Data Analysis & Visualization Lab

Name: Alle Pranav Sudhan

Class: AI &DS (B2)

UGID: 21WU0102058

Year: 2021-2025

lab-1

In [1]:

```
1
2  num1 = float(input(" Please Enter the First Value Number 1: "))
3  num2 = float(input(" Please Enter the Second Value Number 2: "))
4
5  # Add Two Numbers
6  add = num1 + num2
7
8  sub = num1 - num2
9
10
11 multi = num1 * num2
12
13
14 div = num1 / num2
15
16 # Modulus of num1 and num2
17 mod = num1 % num2
18
19 # Exponent of num1 and num2
20 expo = num1 ** num2
21
22 print("The Sum of {0} and {1} = {2}".format(num1, num2, add))
23 print("The Subtraction of {0} from {1} = {2}".format(num2, num1, sub))
24 print("The Multiplication of {0} and {1} = {2}".format(num1, num2, multi))
25 print("The Division of {0} and {1} = {2}".format(num1, num2, div))
26 print("The Modulus of {0} and {1} = {2}".format(num1, num2, mod))
27 print("The Exponent Value of {0} and {1} = {2}".format(num1, num2, expo))
```

```
 Please Enter the First Value Number 1: 10
 Please Enter the Second Value Number 2: 20
The Sum of 10.0 and 20.0 = 30.0
The Subtraction of 20.0 from 10.0 = -10.0
The Multiplication of 10.0 and 20.0 = 200.0
The Division of 10.0 and 20.0 = 0.5
The Modulus of 10.0 and 20.0 = 10.0
The Exponent Value of 10.0 and 20.0 = 1e+20
```

In [51]:

```python
# To Display a basic calculator
# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
    return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

print("Please select operation -\n" \
        "1. Add\n" \
        "2. Subtract\n" \
        "3. Multiply\n" \
        "4. Divide\n")


# Take input from the user
select = int(input("Select operations form 1, 2, 3, 4 :"))

number_1 = int(input("Enter first number: "))
number_2 = int(input("Enter second number: "))

if select == 1:
    print(number_1, "+", number_2, "=",
                    add(number_1, number_2))

elif select == 2:
    print(number_1, "-", number_2, "=",
                    subtract(number_1, number_2))

elif select == 3:
    print(number_1, "*", number_2, "=",
                    multiply(number_1, number_2))

elif select == 4:
    print(number_1, "/", number_2, "=",
                    divide(number_1, number_2))
else:
    print("Invalid input")
```

```
Please select operation -
1. Add
2. Subtract
3. Multiply
```

```
4. Divide

Select operations form 1, 2, 3, 4 :4
Enter first number: 2
Enter second number: 3
2 / 3 = 0.6666666666666666
```

In [52]:
```
# c) Calculate the net salary of an employee
name= str(input("Enter name of employee:"))
basic=float(input("Enter Basic Salary :"))
da=float(basic*0.25)
hra=float(basic*0.15)
pf=float((basic+da)*0.12)
ta=float(basic*0.075)
netpay=float(basic+da+hra+ta)
grosspay=float(netpay-pf)

print("\n\n") print("S A L A R Y   D E T A I L E D   B R E A K U P ")
print("================================================") print(" NAME OF EMPLOYEE : ",name) print(" BASIC SALARY : ",basic) print(" DEARNESS ALLOW. : ",da) print(" HOUSE RENT ALLOW.: ",hra) print(" TRAVEL ALLOW. : ",ta)
print("================================================") print(" NET SALARY PAY : ",netpay) print(" PROVIDENT FUND : ",pf)
print("================================================") print(" GROSS PAYMENT : ",grosspay)
print("================================================")
```

```
Enter name of employee: pranav
Enter Basic Salary :50000



S A L A R Y   D E T A I L E D   B R E A K U P
================================================
 NAME OF EMPLOYEE :  pranav
 BASIC SALARY :  50000.0
 DEARNESS ALLOW. :  12500.0
 HOUSE RENT ALLOW.:  7500.0
 TRAVEL ALLOW. :  3750.0
================================================
 NET SALARY PAY :  73750.0
 PROVIDENT FUND :  7500.0
================================================
 GROSS PAYMENT :  66250.0
================================================
```

[53]:
```
# d) Print factorial of n numbers
num = int(input("Enter a number: "))
factorial = 1    if num < 0:
```

In

```
    print(" Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
        for i in range(1,num + 1):
        factorial = factorial*i
        print("The factorial
        of",num,"is",factorial)
```

```
Enter a number: 5
```

In [54]:

```
The factorial of 5 is 120
```

```
# e)    Circulate the values of n variables
n = int(input("Enter number of values : "))
list1 = []
for val in range(0,n,3):
    ele = int(input("Enter integer : "))
    list1.append(ele)
print("Circulating the elements of list ", list1)
for val in range(0,n,1):
    ele = list1.pop(0)
    list1.append(ele)
    print(list1)
```

```
Enter number of values : 5
Enter integer : 2
        Enter integer : 3
        Circulating the elements of list  [2, 3]
        [3, 2]
        [2, 3]
        [3, 2]
        [2, 3]
        [3, 2]
```

In [55]:

```
# Python program to check if year is a leap year or not

year=int(input("Enter the year: "))

# To get year (integer input) from the user
# year = int(input("Enter a year: "))

    # divided by 100 means century year (ending with 00)
    # century year divided by 400 is leap year 10 if (year % 400 == 0) and (year % 100 == 0):
   print("{0} is a leap year".format(year))

# not divided by 100 means not a century year
# year divided by 4 is a leap year
elif (year % 4 ==0) and (year % 100 != 0):
   print("{0} is a leap year".format(year))

# if not divided by both 400 (century year) and 4 (not century year)
# year is not leap year 20 else:
   print("{0} is not a leap year".format(year))
```

```
Enter the year: 2016
```

In [56]:

```
2016 is a leap year
```

```
# g)    If the given number is Palindrome or not
num=int(input("Enter a number: "))
temp=num
rev=0
while(num>0):
    dig=num%10
    rev=rev*10+dig
    num=num//10
if(temp==rev):
    print("The number is palindrome!")
else:
    print("Not a palindrome!")
```

```
Enter a number: 203
Not a palindrome!
```

In [57]:

```python
#perform 2x2 matrix operations using python library
#import numpy
import numpy as np
mat1=np.array([[12,11],[32,31]])
mat2=np.array([[34,55],[17,30]])
print("Matrix1\n",mat1)
print("Matrix2\n",mat2)

#addition
print ("\nAddition of two matrices: ")
print (np.add(mat1,mat2))

#multiplication
print ("\nMultiplication of two matrices: ")
print (np.multiply(mat1,mat2))

#transpose
print("Transpose of 2x2 matrix:\n",mat1.T)
```

```
Matrix1
 [[12 11]
 [32 31]]
Matrix2
 [[34 55]
 [17 30]]

Addition of two matrices:
[[46 66]
 [49 61]]

Multiplication of two matrices:
[[408 605]
 [544 930]] Transpose of
2x2 matrix:
 [[12 32]
 [11 31]]
```

In [58]:

```
List_Size = int(input("Enter the list Size: "))
Position =0
aList=[]
while(Position < List_Size):
    avalue=int(input("Enter a value "))
    aList.append(avalue)
    Position+=1

print(aList)
aList.append(0)
avalue=int(input("Enter a card to insert ")) # a new card

Position = List_Size-1
while(Position >=0):
    if(avalue<aList[Position]):
        aList[Position+1]=aList[Position]
        aList[Position]=0
    else:
        aList[Position+1]=avalue
        break

    Position-=1

print(aList)
```

```
Enter the list Size: 5
Enter a value 2
       Enter a value 3
       Enter a value 6
       Enter a value 9
       Enter a value 8
       [2, 3, 6, 9, 8]
       Enter a card to insert 5
```

In
In [59]:
[2, 3, 5, 6, 9, 8]

```
a=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt", 'r')
line = 0
word = 0
character = 0
#count = 0
for count, line in enumerate(a):
    character = character +len(line)
    words = line.split (  )
    word = word + len(words)

#print('Number of line', line)
print('Number of character', character)
print('Number of words', word)
print('Total Number of lines:', count + 1)

```

Number of character 38
Number of words 7
Total Number of lines: 1

[60]:

```
#read
aa=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt", 'r')
for line in aa:
    print(line)

 #Splitting line in a text line:
aa=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt", 'r+')
for line in aa:
    words=line.split()
    print(words)


```

this file is pranav.txtto add more lines ['this', 'file',
'is', pranav.txtto', 'add', 'more', 'lines']

In

In [48]:

```

#write to a file
aa=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt", 'w+')
aa.write('this file is pranav.txt')
aa.write('to add more lines')
aa.close()

```

In [61]:

```
#copy the contents of one file to another
source=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt", 'r')
destination=open("C:\\Users\\ pranav \\OneDrive\\Desktop\\ pranav.txt" , 'w
for line in source:
    destination.write(line)
source.close()
destination.close()
```

```

```

In [ ]:

lab-2

In [15]:
```
# 4) Perform the following data exploring analysis on the download dataset.
```

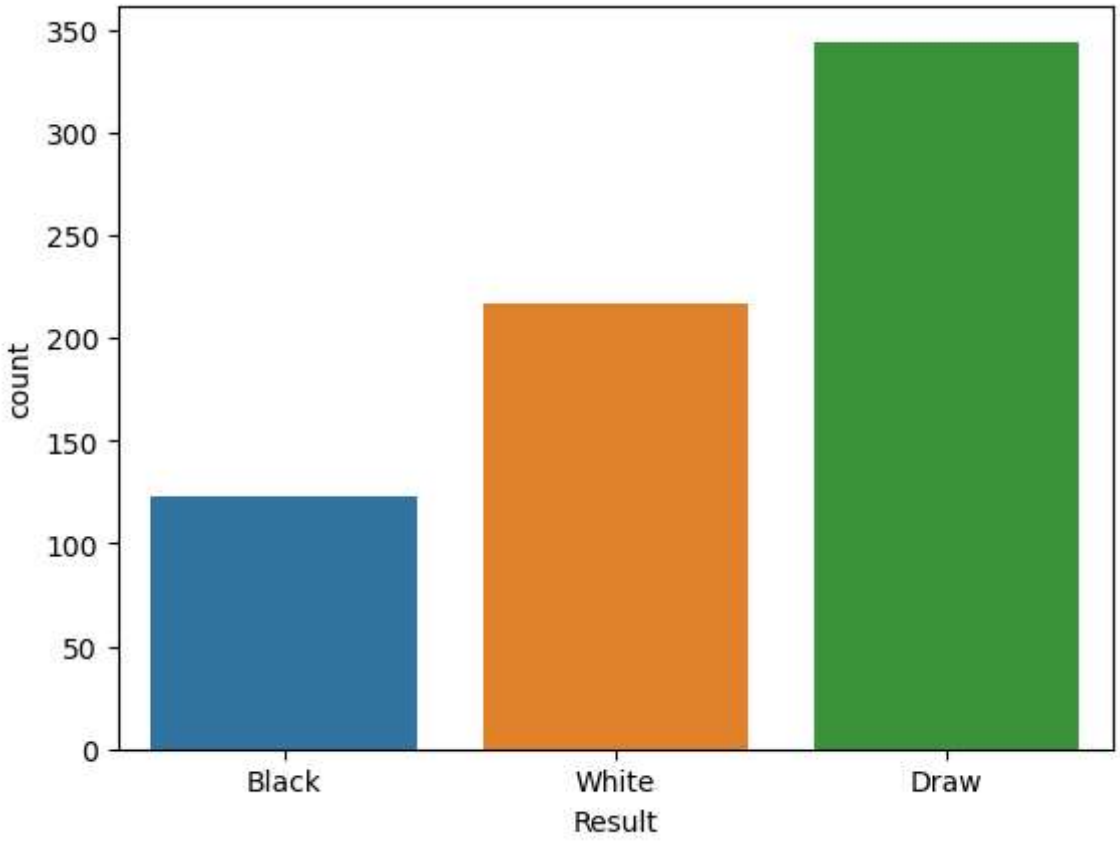
In [16]:
```
df = pd.read_csv(r"C:\Users\prana\Downloads\Game_Nodes.csv")
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [17]:
```
df1 = pd.read_csv(r"C:\Users\prana\Downloads\Game_Nodes.csv")
df1.dtypes
```

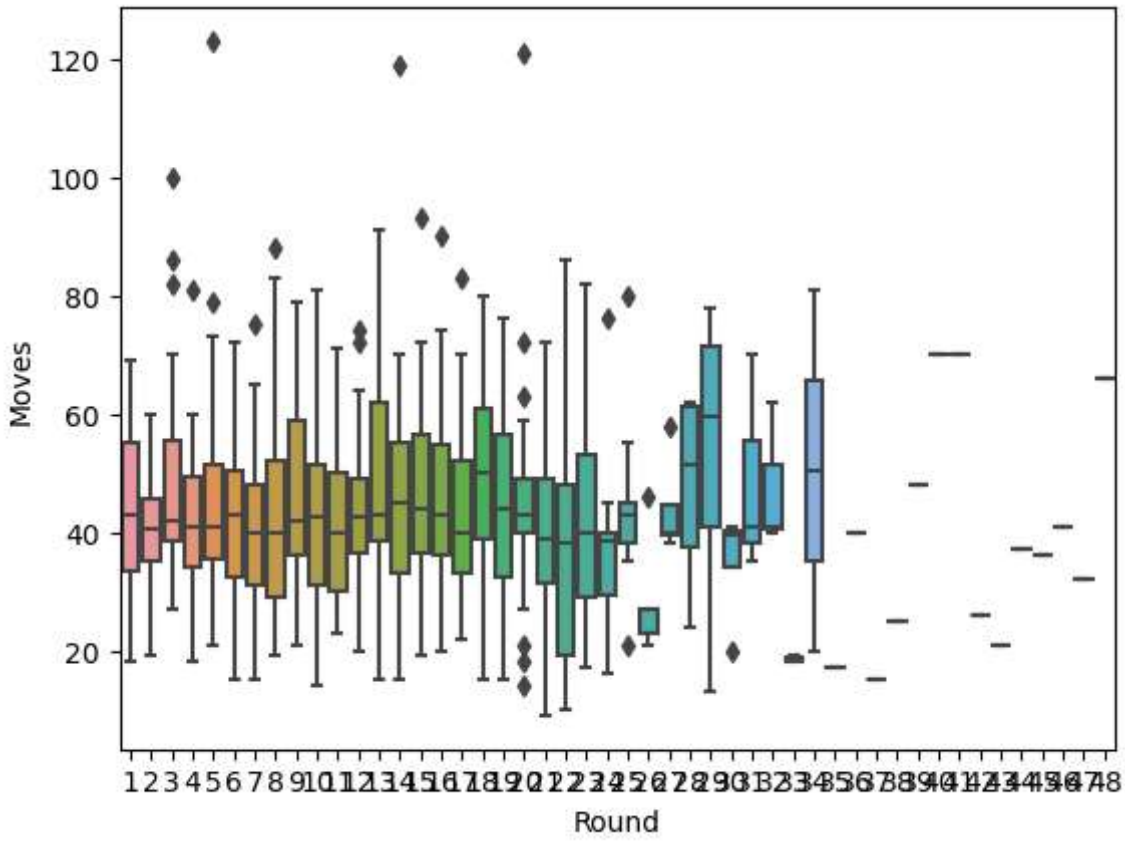
Out[17]:
```
Unnamed: 0     int64
Black         object
BlackElo       int64
Date          object
ECO           object
Event         object
EventDate     object
GameNumber     int64
HalfMoves      int64
Moves          int64
Opening       object
Result        object
Round          int64
Site          object
White         object
WhiteElo       int64
dtype: object
```

In [22]:
```
# 1.COUNT PLOT
sns.countplot(df1['Result'])
plt.show()
```

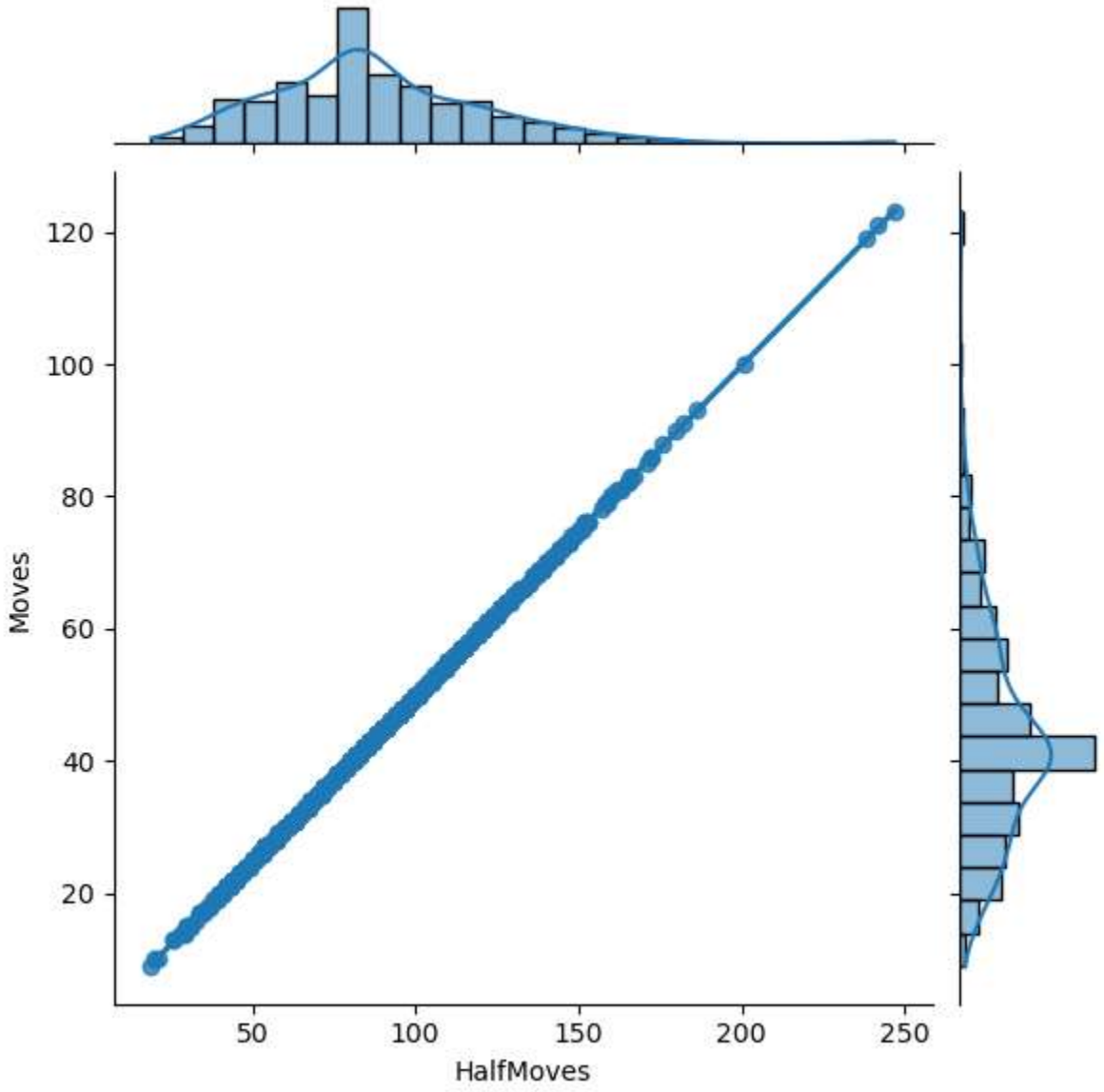
```
C:\Users\prana\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [30]:
```
# 2.Box Plot
sns.boxplot(x='Round', y='Moves', data=df1 , showfliers=True)
plt.show()
```



In [33]:
```
# 3.Joint Plot
sns.jointplot(x='HalfMoves',y='Moves',data=df1, kind='reg')
plt.show()
```



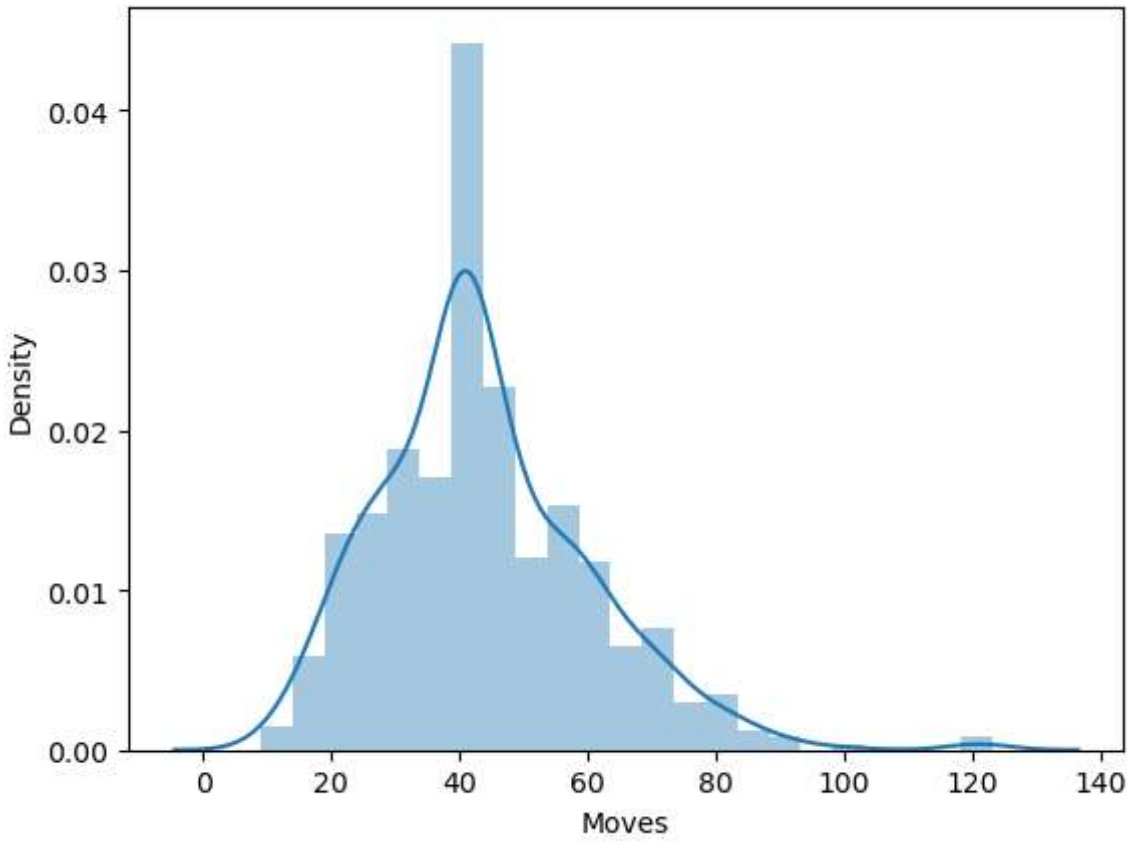
In [34]:
```
# 4.Correlation
from scipy.stats import pearsonr
def get_correlation(column1, column2, df):
    pearson_corr, p_value = pearsonr(df[column1], df[column2])
    print("Correlation between {} and {} is {}".format(column1, column2, pears
    print("P-value of this correlation is {}".format(p_value))
```

In [36]:
```
get_correlation('Round','Moves', df1)
```

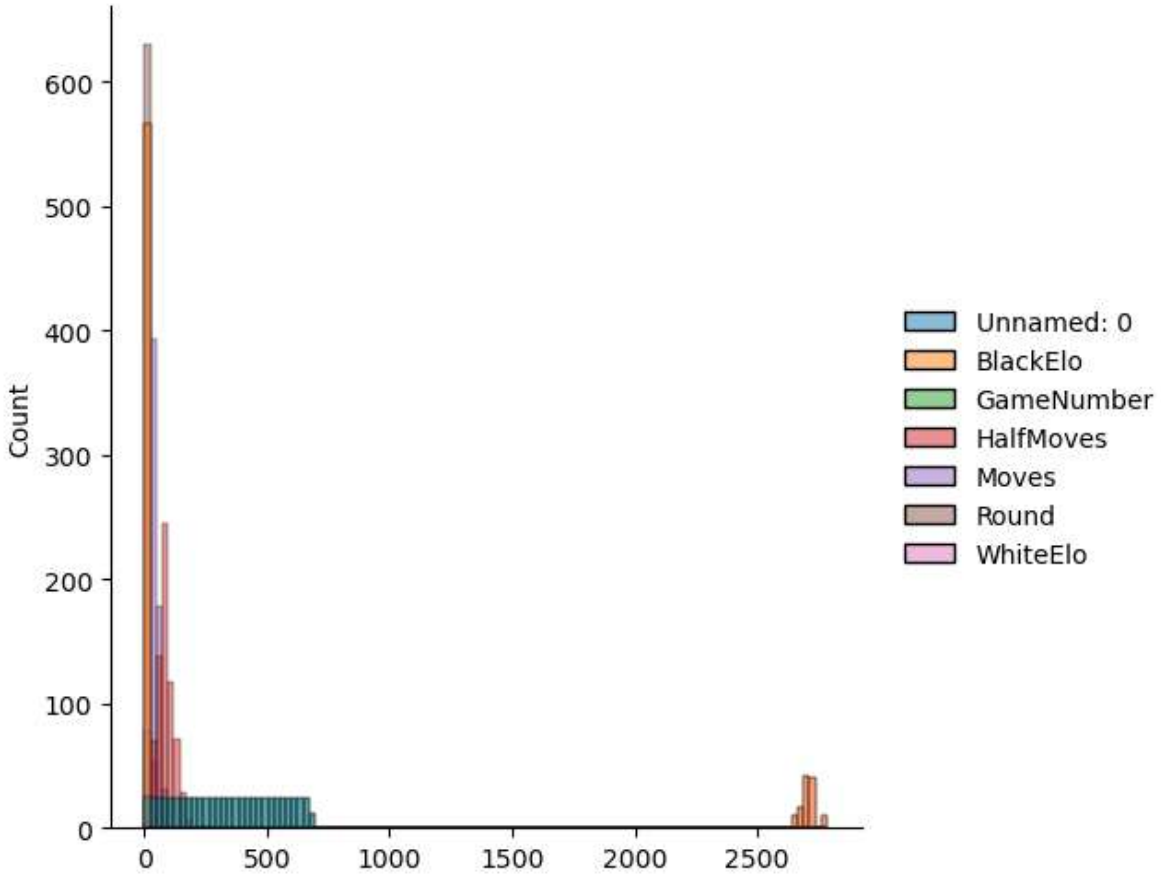
```
Correlation between Round and Moves is -0.03340444882483328
P-value of this correlation is 0.38305278900972234
```

In [37]:
```
# 6.Distribution Plot
sns.distplot(df1['Moves'])
plt.show()
```

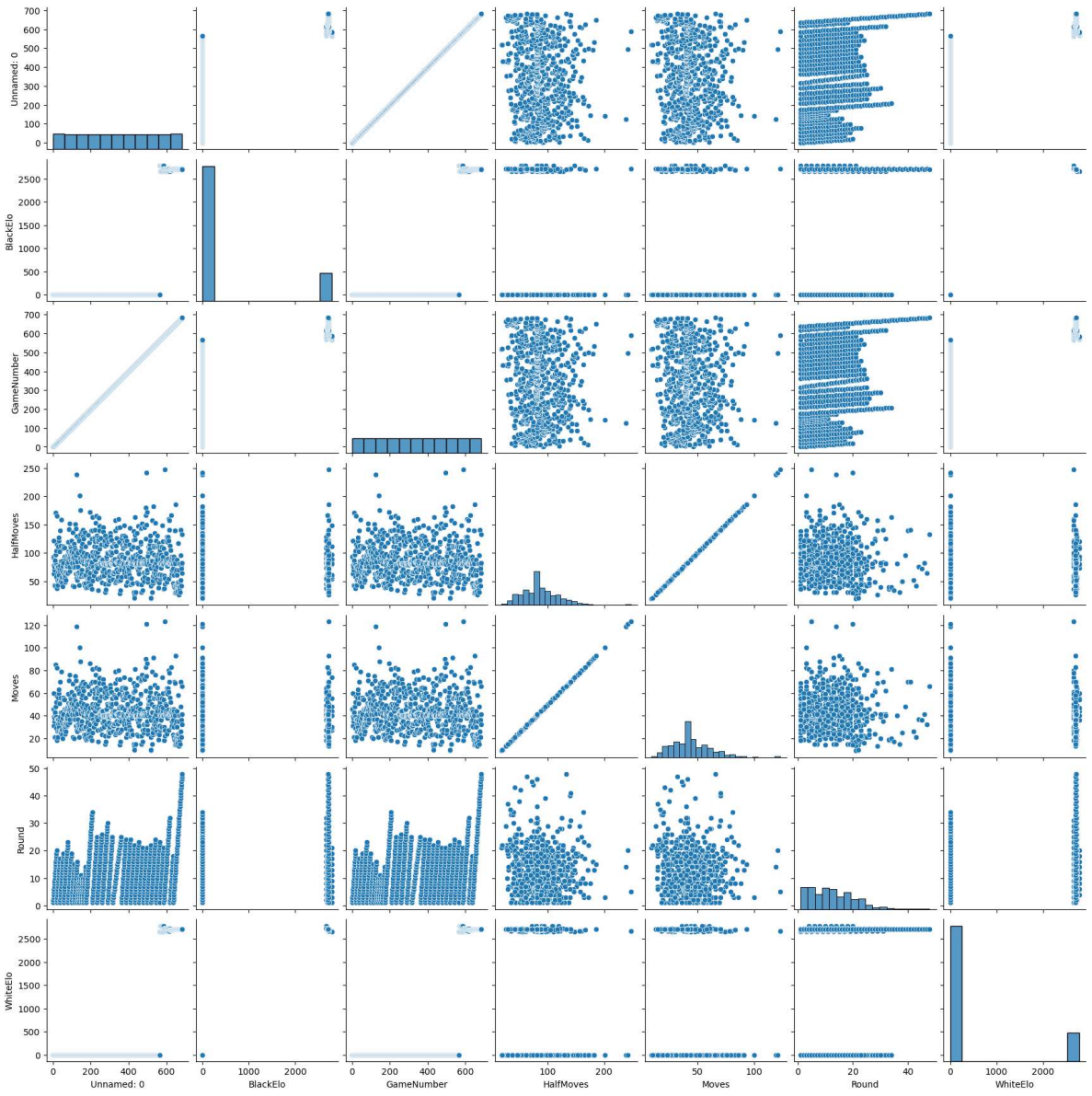
```
C:\Users\prana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```



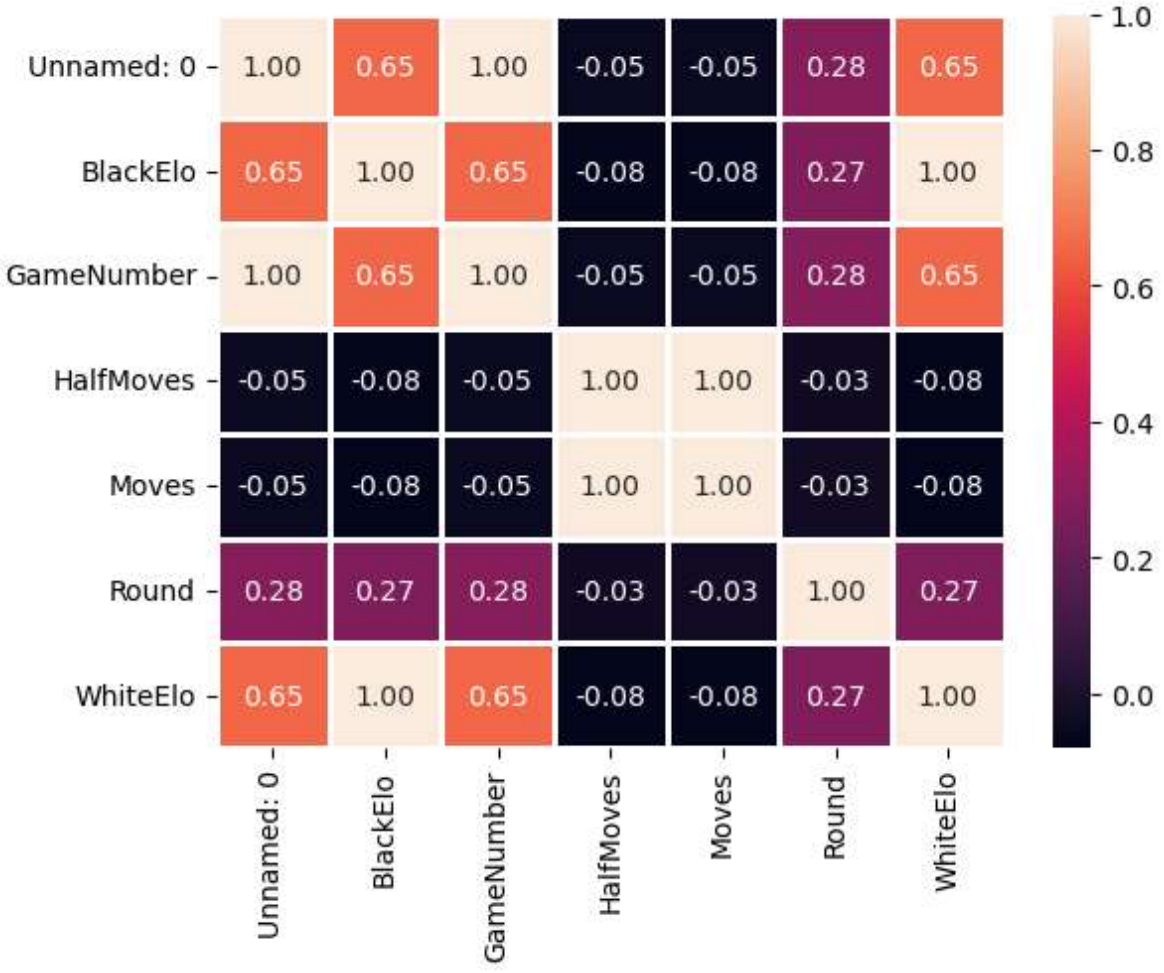
In [38]:
```
sns.displot(df)
plt.show()
```



In [39]:
```
# 7.Pair Plot
sns.pairplot(df1)
plt.show()
```



In [40]:
```
# 8.Heatmap
sns.heatmap(df1.corr(), annot=True, fmt='.2f', linewidths=2)
plt.show()
```



In [41]:
```
# 5) [Frame a problem statement based on the attributes of the dataset and pre
```

In [42]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC,SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split,cross_validate
from sklearn.preprocessing import MinMaxScaler,StandardScaler,LabelEncoder
from sklearn.metrics import accuracy_score
```

In [44]:
```
label_quality = LabelEncoder()
df['HalfMoves'] = label_quality.fit_transform(df['HalfMoves'])
```

In [45]:
```
df.head(10)
```

Out[45]:

	Unnamed: 0	Black	BlackElo	Date	ECO	Event	EventDate	GameNumber	HalfMov
0	0	Steinitz Wilhelm	0	1886-01-11	D11	World Championship 1st	1886.01.11	1	
1	1	Zukertort Johannes H	0	1886-01-13	C47	World Championship 1st	1886.01.11	2	
2	2	Steinitz Wilhelm	0	1886-01-15	D10	World Championship 1st	1886.01.11	3	
3	3	Zukertort Johannes H	0	1886-01-18	C67	World Championship 1st	1886.01.11	4	
4	4	Steinitz Wilhelm	0	1886-01-20	D10	World Championship 1st	1886.01.11	5	
5	5	Zukertort Johannes H	0	1886-02-03	C67	World Championship 1st	1886.01.11	6	
6	6	Steinitz Wilhelm	0	1886-02-05	D40	World Championship 1st	1886.01.11	7	
7	7	Zukertort Johannes H	0	1886-02-08	C67	World Championship 1st	1886.01.11	8	
8	8	Steinitz Wilhelm	0	1886-02-10	D26	World Championship 1st	1886.01.11	9	
9	9	Zukertort Johannes H	0	1886-02-26	C67	World Championship 1st	1886.01.11	10	

In [46]:
```
x_train,x_test,y_train,y_test=train_test_split(df1.drop(['EventDate','GameNumb

models=[LogisticRegression(),
        LinearSVC(),
        SVC(kernel='rbf'),
        KNeighborsClassifier(),
        RandomForestClassifier(),
        DecisionTreeClassifier(),
        GradientBoostingClassifier(),
        GaussianNB()]
```

In [ ]:

Lab 4: MM:10

1. Create a nd array of size 3 x 3 x 3 and perform subtraction operation as per x axis, y axis and z axis.

2. Demonstrate the following on the above created nd array:

1. basic indexing

2. slicing

3. boolean indexing

4. fancy indexing

3. Demonstrate the use of the following universal functions:

1. Trigonometric functions (atleast 2)

2. Statistical Functions (atleast 2)

3. Bit-twiddling Functions (atleast 2)

4. Download the dataset of your choice and demonstrate the following operations on it:

1. Read the dataset in the form of array

2. Perform any 5 nd array operations on the above dataset (for ex. reshaping, ravel, transpose, vstack, concatenate etc.)

5. Perform file input and output operations on nd array using save, load, savetxt and loadtxt methods.

1. Create a nd array of size 3 x 3 x 3 and perform subtraction operation as per x axis, y axis and z axis.

In [1]:

```
import numpy as np

# create a 3x3x3 numpy array
arr = np.array([[[1, 5, 6], [5, 7, 9], [7, 8, 9]],
                [[15, 22, 16], [14, 15, 18], [6, 7, 8]],
                [[13,80, 27], [21, 32, 22], [29, 21, 41]]])

# subtract along the x axis
x_subtracted = np.diff(arr, axis=0)

# subtract along the y axis
y_subtracted = np.diff(arr, axis=1)

# subtract along the z axis
z_subtracted = np.diff(arr, axis=2)inal array:")
print(arr)/

print("\nSubtracted along x axis:")
print(x_subtracted)

print("\nSubtracted along y axis:")
print(y_subtracted)

print("\nSubtracted along z axis:")
print(z_subtracted)
```

```
Original array:
[[[ 1  5  6]
  [ 5  7  9]
  [ 7  8  9]]

 [[15 22 16]
  [14 15 18]
  [ 6  7  8]]

 [[13 80 27]
  [21 32 22]
  [29 21 41]]]

Subtracted along x axis:
[[[14 17 10]
  [ 9  8  9]
  [-1 -1 -1]]

 [[-2 58 11]
  [ 7 17  4]
  [23 14 33]]]

Subtracted along y axis:
[[[  4   2   3]
  [  2   1   0]]

 [[ -1  -7   2]
  [ -8  -8 -10]]

 [[  8 -48  -5]
  [  8 -11  19]]]

Subtracted along z axis:
[[[  4   1]
  [  2   2]
  [  1   1]]

 [[  7  -6]
  [  1   3]
  [  1   1]]

 [[ 67 -53]
  [ 11 -10]
  [ -8  20]]]
```

2. Demonstrate the following on the above created nd array:¶

```
###1. basic indexing
```

In [2]:
```
# access a single element
print(arr[1, 2, 1])

# access an entire row
print(arr[0, 1, :])

# access a column
print(arr[2, :, 1])
```

```
7
[5 7 9]
[80 32 21]
```

2. slicing

In [3]:
```
# get a sub-array
print(arr[1:, 1:, 1:])

# slice along x-axis
print(arr[1, :, :])

# slice along y-axis
print(arr[:, 1, :])
```

```
[[[15 18]
  [ 7  8]]

 [[32 22]
  [21 41]]]
[[15 22 16]
 [14 15 18]
 [ 6  7  8]]
[[ 5  7  9]
 [14 15 18]
 [21 32 22]]
```

###3. boolean indexing

In [4]:
```
# create a boolean mask
mask = arr > 10

# use the mask to get elements that satisfy the condition
print(arr[mask])

# assign a new value to the elements that satisfy the condition
arr[mask] = 0
print(arr)
```

```
[15 22 16 14 15 18 13 80 27 21 32 22 29 21 41]
[[[1 5 6]
  [5 7 9]
  [7 8 9]]

 [[0 0 0]
  [0 0 0]
  [6 7 8]]

 [[0 0 0]
  [0 0 0]
  [0 0 0]]]
```

In [5]:
```
# select specific elements using fancy indexing
indices = [[0, 1], [2, 0], [1, 1]]
print(arr[indices])

# assign a new value to the elements selected using fancy indexing
arr[indices] = 100
print(arr)
```

```
[8 0]
[[[  1   5   6]
  [  5   7   9]
  [  7 100   9]]

 [[  0 100   0]
  [  0   0   0]
  [  6   7   8]]

 [[  0   0   0]
  [  0   0   0]
  [  0   0   0]]]

C:\Users\prana\AppData\Local\Temp\ipykernel_36204\3422740146.py:3: FutureWarn
ing: Using a non-tuple sequence for multidimensional indexing is deprecated;
use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be inter
preted as an array index, `arr[np.array(seq)]`, which will result either in a
n error or a different result.
  print(arr[indices])
C:\Users\prana\AppData\Local\Temp\ipykernel_36204\3422740146.py:6: FutureWarn
ing: Using a non-tuple sequence for multidimensional indexing is deprecated;
use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be inter
preted as an array index, `arr[np.array(seq)]`, which will result either in a
n error or a different result.
  arr[indices] = 100
```

3. Demonstrate the use of the following universal functions:

1. Trigonometric functions (atleast 2)

In [6]:
```
# Python code to demonstrate trigonometric function
import numpy as np

# create an array of angles
angles = np.array([0, 30, 45, 60, 90, 180])

# conversion of degree into radians
# using deg2rad function
radians = np.deg2rad(angles)

# sine of angles
print('Sine of angles in the array:')
sine_value = np.sin(radians)
print(np.sin(radians))

# inverse sine of sine values
print('Inverse Sine of sine values:')
print(np.rad2deg(np.arcsin(sine_value)))

# hyperbolic sine of angles
print('Sine hyperbolic of angles in the array:')
sineh_value = np.sinh(radians)
print(np.sinh(radians))
```

```
Sine of angles in the array:
[0.00000000e+00 5.00000000e-01 7.07106781e-01 8.66025404e-01
 1.00000000e+00 1.22464680e-16]
Inverse Sine of sine values:
[0.0000000e+00 3.0000000e+01 4.5000000e+01 6.0000000e+01 9.0000000e+01
 7.0167093e-15]
Sine hyperbolic of angles in the array:
[ 0.          0.54785347  0.86867096  1.24936705  2.3012989  11.54873936]
```

2. Statistical Functions (atleast 2

In [10]:
```python
# Python code demonstrate statistical function
import numpy as np

# construct a weight array
weight = np.array([50, 82.5, 100, 81, 55, 73, 51, 49])

# range of weight i.e. max weight-min weight
print('Range of the weight of the students: ')
print(np.ptp(weight))

# percentile
print('Weight below which 70 % student fall: ')
print(np.percentile(weight, 70))

# mean
print('Mean weight of the students: ')
print(np.mean(weight))

# median
print('Median weight of the students: ')
print(np.median(weight))
```

```
Range of the weight of the students:
51.0
Weight below which 70 % student fall:
80.19999999999999
Mean weight of the students:
67.6875
Median weight of the students:
64.0
```

3. Bit-twiddling Functions (atleast 2)

In [9]:
```
# Python code to demonstrate bitwise-function
import numpy as np

# construct an array of even and odd numbers
even = np.array([1, 2, 3, 4, 5, 6, 7])
odd = np.array([0, 2, 4, 8, 10, 18, 21])

# invert or not
print('inversion of even no. array: ')
print(np.invert(even))

# left_shift
print('left_shift of even no. array: ')
print(np.left_shift(even, 1))

# right_shift
print('right_shift of even no. array: ')
print(np.right_shift(even, 1))
```

```
inversion of even no. array:
[-2 -3 -4 -5 -6 -7 -8]
left_shift of even no. array:
[ 2  4  6  8 10 12 14]
right_shift of even no. array:
[0 1 1 2 2 3 3]
```

4. Download the dataset of your choice and demonstrate the following operations on it:

1. Read the dataset in the form of array

In [12]:
```
import csv
import numpy as np

data = np.genfromtxt(r'C:\Users\prana\Downloads\DEM_lab\color code1.csv', deli
```

2. Perform any 5 nd array operations on the above dataset (for ex. reshaping, ravel, transpose, vstack, concatenate etc.)

In [19]:
```
reshaped_data = data.reshape((8, 1))

reshaped_data
```

Out[19]:
```
array([[nan],
       [nan],
       [nan],
       [12.],
       [nan],
       [13.],
       [nan],
       [14.]])
```

In [20]:
```
flattened_data = data.ravel()
flattened_data
```

Out[20]:
```
array([nan, nan, nan, 12., nan, 13., nan, 14.])
```

In [21]:
```
transposed_data = data.transpose()
transposed_data
```

Out[21]:
```
array([[nan, nan, nan, nan],
       [nan, 12., 13., 14.]])
```

In [22]:
```
stacked_data = np.vstack((data, data))
stacked_data
```

Out[22]:
```
array([[nan, nan],
       [nan, 12.],
       [nan, 13.],
       [nan, 14.],
       [nan, nan],
       [nan, 12.],
       [nan, 13.],
       [nan, 14.]])
```

In [23]:
```
concatenated_data = np.concatenate((data, data), axis=1)
concatenated_data
```

Out[23]:
```
array([[nan, nan, nan, nan],
       [nan, 12., nan, 12.],
       [nan, 13., nan, 13.],
       [nan, 14., nan, 14.]])
```

In [ ]:

lab-5

1

In [1]:
```
import requests
import re
from bs4 import BeautifulSoup
from urllib.request import urlopen
import csv
```

In [5]:
```
url = "https://en.wikipedia.org/wiki/R_(programming_language)"
page = urlopen(url)

html_bytes = page.read()
html = html_bytes.decode("utf-8")
title_index = html.find("<title>")
start_index = title_index + len("<title>")
end_index = html.find("</title>")
title = html[start_index:end_index]
print(title)
```

```
R (programming language) - Wikipedia
```

2

In [6]:
```
url = "https://en.wikipedia.org/wiki/R_(programming_language)"
response = requests.get(url)


pattern = re.compile('[A-Z]+')
matches = pattern.findall(response.text)


for match in matches:
    print(match)
```

```
DOCTYPE
UTF
R
W
E
N
N
E
N
E
N
N
N
RLCONF
B
F
S
T
T
```

3

In [7]:
```
url = "https://en.wikipedia.org/wiki/R_(programming_language)"
response = requests.get(url)
html_content = response.text


soup = BeautifulSoup(html_content, 'html.parser')
p_tags = soup.find_all('p')


for p_tag in p_tags:
    print(p_tag.text)
```

```
R is a programming language for statistical computing and graphics support
ed by the R Core Team and the R Foundation for Statistical Computing. Crea
ted by statisticians Ross Ihaka and Robert Gentleman, R is used among data
miners, bioinformaticians and statisticians for data analysis and developi
ng statistical software.[7] Users have created packages to augment the fun
ctions of the R language.

According to user surveys and studies of scholarly literature databases, R
is one of the most commonly used programming languages in data mining.[8]
As of April 2023,[update] R ranks 16th in the TIOBE index, a measure of pr
ogramming language popularity, in which the language peaked in 8th place i
n August 2020.[9][10]

The official R software environment is an open-source free software enviro
nment within the GNU package, available under the GNU General Public Licen
se. It is written primarily in C, Fortran, and R itself (partially self-ho
sting). Precompiled executables are provided for various operating system
```

4

In [8]:
```
url = "https://en.wikipedia.org/wiki/R_(programming_language)"
response = requests.get(url)
html_content = response.text

soup = BeautifulSoup(html_content, 'html.parser')
p_tags = soup.find_all('p')

with open("show.txt", 'w', encoding='utf-8') as file:
    for p_tag in p_tags:
        file.write(p_tag.text + '\n')
```

5

In [9]:
```
url = "https://en.wikipedia.org/wiki/R_(programming_language)"
response = requests.get(url)
html_content = response.text


soup = BeautifulSoup(html_content, 'html.parser')
p_tags = soup.find_all('p')


with open("output.csv", 'w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    for p_tag in p_tags:
        writer.writerow([p_tag.text])
```

In [ ]:

lab-6

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Create random data
np.random.seed(60)  # for reproducibility
x = np.random.rand(50)
y = x * 2 + np.random.rand(50)

# Create dataframe
df = pd.DataFrame({'x': x, 'y': y})

# Fit linear regression model
model = LinearRegression()
model.fit(df[['x']], df['y'])

# Calculate predictions
y_pred = model.predict(df[['x']])

# Plot data and regression line
plt.scatter(df['x'], df['y'])
plt.plot(df['x'], y_pred, color='yellow')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Regression Example')
plt.show()
```