**PANDAS IN PYTHON**

To handle the big data, requires data structures that can be easily customized. Big data comes in multiple type, which can have missing data that needs to be handled.

Additionally, there will be various functions and mathematical operations to be applied to this big data.

That’s where Pandas comes in. panda is derived from panel data.

Pandas are classified into two main data structures:

* Series is used to handle one dimensional data
* DataFrames are used to handle two-dimensional data

Pandas uses array behind the scenes and is very closely related to Numpy library.

Several Numpy libraries functions accepts Series and DataFrames as functional arguments, allowing us to use pandas with the Numpy library.

**SERIES**

Its an enhanced one dimensional array, while arrays use zero-based indexing, which is numeric.

Series supports custom indexing, just like strings.

Series also handle missing data, similar to many functions on the series.

We can create a series using Numpy Array, List, Map etc.

**INSTALL PANDAS FROM COMMAND LINE**

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) PS C:\Users\kamal\PycharmProjects\pandasExample> pip install pandas

Collecting pandas

Obtaining dependency information for pandas from https://files.pythonhosted.org/packages/61/11/1812ef6cbd7433ad240f72161ce5f84c4c450cede4db080365d371d29117/pandas-2.2.1-cp311-cp311-win\_amd64.whl.metadata

Downloading pandas-2.2.1-cp311-cp311-win\_amd64.whl.metadata (19 kB)

Collecting numpy<2,>=1.23.2 (from pandas)

Obtaining dependency information for numpy<2,>=1.23.2 from https://files.pythonhosted.org/packages/3f/6b/5610004206cf7f8e7ad91c5a85a8c71b2f2f8051a0c0c4d5916b76d6cbb2/numpy-1.26.4-cp311-cp311-win\_amd64.whl.metadata

Using cached numpy-1.26.4-cp311-cp311-win\_amd64.whl.metadata (61 kB)

Collecting python-dateutil>=2.8.2 (from pandas)

Obtaining dependency information for python-dateutil>=2.8.2 from https://files.pythonhosted.org/packages/ec/57/56b9bcc3c9c6a792fcbaf139543cee77261f3651ca9da0c93f5c1221264b/python\_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata

Downloading python\_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)

Collecting pytz>=2020.1 (from pandas)

Obtaining dependency information for pytz>=2020.1 from https://files.pythonhosted.org/packages/9c/3d/a121f284241f08268b21359bd425f7d4825cffc5ac5cd0e1b3d82ffd2b10/pytz-2024.1-py2.py3-none-any.whl.metadata

Downloading pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)

Collecting tzdata>=2022.7 (from pandas)

Obtaining dependency information for tzdata>=2022.7 from https://files.pythonhosted.org/packages/65/58/f9c9e6be752e9fcb8b6a0ee9fb87e6e7a1f6bcab2cdc73f02bb7ba91ada0/tzdata-2024.1-py2.py3-none-any.whl.metadata

Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)

Collecting six>=1.5 (from python-dateutil>=2.8.2->pandas)

Obtaining dependency information for six>=1.5 from https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a01506781dbcf25c91daf11/six-1.16.0-py2.py3-none-any.whl.metadata

Downloading six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)

Downloading pandas-2.2.1-cp311-cp311-win\_amd64.whl (11.6 MB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 9.6 MB/s eta 0:00:00

Using cached numpy-1.26.4-cp311-cp311-win\_amd64.whl (15.8 MB)

Downloading python\_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 229.9/229.9 kB 13.7 MB/s eta 0:00:00

Downloading pytz-2024.1-py2.py3-none-any.whl (505 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 505.5/505.5 kB 33.0 MB/s eta 0:00:00

Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 345.4/345.4 kB 20.9 MB/s eta 0:00:00

Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)

Installing collected packages: pytz, tzdata, six, numpy, python-dateutil, pandas

Successfully installed numpy-1.26.4 pandas-2.2.1 python-dateutil-2.9.0.post0 pytz-2024.1 six-1.16.0 tzdata-2024.1

[notice] A new release of pip is available: 23.2.1 -> 24.0

[notice] To update, run: python.exe -m pip install --upgrade pip

(.venv) PS C:\Users\kamal\PycharmProjects\pandasExample>

**INSTALL PANDAS FROM PYCHARM**

A screenshot of a computer

Description automatically generated

**USING DEFAULT INDICES (INDEX)**

**Code:**

import pandas as pd  
  
reviews = pd.Series([4.6,4.4,4.8,5,4.2])  
  
print("Series : ")  
print(reviews)  
print()  
  
print("Single Series data to retrieve:")  
print(reviews[1])  
print()  
  
print("Mean of elements in the series:")  
print(reviews.mean())  
print()  
  
print("Min value of elements in the series:")  
print(reviews.min())  
print()  
  
print("Max value of elements in the series:")  
print(reviews.max())  
print()  
  
print("Standard Deviation of elements in the series:")  
print(reviews.std())  
print()

**Output:**

C:\Users\kamal\PycharmProjects\pandasExample\.venv\Scripts\python.exe C:\Users\kamal\PycharmProjects\pandasExample\seriesdemo.py

Series :

0 4.6

1 4.4

2 4.8

3 5.0

4 4.2

dtype: float64

Single Series data to retrieve:

4.4

Mean of elements in the series:

4.6

Min value of elements in the series:

4.2

Max value of elements in the series:

5.0

Standard Deviation of elements in the series:

0.31622776601683783

**USING CUSTOM INDICES (INDEX)**

**Code:**

import pandas as pd  
  
reviews = pd.Series([4.6,4.4,4.8,5,4.2],index=['python','sqlserver','devops','aws','.Net'])  
print("Series : ")  
print(reviews)  
print()  
  
*# USING DICTIONARY TO THE ABOVE SERIES*reviews= pd.Series({'augusta':48.8, 'kamalkumar':48.7, 'jazlyn':16.6, 'jerrick':11.5})  
print("Series using Dictionary: ")  
print(reviews)  
print()  
  
*#ACCESSING USING THE CUSTOM INDEX*print("Single Series data to retrieve:")  
print(reviews['augusta'])  
*# or*print(reviews.jazlyn)  
print()  
*#PRINT THE VALUES ONLY*print("Values Only")  
print(reviews.values)  
print()

*#PRINT THE INDEXES ONLY*print("Indexes Only")  
print(reviews.index)  
print()

**Output:**

Series :

python 4.6

sqlserver 4.4

devops 4.8

aws 5.0

.Net 4.2

dtype: float64

Series using Dictionary:

augusta 48.8

kamalkumar 48.7

jazlyn 16.6

jerrick 11.5

dtype: float64

Single Series data to retrieve:

48.8

16.6

Values Only

[48.8 48.7 16.6 11.5]

Indexes Only

Index(['augusta', 'kamalkumar', 'jazlyn', 'jerrick'], dtype='object')

**STRING TYPE DATA IN PANDAS**

**Code:**

import pandas as pd  
  
phones = pd.Series(['iPhone','Samsung','OnePlus','Oppo','Readme','Nokia','Huawei'])  
*# PRINT THE SERIES*print("Single Series data to retrieve:")  
print(phones)  
print()  
  
*# CONVERT THE SERIES TO UPPER CASE*print("Series data in Upper Case")  
print(phones.str.upper())  
print()  
  
*# CONTAINS THE SERIES TO UPPER CASE*print("Contains any string in Series")  
print(phones.str.contains('o'))  
print()

**Output:**

Single Series data to retrieve:

0 iPhone

1 Samsung

2 OnePlus

3 Oppo

4 Readme

5 Nokia

6 Huawei

dtype: object

Series data in Upper Case

0 IPHONE

1 SAMSUNG

2 ONEPLUS

3 OPPO

4 README

5 NOKIA

6 HUAWEI

dtype: object

Contains any string in Series

0 True

1 False

2 False

3 True

4 False

5 True

6 False

dtype: bool

**DESCRIBE() IN SERIES**

Describe method return descriptive statistics for the given series of data.

It will handle the NaN values.

The Percentile rank of a value tells us the percentage of values in a dataset that rank equal to or below a given value.

25th Percentile : First or Lower, Quartile. The 25th Percentile is the value at which 25% of the answers lie between the value and 75 Percentile of the answers lie above that value.

**Code:**

import pandas as pd  
  
phones = pd.Series(['iPhone','Samsung','OnePlus','Oppo','Readme','Nokia','Huawei'])  
price = pd.Series([1079.99,1010.29,899.95,799.99,500.79,750.00,1011.78])  
  
*# PRINT THE SERIES*print("Single Series data to retrieve: (String Data)")  
print(phones)  
print()  
  
*# DESCRIBE METHOD*print("Descriptive Value for the data : (String Data)")  
print(phones.describe())  
print()  
  
*# PRINT THE SERIES*print("Single Series data to retrieve: (Float Data)")  
print(price)  
print()  
  
*# DESCRIBE METHOD*print("Descriptive Value for the data : (Float Data)")  
print(price.describe())  
print()

**Output:**

Single Series data to retrieve: (String Data)

0 iPhone

1 Samsung

2 OnePlus

3 Oppo

4 Readme

5 Nokia

6 Huawei

dtype: object

Descriptive Value for the data : (String Data)

count 7

unique 7

top iPhone

freq 1

dtype: object

Single Series data to retrieve: (Float Data)

0 1079.99

1 1010.29

2 899.95

3 799.99

4 500.79

5 750.00

6 1011.78

dtype: float64

Descriptive Value for the data : (Float Data)

count 7.000000

mean 864.684286

std 200.081351

min 500.790000

25% 774.995000

50% 899.950000

75% 1011.035000

max 1079.990000

dtype: float64

**DATAFRAMES**

Dataframes are improved two dimensional arrays.

They allow custom row and column indexing. They have various operations required for data science projects.

Each column in a dataframe is a series.

**Code:**

import pandas as pd  
  
salary\_dict = {  
 'Kumar':[62,67,75,85],  
 'Ragav':[65,72,77,82],  
 'Purusoth':[75,82,85,90],  
 'Sathya':[65,70,75,80]  
}  
  
*#CREATE A DATA FRAME  
#salary = pd.DataFrame(salary\_dict, index=[2022,2023,2024,2025])  
# OTHER WAY OF PASSING THE INDEXES*salary = pd.DataFrame(salary\_dict)  
salary.index = [2022,2023,2024,2025]  
  
*#PRINT THE DATA FRAME*print("The output as Dataframe")  
print(salary)  
print()  
  
*#ACCESSING THE COLUMN DATA*print("Print the Columne Data of a Purusoth")  
*#AS THE COLUMN NAMES ARE VALID PYTHON IDENTIFIER STRINGS, THIS WILL ALSO WORK*print(salary.Purusoth) *# OR #print(salary['Purusoth'])*print()  
  
*#ACCESSING THE ROW DATA*print("Print the Row Data of a Salary - Index 2024")  
print(salary.loc[2024])  
print()  
*#USING ILOC*print("Print the Row Data of a Salary - Row 1")  
print(salary.iloc[1])

**Output:**

The output as Dataframe

Kumar Ragav Purusoth Sathya

2022 62 65 75 65

2023 67 72 82 70

2024 75 77 85 75

2025 85 82 90 80

Print the Columne Data of a Purusoth

2022 75

2023 82

2024 85

2025 90

Name: Purusoth, dtype: int64

Print the Row Data of a Salary - Index 2024

Kumar 75

Ragav 77

Purusoth 85

Sathya 75

Name: 2024, dtype: int64

Print the Row Data of a Salary - Row 1

Kumar 67

Ragav 72

Purusoth 82

Sathya 70

Name: 2023, dtype: int64

**SLICING USING LOC AND ILOC**

**Code:**

import pandas as pd  
  
salary\_dict = {  
 'Kumar':[62,67,75,85],  
 'Ragav':[65,72,77,82],  
 'Purusoth':[75,82,85,90],  
 'Sathya':[65,70,75,80]  
}  
  
*#CREATE A DATA FRAME  
#salary = pd.DataFrame(salary\_dict, index=[2022,2023,2024,2025])  
# OTHER WAY OF PASSING THE INDEXES*salary = pd.DataFrame(salary\_dict)  
salary.index = [2022,2023,2024,2025]

*#USING SLICING IN LOC*print("Print the Row Data of a Salary - using Slicing (loc)")  
print(salary.loc[2023:2025])  
print()  
  
*#USING SLICING IN ILOC*print("Print the Row Data of a Salary - using Slicing (iloc)")  
print(salary.iloc[1:3]) *#LAST ROW (3) WILL EXCLUDE 3*print()  
  
*#USING LIST IN LOC*print("Print the Row Data of a Salary - using lISTING (loc)")  
print(salary.loc[[2023,2025]])  
print()  
  
*#USING LIST IN iLOC*print("Print the Row Data of a Salary - using lISTING (Iloc)")  
print(salary.iloc[[2,3]])  
print()

**Output:**

Print the Row Data of a Salary - using Slicing (loc)

Kumar Ragav Purusoth Sathya

2023 67 72 82 70

2024 75 77 85 75

2025 85 82 90 80

Print the Row Data of a Salary - using Slicing (iloc)

Kumar Ragav Purusoth Sathya

2023 67 72 82 70

2024 75 77 85 75

Print the Row Data of a Salary - using lISTING (loc)

Kumar Ragav Purusoth Sathya

2023 67 72 82 70

2025 85 82 90 80

Print the Row Data of a Salary - using lISTING (Iloc)

Kumar Ragav Purusoth Sathya

2024 75 77 85 75

2025 85 82 90 80

**USING SUBSET**

**Code:**

*#USING SUBSET IN LOC*print("Print the Row Data of a Salary - using Subset (loc)")  
print(salary.loc[2023:2025, ['Ragav','Sathya']])  
print()  
  
*#USING SUBSET IN LOC*print("Print the Row Data of a Salary - using Subset (iloc)")  
print(salary.iloc[[2,3],1:3])  
print()

**Output:**

Print the Row Data of a Salary - using Subset (loc)

Ragav Sathya

2023 72 70

2024 77 75

2025 82 80

Print the Row Data of a Salary - using Subset (iloc)

Ragav Purusoth

2024 77 85

2025 82 90

**BOOLEAN INDEXING**

**Code:**

*#USING Nan FOR SALARY > 77 (NaN --> Not a Number)*print("Salary which is greater than 77")  
print(salary[salary>77])  
print()  
  
*#USING Nan FOR MULTIPLE CONDITON (NaN --> Not a Number)*print("Salary using Multiple Condition")  
print(salary[(salary >75) & (salary <85)])  
print()

**Ouput**

Salary which is greater than 77

Kumar Ragav Purusoth Sathya

2022 NaN NaN NaN NaN

2023 NaN NaN 82.0 NaN

2024 NaN NaN 85.0 NaN

2025 85.0 82.0 90.0 80.0

Salary using Multiple Condition

Kumar Ragav Purusoth Sathya

2022 NaN NaN NaN NaN

2023 NaN NaN 82.0 NaN

2024 NaN 77.0 NaN NaN

2025 NaN 82.0 NaN 80.0

**USING at AND iat – FOR QUERY AND UPDATE**

**Code:**

*#USING at TO RETRIEVE A DATA BASED ON ROW AND COLUMN*print("Salary retrival using at....")  
print("Ragav - 2024 : ",salary.at[2024,'Ragav'])  
print("Kumar - 2024 : ",salary.at[2024,'Kumar'])  
print()  
print("Salary retrival using iat....")  
print("Sathya - 2024 : ",salary.iat[2,3])  
print("Purusoth - 2024 : ",salary.iat[2,2])  
print()  
  
*#OVERRIDE THE EXISTING DATA USING at*print("Salary retrival using at before update....")  
print("Ragav - 2025 : ",salary.at[2025,'Ragav'])  
print("Kumar - 2025 : ",salary.at[2025,'Kumar'])  
salary.at[2025,'Kumar'] = 90  
salary.at[2025,'Ragav'] = 85  
print("Salary retrival using at after update....")  
print("Ragav - 2025 : ",salary.at[2025,'Ragav'])  
print("Kumar - 2025 : ",salary.at[2025,'Kumar'])  
print()  
*#OVERRIDE THE EXISTING DATA USING iat*print("Salary retrival using iat before update....")  
print("Sathya - 2025 : ",salary.at[2025,'Sathya'])  
print("Purusoth - 2025 : ",salary.at[2025,'Kumar'])  
salary.iat[3,3] = 80  
salary.iat[3,2] = 85  
print("Salary retrival using at after update....")  
print("Sathya - 2025 : ",salary.at[2025,'Sathya'])  
print("Purusoth - 2025 : ",salary.at[2025,'Purusoth'])  
print()

**Output**

Salary retrival using at....

Ragav - 2024 : 77

Kumar - 2024 : 75

Salary retrival using iat....

Sathya - 2024 : 75

Purusoth - 2024 : 85

Salary retrival using at before update....

Ragav - 2025 : 82

Kumar - 2025 : 85

Salary retrival using at after update....

Ragav - 2025 : 85

Kumar - 2025 : 90

Salary retrival using iat before update....

Sathya - 2025 : 80

Purusoth - 2025 : 90

Salary retrival using at after update....

Sathya - 2025 : 80

Purusoth - 2025 : 85

**Code:**

print("...........Dataframes values ...............")  
print("Mean Value for the All Employees")  
print(salary.mean())  
print()  
pd.set\_option('display.precision',2)  
print("Describe for the All Employees")  
print(salary.describe())  
print()  
print("Transpose the values - Rows to Columns and Columns to Rows")  
print(salary.T)

**Output:**

...........Dataframes values ...............

Mean Value for the All Employees

Kumar 73.50

Ragav 74.75

Purusoth 81.75

Sathya 72.50

dtype: float64

Describe for the All Employees

Kumar Ragav Purusoth Sathya

count 4.00 4.00 4.00 4.00

mean 73.50 74.75 81.75 72.50

std 12.23 8.42 4.72 6.45

min 62.00 65.00 75.00 65.00

25% 65.75 70.25 80.25 68.75

50% 71.00 74.50 83.50 72.50

75% 78.75 79.00 85.00 76.25

max 90.00 85.00 85.00 80.00

Transpose the values - Rows to Columns and Columns to Rows

2022 2023 2024 2025

Kumar 62 67 75 90

Ragav 65 72 77 85

Purusoth 75 82 85 85

Sathya 65 70 75 80

**USING SORT\_INDEX, AXIS, ASCENDING**

**Code:**

print("...........Dataframes values ...............")  
print("Default Dataframe Value - salary")  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
print(salary)  
print()  
print("ROW INNDEX SORTING")  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
print("Sort the Dataframe")  
print(salary.sort\_index())  
print()  
print("Sort the Dataframe - Descending Order")  
print(salary.sort\_index(ascending=False))  
print()  
print("COL INNDEX SORTING")  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
print("Sort the Dataframe (using axis)")  
print(salary.sort\_index(axis=1))  
print("Sort the Dataframe - Descending Order (using axis and ascending)")  
print(salary.sort\_index(axis=1,ascending=False))

**Output:**

...........Dataframes values ...............

Default Dataframe Value - salary

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Kumar Ragav Purusoth Sathya

2022 62 65 75 65

2023 67 72 82 70

2024 75 77 85 75

2025 90 85 85 80

ROW INNDEX SORTING

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sort the Dataframe

Kumar Ragav Purusoth Sathya

2022 62 65 75 65

2023 67 72 82 70

2024 75 77 85 75

2025 90 85 85 80

Sort the Dataframe - Descending Order

Kumar Ragav Purusoth Sathya

2025 90 85 85 80

2024 75 77 85 75

2023 67 72 82 70

2022 62 65 75 65

COL INNDEX SORTING

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sort the Dataframe (using axis)

Kumar Purusoth Ragav Sathya

2022 62 75 65 65

2023 67 82 72 70

2024 75 85 77 75

2025 90 85 85 80

Sort the Dataframe - Descending Order (using axis and ascending)

Sathya Ragav Purusoth Kumar

2022 65 65 75 62

2023 70 72 82 67

2024 75 77 85 75

2025 80 85 85 90

**READING / WRITING DATA IN THE CSV FILE USING DATAFRAME**

**Code:**

import pandas as pd  
  
*#CREATE A DATAFRAME AND READ THE CSV FILE.*employee\_df = pd.read\_csv('employee.csv')  
print(employee\_df)  
print()  
*#IF THE COLUMN HEADINGIS NOT THERE THEN WE CAN PROVIDE AS BELOW  
#emp1 = pd.read\_csv('employee.csv', names=['Employee\_FullName','Employee\_Location','Employee\_Email','Employee\_Company'])  
#print(emp1)  
  
#WRITE A DATAFRAME AND READ THE CSV FILE.*employee\_df.to\_csv('employee1.csv',index=False)

**Output:**

emp\_name emp\_location emp\_email emp\_company

0 Kamal Kumar Dublin kk\_test@gmail.com BNY Mellon

1 Ragav London rg\_text@gmail.com TCS

2 Purusoth Dubai pr\_test@gmail.com Al Jaz Tech

3 Kingsly Pune kg\_test@gmail.com Scope

4 Anwar Seattle an\_test@gmail.com Royal Bank

**File Created**

A screenshot of a computer

Description automatically generated

**QUIZ**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated