# **Practical – 1**

**Aim:** Study of Python Basic Libraries like a Numpy. Perform the following task using Numpy library.

- Creating blank array, with predefined data, with pattern specific data

- Slicing and Updating elements,

- Shape manipulations

- Looping over arrays.

* **Code:**

import numpy as np

a = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])

print(a)

print(f"Slicing array: {a[5:]}")

print(f"Slicing array: {a[:5]}")

print(f"Slicing array: {a[3:8]}")

a[5] = 999

a[2] = 888

print(a)

b = a.reshape(2, 5)

print(f"reshaping into 2x5: \n{b}")

c = a.reshape(5, 2)

print(f"reshaping into 5x2: \n{c}")

print("\nLooping over Array:\n")

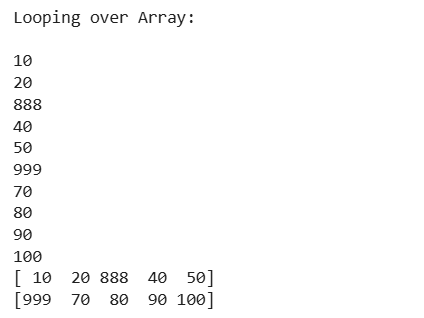
for i in a:

print(i)

for i in b:

print(i)

* A screenshot of a computer code

  AI-generated content may be incorrect.**Output:**

# **Practical – 2**

**Aim:** Study of Python Libraries for ML application such as Pandas and Matplotlib Perform the following task using Pandas & Matplotlib library.

* Creating data frame
* Reading files
* Slicing manipulations
* Exporting data to files
* Columns and row manipulations with loops
* Use pandas for masking data and reading in Boolean format.
* Importing matplotlib
* Simple line chart
* Correlation chart
* Histogram
* Plotting of Multivariate data
* Plot Pi Chart
* **Code:**

import kagglehub

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

path = kagglehub.dataset\_download("sidtwr/videogames-sales-dataset")

# print("Path to dataset files:", path)

df = pd.read\_csv("/kaggle/input/videogames-sales-dataset/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv")

# df.head()

temp = df.iloc[30:50]

# print(temp)

# temp.to\_csv('temp.csv', index=False)

# from google.colab import files

# files.download('temp.csv')

colManipulation = df.iloc[50:60:5]

# print(colManipulation)

sales\_by\_year = temp.groupby('Year\_of\_Release')['Global\_Sales'].sum().reset\_index()

sales\_by\_year['Year\_of\_Release'] = pd.to\_numeric(sales\_by\_year['Year\_of\_Release'], errors='coerce')

sales\_by\_year.dropna(subset=['Year\_of\_Release'], inplace=True)

sales\_by\_year.sort\_values('Year\_of\_Release', inplace=True)

#line chart

plt.figure(figsize=(12, 6))

plt.plot(sales\_by\_year['Year\_of\_Release'], sales\_by\_year['Global\_Sales'], marker='o', linestyle='-')

plt.title('Global Video Game Sales Over Time')

plt.xlabel('Year of Release')

plt.ylabel('Global Sales (Millions)')

plt.grid(True)

plt.show()

#corelation chart

numeric\_cols = temp.select\_dtypes(include=['float64', 'int64'])

correlation\_matrix = numeric\_cols.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix of Video Game Sales Data')

plt.show()

#histogram

plt.figure(figsize=(10, 6))

plt.hist(temp['User\_Score'], bins=50, color='lightblue', edgecolor='black')

plt.title('Distribution of User\_Score')

plt.xlabel('User\_Score')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

#multivariant data

cols\_for\_pairplot = ['NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Global\_Sales', 'Critic\_Score', 'User\_Score']

sns.pairplot(temp[cols\_for\_pairplot].dropna())

plt.suptitle('Pair Plot of Select Numeric Columns', y=1.02)

plt.show()

games\_per\_platform = temp['Platform'].value\_counts().reset\_index()

games\_per\_platform.columns = ['Platform', 'Number\_of\_Games']

top\_platforms = games\_per\_platform.head(10)

# pie chart

plt.figure(figsize=(10, 10))

plt.pie(top\_platforms['Number\_of\_Games'], labels=top\_platforms['Platform'], autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Game Releases by Platform (Top 10)')

plt.axis('equal')

plt.show()

* A graph with blue lines and numbers

  AI-generated content may be incorrect.**Output:**

A chart with numbers and a number of numbers

AI-generated content may be incorrect.

A graph with blue lines

AI-generated content may be incorrect.

A pie chart with numbers and text

AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.

# **Practical – 8**

**Aim:** Write a program to implement K-mean clustering in python.

* **Code:**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import numpy as np

df = pd.read\_csv("/kaggle/input/videogames-sales-dataset/Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv")

df = df[['Global\_Sales', 'Critic\_Score']].dropna()

df\_sample = df.sample(n=100, random\_state=42)

X = df\_sample[['Global\_Sales', 'Critic\_Score']]

# Loop through K = 1 to 5

for k in range(1, 6):

kmeans = KMeans(n\_clusters=k, init='k-means++', n\_init=10, random\_state=42)

kmeans.fit(X)

centroids = kmeans.cluster\_centers\_

labels = kmeans.labels\_

plt.figure(figsize=(6, 5))

plt.scatter(X['Global\_Sales'], X['Critic\_Score'], c=labels, cmap='viridis')

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.title(f"KMeans Clustering with K={k}")

plt.xlabel("Global Sales (millions)")

plt.ylabel("Critic Score")

plt.legend()

plt.grid(True)

plt.show()

* A graph with red and blue dots

  AI-generated content may be incorrect.A graph with red and yellow dots

  AI-generated content may be incorrect.**Output:**

A graph with colored dots

AI-generated content may be incorrect.A graph with colored dots

AI-generated content may be incorrect.A graph with colored dots

AI-generated content may be incorrect.