1. Numpy

NumPy is a foundational Python library for numerical computing. It offers support for multi-dimensional arrays and matrices, along with a suite of mathematical functions to operate on these data structures. Essential for data science and scientific computing tasks, NumPy provides efficient array operations and interfaces seamlessly with other libraries in the Python ecosystem. Its capabilities form the backbone of the data manipulation in Python.

importing the library

In [1]:
```
import numpy as np
```

Creating Arrays

In [2]:
```
# Create a simple array from a list
a = np.array([1, 2, 3, 4, 5])
print("Array a:", a)
```

```
Array a: [1 2 3 4 5]
```

In [3]:
```
# Create a 2D array (matrix) from a list of lists
b = np.array([[1, 2], [3, 4], [5, 6]])
print("\n2D Array b:\n", b)
```

```
2D Array b:
 [[1 2]
 [3 4]
 [5 6]]
```

In [4]:
```
# Generate an array of zeros
c = np.zeros((3, 3))
print("\nArray of zeros:\n", c)
```

```
Array of zeros:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Basic Array Operations

In [5]:
```
# You can perform element-wise operations on arrays,
# as well as use built-in methods for calculations.
```

In [6]:
```
# Element-wise addition
sum_array = a + np.array([5, 4, 3, 2, 1])
print("\nSum of arrays:", sum_array)
```

```
Sum of arrays: [6 6 6 6 6]
```

In [7]:
```
# Element-wise multiplication
product_array = a * 2
print("\nProduct of array with 2:", product_array)
```

```
Product of array with 2: [ 2  4  6  8 10]
```

In [8]:
```
# Calculate mean of the array
mean_a = np.mean(a)
print("\nMean of array a:", mean_a)
```

```
Mean of array a: 3.0
```

Reshaping and Slicing

In [9]:
```
# NumPy allows easy reshaping of arrays and extracting portions of arr
```

In [10]:
```
# Reshape 'a' to a 5x1 matrix
reshaped_a = a.reshape(5, 1)
print("\nReshaped array:\n", reshaped_a)
```

```
Reshaped array:
 [[1]
 [2]
 [3]
 [4]
 [5]]
```

In [11]:
```
# Slice 'b' to extract its first 2 rows
sliced_b = b[:2, :]
print("\nSliced array b:\n", sliced_b)
```

```
Sliced array b:
 [[1 2]
 [3 4]]
```

Mathematical Functions

In [12]:
```
# NumPy provides numerous mathematical functions to perform operations
```

In [13]:
```
# Calculate the square root of each element
sqrt_a = np.sqrt(a)
print("\nSquare root of array a:", sqrt_a)
```

```
Square root of array a: [1.         1.41421356 1.73205081 2.
2.23606798]
```

In [14]:
```
# Dot product of two matrices
dot_product = np.dot(np.array([[1,2],[3,4]]), np.array([[10],[20]]))
print("\nDot product:\n", dot_product)
```

```
Dot product:
 [[ 50]
 [110]]
```

2. Pandas

Pandas is a powerful Python library for data analysis and manipulation. It introduces two primary data structures: "Series" for one-dimensional data and "DataFrame" for tabular data. With functions for reading/writing data, handling missing values, and filtering rows/columns, Pandas simplifies complex data operations. It's an essential tool for data scientists and analysts using Python.

Importing Libraries and Loading the Dataset

In [15]:
```
import pandas as pd
import seaborn as sns

# Load the Titanic dataset
df = sns.load_dataset('titanic')
```

Exploring the Dataset

In [16]:
```
# Display the first few rows of the dataset
print(df.head())

# Get information about the dataset
print(df.info())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  c
lass  \
0         0       3    male  22.0      1      0   7.2500        S  T
hird
1         1       1  female  38.0      1      0  71.2833        C  F
irst
2         1       3  female  26.0      0      0   7.9250        S  T
hird
3         1       1  female  35.0      1      0  53.1000        S  F
irst
4         0       3    male  35.0      0      0   8.0500        S  T
hird

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

Data Selection and Filtering

In [17]:
```
# Selecting the 'age' column
ages = df['age']

# Filtering passengers who survived
survived_passengers = df[df['survived'] == 1]
```

Handling Missing Data

In [18]:
```
# Fill missing ages with median age
df['age'].fillna(df['age'].median(), inplace=True)
```

Feature Engineering: Creating a New Column

In [19]:
```
# Categorize age into 'child', 'adult', 'elderly'
df['age_group'] = pd.cut(df['age'], bins=[0, 18, 60, 100], labels=['ch
```

Aggregation: Average Fare by Class

In [20]:
```
average_fare_by_class = df.groupby('class')['fare'].mean()
print(average_fare_by_class)
```

```
class
First     84.154687
Second    20.662183
Third     13.675550
Name: fare, dtype: float64
```

Saving Processed Data

In [21]:
```
df.to_csv('processed_titanic_data.csv', index=False)
```

In [ ]: