Q1. WAP to print all the integers that are not divisible by either 2 or 3 and lie between 1 and 50.

```
for i in range(1, 51):
    if i % 2 != 0 and i % 3 != 0:
        print(i)
```

Output

```
11
13
17
19
23
25
29
31
35
37
41
43
47
49

Process finished with exit code 0
```

Q2. Write a Python program to check the validity of a password given by the user. The password should satisfy the following criteria: 1. Contains at least one letter between a and z 2. Contains at least one number between 1 and 9 3. Contains at least one letter between A and Z 4. Contains at least one special character from @, $, *

```
def is_valid_pass(password):
    has_lower = False
    has_upper = False
    has_digit = False
    has_special = False
    special_chars = ['@', '$', '*']

    for char in password:
        if char.islower():
            has_lower = True
        elif char.isupper():
            has_upper = True
        elif char.isdigit():
            has_digit = True
        elif char in special_chars:
            has_special = True

    return has_lower and has_upper and has_digit and has_special


password = input("Enter a password: ")

if is_valid_pass(password):
    print("The password is valid.")
```

```
else:
    print("The password is not valid.")
```

Result

```
Enter a password: iamsharukhkahn@
The password is not valid.

Process finished with exit code 0
```

Q3. Write a python program to print factorial of a number. Take input from the user.

```
def f(n):
    if n <= 1:
        return 1
    return f(n - 1) * n


x = int(input('Please enter a value: '))
print(f(x))
```

Result

```
Please enter a value: 3
6

Process finished with exit code 0
```

Q4. Write a Python program that accepts a list of numbers. Count the negative numbers and compute the sum of the positive numbers of the said list. Return these values through a list. (For example: if input is [1, 2, 3, -4, -5] then output should be: Number of negative of numbers and sum of the positive numbers of the said list: [2, 6])

```
def process(n):
    negative_count = 0
    positive_sum = 0

    for number in n:
        if number < 0:
            negative_count += 1
        else:
            positive_sum += number

    return [negative_count, positive_sum]
```

```
input_numbers = [1, 2, 3, -4, -5,-6.-7,-8,8,8,7]
result = process(input_numbers)

print(f"Number of negative numbers and sum of the positive numbers of the said list respectively : {result}")
```

Output

```
Number of negative numbers and sum of the positive numbers of the said list respectively : [4, 29]

Process finished with exit code 0
```

Q5. Write the python statement for the following question on the basis of given dataset:

Name	Degree	Score
RAM	BCA	90
EKLAVAY	MCA	60
HERRY	BCA	NaN
MARIYAM	BTECH	70
LINDA	MCA	80
BILL	BCA	75

a) To create the above DataFrame.

b) To print the Degree and maximum marks in each stream.

c) To fill the NaN with 76.

d) To count the number of students in MCA.

e) To print the MEAN marks for BCA

```
            import pandas as pd

data = {'Name': ['Ram', 'Eklavya', 'Herry', 'Mariyam', 'Linda', 'Bill'],
       'Degree': ['BCA', 'MCA', 'BCA', 'BTECH', 'MCA', 'BCA'],
       'Score': [90, 60, None, 70, 80, 75]}
df = pd.DataFrame(data)

print(df.groupby('Degree').max())
print()

df.fillna(76, inplace=True)
print(df)
print()

print('No. of students in MCA: ' + str(df.groupby('Degree').count().loc['MCA', 'Name']))

print('Mean marks in BCA: ' + str(df.groupby('Degree')['Score'].mean()['BCA']))
```

Result

```
            Name  Score
Degree
BCA          Ram   90.0
BTECH   Mariyam   70.0
MCA       Linda   80.0

       Name Degree  Score
0       Ram    BCA   90.0
1  Eklavya    MCA   60.0
2    Herry    BCA   76.0
3  Mariyam  BTECH   70.0
4    Linda    MCA   80.0
5     Bill    BCA   75.0

No. of students in MCA: 2
Mean marks in BCA: 80.33333333333333
```

Q6. Following is a dataset provided by a car manufacturing company:

Sales	Price	Advertising	Model	Fuel
Yes	Expensive	High	Sedan	Electric
No	Affordable	Low	Sedan	CNG
No	Expensive	High	SUV	Petrol
Yes	Affordable	High	Hatchback	CNG
No	Affordable	Low	SUV	Petrol
Yes	Inexpensive	High	Hatchback	Petrol
No	Expensive	High	SUV	Electric
Yes	Inexpensive	High	Hatchback	Petrol
Yes	Inexpensive	Low	Hatchback	Electric

Fit an appropriate model on the given data to predict whether the sales would be or not. Also, check the accuracy score of the model. Additionally, the company is planning to launch a new electric SUV car, which will be expensive and hence company is planning to do high advertising for the same. Predict what will be the situation of sales for the same.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

data = {'Sales': ['Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes'],
        'Price': ['Expensive', 'Affordable', 'Expensive', 'Affordable', 'Affordable', 'Inexpensive', 'Expensive',
                  'Inexpensive', 'Inexpensive'],
        'Advertising': ['High', 'Low', 'High', 'High', 'Low', 'High', 'High', 'High', 'Low'],
        'Model': ['Sedan', 'Sedan', 'SUV', 'Hatchback', 'SUV', 'Hatchback', 'SUV', 'Hatchback', 'Hatchback'],
        'Fuel': ['Electric', 'CNG', 'Petrol', 'CNG', 'Petrol', 'Petrol', 'Electric', 'Petrol', 'Electric']}
df = pd.DataFrame(data)
```

```
df['Price'] = df['Price'].replace({'Inexpensive': 0, 'Affordable': 1, 'Expensive': 2})
df['Advertising'] = df['Advertising'].replace({'Low': 0, 'High': 1})
df['Model'] = df['Model'].replace({'Hatchback': 0, 'Sedan': 1, 'SUV': 2})
df['Fuel'] = df['Fuel'].replace({'Petrol': 0, 'CNG': 1, 'Electric': 2})

X = df[['Price', 'Advertising', 'Model', 'Fuel']]
y = df['Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict on the testing data
y_pred = model.predict(X_test)

# Calculate accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Predicting for our own values
prediction = model.predict(pd.DataFrame([[2, 1, 2, 2]], columns=['Price', 'Advertising', 'Model', 'Fuel']))

print('Will sales happen? ', prediction[0])
```

Result

```
Accuracy: 0.5
Will sales happen?  No


Process finished with exit code 0
```

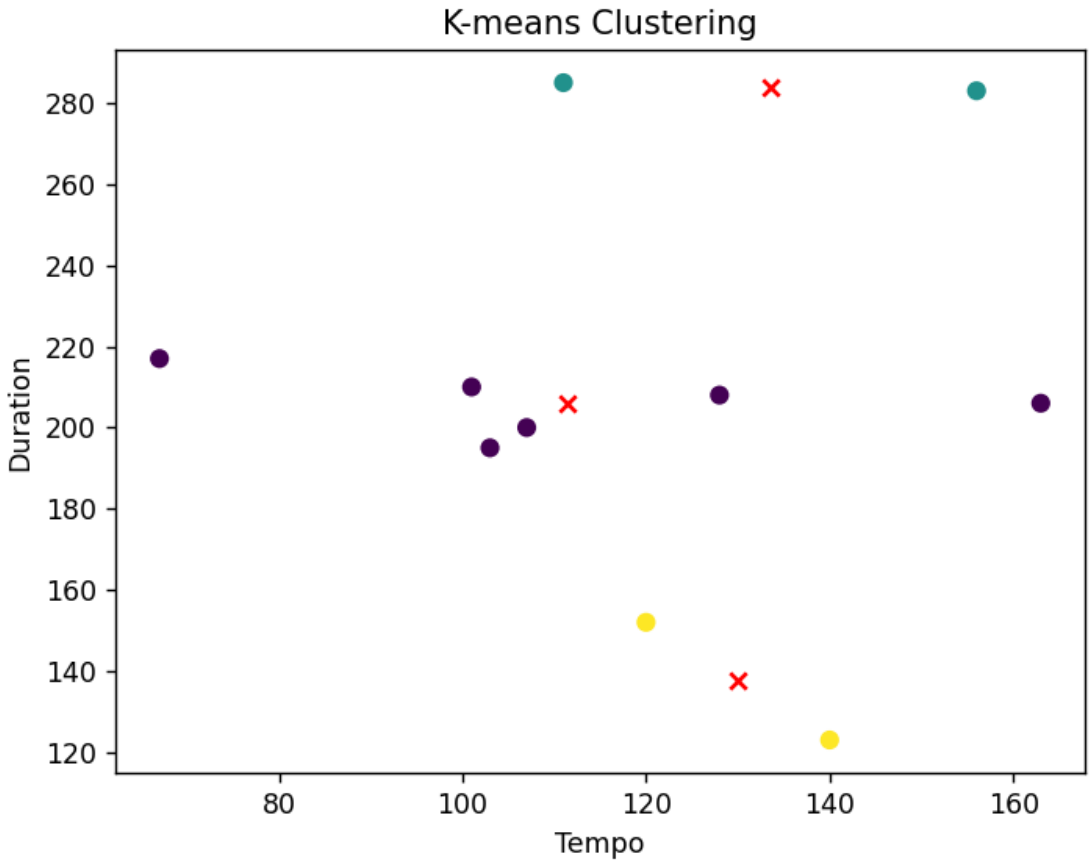
Q7. The following data set is for 10 random songs. Divide the data into 3 clusters and find the accuracy of the model. (Tempo: Beats per minute of a song)

Tempo	Duration
101	210
111	285
67	217
120	152
107	200
140	123
156	283
163	206
103	195
128	208

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import warnings
from sklearn.metrics import silhouette_score

data = {'Tempo': [101, 111, 67, 120, 107, 140, 156, 163, 103, 128],
        'Duration': [210, 285, 217, 152, 200, 123, 283, 206, 195, 208]}

df = pd.DataFrame(data)

# Suppress the FutureWarning for n_init
with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
    kmeans.fit(df.values)

labels = kmeans.labels_
centroids = kmeans.cluster_centers_

df = df.values
```

```
plt.scatter(df[:, 0], df[:, 1], c=labels, cmap='viridis')
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x')
plt.xlabel('Tempo')
plt.ylabel('Duration')
plt.title('K-means Clustering')
plt.show()

# Accuracy of the model
# Silhouette Score - ranges from -1 to 1 - higher the better
sil_score = silhouette_score(df, labels)
print('Silhouette Score:', sil_score)
```

Result




```
Silhouette Score: 0.4861256837417036

Process finished with exit code 0
```

Q8. Following is the data set of a company that produces automobile.

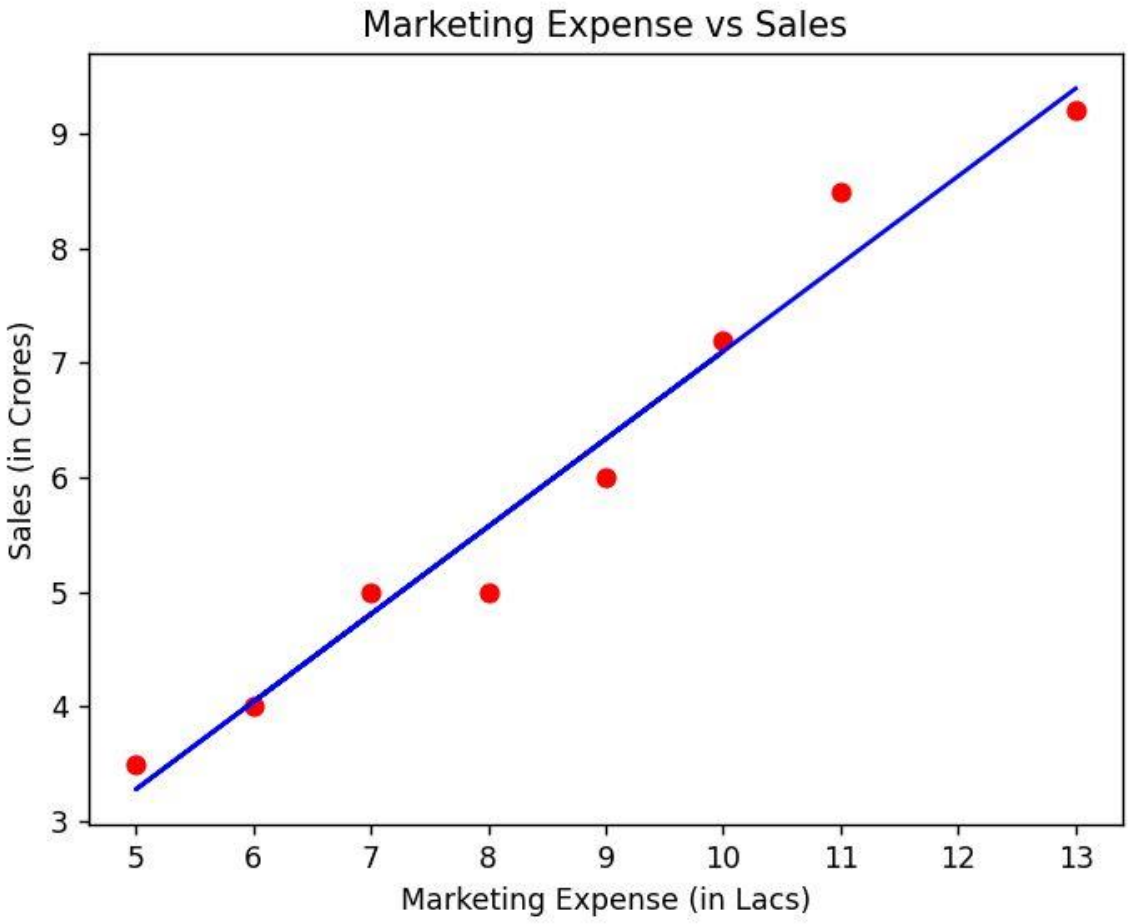
Year	Marketing Expense in lacs	Sales in the Crores
2011	5	3.5
2012	6	4
2013	4	3
2014	8	5
2015	13	9.2
2016	7	5
2017	9	6
2018	10	7.2
2019	9.5	6.9
2020	11	8.5

As a marketing head of this company build a suitable regression model using the python libraries to predict the sales of company in future years. Also, check the accuracy score of the model.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

data = {'Year': np.arange(2011, 2021),
        'Marketing_Expense_in_Lacs': [5, 6, 4, 8, 13, 7, 9, 10, 9.5, 11],
        'Sales_in_Crores': [3.5, 4, 3, 5, 9.2, 5, 6, 7.2, 6.9, 8.5]}

df = pd.DataFrame(data)

X = df[['Marketing_Expense_in_Lacs']]
y = df[['Sales_in_Crores']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
lr = LinearRegression()
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, lr.predict(X_train), color='blue')
plt.title('Marketing Expense vs Sales')
plt.xlabel('Marketing Expense (in Lacs)')
plt.ylabel('Sales (in Crores)')
plt.show()

# R-squared score of model
print('R-squared score of this model: ', str(r2_score(y_test, y_pred)))

# Predicting Sales for year 2021 assuming Marketing Expense of 15 Lacs
sales_in_2021 = lr.predict([[15]])
print('Sales in 2021 (given Marketing Expense is 15 Lacs): ', str(sales_in_2021))
```

Result




```
R-squared score of this model:  0.9644282395898277
Sales in 2021 (given Marketing Expense is 15 Lacs):  [[10.92631579]]
```

Q9. Following is the dataset given by a school of a random sample of 10 students.

IS_PCM	Aggregate Marks	Science Marks	Maths Marks	Gender
1	98	97	98	1
1	97.8	99	95	1
1	95.6	94	98	0
0	89	87	94	1
0	85	86	84	1
0	98.9	97	99	0
0	99	99	100	0
1	95	90	95	1
0	87.8	94	80	1
1	94	90	98	1

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

data = {'IS_PCM': [1, 1, 1, 0, 0, 0, 0, 1, 0, 1],
        'Aggregate_Marks': [98, 97.8, 95.6, 89, 85, 98.9, 99, 95, 87.8, 94],
        'Science_Marks': [97, 99, 94, 87, 86, 97, 99, 90, 94, 90],
        'Maths_Marks': [98, 95, 98, 94, 84, 99, 100, 95, 80, 98],
        'Gender': [1, 1, 0, 1, 1, 0, 0, 1, 1, 1]}
df = pd.DataFrame(data)

X = df[['Aggregate_Marks', 'Science_Marks', 'Maths_Marks', 'Gender']]
y = df['Gender']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()
```

```
# Fit the model on the training data
model.fit(X_train, y_train)

# Predict on the testing data
y_pred = model.predict(X_test)

# Calculate accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Prediction for Harsh
harsh = model.predict(
    pd.DataFrame([[90, 95, 80, 1]], columns=['Aggregate_Marks', 'Science_Marks', 'Maths_Marks', 'Gender']))
harsh_pcm = 'Yes' if harsh[0] == 1 else 'No'
print('Is Harsh a PCM Student?: ', harsh_pcm)
```

Result

```
Accuracy: 1.0
Is Harsh a PCM Student?:  Yes

Process finished with exit code 0
```

Q10. Following is the dataset regarding the survival of passengers in a train accident

Class	Age	Gender	Survived
1st	29	female	1
1st	30	male	0
1st	47	male	1
2nd	32	female	1
2nd	57	male	0
2nd	18	male	0
2nd	36	female	1
3rd	25	male	0
3rd	18	female	0
3rd	38	female	1

Fit an appropriate Machine learning model to predict whether the person has survived or not. Also check the accuracy of the model by using the following dataset

Class	Age	Gender	Survived
1st	13	male	0
2nd	33	female	1
3rd	26	male	0

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
data = {'Class': [1, 1, 1, 2, 2, 2, 2, 3, 3, 3],
        'Age': [29, 30, 47, 32, 57, 18, 36, 25, 18, 38],
        'Gender': ['female', 'male', 'male', 'female', 'male', 'male', 'female', 'male', 'female', 'female'],
        'Survived': [1, 0, 1, 1, 0, 0, 1, 0, 0, 1]}
df = pd.DataFrame(data)
df['Gender'] = df['Gender'].replace({'male': 1, 'female': 0})

X = df[['Age', 'Class', 'Gender']]
y = df['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

pred_data = {
    'Class': [1, 2, 3],
    'Age': [13, 33, 26],
    'Gender': ['male', 'female', 'male'],
    'Survived': [0, 1, 0]
}
pred_df = pd.DataFrame(pred_data)
pred_df['Gender'] = pred_df['Gender'].replace({'male': 1, 'female': 0})

# Prediction
y_pred = model.predict(pred_df[['Age', 'Class', 'Gender']])

# Calculate accuracy of the model
accuracy = accuracy_score(pred_df['Survived'], y_pred)
print('Accuracy:', accuracy)
```

Result