1. calculator

```
print("Operation: +, -, *, /")
select = input("Select operations: ")

#get inputs

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))


# check operations and display result
# add(+) two numbers
if select == "+":
print(num1, "+", num2, "=", num1+num2)
# subtract (-) two numbers
elif select == "-":
        print(num1, "-", num2, "=", num1-num2)
# multiply (*) two numbers
elif select == "*":
  print(num1, "*", num2, "=", num1*num2)
# divide (/) one number by another
elif select == "-":
         print(num1, "/", num2, "=", num1/num2)
else:
         print ("Invalid input")
```

2.ARMSTRONG SERIES

```
lower = int(input("Enter the lower range : "))
upper = int(input("Enter the upper range : "))
for num in range(lower, upper + 1):
  # order of number
  order = len(str(num))

#initialize sum
  sum = 0
  temp = num
  while temp > 0:
   digit = temp%10
   sum+=digit**order
   temp//=10
  if num == sum:
    print (num)
```

3.FIBONACCI SERIES

```
nterms = int(input("How many terms? "))
# first two terms
n1, n2 = 0, 1
count = 0
# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence
else:
  print("Fibonacci sequence:")
  while count < nterms:

    print(n1)
    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1
```

4.MODULES AND FUNCTIONS

```
def summation(a,b):
    return a+b
def multiplication(a,b):
    return a*b
def subtraction(a,b):
    return a-b
def division(a,b):
    return a/b
a=int(input("Enter the first number"))
b=int(input("Enter the second number"))
print("Sum = ",summation(a,b))
print("product = ",multiplication(a,b))
print("subtract = ",subtraction(a,b))
print("divisor = ",division(a,b))
```

5(A).WORKING WITH STRINGS

```
#Working with strings
#a.from input string count special character.....
def Count(str):
    alpha,upper,lower,number,special = 0,0,0,0,0
    for i in range(len(str)):
        if str[i].isalpha():
            alpha+= 1
        if str[i].isupper():
            upper+= 1
        elif str[i].islower():
            lower+= 1
        elif str[i].isdigit():
            number+= 1
        elif str[i]!="":
            special += 1
    print('Digits:',number)
    print('Alphabets:',alpha)
    print('Special characters:',special)
    print('lowercase:',lower)
    print('uppercase:',upper)

str =input("Enter a string:")
Count(str)
```

5(B).WELCOME

```
import math
N, M = map(int,input("Enter N and M: ").split())
for i in range(0,math.floor(N/2)):
    s = '.|.'*i
    print(s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
print('WELCOME'.center(M,'-'))
for i in reversed(range(0,math.floor(N/2))):
    s = '.|.'*i
    print(s.rjust(math.floor((M-2)/2),'-')+'.|.'+('.|.'*i).ljust(math.floor((M-2)/2),'-'))
```

6.DATA PREPROCESSING

```
import pandas as pd

df = pd.read_csv("/Heart.csv")
df

df.describe()

df.info()

df.replace(0,'NAN')

df.dropna()

df.fillna(df.mean())

x = df.iloc[:,0:14].values
x

y = df.iloc[:,14].values
y

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=0)
print(x_train.shape)
print(x_test.shape)
```

7.MANIPULATE THE TWITTER DATASET

```
import pandas as pd
import numpy as np
import re
data = pd.read_csv("tweets1.csv")
data


def remove_pattern(input_txt, pattern):
 r = re.findall(pattern,input_txt)
 for i in r:
   input_txt = re.sub(i,'',input_txt)
 return input_txt
print(data)


data['new'] = np.vectorize(remove_pattern)(data ['text'],"@[\w]*")
print(data)


data['new'] = data['new'].str.replace("[^a-zA-Z#]"," ")
print(data)



data['new'] = data['new'].apply(lambda x:' '.join([w for w in x.split() if
len(w) > 3]))
print (data)



tokenized_tweet = data['new'].apply (lambda x:x.split())
print (tokenized_tweet.head())



from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x:[stemmer.stem(i) for i in
    x])
print (tokenized_tweet.head())
```

8.EVALUATING THE RESULTS OF ML

```
y=['0','1','0','1','1','1','0','1','0','1','0','0','0','1','1','1','0','1','1','0']
y_pred = ['0','0','0','0','1','0','1','1','1','1','0','0','0','0','0','1','0','1','1','0']
print (y)
print(y_pred)

j=0
TP,TN,FP,FN = 0,0,0,0
for i in y:
  if i == '1' and y_pred[j] =='1':
    TP +=1
  elif i == '0' and y_pred[j] =='0':
    TN +=1
  elif i == '1' and y_pred[j] =='0':
    FP +=1
  elif i == '0' and y_pred[j] =='1':
    FN+=1
  j+=1
confusion_matrix = [TP,TN,FP,FN]
print ("Confusion Matrix : ", confusion_matrix)
ACC = (TP+TN) / (TP+FP+TN+FN)
print ("ACCURACY : ", ACC)
PREC = TP / (TP+FP)
print ("PRECISION : ", PREC)
REC = TP / (TP+FN)
print ("RECALL : ", REC)
SN = TP/ (TP+FN)
print ("SENSITIVITY : ", SN)
SP = TN/ (TN+FP)
print ("SPECIFICITY : ", SP)
MCE = 1-ACC
print ("MISCLASSIFICATION ERROR : ", MCE)
```

9.IMPLEMENT CORELATION AND REGRESSION TECNIQUES

```
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt


df=pd.read_csv(r"Salary_Data.csv")
x=list(df["YearsExperience"])
y=list(df["Salary"])
df


def LinearRegressor(x,y):
    sumX=sum(x)
    sumY=sum(y)
    xMean=sumX/len(x)
    yMean=sumY/len(y)
    x_minus_xmean=[val-xMean for val in x]
    y_minus_ymean=[val-yMean for val in y]
    zip_li=zip(x_minus_xmean,y_minus_ymean)
    val=[x*y for x,y in zip_li]
    b1=sum(val)/sum([x**2 for x in x_minus_xmean])
    b0=yMean-b1*xMean
    return b0,b1


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/2,shuffle=True)
b=LinearRegressor(x_train,y_train)
y_pred=[b[0]+b[1]*val for val in x_test]


r2_score(y_test,y_pred)

plt.plot(x_test,y_pred)
plt.scatter(x_test, y_test,c="k")


from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np



print( "RMSE: ",np.sqrt( mean_squared_error( y_test, y_pred ) ))
#R-squared value
print( "R-squared: ",r2_score( y_test, y_pred ) )
```

10.NAVIE BAYESIAN CLASSIFICATION

```
import pandas as pd
data=pd.read_csv("Iris.csv")
X =data.iloc[:,[1,2,3,4]].values
y =data.iloc[:,5].values


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)



from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred=gnb.predict(X_test)


from sklearn import metrics
print("Classification Accuracy:", metrics.accuracy_score(y_test, y_pred)*100)
cm=metrics.confusion_matrix(y_test,y_pred)
print(cm)



import seaborn as sn
from matplotlib import pyplot as plt



plt.figure(figsize=(5,4))
plt.xlabel('Predicted value')
plt.ylabel('Actual value')
plt.show()


sn.heatmap(cm,annot=True)
```

11.K MEANS CLUSTERING

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("Mall_Customers.csv")
df.head(3)



len(df)

X = df.iloc[:, [3,4]].values
X[0:5]


from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init ='k-means++', max_iter=300, n_init=10, random_state=0 )
kmeans.n_clusters

y_kmeans = kmeans.fit_predict(X)


df['cluster'] = y_kmeans
df.head()

print(y_kmeans.shape)

plt.scatter(X[y_kmeans==0, 0], X[y_kmeans==0, 1], s=100, c='red', label ='Cluster 1')
plt.scatter(X[y_kmeans==1, 0], X[y_kmeans==1, 1], s=100, c='blue', label ='Cluster 2')
plt.scatter(X[y_kmeans==2, 0], X[y_kmeans==2, 1], s=100, c='green', label ='Cluster 3')
plt.scatter(X[y_kmeans==3, 0], X[y_kmeans==3, 1], s=100, c='cyan', label ='Cluster 4')
plt.scatter(X[y_kmeans==4, 0], X[y_kmeans==4, 1], s=100, c='magenta', label ='Cluster 5')




plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='yellow', label = 'Centroids')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income(k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

4. modules and functions