Pandas: A Python library used for data manipulation and analysis. It provides data structures and functions designed to work efficiently with structured data like tables.

DataFrame: A two-dimensional tabular data structure with labeled axes (rows and columns) in pandas.

Series: A one-dimensional array with labels. It can hold any data type (integers, strings, floats, Python objects, etc.). A Series is essentially a single column of a DataFrame.

In [1]:
```
import pandas as pd
```

Creating DataFrames

You can create a DataFrame from various data sources, like a Python dictionary, lists, or external data files.

In [2]:
```
#From Lists
data = [[1, 'Alice'], [2, 'Bob'], [3, 'Charlie']]
df = pd.DataFrame(data, columns=['ID', 'Name'])
df
```

Out[2]:

	ID	Name
0	1	Alice
1	2	Bob
2	3	Charlie

In [3]:
```
#From Dictionaries
data = {'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']}
df = pd.DataFrame(data)
df
```

Out[3]:

	ID	Name
0	1	Alice
1	2	Bob
2	3	Charlie

In [4]:
```
'''
You can specify an index when creating a DataFrame.
This is useful if you need to align your data
with a specific set of labels!'''

data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]}
index = ['Person1', 'Person2', 'Person3']
df = pd.DataFrame(data, index=index)
df
```

Out[4]:

	Name	Age
Person1	Alice	25
Person2	Bob	30
Person3	Charlie	35

In [5]:
```
#From a Series
series = pd.Series([1, 2, 3, 4], name='Column Name')
df = series.to_frame()
df
```

Out[5]:

	Column Name
0	1
1	2
2	3
3	4

In [6]:
```
'''The pd.read_csv() function reads data from a
CSV file located at the specified path and converts it into a DataFrame
allowing you to work with the data in Python.'''

df = pd.read_csv('C:/Users/Tarek/Desktop/archive/Iris.csv')
df
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [7]:
```
'''To determine the data type of the entire tips table
you would use the type() function:'''

type(df)
```

Out[7]:
```
pandas.core.frame.DataFrame
```

In [8]:
```
# Display the first few rows
df.head()
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [9]:
```
# Display the last few rows
df.tail()
```

Out[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [10]:
```
# Summary statistics for numeric columns
df.describe()
```

Out[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [11]:
```
#Information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [12]:
```
# Number of rows and columns
df.shape
```

Out[12]:
```
(150, 6)
```

In [13]:
```
# Unique values in a column
df['Species'].unique()
```

Out[13]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [14]:
```
# Value counts for a column
df['Species'].value_counts()
```

Out[14]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

In [15]:
```
# Checking for Missing Values
df.isnull().sum()
```

Out[15]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [16]:
```
# Unique Counts
df.nunique()
```

Out[16]:
```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

In [17]:
```
# Remove duplicate rows if exsists
df.drop_duplicates()
```

Out[17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [18]:
```
# Using .loc[] to select rows where species is 'versicolor' and only the 'petal length (cm)' column
selected_rows_loc = df.loc[df['Species'] == 'Iris-versicolor']
selected_rows_loc.head()
```

Out[18]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
51	52	6.4	3.2	4.5	1.5	Iris-versicolor
52	53	6.9	3.1	4.9	1.5	Iris-versicolor
53	54	5.5	2.3	4.0	1.3	Iris-versicolor
54	55	6.5	2.8	4.6	1.5	Iris-versicolor

In [19]:
```
# To select specific columns:
selected_rows_loc = df.loc[df['Species'] == 'Iris-versicolor', ['PetalLengthCm', 'PetalWidthCm']]
selected_rows_loc.head()
```

Out[19]:

	PetalLengthCm	PetalWidthCm
50	4.7	1.4
51	4.5	1.5
52	4.9	1.5
53	4.0	1.3
54	4.6	1.5

In [20]:
```
# .loc[] multiple conditions
selected_rows_loc = df.loc[(df['Species'] == 'Iris-versicolor') &
                           ((df['PetalLengthCm'] > 4.0) |
                           (df['PetalWidthCm'] < 1.5)),
                           ['PetalLengthCm', 'PetalWidthCm']]
selected_rows_loc.head()

'''This means a row will be included if it corresponds to an 'Iris-versicolor' species
and either has a petal length greater than 4.0 cm or a petal width less than 1.5 cm.
The parentheses around conditions 2 and 3 ensure that the OR operation is evaluated correctly'''
```

Out[20]:
```
"This means a row will be included if it corresponds to an 'Iris-versicolor' species\nand either has a p
etal length greater than 4.0 cm or a petal width less than 1.5 cm.\nThe parentheses around conditions 2
and 3 ensure that the OR operation is evaluated correctly"
```

In [21]:
```
'''Boolean indexing in Pandas allows you to select rows
from a DataFrame based on the values of a boolean condition.'''

# Boolean indexing to select rows where petal length is greater than 4.5 cm
df[df['PetalLengthCm'] > 4.5]
```

Out[21]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
52	53	6.9	3.1	4.9	1.5	Iris-versicolor
54	55	6.5	2.8	4.6	1.5	Iris-versicolor
56	57	6.3	3.3	4.7	1.6	Iris-versicolor
58	59	6.6	2.9	4.6	1.3	Iris-versicolor
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

63 rows × 6 columns

In [22]:
```
# Boolean indexing with multiple conditions
df[(df['PetalLengthCm'] > 4.5) & (df['Species'] == 'Iris-versicolor')].head()
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
52	53	6.9	3.1	4.9	1.5	Iris-versicolor
54	55	6.5	2.8	4.6	1.5	Iris-versicolor
56	57	6.3	3.3	4.7	1.6	Iris-versicolor
58	59	6.6	2.9	4.6	1.3	Iris-versicolor

In [23]:
```
# Use .iloc[] to select rows/columns by integer index from the filtered DataFrame.
selected_rows_iloc = df.iloc[:5, [2, 3]]  # This selects the first 5 rows and columns at index 2 and 3
selected_rows_iloc
```

Out[23]:

	SepalWidthCm	PetalLengthCm
0	3.5	1.4
1	3.0	1.4
2	3.2	1.3
3	3.1	1.5
4	3.6	1.4

In [ ]:
```

```