AI ASSIGNMENT-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.

In [1]:
```
a=list(input("give numbers").split())
b=[]
for i in range(len(a)):
    a[i]=int(a[i])
    b.append(int((a[i])**(1/2)))
print(b)
```

```
[2, 4, 5]
```

here i am not calling function as it is a infinite loop

In [3]:
```
def operation():
    i=0
    while i>=0:
        if i%3==0 | i%5==0:
            print(i)
        i+=1
```

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

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

In [4]:
```
a=input("give a letter")
b=['a','e','i','o','u']
if a in b:
    print("letter is a vowel")
elif(a.isalpha()):
    print("letter is a consonant")
else:
    print("given input is not a letter")
```

```
given input is not a letter
```

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

In [6]:
```
a=input(list(("give two characters").split()))
z=ord(a[0])
y=ord(a[2])
x=z-y
if(x<0):
    x=-x
print(x)
```

```
3
```

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

In [ ]:
```
a=input("give a string")
b=['a','e','i','o','u']
def vowels(a):
    count=0
    for i in range(len(a)):
        if a[i] in b:
            count+=1
    return count
z=vowels(a)
print(z)
```

```
1
```

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

we are using ascii table to determine range

In [ ]:
```
for i in range(65,91):
    print(chr(i))
```

```
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
```

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

In [ ]:
```
a=list(input("give list of numbers").split())
sum=0
print(a)
for i in range(len(a)):
    a[i]=int(a[i])
    if a[i]%2==0:
```

```
        sum+=a[i]
print(sum)
```

```
['2', '4', '6', '5', '7']
12
```

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

In [ ]:

```
for i in range(20):
    if i%3!=0:
        print(i**2)
```

```
1
4
16
25
49
64
100
121
169
196
256
289
361
```

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

In [ ]:

```
a=input("give string")
b=input("give string2")
c=''
for i in a:
    if i=='A':
        i='a'
    c+=i
d=''
for i in b:
    if i=='A':
        i='a'
    d+=i
e=c+d
print(e)
```

```
aaaaaaDSaDaadaa
```

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

In [ ]:

```
a=[int(i) for i in input("give numbers").split() if int(i) % 2!=0]
print(a)
```

```
[5, 7]
```

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”)

In [ ]:
```
a=input("give a string")
b=''
for i in a:
    if i.isupper():
        c=i.lower()
    elif i.islower():
        c=i.upper()
    b+=c
print(b)
```

```
acdA
```

part 2 IRIS CLASSIFIER PROJECT we are importing dataset of some of flower parts

In [ ]:
```
from sklearn.datasets import load_iris
x,y=load_iris(return_X_y=True)
```

we can input from local files we need to give the path where it is located

In [ ]:
```
file_1="C:\Users\admin\Downloads\plot_iris_dataset.ipynb"
```

In [ ]:
```
x[0:9]
```

Out[ ]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2]])
```

In [ ]:
```
y
```

Out[ ]:
```
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, 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, 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, 2])
```

here 0,1,2 denote different names of flowers

In [ ]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.1)
```

we are using this module to divide our data into training data to get a relation and testing data to check if predicted value is equal to orginal output in data

In [ ]:
```
from sklearn.linear_model import LinearRegression
s=LinearRegression()
d=s.fit(xtrain,ytrain)
```

In [ ]:
```
predict=s.predict(xtest)
for i in range(len(predict)):
    predict[i]=round(predict[i])
predict
```

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

In [ ]:
```
from sklearn.metrics import accuracy_score
print((accuracy_score(predict,ytest))*100)
```

```
93.33333333333333
```

we get 93.3333% of accuracy using linear regression method

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
a=DecisionTreeClassifier()
z=a.fit(xtrain,ytrain)
prediction=a.predict(xtest)
prediction
```

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

In [ ]:
```
from sklearn.metrics import accuracy_score
print((accuracy_score(prediction,ytest))*100)
```

```
86.66666666666667
```

we get 86.66% using DecisionTreeClassifer method

now upon changing split ratio from0.1 to 0.2

In [ ]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
a=DecisionTreeClassifier()
z=a.fit(xtrain,ytrain)
prediction=a.predict(xtest)
prediction
from sklearn.metrics import accuracy_score
print((accuracy_score(prediction,ytest))*100)
```

```
83.33333333333334
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
s=LinearRegression()
d=s.fit(xtrain,ytrain)
```

```
predict=s.predict(xtest)
for i in range(len(predict)):
    predict[i]=round(predict[i])
from sklearn.metrics import accuracy_score
print((accuracy_score(prediction,ytest))*100)
```

```
83.33333333333334
```

we get 83.3% on changing split ratio()decisiontree method and linear regression method for 0.2 of total size as test size

In [ ]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.6)
from sklearn.linear_model import LinearRegression
s=LinearRegression()
d=s.fit(xtrain,ytrain)
predict=s.predict(xtest)
for i in range(len(predict)):
    predict[i]=round(predict[i])
from sklearn.metrics import accuracy_score
print((accuracy_score(predict,ytest))*100)
```

```
96.66666666666667
```

keeping 0.6 of total data as test size gives us 96.66%(highest accuracy so far)

part 3 haarcascade

In [ ]:

```
import cv2
a=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

image is converted inform of matrices

In [ ]:

```
from matplotlib.image import imread
img=imread("C:\\Users\\admin\\Pictures\\Saved Pictures\\1085-carlos-fuentes.jpg",0)
print(img)
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
[[[ 81  81  81]
  [ 82  82  82]
  [ 84  84  84]
  ...
  [198 198 198]
  [188 188 188]
  [188 188 188]]

 [[ 87  87  87]
  [ 88  88  88]
  [ 89  89  89]
  ...
  [200 200 200]
  [192 192 192]
  [192 192 192]]

 [[ 99  99  99]
  [ 99  99  99]
  [ 98  98  98]
  ...
```

```
 [202 202 202]
 [198 198 198]
 [198 198 198]]

...

[[ 13  13  13]
 [ 15  15  15]
 [ 17  17  17]
 ...
 [  3   3   3]
 [  0   0   0]
 [  9   9   9]]

[[  0   0   0]
 [  0   0   0]
 [  0   0   0]
 ...
 [  0   0   0]
 [  0   0   0]
 [ 11  11  11]]

[[ 12  12  12]
 [ 10  10  10]
 [  7   7   7]
 ...
 [  0   0   0]
 [  1   1   1]
 [ 13  13  13]]]
```

cvtColor converts color of img from one to another (rgb to gray shades)

In [ ]:
```
faces=a.detectMultiScale(gray,1.1,4)
```

detectMultiScale divides the image into smaller rectangles ,its arguments include size of the object and no of objects

In [ ]:
```
for (a,b,c,d) in faces:
    cv2.rectangle(img,(a,b),(a+d,b+c),(0,255,0),4)
cv2.imshow('image',img)
cv2.waitKey(20000)
cv2.destroyAllWindows()
```

this gives us the mark of green color box(rectangle)