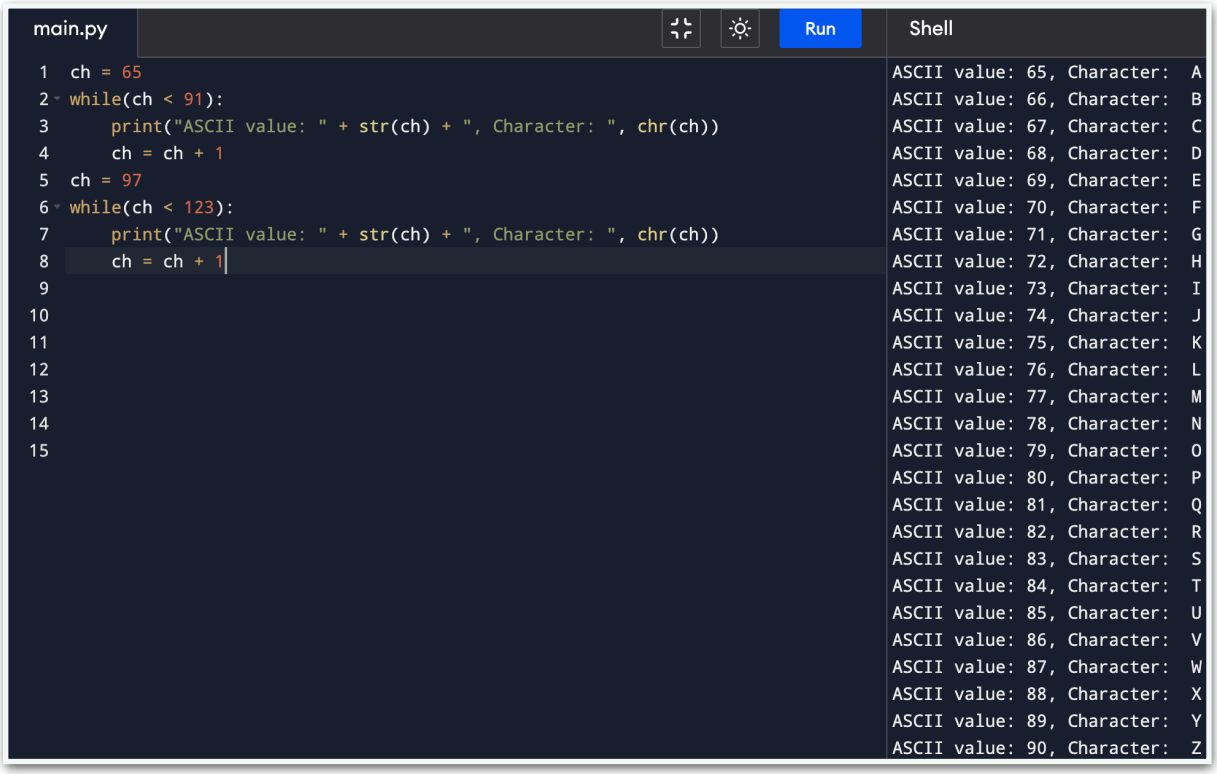

5.Write a function which returns the number of vowels present in the given string.

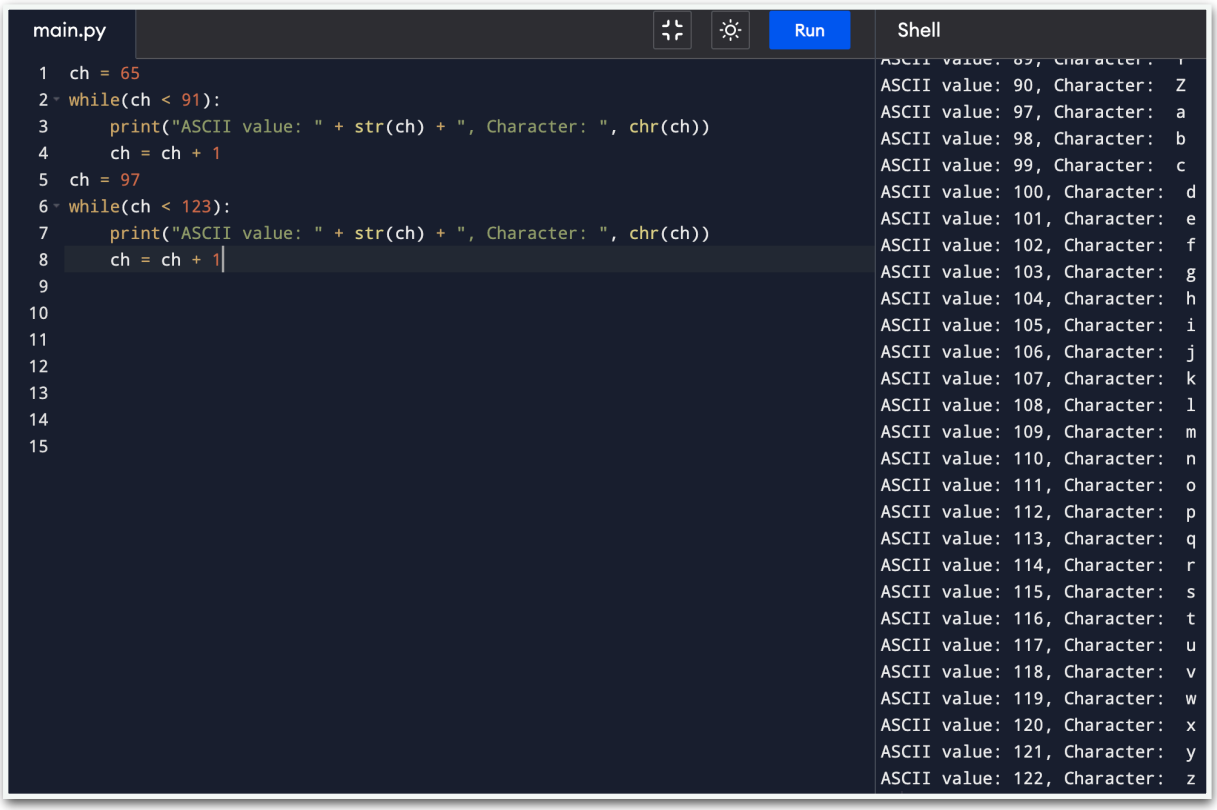
**Code & Output:**


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

**Code & Output:**

**Student Name : Varshitha chintareddy.**




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

**Code & Output:**

**Student Name : Varshitha chintareddy.**

main.py

```
nums = []
print("Enter 5 elements for the list: ")
for i in range(5):
    val = int(input())
    nums.append(val)
sum = 0
for i in range(5):
    if nums[i]%2 == 0:
        sum = sum + nums[i]
print("\nSum of Even Numbers is", sum)

```

Shell

```
Enter 5 elements for the list:
4
3
6
7
9
Sum of Even Numbers is 10
>
```

8.Write a program for print the squares of all the numbers, except for factors of 3 .

## Code & Output:

main.py

```
def print_squares(n):
    for i in range(1, n+1):
        if i % 3 == 0:
            continue
        print(i**2)

n = int(input("Enter a number: "))
print_squares(n)

```

Shell

```
Enter a number: 15
1
4
16
25
49
64
100
121
169
196
```

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

## Code & Output:

main.py

```
s1=input()
s2=input()
s1 = s1.replace('A', 'a')
s2 = s2.replace('A', 'a')
print(s1+s2)

```

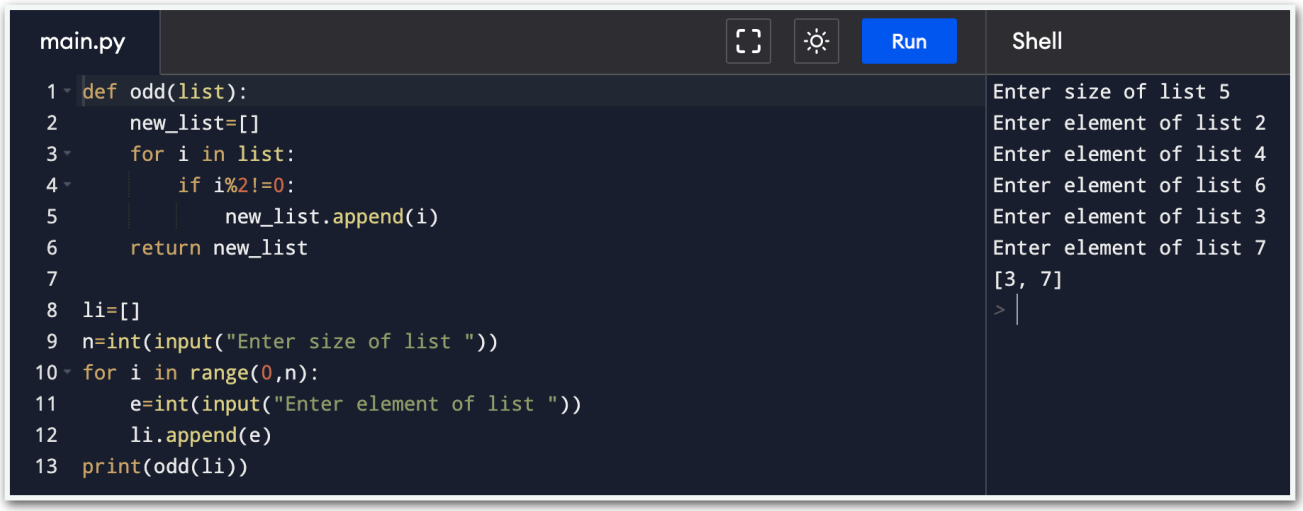
Shell

```
VARSHITHA
CHINTAREDDY
VaRSHITHaCHINTaREDDY
>
```

**Student Name : Varshitha chintareddy.**

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

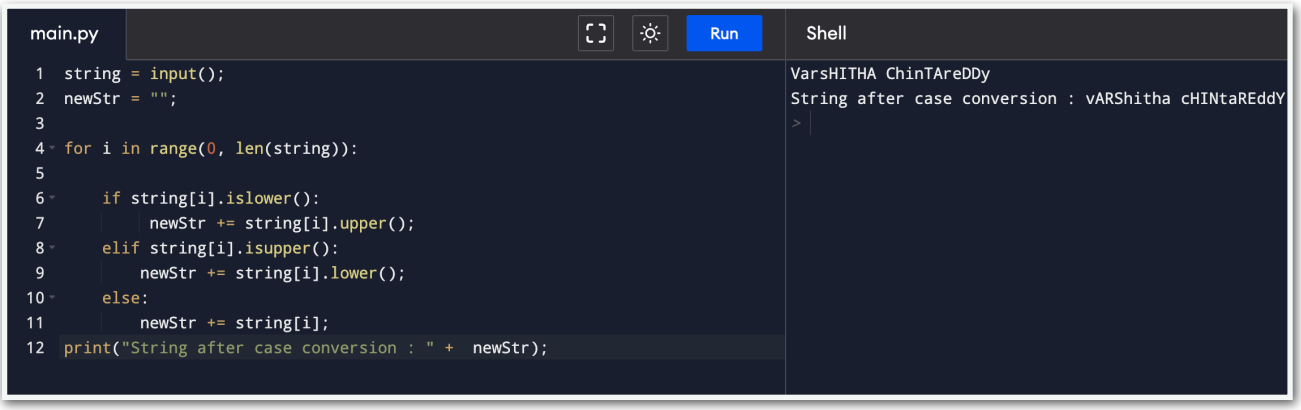
## Code & Output:




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

(if your input is “aBcD” your output should be “AbCd”)

## Code & Output:



STUDENT NAME : VARSHITHA CHINTAREDDY.

## Part 2:

1.Implement Iris classifier project.

2.Get the data from local system not from web

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

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

## IMPORT THE LIBRARIES.

```
import numpy as np
import matplotlib.pyplot as plt


from sklearn.datasets import load_iris


dataset = load_iris()


print(dataset.DESCR)
```

```
    **Data Set Characteristics:**

        :Number of Instances: 150 (50 in each of three classes)
        :Number of Attributes: 4 numeric, predictive attributes and the class
        :Attribute Information:
            - sepal length in cm
            - sepal width in cm
            - petal length in cm
            - petal width in cm
            - class:
                    - Iris-Setosa
                    - Iris-Versicolour
                    - Iris-Virginica

        :Summary Statistics:

        ============== ==== ==== ======= ===== ====================
                        Min  Max   Mean    SD   Class Correlation
        ============== ==== ==== ======= ===== ====================
        sepal length:   4.3  7.9   5.84   0.83    0.7826
        sepal width:    2.0  4.4   3.05   0.43   -0.4194
        petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
        petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
        ============== ==== ==== ======= ===== ====================

        :Missing Attribute Values: None
        :Class Distribution: 33.3% for each of 3 classes.
        :Creator: R.A. Fisher
        :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
        :Date: July, 1988

    The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
    from Fisher's paper. Note that it's the same as in R, but not as in the UCI
    Machine Learning Repository, which has two wrong data points.

    This is perhaps the best known database to be found in the
    pattern recognition literature.  Fisher's paper is a classic in the field and
    is referenced frequently to this day.  (See Duda & Hart, for example.)  The
    data set contains 3 classes of 50 instances each, where each class refers to a
    type of iris plant.  One class is linearly separable from the other 2; the
    latter are NOT linearly separable from each other.

    .. topic:: References

       - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
         Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
         Mathematical Statistics" (John Wiley, NY, 1950).
       - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
         (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
       - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
         Structure and Classification Rule for Recognition in Partially Exposed
```

```
        conceptual clustering system finds 3 classes in the data.
    - Many, many more ...
```

```
x = dataset.data
```

```
y = dataset.target
```

```
y
```

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

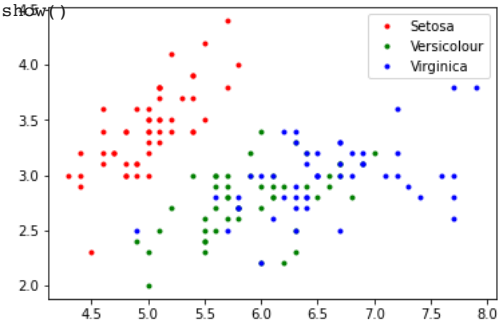
```
x
```

```
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
[6.3, 2.5, 5. , 1.9],
[6.5, 3. , 5.2, 2. ],
[6.2, 3.4, 5.4, 2.3],
[5.9, 3. , 5.1, 1.8]])
```

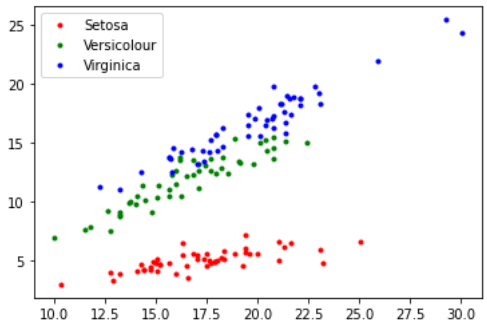
## VISUALIZE THE DATA.

```
plt.plot(x[:,0][y == 0],x[:,1][y == 0],'r.',label='Setosa')
plt.plot(x[:,0][y == 1],x[:,1][y == 1],'g.',label='Versicolour')
plt.plot(x[:,0][y == 2],x[:,1][y == 2],'b.',label='Virginica')
```

```
plt.legend()
plt.show()
```




```
plt.plot(x[:,0][y == 0]*x[:,1][y == 0], x[:,1][y == 0]*x[:,2][y == 0],'r.',label='Setosa')
plt.plot(x[:,0][y == 1]*x[:,1][y == 1], x[:,1][y == 1]*x[:,2][y == 1],'g.',label='Versicolour')
plt.plot(x[:,0][y == 2]*x[:,1][y == 2], x[:,1][y == 2]*x[:,2][y == 2],'b.',label='Virginica')
plt.legend()
plt.show()
```




## STANDARDIZED THE DATA

```
from sklearn.preprocessing import StandardScaler
x = StandardScaler().fit_transform(x)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

## CLASSIFY THE THREE TYPES OF FLOWERS BASED ON MACHINE LEARNING.

```
from sklearn.linear_model import LogisticRegression
```

```
log_reg = LogisticRegression()
```

```
log_reg.fit(x_train, y_train)
```

```
LogisticRegression()
```

## TO CHECK THE HOW ACCURATE IT IS!

```
log_reg.score(x_test, y_test)
```

```
0.9736842105263158
```

## TO SEE THE SCORE OF WHOLE DATASET!

```
log_reg.score(x, y)
```

```
0.9733333333333334
```

**Student Name : Varshitha chintareddy.**

## Part 3:

### 1. Study about haar cascade algorithm.

Haar-cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location.

This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, etc.

Haar cascade works as a classifier. It classifies positive data points → that are part of our detected object and negative data points → that don't contain our object.

- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

The OpenCV library manages a repository containing all popular haar cascades that can be used for:

- Human face detection
- Eye detection
- Nose / Mouth detection
- Vehicle detection

### 2. Try to import haar cascade algorithm for face detection in ide (.xml).

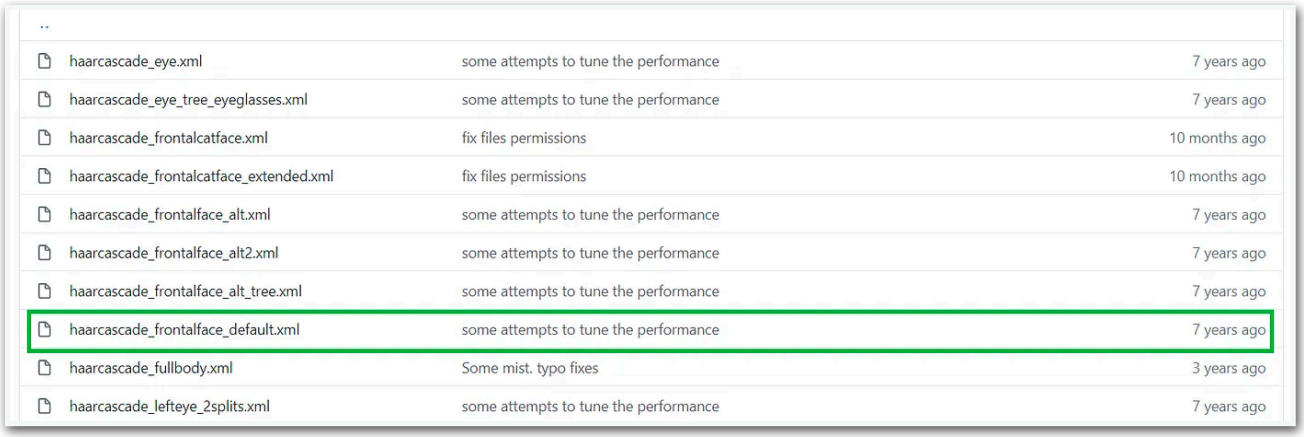
### 3. Prepare a model which will detect the face and boundary it using green colour box.

**Steps to Face Detection :**

1. Install and Import OpenCV .
2. Download the files from Github without downloading the whole project.

3. Here is the link : https://github.com/opencv/opencv/tree/master/data/haarcascades.

4. Haarcascade_frontalface_default.xml is the only file needed for this piece of code.



| | | |
|---|---|---|
| haarcascade_eye.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_eye_tree_eyeglasses.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_frontalcatface.xml | fix files permissions | 10 months ago |
| haarcascade_frontalcatface_extended.xml | fix files permissions | 10 months ago |
| haarcascade_frontalface_alt.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_frontalface_alt2.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_frontalface_alt_tree.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_frontalface_default.xml | some attempts to tune the performance | 7 years ago |
| haarcascade_fullbody.xml | Some mist. typo fixes | 3 years ago |
| haarcascade_lefteye_2splits.xml | some attempts to tune the performance | 7 years ago |

5. Click on the file and then click on raw and save the file.

6. **Let's jump to the code now :**

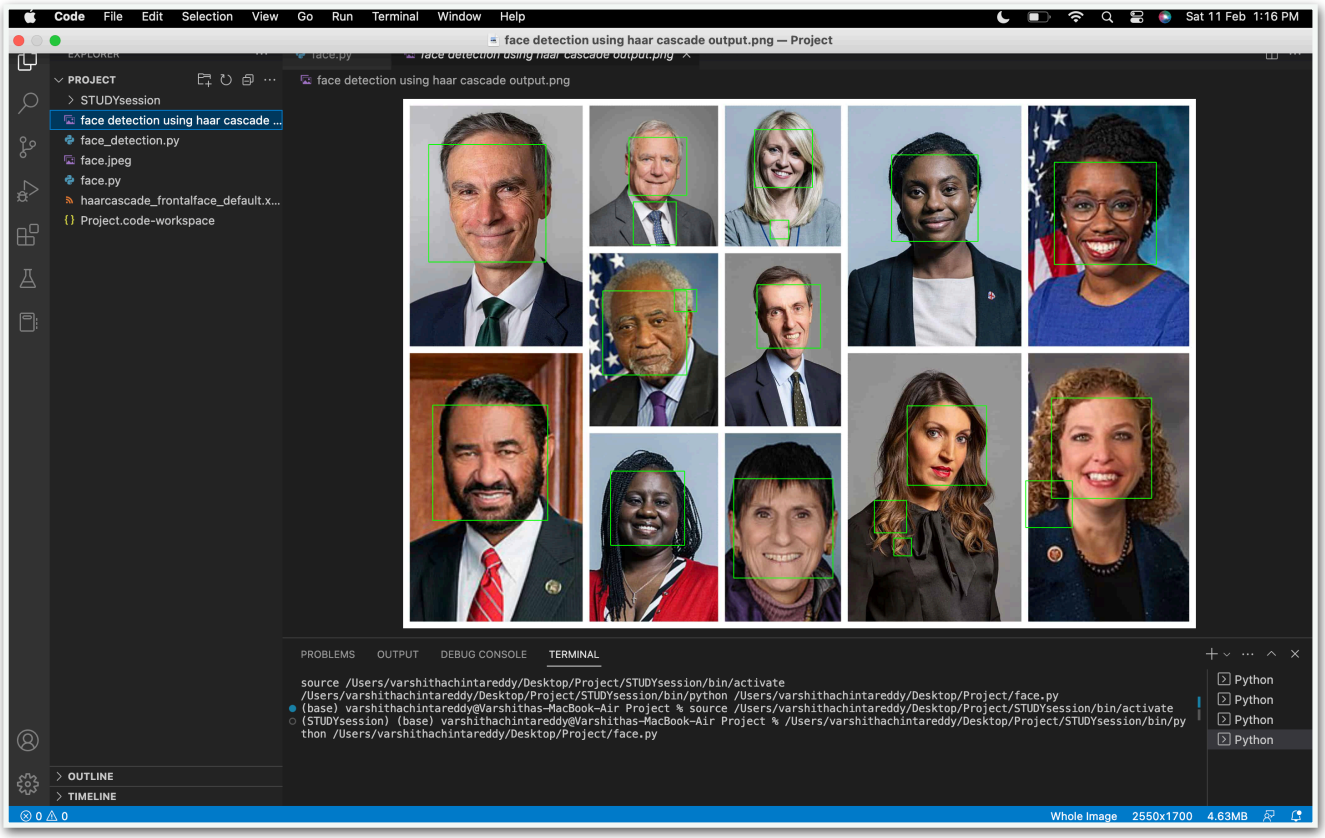
```
# Import OpenCV

import cv2


# Define haar cascade classifier for face detection

face_classifier =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


# Read input image to recognize face

img = cv2.imread('face.jpeg')


# Convert image to gray scale OpenCV

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
# Detect face using haar cascade classifier
faces_coordinates = face_classifier.detectMultiScale(gray_img)

# Draw a rectangle around the faces
for (x, y, w, h) in faces_coordinates:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Saving the image
cv2.imwrite('face detection using haar cascade output.png', img)

# show output image
cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

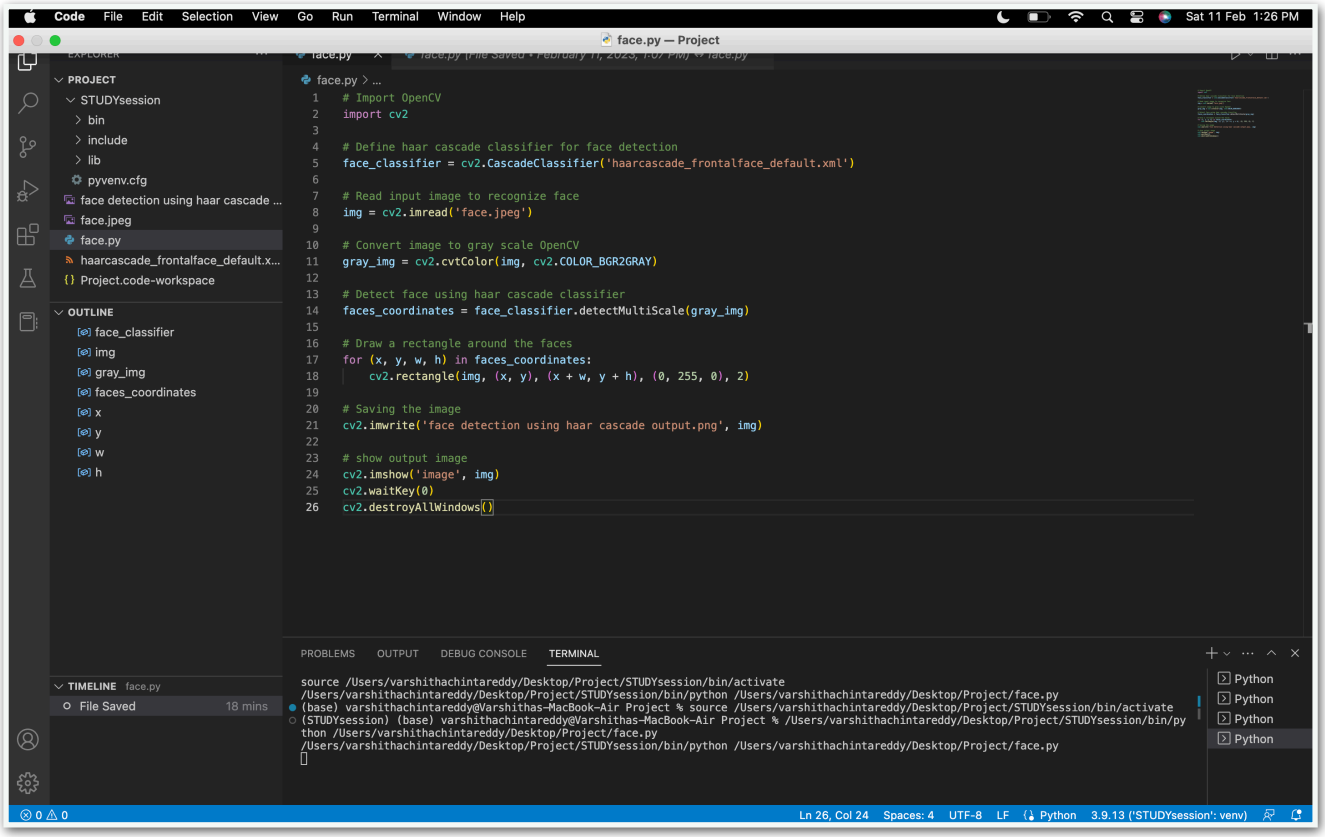
**7. Input Image :**

**Student Name : Varshitha chintareddy.**

## 8. Output Image :




## 9. Code In Vs code with Good Environment.



**Student Name : Varshitha chintareddy.**

## 10. Conclusion (What I have observed /learn from this project) :

- Haar Cascade Detection is one of the oldest yet powerful face detection algorithms invented. It has been there since long, long before Deep Learning became famous. Haar Features were not only used to detect faces, but also for eyes, lips, license number plates etc.
- We can see the Output Image in this , that was not so accurate because in that image it perfectly detecting faces and also some other things like hair ,tie..etc.Even though it is very good project because it detecting faces perfectly.
- I have learnt so much form this project and enjoyed while doing this.This was a great experience for me.

**…THANK YOU…**