Module - 2

Data Analytics

Units for Discussion

Fundamentals of Data Analytics

Unit - 1

Data Analysis using NumPy

Unit - 2

Data Analysis using Pandas

Unit - 3

Data Visualization using Matplotlib

Unit - 4

Data Visualization using Seaborn

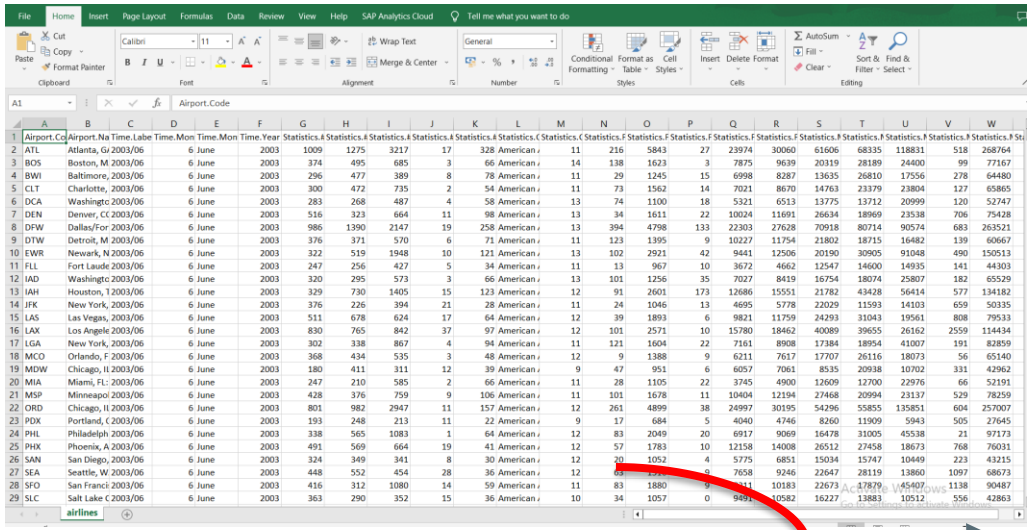
Unit - 5

Unit - 3

Data Analysis using Pandas

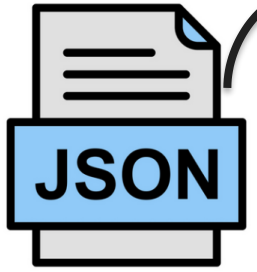
DISCLAIMER

The content is curated from online/offline resources and used for educational purpose only.



```python
# importing pandas package
import pandas as pd
data = pd.read_csv("airlines.csv")
data
```

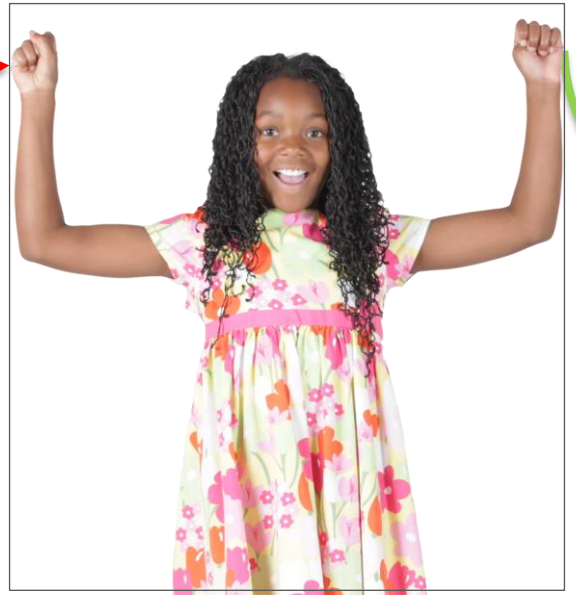
	Airport.Code	Airport.Name	Time.Label	Time.Month	Time.Month Name	Time.Year	Statistics.# of Delays.Carrier	Statistics.# of Delays.Late Aircraft	Statistics.# of Delays.National Aviation System	Statistics.# of Delays.Security	...	Statistic
0	ATL	Atlanta, GA: Hartsfield-Jackson Atlanta Intern...	2003/06	6	June	2003	1009	1275	3217	17	...	
1	BOS	Boston, MA: Logan International	2003/06	6	June	2003	374	495	685	3	...	
2	BWI	Baltimore, MD: Baltimore/Washington Internatio...	2003/06	6	June	2003	296	477	389	8	...	
3	CLT	Charlotte, NC: Charlotte Douglas International	2003/06	6	June	2003	300	472	735	2	...	
4	DCA	Washington, DC: Ronald Reagan Washington National	2003/06	6	June	2003	283	268	487	4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
4403	SAN	San Diego, CA: San Diego International	2016/01	1	January	2016	280	397	171	2	...	
4404	SEA	Seattle, WA: Seattle/Tacoma International	2016/01	1	January	2016	357	513	351	2	...	
4405	SFO	San Francisco, CA: San Francisco International	2016/01	1	January	2016	560	947	2194	2	...	



Pandas can Import Many More Data files..

Age(yrs)	Ht>5	wt(lbs)	obese
20	no	130	no
24	yes	160	yes
26	no	150	no
25			no

Missing Values



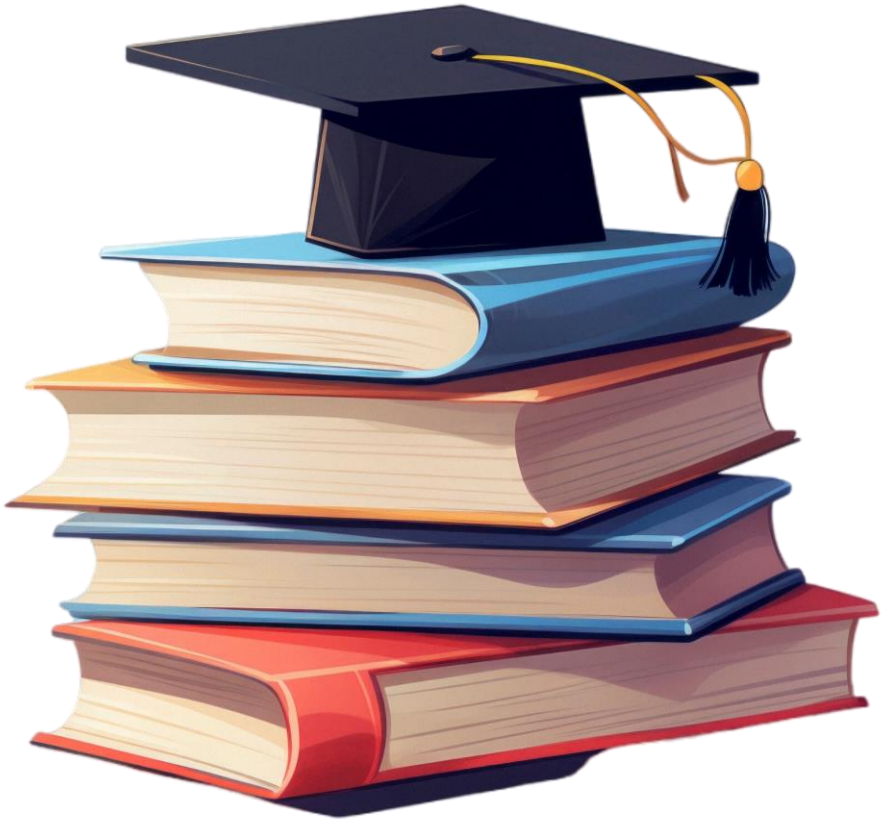
Give your Data to Pandas with the correct code

Age(yrs)	Ht>5	wt(lbs)	obese
20	no	130	no
24	yes	160	yes
26	no	150	no
25	no	150	no

Pandas solved your Problem

Learning Objectives

- Introduction to Pandas
- Why Pandas?
- Applications of Pandas
- Installation of Pandas
- Pandas Objects
- Filtering and Sorting in Pandas
- Handling missing and duplicates values
- The Groupby Function
- Statistical Function

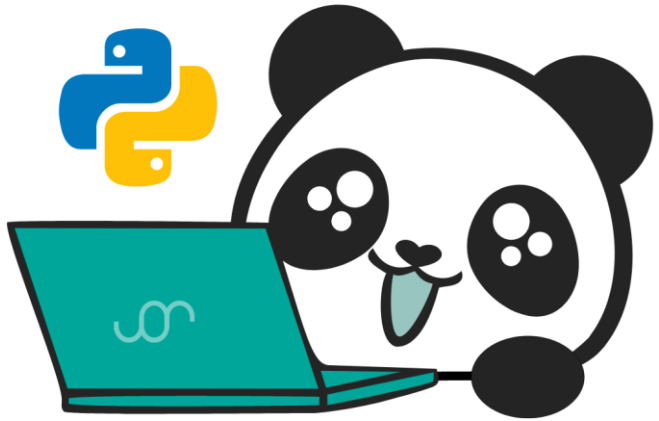


Source :

Reference

Introduction to Pandas

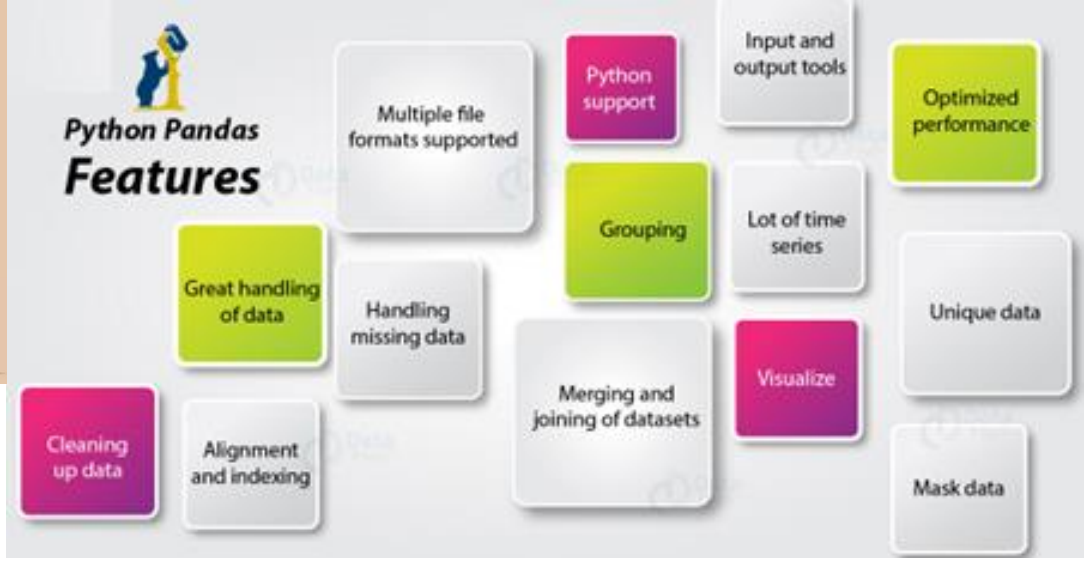
- Pandas is an open-source Python library that uses powerful data structures to provide high-performance data manipulation and analysis.
- It provides a variety of data structures and operations for manipulating numerical data and time series.
- This library is based on the NumPy library.



Why Pandas?

- Pandas allows you to become familiar with your data by cleaning, transforming, and analysing it.
- Pandas have so many applications that it might be more useful to list what it can't do than what it can.
- This tool is essentially the home of your data.

```python
import pandas as pd

df = pd.DataFrame(
{"col1": [1, 2, 3, 4],
"col2": ["a", "b", "c", "c"]
})
```



Source :

[Reference](#)
[Reference](#)
[Reference](#)

Source :

[Reference](#)
[Reference](#)

Installation of Pandas

- The first step in using pandas is to check whether it is installed in the Python folder.
- If not, we must install it on our system using the pip command.

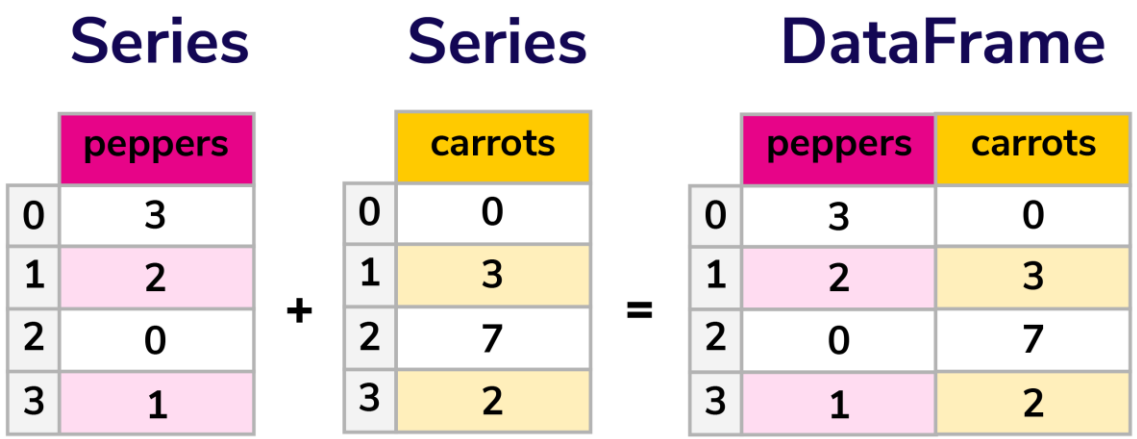
```
pip install pandas
```

- After installing pandas on your system, you'll need to import the library.
- This module is typically imported as follows:

```
import pandas as pd
```

Introducing Pandas Objects

- Pandas objects can be thought of as enhanced versions of NumPy structured arrays in which the rows and columns are identified with labels rather than simple integer indices
- There are two fundamental data structures in Pandas:
    - Series
    - DataFrame

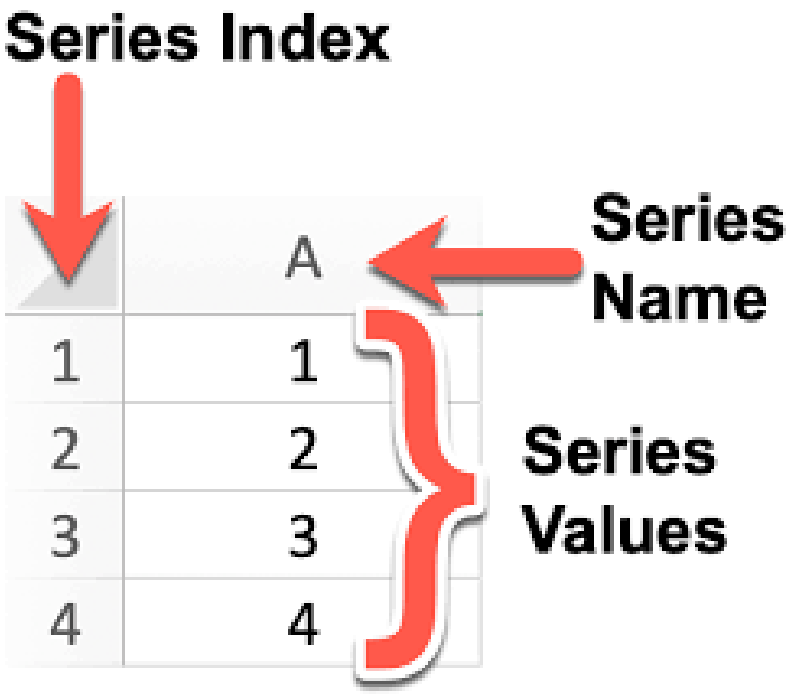


Source :

Reference

Pandas Series

- Pandas Series is a labelled one-dimensional array that can hold any type of data (integer, string, float, Python objects, and so on).
- Pandas Series is simply a column in an Excel spreadsheet.
- Using the Series() method, we can easily convert a list, tuple, or dictionary into a Series.



Source :

Reference

Pandas Series

Creating a Series

```
import pandas as pd
import numpy as np


# Creating empty series
ser = pd.Series()

print(ser)

# simple array
data = np.array(['g', 'e', 'e', 'k', 's'])

ser = pd.Series(data)
print(ser)
```

```
Series([], dtype: float64)
0    g
1    e
2    e
3    k
4    s
dtype: object
```

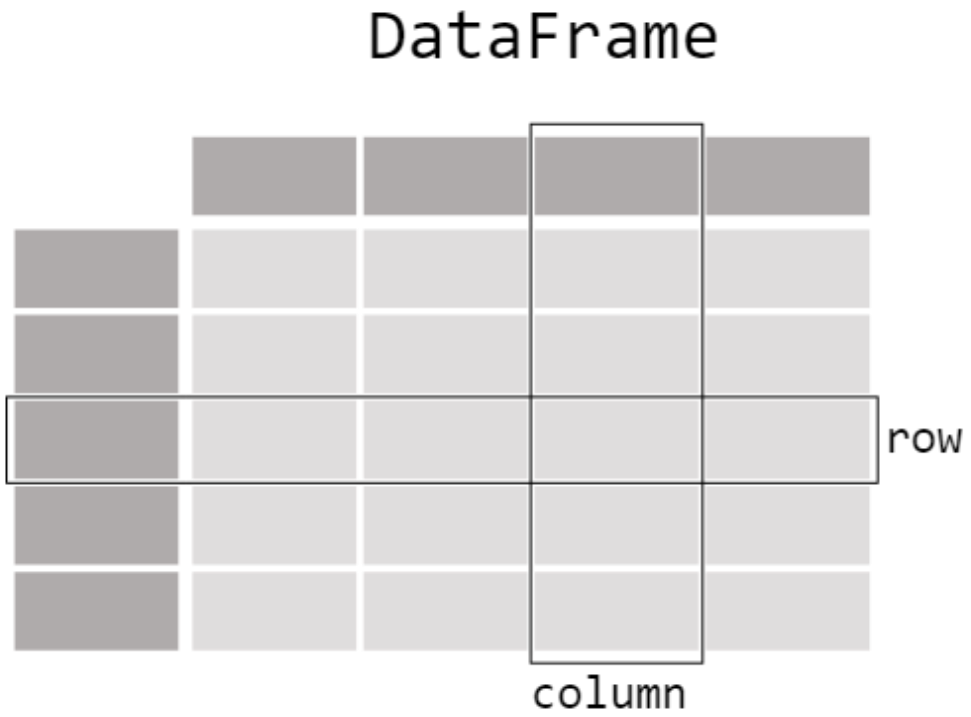
Creating a series from Lists

```
import pandas as pd

# a simple list
list = ['g', 'e', 'e', 'k', 's']

# create series form a list
ser = pd.Series(list)
print(ser)
```

```
0    g
1    e
2    e
3    k
4    s
dtype: object
```

Pandas DataFrame

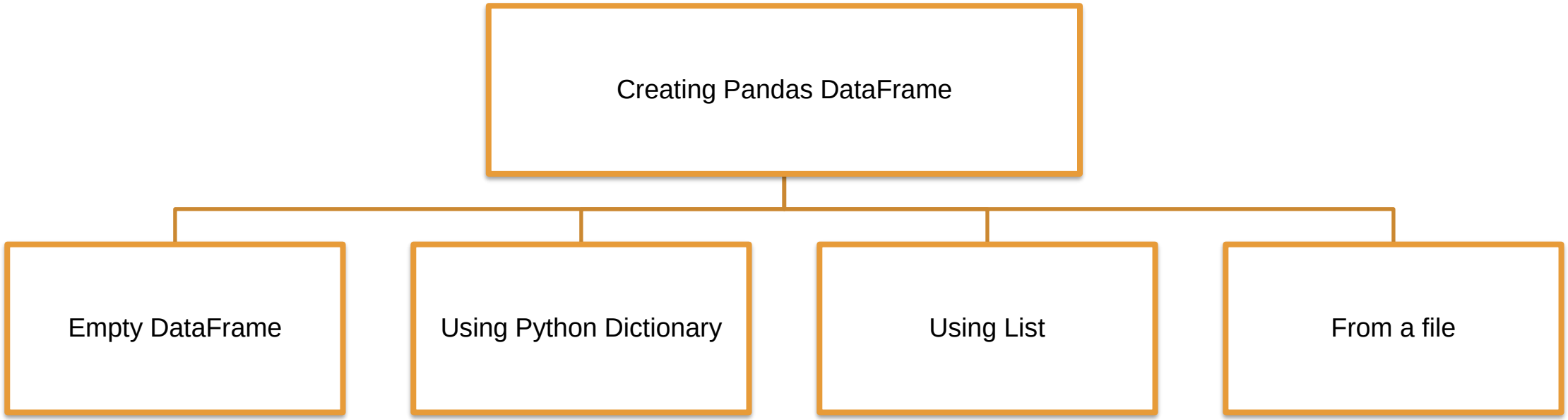
- Panda has A two-dimensional data structure with corresponding labels is known as a dataframe.
- Spreadsheets used in Excel or Calc or SQL tables are similar to DataFrames.
- Pandas DataFrame consists of three main components: the data, the index, and the columns.



Source :

Reference

Creating Pandas DataFrame



Creating Pandas DataFrame

Creating a Empty DataFrame

- Sometimes, you might want to create an empty DataFrame with just the column names, without any data.
- Empty dataframe can be useful when you want to define the structure of your DataFrame before filling it with data.
- To create an empty DataFrame with only column names, you can use the pandas.DataFrame() constructor and specify the column names as a list.

```python
# Lets create empty dataframe
import pandas as pd

# Create the pandas DataFrame
empty_df=pd.DataFrame(columns=['Column 1', 'Column 2', 'Column 3'])

# Print the output
print(empty_df)
```

```
Empty DataFrame
Columns: [Column 1, Column 2, Column 3]
Index: []
```

Creating Pandas DataFrame

Creating a Pandas DataFrame Using List

- We can create dataframe from list by using DataFrame() method.
- In this example, we created a two-dimensional list called list_of_list containing nested lists.
- The DataFrame() function converts the 2-D list to a DataFrame.
- Each nested list behaves like a row of data in the DataFrame.

```
# Creating a dataframe using a conventional list of lists
# Initialize list of lists
list_of_lists = [['Tom', 20], ['Nick', 21], ['Krish', 19],['Jack',18]]

# Create the pandas DataFrame
df = pd.DataFrame(list_of_lists, columns = ['Name', 'Age'])

# Print dataframe
df
```

	Name	Age
0	Tom	20
1	Nick	21
2	Krish	19
3	Jack	18

Creating Pandas DataFrame

Creating a Pandas DataFrame Using Dictionary

- We can create dataframe from dictionary by using DataFrame() method.

```
import pandas as pd

# Store the student data in dictionary
std_dict = { 'Name': ['Tom', 'Nick', 'Krish', 'Jack'], 'Age': [20, 21, 19, 18]}

# Create the pandas DataFrame
df = pd.DataFrame(std_dict)

# Print the output.
print(df)
```

```
    Name  Age
0    Tom   20
1   Nick   21
2  Krish   19
3   Jack   18
```

Reading Data From a File

- We can create a DataFrame by loading data from a CSV (Comma-Separated Values) file.
- For example, we create dataframe from Airlines.csv file by using read_csv() method.

```python
# importing pandas package
import pandas as pd
data = pd.read_csv("Airlines.csv")
data
```

	Callsign	Tag	Number	unnamed
0	Dornier Aviation Nigeria Aiep	DO / DAV	3 aircraft	NaN
1	LATAM Airlines Ecuador	XL / LNE	6 aircraft	
2	9 Air	AQ / JYH	8 aircraft	NaN
3	ABX Air	GB / ABX	14 aircraft	NaN
4	Adria Airways	JP / ADR	13 aircraft	NaN
...	...	...	...	...
918	Yangtze River Express	Y8 / YZR	25 aircraft	NaN
919	Yemenia	IY / IYE	5 aircraft	NaN
920	Yeti Airlines	YT / NYT	2 aircraft	NaN
921	Zagros Airlines	ZO / IZG	18 aircraft	NaN
922	ZagrosJet	Z4 / GZQ	1 aircraft	NaN

923 rows × 4 columns

Pandas Index

- Pandas Index is an efficient tool for extracting particular rows and columns of data from a DataFrame.
- Its job is to organize data and make it easily accessible.
- We can also define an index, similar to an address, through which we can access any data in the Series or DataFrame.

	office	candidates	hired
0	Atlanta	100	34.0
1	Boston	120	43.0
2	New York	130	32.0
3	Miami	105	NaN

Source :

Pandas Index

Reindexing

- Reindexing modifies the row and column labels of a DataFrame.
- It denotes verifying that the data corresponds to a specific set of labels along an established axis.
- Indexing enables us to carry out a variety of operations, including:-
    - Insert missing value (NaN) markers in label locations where there was previously no data for the label.
    - To reorder existing data to correspond to a new set of labels.

Reindexing

- To reindex the dataframe, use the reindex() function.
- Values in the new index that do not have matching records in the dataframe are by default given the value NaN.

```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({
    'A': [1, 2, 3],
    'B': [4, 5, 6]
}, index=['a', 'b', 'c'])
print(df)
```

```
   A  B
a  1  4
b  2  5
c  3  6
```

Now, we can use the dataframe.reindex() function to reindex the dataframe.

```
new_index = ['c', 'a', 'e']
df_reindexed = df.reindex(new_index)

print(df_reindexed)
```

```
     A    B
c  3.0  6.0
a  1.0  4.0
e  NaN  NaN
```

Exploring Pandas DataFrame

- Attributes are the properties of a DataFrame that can be used to fetch data or any information related to a particular dataframe. Some important attributes are given below:

Attribute	Description
index	The index property returns the index information of the DataFrame.
columns	The columns property returns the label of each column in the DataFrame.
dtypes	The dtype attributes display the data type for each column of a particular dataframe.
size	This attribute is used to display the total number of elements or items present in a data frame.
shape	This attribute is used to display the total number of rows and columns of a particular data frame
ndim	this attribute is used to display the number of dimensions of a particular data frame

Pandas Sort

There are two kinds of sorting available in Pandas. They are –

- By label
- By Actual Value

By Label - When using the sort_index() method, DataFrame can be sorted by passing the axis arguments and the sorting order. Row labels are sorted by default in ascending order.

```
import pandas as pd

data = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                     'Age': [25, 50, 35, 48],
                     'Salary': [10000, 96000, 54000, 52000]})
print(data)

#Sort the rows on the basis of their index
sorted_data = data.sort_index(axis=0, ascending=False)
print("\n",sorted_data)
```

```
      Name  Age  Salary
0    Alice   25   10000
1      Bob   50   96000
2  Charlie   35   54000
3    David   48   52000

       Name  Age  Salary
3    David   48   52000
2  Charlie   35   54000
1      Bob   50   96000
0    Alice   25   10000
```

Pandas Sort

Sort the Columns

- Sorting on the column labels is possible by passing the axis argument a value of 0 or 1. Sort by row by default, axis=0. To better understand this, consider the following example.

```
import pandas as pd

data = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                     'Age': [25, 50, 35, 48],
                     'Salary': [10000, 96000, 54000, 52000]})
print(data)

# Lets sort the DataFrame based on column names,
sorted_data = data.sort_index(axis=1, ascending=True)
print(sorted_data)
```

```
      Name  Age  Salary
0    Alice   25   10000
1      Bob   50   96000
2  Charlie   35   54000
3    David   48   52000
   Age     Name  Salary
0   25    Alice   10000
1   50      Bob   96000
2   35  Charlie   54000
3   48    David   52000
```

Pandas Sort

By Value

- Sort_values(), like index sorting, is a method for sorting by values. It accepts a 'by' argument, which will use the column name of the DataFrame to sort the values.

```python
import pandas as pd

data = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie', 'David'],
                     'Age': [25, 50, 35, 48],
                     'Salary': [10000, 96000, 54000, 52000]})
print(data)

# Lets sort the rows of a dataframe based on Single column
sorted_data = data.sort_values(by='Age', ascending=True)
print("\n",sorted_data)
```

```
      Name  Age  Salary
0    Alice   25   10000
1      Bob   50   96000
2  Charlie   35   54000
3    David   48   52000

       Name  Age  Salary
0    Alice   25   10000
2  Charlie   35   54000
3    David   48   52000
1      Bob   50   96000
```

Filtering a DataFrame

- Filtering is one of the most commonly performed operation on dataframe.
- It essentially involves excluding some values to meet a condition or set of conditions.
- Pandas provides loc and iloc methods for filtering, selecting, and manipulating data. They allow us to access the desired combination of rows and columns.
- The main difference between them is the way they access rows and columns:

  loc uses row and column labels.

  iloc uses row and column indexes.

	Column index	0	1	2
Row index	Label	A	B	C
0	Day 1			
1	Day 2			
2	Day 3			

Filtering a DataFrame

Difference between loc and iloc

- The loc and iloc methods are frequently used to select or extract a part of a data frame.
- The main difference is that loc works with labels whereas iloc works with indices.
- Consider the DataFrame shown in the following illustration.
- To access the rows and columns using the iloc method, we use the index values (0, 1, and 2).
- If we want to use the loc method, we need to use the labels A, Day 1, and so on).

	Column index	0	1	2
Row index	Label	A	B	C
0	Day 1			
1	Day 2			
2	Day 3			

Filtering a DataFrame

Selecting a Subset of columns by Using Square bracket notation

- To select multiple columns in a pandas dataframe, we can use bracket notation.
- We can pass a list of column names inside the brackets to select those specific columns.
- For Example: To select column Name and age of dataframe : df[[“Name”,”Age”]]

Dataframe

	Name	Age	City	Marks	
0					
1					
2					
3					

df[[“Name”,”Age”]]

Subset of Dataframe

	Name	Age
0		
1		
2		
3		

Handling Missing values

In Pandas, missing values, often represented as NaN (Not a Number), can cause problems during data processing and analysis. These gaps in data can lead to incorrect analysis and misleading conclusions.

Depending upon the problem and nature of data ,We have three approaches for handling missing values.

- First, we can drop rows with null values
- Second, we can drop columns that contain missing values.
- Replacing null values

Finding Missing values

- The first step in handling missing values is to find them.
- We can use either the isna() or isnull() function to detect missing values.
- The isnull() function evaluates each cell in a DataFrame and returns True to indicate a missing value.

	A	B	C	D
0	1.0	2.4	NaN	11.5
1	2.0	NaN	foo	NaN
2	3.0	5.1	zoo	6.2
3	NaN	NaN	bar	21.1
4	7.0	2.6	NaN	8.7

Data Frame with missing values

	A	B	C	D
0	False	False	True	False
1	False	True	False	True
2	False	False	False	False
3	True	True	False	False
4	False	False	True	False

Dropping rows and columns with Null values

- After finding missing values , depending upon the data we can drop rows or columns that contain missing values.
- The dropna function is used to drop rows and columns with missing values.

	A	B	C	D	E
0	1.0	2.4	NaN	11.5	1
1	2.0	NaN	foo	NaN	2
2	3.0	5.1	zoo	6.2	3
3	NaN	NaN	bar	21.1	4
4	7.0	2.6	NaN	8.7	5

Data Frame with missing values

	A	B	C	D	E
2	3.0	5.1	zoo	6.2	3

Data Frame after dropping rows

Replacing Missing values

- Dropping may not be the best option in many cases.
- We can replace missing values with an actual value with the fillna function.

	A	B	D	E
0	1.0	2.4	11.5	1
1	2.0	NaN	NaN	2
2	3.0	5.1	6.2	3
3	NaN	NaN	21.1	4
4	7.0	2.6	8.7	5

Data Frame with missing values

	A	B	D	E
0	1.00	2.400000	11.500	1
1	2.00	3.366667	11.875	2
2	3.00	5.100000	6.200	3
3	3.25	3.366667	21.100	4
4	7.00	2.600000	8.700	5

Data Frame after replacing missing values

Handling Duplicates values

Finding Duplicates values:

- In large datasets, we often encounter duplicate entries in tables. These duplicate entries can throw off our analysis and skew the results.
- Pandas provides duplicated() function to find duplicates values in dataframe.
- The function returns a series of boolean values depicting whether a record is duplicated.

```python
import pandas as pd

# create dataframe
data = {
    'Name': ['Aman', 'Ram', 'Aman', 'Ram', 'Aman'],
    'Age': [27, 24, 27, 24, 19],
    'City': ['Haryana', 'Surat', 'Haryana', 'New Delhi', 'Rajkot']
}
df = pd.DataFrame(data)
print(df)
# check for duplicate entries
print("\n",df.duplicated())
```

```
   Name  Age       City
0  Aman   27    Haryana
1   Ram   24      Surat
2  Aman   27    Haryana
3   Ram   24  New Delhi
4  Aman   19     Rajkot

 0    False
1    False
2     True
3    False
4    False
dtype: bool
```

Handling Duplicates values

Dropping Duplicates values:

- To drop duplicate rows, pandas provides drop_duplicates() method. By default, this method keeps the first occurrence of each duplicated row and drops the rest.
- If we want to keep the last occurrence of each duplicated row and drop the rest, use the keep parameter.

```
# remove duplicates
df.drop_duplicates(inplace=True)
df
```

	Name	Age	City
0	Aman	27	Haryana
1	Ram	24	Surat
3	Ram	24	New Delhi
4	Aman	19	Rajkot

The Groupby function

- Pandas provides groupby method, which allows you to perform efficient grouping and aggregation operations on data stored in a DataFrame object.
- The groupby operation involves the “split-apply-combine” approach, which consists of three steps:

Split: Data is divided into groups based on specified criteria.

Apply: A function is applied to each group independently.

Combine: The results from the applied function are combined into a new DataFrame

Stages in Groupby

- The following drawing illustrates how the groupby function operates.

Product_group	Product_code	price
A	1001	9
A	1002	14
B	1101	21
A	1003	12
B	1104	19
C	1201	7
B	1105	25

Product_group	Product_code	price
A	1001	9
A	1002	14
A	1103	12

mean() → 11.67

Product_group	Product_code	price
B	1101	21
B	1104	19
B	1105	25

mean() → 22

Product_group	Product_code	price
C	1101	21

mean() → 7

Statistical Functions

- Using pandas, it is simple to simplify numerous complex statistical operations in Python to a single line of code.
- Some of the most popular and practical statistical operations will be covered.

sum():	Return the sum of the values.
count():	Return the count of non-empty values.
max():	Return the maximum of the values.
min():	Return the minimum of the values.
mean():	Return the mean of the values.
median():	Return the median of the values.
std():	Return the standard deviation of the values.
describe():	Return the summary statistics for each column

Statistical Functions

Pandas sum() method

```python
import pandas as pd

# Dataset
data = {
  'Maths': [90, 85, 98, 80, 55, 78],
  'Science': [92, 87, 59, 64, 87, 96],
  'English': [95, 94, 84, 75, 67, 65]
}

# DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame = \n",df)

# Display the Sum of Marks in each column
print("\nSum = \n",df.sum())
```

```
DataFrame =
    Maths  Science  English
0      90       92       95
1      85       87       94
2      98       59       84
3      80       64       75
4      55       87       67
5      78       96       65

Sum =
 Maths      486
Science    485
English    480
dtype: int64
```

Pandas count() method

```python
import pandas as pd

# Dataset
data = {
  'Maths': [90, 85, 98, None, 55, 78],
  'Science': [92, 87, 59, None, None, 96],
  'English': [95, None, 84, 75, 67, None]
}

# DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame = \n",df)

# Display the Count of non-empty values in each column
print("\nCount of non-empty values = \n",df.count())
```

```
DataFrame =
    Maths  Science  English
0   90.0     92.0     95.0
1   85.0     87.0      NaN
2   98.0     59.0     84.0
3    NaN      NaN     75.0
4   55.0      NaN     67.0
5   78.0     96.0      NaN

Count of non-empty values =
 Maths      5
Science    4
English    4
dtype: int64
```

Statistical Functions

Pandas max() method

```python
import pandas as pd

# Dataset
data = {
  'Maths': [90, 85, 98, 80, 55, 78],
  'Science': [92, 87, 59, 64, 87, 96],
  'English': [95, 94, 84, 75, 67, 65]
}

# DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame = \n",df)

# Display the Maximum of Marks in each column
print("\nMaximum Marks = \n",df.max())
```

```
DataFrame =
    Maths  Science  English
0      90       92       95
1      85       87       94
2      98       59       84
3      80       64       75
4      55       87       67
5      78       96       65

Maximum Marks =
 Maths      98
Science    96
English    95
dtype: int64
```

Pandas min() method

```python
import pandas as pd

# Dataset
data = {
  'Maths': [90, 85, 98, 80, 55, 78],
  'Science': [92, 87, 59, 64, 87, 96],
  'English': [95, 94, 84, 75, 67, 65]
}

# DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame = \n",df)

# Display the Minimum of Marks in each column
print("\nMinimum Marks = \n",df.min())
```

```
DataFrame =
    Maths  Science  English
0      90       92       95
1      85       87       94
2      98       59       84
3      80       64       75
4      55       87       67
5      78       96       65

Minimum Marks =
 Maths      55
Science    59
English    65
dtype: int64
```

Statistical Functions

Pandas median() method

```python
import pandas as pd

# Dataset
data = {
  'Maths': [90, 85, 98, 80, 55, 78],
  'Science': [92, 87, 59, 64, 87, 96],
  'English': [95, 94, 84, 75, 67, 65]
}

# DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("DataFrame = \n",df)

# Display the Median of Marks in each column
print("\nMedian = \n",df.median())
```

```
DataFrame =
    Maths  Science  English
0      90       92       95
1      85       87       94
2      98       59       84
3      80       64       75
4      55       87       67
5      78       96       65

Median =
 Maths      82.5
Science    87.0
English    79.5
dtype: float64
```

Summary Statistics

Hip Hip Hurray!

Just One command and get All Insights from Data

Pandas Describe

pd.DataFrame.describe()

Index	rand_num
0	7
1	1
2	6
3	2
4	6

count	5	The number of values in your dataset
mean	4.4	The average of your values
std	2.701	The Standard Deviation of your values
min	1	The smallest value
25%	2	The value at the 25% percentile
50%	6	The value at the 50% percentile
75%	6	The value at the 75% percentile
max	7	The largest value

Lab - 1

Exploring Pandas for Data Manipulation

Conclusion

We have completed this section and now we have understood about:

- What is Pandas
- Application of Pandas
- Objects of Pandas –Series and DataFrame
- How to import Pandas Library
- How to import files using Pandas
- Indexing in Pandas
- Sorting and filtering method in Pandas
- How to handle missing and duplicates values
- This Knowledge we will use in Machine Learning, Data Analysis, Visualization and Mathematical Operation.

Source :

Reference

References

- https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
- https://docs.python.org/3/library/
- https://pandas.pydata.org/docs/user_guide/10min.html
- https://www.geeksforgeeks.org/python-pandas-series/
- https://towardsdatascience.com/pandas-index-explained-b131beaf6f7b
- https://medium.com/analytics-vidhya/introduction-to-pandas-90b75a5c2278
- https://mode.com/python-tutorial/libraries/pandas/
- https://www.freepik.com/


Q&A
Let’s Start

Quiz

1. Pandas Stands For_____

a) Panel Data Analysis

b) Panel Data Analyst

c) Panel Data

d) Panel Dashboard

Answer: C

Panel Data

Quiz

2. _____is in important library used for analyzing data.

a) Math

b) Random

c) Pandas

d) None of the above

Answer: C

Pandas

Quiz

3. _____is used when data in Tabular Format

a) NumPy

b) Pandas

c) Matplotlib

d) All of the above

Answer: B

Pandas

Quiz

4. Which of the following command is used to install Pandas?

a) pip install pandas

b) install pandas

c) pip pandas

d) None of the above

Answer: A

pip install pandas

Quiz

5. A _____is a One-dimensional array.

a) Data Frame

b) Series

c) Both of the above

d) None of the above

Answer: B

Series

Thank You