In [2]:
```
#Data Handling and Analysis with Pandas
#1. Getting Familiar with Pandas
```

In [3]:
```
import pandas as pd #importing pandas
```

In [4]:
```
#data Types in pandas
```

In [12]:
```
# Creating a Series
data = [10, 20, 30, 40, 50]
series = pd.Series(data)
print("Series:\n", series)

# Creating a DataFrame from a dictionary
data = {
    'Name': ['prem', 'yeshwanth', 'sanjana'],
    'Age': [19, 24, 30],
    'City': ['Eluru', 'Vijyawada', 'Vizag']
}
df = pd.DataFrame(data)
print("\nDataFrame:\n", df)
```

```
Series:
 0    10
1    20
2    30
3    40
4    50
dtype: int64

DataFrame:
         Name  Age       City
0       prem   19      Eluru
1  yeshwanth   24  Vijyawada
2    sanjana   30      Vizag
```

In [13]:
```
#accesing elements in data frame
```

In [14]:
```
# Selecting a single column
print("\nSelect 'Name' column:\n", df['Name'])

# Selecting multiple columns
print("\nSelect 'Name' and 'City' columns:\n", df[['Name', 'City']])

# Selecting rows by index
print("\nSelect first row:\n", df.iloc[0])

# Selecting rows by condition
print("\nSelect rows where Age > 30:\n", df[df['Age'] > 30])
```

```
Select 'Name' column:
 0         prem
1    yeshwanth
2      sanjana
Name: Name, dtype: object

Select 'Name' and 'City' columns:
         Name       City
0       prem      Eluru
1  yeshwanth  Vijyawada
2    sanjana      Vizag

Select first row:
 Name     prem
Age        19
City    Eluru
Name: 0, dtype: object

Select rows where Age > 30:
 Empty DataFrame
Columns: [Name, Age, City]
Index: []
```

In [15]:
```
#modification of the dataframe
```

In [16]:
```
# Adding a new column
df['Occupation'] = ['Engineer', 'Doctor', 'Scientist']
print("\nDataFrame with new column:\n", df)

# Modifying existing data
df.loc[1, 'City'] = 'Bezawada'
print("\nDataFrame after modification:\n", df)
```

```
DataFrame with new column:
         Name  Age       City Occupation
0       prem   19      Eluru   Engineer
1  yeshwanth   24  Vijyawada     Doctor
2    sanjana   30      Vizag  Scientist

DataFrame after modification:
         Name  Age      City Occupation
0       prem   19     Eluru   Engineer
1  yeshwanth   24  Bezawada     Doctor
2    sanjana   30     Vizag  Scientist
```

In [17]:
```
#2. Data Handling with Pandas
```

In [18]:
```
# Reading data from a CSV file
df = pd.read_csv('student.csv')  # Ensure 'data.csv' is present in your working directory
print("\nDataFrame from CSV:\n", df.head())
```

```
DataFrame from CSV:
         name  roll no class
0       siva       48   csd
1     lokesh       30   csd
2  yeshwanth       37   csd
3       teja       49   csd
```

In [19]:
```
# Checking for missing values
print("\nMissing values:\n", df.isna().sum())

# Dropping rows with missing values
df_cleaned = df.dropna()
print("\nDataFrame after dropping missing values:\n", df_cleaned)
```

```
Missing values:
 name       0
roll no    0
class      0
dtype: int64

DataFrame after dropping missing values:
         name  roll no class
0       siva       48   csd
1     lokesh       30   csd
2  yeshwanth       37   csd
3       teja       49   csd
```

In [20]:
```
# Converting data types
df['roll no'] = df['roll no'].astype(float)
print("\nDataFrame with 'Age' as float:\n", df.dtypes)
```

```
DataFrame with 'Age' as float:
 name        object
roll no    float64
class       object
dtype: object
```

In [ ]:
```
#3. Data Analysis with Pandas
```

In [21]:
```
print("\nSummary statistics:\n", df.describe())
```

```
Summary statistics:
          roll no
count   4.000000
mean   41.000000
std     9.128709
min    30.000000
25%    35.250000
50%    42.500000
75%    48.250000
max    49.000000
```

In [24]:
```
# Merging two DataFrames
df2 = pd.DataFrame({
    'name': ['girish', 'prem'],
    'roll no': [79, 88],
    'class':['csd','csd']
})
merged_df = pd.merge(df, df2, on='name')
print("\nMerged DataFrame:\n", merged_df)

# Concatenating two DataFrames
concat_df = pd.concat([df, df2], axis=0)
print("\nConcatenated DataFrame:\n", concat_df)
```

```
Merged DataFrame:
 Empty DataFrame
Columns: [name, roll no_x, class_x, roll no_y, class_y]
Index: []

Concatenated DataFrame:
         name  roll no class
0       siva     48.0   csd
1     lokesh     30.0   csd
2  yeshwanth     37.0   csd
3       teja     49.0   csd
0     girish     79.0   csd
1       prem     88.0   csd
```

In [ ]:
```
#4. Application in Data Science
```

In [27]:
```
print("""Pandas provides powerful and flexible data structures for data manipulation and analysis. The key adva
Efficient Data Handling: Pandas DataFrames and Series offer efficient data storage and manipulation capabilitie
Convenience: Built-in functions for data cleaning (e.g., handling missing values, removing duplicates) and data
Data Analysis: Functions for summarizing data, aggregating statistics, and performing group operations simplify
Real-World Examples:
Data Cleaning: In a project involving customer data from multiple sources, Pandas can be used to clean and stan
Exploratory Data Analysis (EDA): Pandas is essential for EDA, allowing data scientists to generate summary stat
Data Merging: In business analytics, combining sales data with customer information using Pandas can help in un
```

```
Pandas provides powerful and flexible data structures for data manipulation and analysis. The key advantages of
using Pandas over traditional Python data structures include:
Efficient Data Handling: Pandas DataFrames and Series offer efficient data storage and manipulation capabilities
, making it easier to handle large datasets.
Convenience: Built-in functions for data cleaning (e.g., handling missing values, removing duplicates) and data
transformation streamline preprocessing tasks.
Data Analysis: Functions for summarizing data, aggregating statistics, and performing group operations simplify
data analysis and exploration.
Real-World Examples:
Data Cleaning: In a project involving customer data from multiple sources, Pandas can be used to clean and stand
ardize data before analysis.
Exploratory Data Analysis (EDA): Pandas is essential for EDA, allowing data scientists to generate summary stati
stics, visualize relationships, and uncover trends in the data.
Data Merging: In business analytics, combining sales data with customer information using Pandas can help in und
erstanding customer behavior and improving decision-making.
```

In [ ]: