PART:1

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**

=> import numpy as np

input\_string = input('Enter elements of a list separated by space ')

print("\n")

user\_list = input\_string.split()

# print list

print('list: ', user\_list)

# convert each item to int type

for i in range(len(user\_list)):

    # convert each item to int type

    user\_list[i] = int(user\_list[i])

    print(i)

    output = np.sqrt(user\_list[i])

print(output)

output:-

Enter elements of a list separated by space 4 9 16 25

list: ['4', '9',’16’ ,’25’]

2

3

4

5

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

=>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)

output:-

Enter lower range limit:1

Enter upper range limit:50

45

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

=>

# taking user input

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")

output:-

Enter a character: h

h is a Consonant

4.calculate the distance between any two characters given by user

=>first=input("enter first character :")

second=input("enter second charactor :")

distance=abs(ord(first)-ord(second))

print(f"ASCII Distance between {first} and {second} is:{distance}")

output

enter first character :k

enter second charactor :j

ASCII Distance between k and j is:1

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

=def vowel\_count(str):

    # Initializing count variable to 0

    count = 0

    # Creating a set of vowels

    vowel = set("aeiouAEIOU")

    # Loop to traverse the alphabet

    # in the given string

    for alphabet in str:

        # If alphabet is present

        # in set vowel

        if alphabet in vowel:

            count = count + 1

    print("No. of vowels :", count)

# Driver code

str = "write a function which returns the number of vowels present in the given string"

# Function Call

vowel\_count(str)

output;

No. of vowels : 22

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

=> for i in range(65,91):

   print(chr(i),end=" ")

for i in range(97,123):

  print(chr(i),end=" ")

output:- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

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

=>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)

output:-  ]

Enter 5 elements for the list:

11

11

22

33

44

Sum of Even Numbers is 66

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

=>for x in range(100):

    if (x % 3==0):

        continue

    print(x,end=' ')

    print('\n')

    print(x\*x)

output:-

1

1

2

4

4

16

5

25

7

49

8

64

10

100

11

121

13

169

14

196

16

256

17

289

19

361

20

400

22

484

23

529

25

625

26

676

28

784

29

841

31

961

32

1024

34

1156

35

1225

37

1369

38

1444

40

1600

41

1681

43

1849

44

1936

46

2116

47

2209

49

2401

50

2500

52

2704

53

2809

55

3025

56

3136

58

3364

59

3481

61

3721

62

3844

64

4096

65

4225

67

4489

68

4624

70

4900

71

5041

73

5329

74

5476

76

5776

77

5929

79

6241

80

6400

82

6724

83

6889

85

7225

86

7396

88

7744

89

7921

91

8281

92

8464

94

8836

95

9025

97

9409

98

9604

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

=>value1 = input("Please enter a string:\n")

print(f'You entered {value}')

value2 = input("Please enter a string:\n")

print(f'You entered {value}')

modified\_str = ''

# iterating over the string

for char in range(0, len(value1)):

    # checking if the character at char index is equivalent to 'a'

    if(value1[char] == 'A'):

        # append $ to modified string

        modified\_str += 'a'

    else:

        # append original string character

        modified\_str += value1[char]

print("Modified string : ")

print(modified\_str)

modified\_str2 = ''

# iterating over the string

for char in range(0, len(value2)):

    # checking if the character at char index is equivalent to 'a'

    if(value2[char] == 'A'):

        # append $ to modified string

        modified\_str2 += 'a'

    else:

        # append original string character

        modified\_str2 += value2[char]

print("Modified string2 : ")

print(modified\_str2)

word = modified\_str+modified\_str2

print(word)

output:-

Please enter a string:

HARSHAL

You entered harshal

Please enter a string:

PAWAR

You entered harshal

Modified string :

HaRSHaL

Modified string2 :

PaWaR

HaRSHaLPaWaR

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

=>input\_string = input('Enter elements of a list separated by space ')

print("\n")

user\_list = input\_string.split()

# print list

print('list: ', user\_list)

# convert each item to int type

for i in range(len(user\_list)):

    # convert each item to int type

    user\_list[i] = int(user\_list[i])

    if (i % 2==0):

        continue

    print(i,end=' ')

output:-

Enter elements of a list separated by space 1 2 3 4

list: ['1', '2', '3', '4']

1 3

1. 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")
2. for i in range(0, len(str1)):
3. #Checks for lower case character
4. if str1[i].islower():
5. #Convert it into upper case using upper () function
6. newStr += str1[i].upper();
7. #Checks for upper case character
8. elif str1[i].isupper():
9. #Convert it into lower case using lower () function
10. newStr += str1[i].upper();
12. else:
13. newStr += str1[i];
14. print("String after case conversion : " +  newStr);

output:-

     HarSHaL

# PART:2 Implement Iris classifier project

=>

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv('Iris.csv')

df.head(50)

|  | **Id** | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **5** | 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| **6** | 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| **7** | 8 | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **8** | 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| **9** | 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **10** | 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **11** | 12 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| **12** | 13 | 4.8 | 3.0 | 1.4 | 0.1 | Iris-setosa |
| **13** | 14 | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| **14** | 15 | 5.8 | 4.0 | 1.2 | 0.2 | Iris-setosa |
| **15** | 16 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| **16** | 17 | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| **17** | 18 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| **18** | 19 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| **19** | 20 | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| **20** | 21 | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| **21** | 22 | 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| **22** | 23 | 4.6 | 3.6 | 1.0 | 0.2 | Iris-setosa |
| **23** | 24 | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |
| **24** | 25 | 4.8 | 3.4 | 1.9 | 0.2 | Iris-setosa |
| **25** | 26 | 5.0 | 3.0 | 1.6 | 0.2 | Iris-setosa |
| **26** | 27 | 5.0 | 3.4 | 1.6 | 0.4 | Iris-setosa |
| **27** | 28 | 5.2 | 3.5 | 1.5 | 0.2 | Iris-setosa |
| **28** | 29 | 5.2 | 3.4 | 1.4 | 0.2 | Iris-setosa |
| **29** | 30 | 4.7 | 3.2 | 1.6 | 0.2 | Iris-setosa |
| **30** | 31 | 4.8 | 3.1 | 1.6 | 0.2 | Iris-setosa |
| **31** | 32 | 5.4 | 3.4 | 1.5 | 0.4 | Iris-setosa |
| **32** | 33 | 5.2 | 4.1 | 1.5 | 0.1 | Iris-setosa |
| **33** | 34 | 5.5 | 4.2 | 1.4 | 0.2 | Iris-setosa |
| **34** | 35 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") | image = cv.imread("/content/photo.jpg") |
| **36** | 37 | 5.5 | 3.5 | 1.3 | 0.2 | Iris-setosa |
| **37** | 38 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **38** | 39 | 4.4 | 3.0 | 1.3 | 0.2 | Iris-setosa |
| **39** | 40 | 5.1 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **40** | 41 | 5.0 | 3.5 | 1.3 | 0.3 | Iris-setosa |
| **41** | 42 | 4.5 | 2.3 | 1.3 | 0.3 | Iris-setosa |
| **42** | 43 | 4.4 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **43** | 44 | 5.0 | 3.5 | 1.6 | 0.6 | Iris-setosa |
| **44** | 45 | 5.1 | 3.8 | 1.9 | 0.4 | Iris-setosa |
| **45** | 46 | 4.8 | 3.0 | 1.4 | 0.3 | Iris-setosa |
| **46** | 47 | 5.1 | 3.8 | 1.6 | 0.2 | Iris-setosa |
| **47** | 48 | 4.6 | 3.2 | 1.4 | 0.2 | Iris-setosa |
| **48** | 49 | 5.3 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **49** | 50 | 5.0 | 3.3 | 1.4 | 0.2 | Iris-setos |

df = df.drop(columns = ['Id'])

df.head()

|  | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** | **Species** |
| --- | --- | --- | --- | --- | --- |
| **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 |

df.describe()

|  | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** |
| --- | --- | --- | --- | --- |
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 SepalLengthCm 150 non-null float64

1 SepalWidthCm 150 non-null float64

2 PetalLengthCm 150 non-null float64

3 PetalWidthCm 150 non-null float64

4 Species 150 non-null object

dtypes: float64(4), object(1)

memory usage: 6.0+ KB

df['Species'].value\_counts()

Iris-setosa 50

Iris-versicolor 50

Iris-virginica 50

Name: Species, dtype: int64

df.isnull().sum()

SepalLengthCm 0

SepalWidthCm 0

PetalLengthCm 0

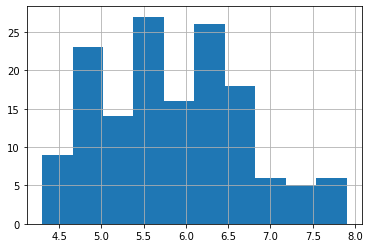
PetalWidthCm 0

Species 0

dtype: int64

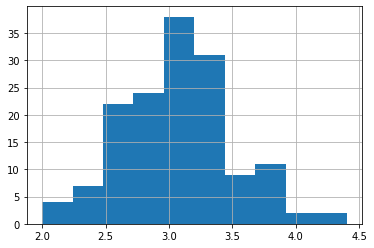
df['SepalLengthCm'].hist()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f94d3a7e670>



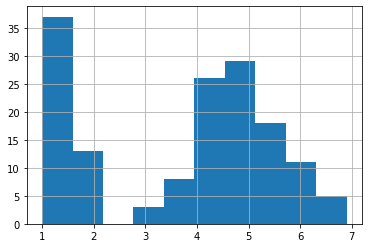
df['SepalWidhistthCm'].()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f94d60c7400>



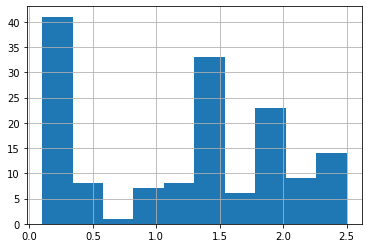
df['PetalLengthCm'].hist()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f94d35b8df0>



df['PetalWidthCm'].hist()

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f94d34db8e0>



colors = ['red', 'orange', 'blue']

species = ['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']

for i in range(3):

    x = df[df['Species'] == species[i]]

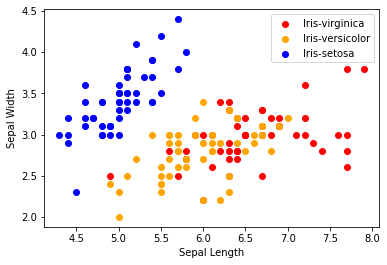
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c = colors[i], label=species[i])

plt.xlabel("Sepal Length")

plt.ylabel("Sepal Width")

plt.legend()

<matplotlib.legend.Legend at 0x7f94d341efd0>



for i in range(3):

    # filter data on each class

    x = df[df['Species'] == species[i]]

    # plot the scatter plot

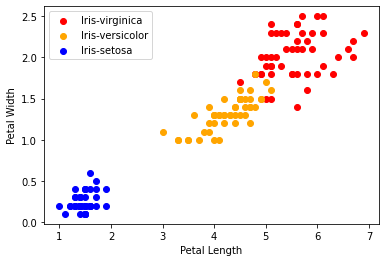
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i], label=species[i])

plt.xlabel("Petal Length")

plt.ylabel("Petal Width")

plt.legend()

<matplotlib.legend.Legend at 0x7f94d3383b80>



for i in range(3):

    # filter data on each class

    x = df[df['Species'] == species[i]]

    # plot the scatter plot

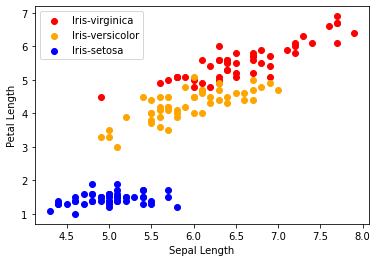
    plt.scatter(x['SepalLengthCm'], x['PetalLengthCm'], c = colors[i], label=species[i])

plt.xlabel("Sepal Length")

plt.ylabel("Petal Length")

plt.legend()

<matplotlib.legend.Legend at 0x7f94d32f4f10>



for i in range(3):

    # filter data on each class

    x = df[df['Species'] == species[i]]

    # plot the scatter plot

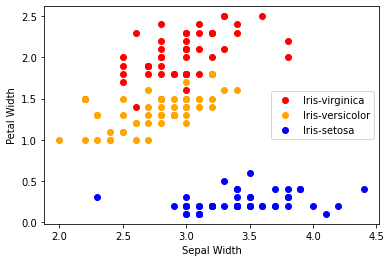
    plt.scatter(x['SepalWidthCm'], x['PetalWidthCm'], c = colors[i], label=species[i])

plt.xlabel("Sepal Width")

plt.ylabel("Petal Width")

plt.legend()

<matplotlib.legend.Legend at 0x7f94d326a910>



# display the correlation matrix

df.corr()

|  | **SepalLengthCm** | **SepalWidthCm** | **PetalLengthCm** | **PetalWidthCm** |
| --- | --- | --- | --- | --- |
| **SepalLengthCm** | 1.000000 | -0.109369 | 0.871754 | 0.817954 |
| **SepalWidthCm** | -0.109369 | 1.000000 | -0.420516 | -0.356544 |
| **PetalLengthCm** | 0.871754 | -0.420516 | 1.000000 | 0.962757 |
| **PetalWidthCm** | 0.817954 | -0.356544 | 0.962757 | 1.000000 |

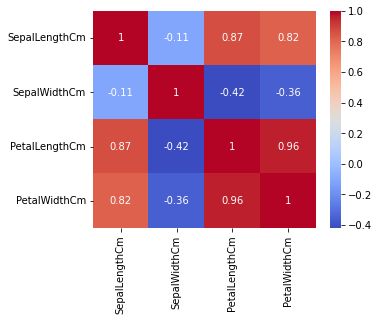
corr = df.corr()

# plot the heat map

fig, ax = plt.subplots(figsize=(5,4))

sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f94d324c910>



from sklearn.model\_selection import train\_test\_split

## train - 70%

## test - 30%

# input data

X = df.drop(columns=['Species'])

# output data

Y = df['Species']

# split the data for train and test

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.30)

# logistic regression

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

# model training

model.fit(x\_train, y\_train)

KNeighborsClassifier()

print("Accuracy: ",model.score(x\_test, y\_test) \* 100)

Accuracy: 95.55555555555556

 decision tree

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()

model.fit(x\_train, y\_train)

model.fit(x\_train, y\_train)

LogisticRegression()

print("Accuracy: ",model.score(x\_test, y\_test) \* 100)

Accuracy: 95.55555555555556

# PART:32. Try to import haarcascade algorithm for face detection in ide (.xml).

import cv2 as cv

from google.colab.patches import cv2\_imshow

from IPython.display import display, Javascript

from google.colab.output import eval\_js

from base64 import b64decode

def take\_photo(filename='photo.jpg', quality=0.8):

  js = Javascript('''

    async function takePhoto(quality) {

      const div = document.createElement('div');

      const capture = document.createElement('button');

      capture.textContent = 'Capture';

      div.appendChild(capture);

      const video = document.createElement('video');

      video.style.display = 'block';

      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);

      div.appendChild(video);

      video.srcObject = stream;

      await video.play();

      // Resize the output to fit the video element.

      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.

      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');

      canvas.width = video.videoWidth;

      canvas.height = video.videoHeight;

      canvas.getContext('2d').drawImage(video, 0, 0);

      stream.getVideoTracks()[0].stop();

      div.remove();

      return canvas.toDataURL('image/jpeg', quality);

    }

    ''')

  display(js)

  data = eval\_js('takePhoto({})'.format(quality))

  binary = b64decode(data.split(',')[1])

  with open(filename, 'wb') as f:

    f.write(binary)

  return filename

from IPython.display import Image

try:

  filename = take\_photo()

  print('Saved to {}'.format(filename))

  # Show the image which was just taken.

  display(Image(filename))

except Exception as err:

  # Errors will be thrown if the user does not have a webcam or if they do not

  # grant the page permission to access it.

  print(str(err))

Saved to photo.jpg



image = cv.imread("/content/photo.jpg")

!wget --no-check-certificate \

https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade\_frontalface\_default.xml \

-O Haarcascade\_frontalface\_default.xml

--2022-12-30 16:57:45-- <https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_frontalface_default.xml>

Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...

Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 930127 (908K) [text/plain]

Saving to: ‘Haarcascade\_frontalface\_default.xml’

Haarcascade\_frontal 100%[===================>] 908.33K --.-KB/s in 0.01s

2022-12-30 16:57:45 (90.5 MB/s) - ‘Haarcascade\_frontalface\_default.xml’ saved [930127/930127]

yhaar\_cascade = cv.CascadeClassifier(cv.samples.findFile

                                    (cv.data.haarcascades+"haarcascade\_frontalface\_default.xml"))

Face\_detect = haar\_cascade.detectMultiScale(image)

for x, y, w, h in Face\_detect:

  cv.rectangle(image,(x, y),(x+w, y+h),(0, 255, 0),thickness=2)

cv2\_imshow(image)

