Exno :1

```
Student = {
    "Roll_no":[],
    "name":[],
    "department":[],
    "Percentage":[]
}
defaultDepartment = ["Computer","IT"]

# This Function Get The Input From The User
def getStudentDetails(how):
    for i in range(how):
        # Roll no Getting
        print("\nStudent no: ",i+1)
        roll_no = input("Enter The Roll Number Of The Student: ")
        while not roll_no.isdigit():
            roll_no = input("Enter The Valid Roll Number Of The
Student: ")
        Student["Roll_no"].append(roll_no)
        # Name Getting
        name = input("Enter The Name Of The Student: ")
        while not name.isalpha():
            name = input("Enter The Correct Name Of The
Student: ")
        Student["name"].append(name)
        # Department Getting
        dept =  input("Enter The Department Of The Student: ")
        while not dept in defaultDepartment:
            dept = input("Enter The Correct Department Of The
Student (Computer/IT): ")
        Student["department"].append(dept)
        # Percentage Getting
        percentage = input("Enter The Last Exam Mark
Percentage Of The Student: ")
```

```
        validatePercentage(percentage)

    # Function For Validate The Percentage
def validatePercentage(percentage):
    if percentage.isdigit():
        percentage = int(percentage)
        if percentage < 0 or percentage > 100:
            percentage = input("Enter The Last Exam Mark 
Percentage Correctly : ")
            validatePercentage(percentage)
            Student["Percentage"].append(percentage)
    else:
        percentage = input("Enter The Last Exam Mark 
Percentage Correctly: ")
        validatePercentage(percentage)

# Function For Print The Student Details
def printStudentDetails():
    print("            2025 Model Exam Mark List")
    print("+-----------------------------------+")
    print("Roll No\t\tStudent 
Name\t\tDepartment\t\tPercentage")
    for i in range(0,len(Student['Roll_no'])):
        print(
            f"{Student['Roll_no'][i]}\t\t"
            f"{Student['name'][i]}\t\t"
            f"{Student['department'][i]}\t\t"
            f"{Student['Percentage'][i]}%"
        )
    print("+--------------------------------------------+")

# This Function Handel The Flow Of Program
def Main():
    how = int(input("Enter the How Many Student Record Want To
Enter: "))
    getStudentDetails(how)
    printStudentDetails()

if __name__ == "__main__":
    Main()
```

Exno:2

```
import numpy as np
StudentMarkList = [65,45,85,95,55,64,77,85,94,35]
StudentMarkTuple = (25,35,48,65,75,48,95,65,48,45)

print("Student Mark List:", StudentMarkList )
print("Student Mark Tuple", StudentMarkTuple)

arrayList = np.array(StudentMarkList )
arrayTuple = np.array(StudentMarkTuple)

print("\nStudent Mark List Array:", arrayList )
print("Student Mark Tuple Array :", arrayTuple)
part_size = (len(arrayList)) // 3

slice1 = arrayList[:part_size]
slice2 = arrayList[part_size:2 * part_size]
slice3 = arrayList[2 * part_size:]

print("\nSlice 1:", slice1)
print("Slice 2:", slice2)
print("Slice 3:", slice3)
```

Exno: 3

```
import numpy as np

data  = np.genfromtxt('StudentDataCsv.csv', delimiter =',',
skip_header = 1)
print(
        f"Length: {len(data)}\n"
        f"Dimension: {data.ndim}\n"
        f"Size: {data.size}\n"
        f"Shape: {data.shape}\n"
        f"Type: {data.dtype}\n"
)
```

Exno:4

```
import pandas as pd
import numpy as np

studentDataFrame = pd.read_csv("StudentDataCsv.csv")

print("The Given Data Frame is: \n",studentDataFrame)

print(f"\n\nThe Minimum Value in The Student List
Is:\n{studentDataFrame.min()}")

print(f"\n\nThe Maximum Value in The Student List
Is:\n{studentDataFrame.max()}")

print(f"\n\nThe CumSum Value of The Student List
Is:\n{studentDataFrame.cumsum()}")

print(f"\n\nThe Average of The Subject 1 List Is:
{studentDataFrame['Subject1'].mean()}")

print(f"\nThe Median of The Subject 1 Is:
{studentDataFrame['Subject1'].median()}")
```

```
print(f"\nThe Standard Division of The Subject 1 Is:
{studentDataFrame['Subject1'].std()}")

print(f"\nThe Co-Efficient of The Student List Is:
\n{np.corrcoef(studentDataFrame['Subject1'],studentDataFrame['
Subject2'])}")
```

Exno:5

```
import pandas as pd
dataFrame =  pd.read_csv("Book1.csv")

print("The orginial data is: \n",dataFrame)
print("The missing data count are: ",
dataFrame.isnull().sum())

dataFrame["mark1"] = dataFrame["mark1"].fillna(0)
dataFrame["mark2"] = dataFrame["mark2"].fillna("Absent")

print("After the data process:\n",dataFrame)
```

Exno:6

```
import pandas as pd
FirstBookDataFrame = pd.read_csv("FirstBook.csv")
SecondBookDataFrame = pd.read_csv("SecondBook.csv")

compained =
pd.concat([FirstBookDataFrame,SecondBookDataFrame]).reset_inde
x(drop = True)
print("The compained Data is: \n",compained)

compained = compained.drop_duplicates(

print("After Removing The Duplicate value: \n",compained)
```

Exno:7

```
import seaborn as sns
import matplotlib.pyplot as plt

dataSet = sns.load_dataset('iris')
setosa = dataSet[dataSet['species'] == 'setosa']
AverageValue = setosa.mean(numeric_only=True)
AverageValue.plot(kind = 'bar',color = 'lightblue',edgecolor=
'black',linewidth = 2)

plt.title("Average Feature Values")
plt.xlabel("Feature Values")
plt.ylabel("Average Values")
plt.show()
```

Exno:8

```
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns. load_dataset('iris')
pair_chart = sns.pairplot(iris, hue="species", palette='husl')
pair_chart.savefig( 'irisPairChart.jpg', format="jpg", dpi=500)
plt.show()
```

Exno:9

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn. tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt

iris = sns. load_dataset("iris")
x = iris.drop( 'species', axis=1)
```

```
y = iris['species' ]

x_train, x_test, y_train, y_test = train_test_split(  x, y,
test_size=0.3, random_state=42)

model = DecisionTreeClassifier(random_state=42)
model. fit(x_train, y_train)
y_pred = model.predict(x_test)

print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy: ", accuracy_score(y_test, y_pred))

plt.figure(figsize=(12,8))
plot_tree(model,filled = True, feature_names= x.columns,
class_names=model.classes_)
plt.title("Decision Tree -Iris Dataset")
plt.show()
```

Exno:10

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans



iris = sns.load_dataset("iris")

x = iris.drop("species",axis=1)

k_mean = KMeans(n_clusters = 3,random_state = 42)
iris['cluster'] = k_mean.fit_predict(x)

print("Cluster Center:\n",k_mean.cluster_centers_)
plt.figure(figsize=(12,8))
plt.scatter(
    x.iloc[:,0],
    x.iloc[:,1],
    c = iris['cluster'],
    cmap = 'viridis',
    s = 50
)
plt.title("K-Mean Clustering on iris dataset")
plt.xlabel("sepal length(cm)")
plt.ylabel("sepal width(cm)")

print(iris.groupby(["cluster","species"]).size())

plt.show()
```