Experiment No:1 Pandas Installation.

Aim: To install Pandas.

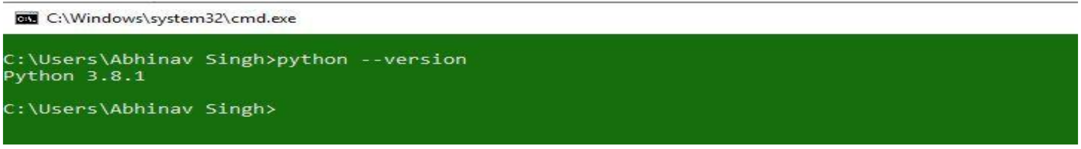
Description:

Pandas in Python : Pandas in Python is a package that is written for data analysis and manipulation. Pandas offer various operations and data structures to perform numerical data manipulations and time series. Pandas is an open-source library that is built over Numpy libraries. Pandas library is known for its high productivity and high performance. Pandas is popular because it makes importing and analyzing data much easier.

Pandas programs can be written on any plain text editor like notepad, notepad++, or anything of that sort and saved with a .py extension. To begin with, writing Pandas Codes and performing various intriguing and useful operations, one must have Python installed on their System. This can be done by following the step by step instructions provided below:

If Python already exists: To check if your device is pre-installed with Python or not, just go to the Command line(search for cmd in the Run dialog (⊞ + R).
Now run the following command:
python --version

If Python is already installed, it will generate a message with the Python version available.



C:\Windows\system32\cmd.exe

```
C:\Users\Abhinav Singh>python --version
Python 3.8.1

C:\Users\Abhinav Singh>
```

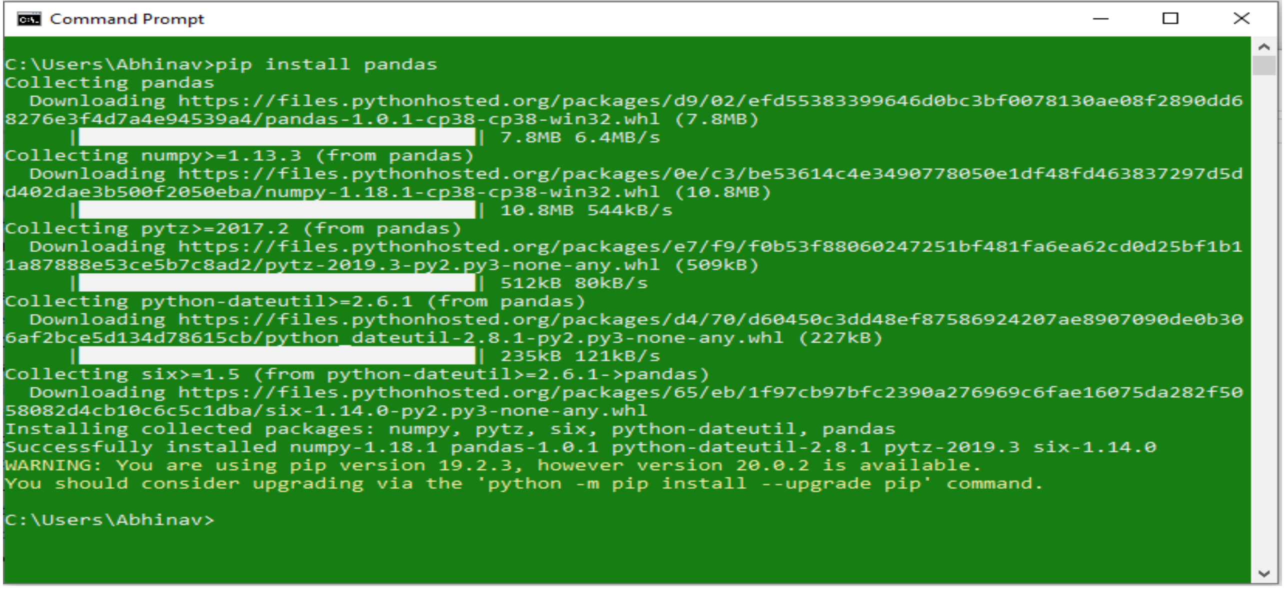
Downloading and Installing Pandas : Pandas can be installed in multiple ways on Windows and on Linux. Various different ways are listed below:

Windows

Python Pandas can be installed on Windows in two ways:

- Using pip
- Using Anaconda

Install Pandas using pip : PIP is a package management system used to install and manage software packages/libraries written in Python. These files are stored in a large “on-line repository” termed as Python Package Index (PyPI).
Pandas can be installed using PIP by the use of the following command: pip install pandas



Command Prompt

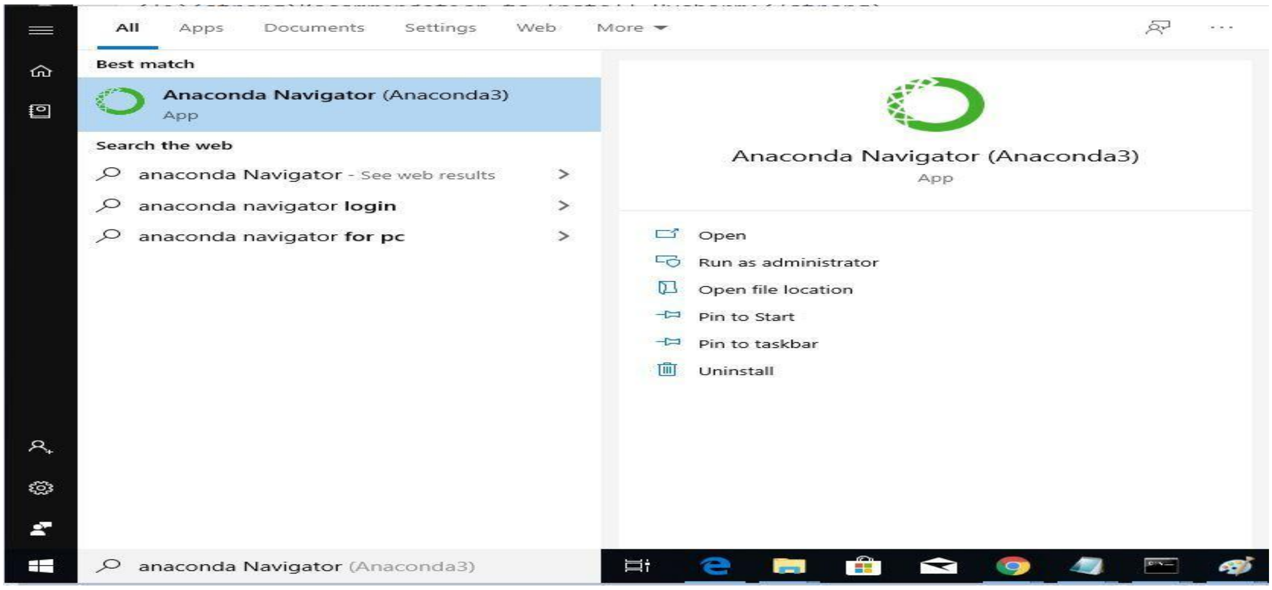
```
C:\Users\Abhinav>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/d9/02/efd55383399646d0bc3bf0078130ae08f2890dd6
8276e3f4d7a4e94539a4/pandas-1.0.1-cp38-cp38-win32.whl (7.8MB)
     |████████████████████████████████| 7.8MB 6.4MB/s
Collecting numpy>=1.13.3 (from pandas)
  Downloading https://files.pythonhosted.org/packages/0e/c3/be53614c4e3490778050e1df48fd463837297d5d
d402dae3b500f2050eba/numpy-1.18.1-cp38-cp38-win32.whl (10.8MB)
     |████████████████████████████████| 10.8MB 544kB/s
Collecting pytz>=2017.2 (from pandas)
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b1
1a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl (509kB)
     |████████████████████████████████| 512kB 80kB/s
Collecting python-dateutil>=2.6.1 (from pandas)
  Downloading https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b30
6af2bce5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any.whl (227kB)
     |████████████████████████████████| 235kB 121kB/s
Collecting six>=1.5 (from python-dateutil>=2.6.1->pandas)
  Downloading https://files.pythonhosted.org/packages/65/eb/1f97cb97bfc2390a276969c6fae16075da282f50
58082d4cb10c6c5c1dba/six-1.14.0-py2.py3-none-any.whl
Installing collected packages: numpy, pytz, six, python-dateutil, pandas
Successfully installed numpy-1.18.1 pandas-1.0.1 python-dateutil-2.8.1 pytz-2019.3 six-1.14.0
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Abhinav>
```

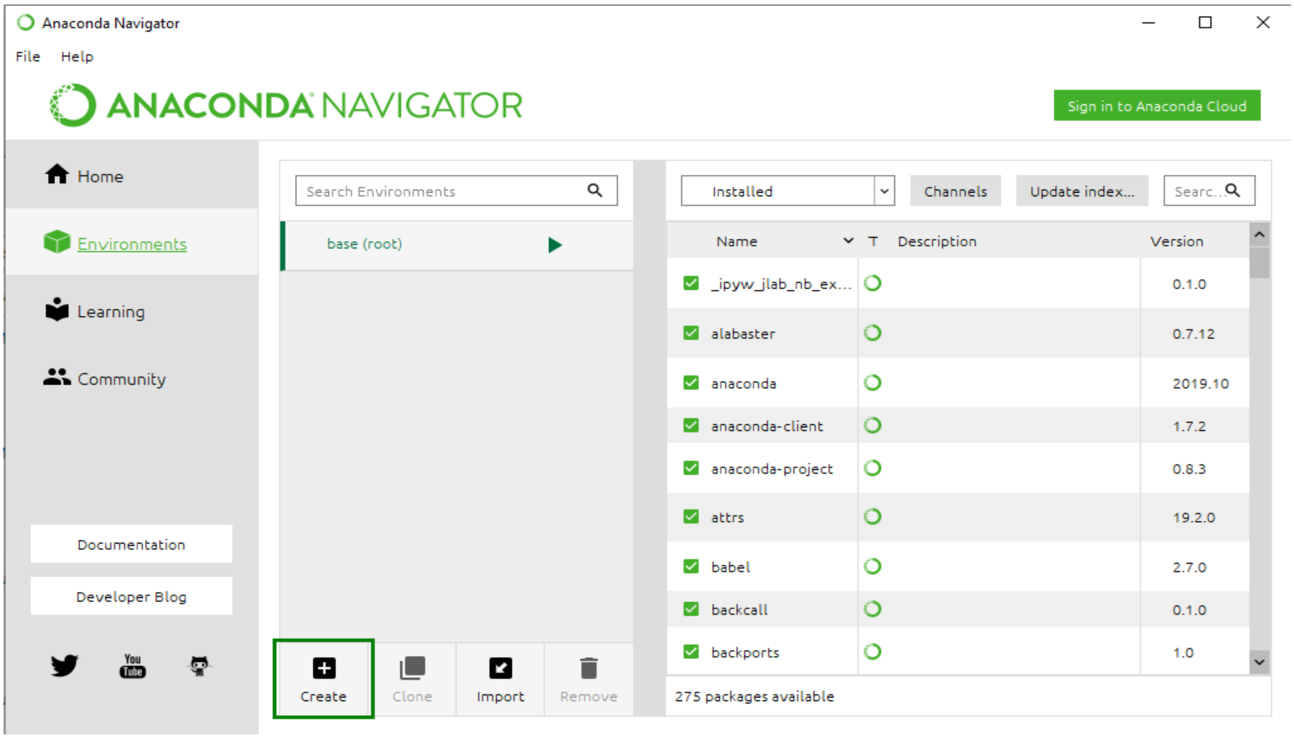
Install Pandas using Anaconda:

Anaconda is open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. If your system is not pre-equipped with Anaconda Navigator.

Steps to Install Pandas using Anaconda Navigator:
Step 1: Search for Anaconda Navigator in Start Menu and open it.



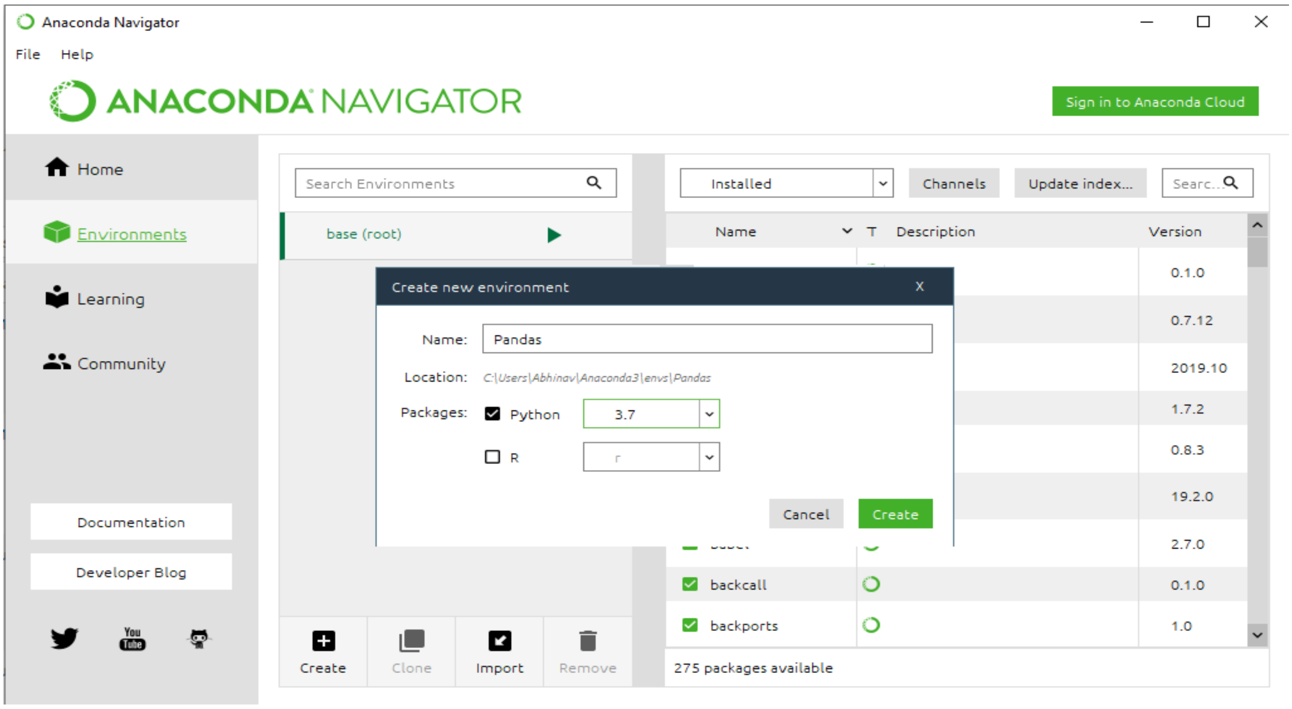
Step 2: Click on the Environment tab and then click on the create button to create a new Pandas Environment.



Name	T	Description	Version
☑ _ipyw_jlab_nb_ex...	○		0.1.0
☑ alabaster	○		0.7.12
☑ anaconda	○		2019.10
☑ anaconda-client	○		1.7.2
☑ anaconda-project	○		0.8.3
☑ attrs	○		19.2.0
☑ babel	○		2.7.0
☑ backcall	○		0.1.0
☑ backports	○		1.0

275 packages available

Step 3: Give a name to your Environment, e.g. Pandas and then choose a python version to run in the environment. Now click on the **Create** button to create Pandas Environment.



Create new environment

Name: Pandas

Location: C:\Users\Abhinav\Anaconda3\envs\Pandas

Packages: ☑ Python 3.7

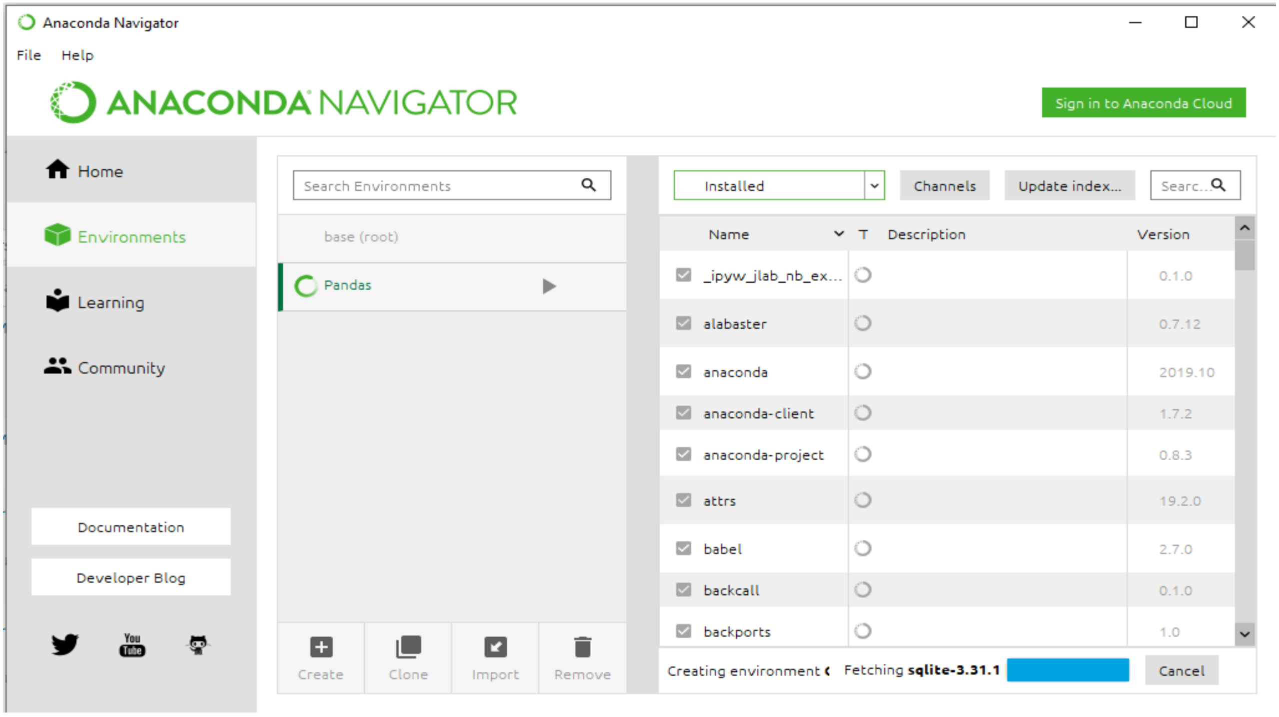
□ R r

Cancel Create

Name	T	Description	Version
			0.1.0
			0.7.12
			2019.10
			1.7.2
			0.8.3
			19.2.0
			2.7.0
☑ backcall	○		0.1.0
☑ backports	○		1.0

275 packages available

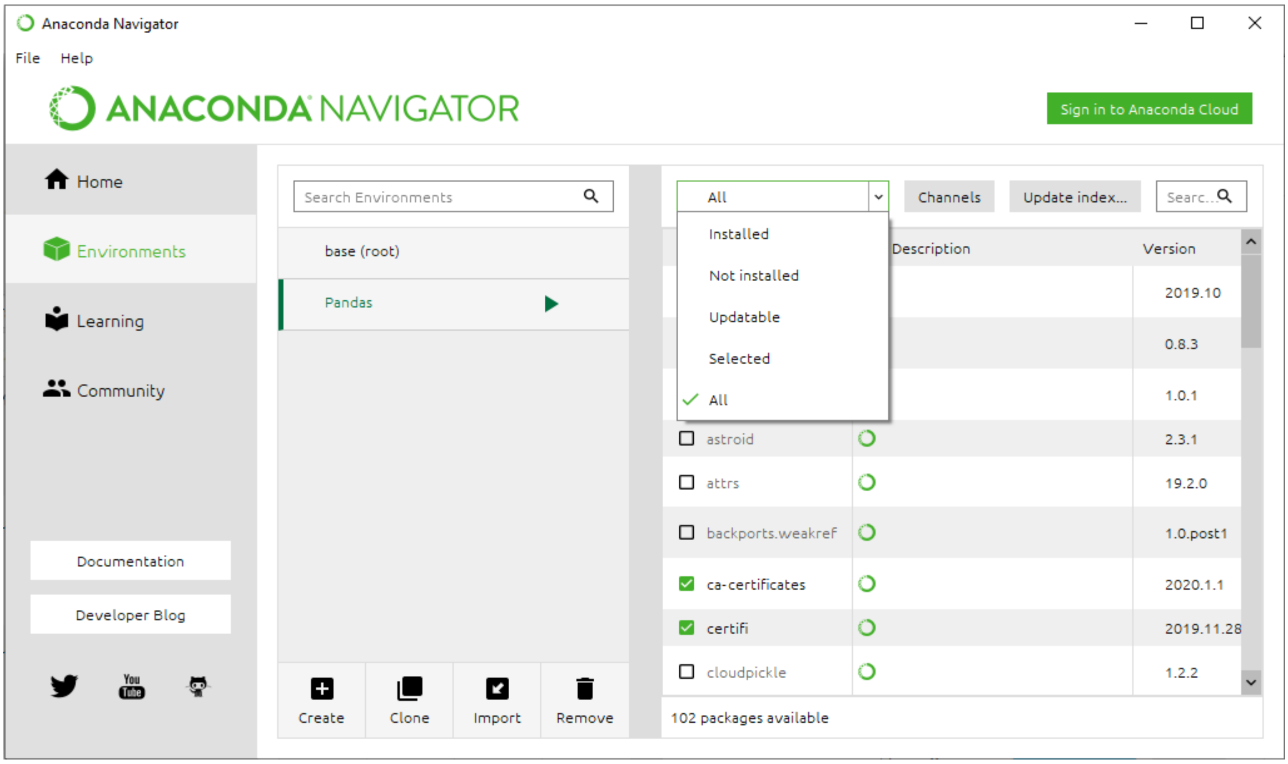
Step 4: Now click on the **Pandas Environment** created to activate it.



Name	T	Description	Version
☑ _ipyw_jlab_nb_ex...	○		0.1.0
☑ alabaster	○		0.7.12
☑ anaconda	○		2019.10
☑ anaconda-client	○		1.7.2
☑ anaconda-project	○		0.8.3
☑ attrs	○		19.2.0
☑ babel	○		2.7.0
☑ backcall	○		0.1.0
☑ backports	○		1.0

Creating environment (Fetching **sqlite-3.31.1**

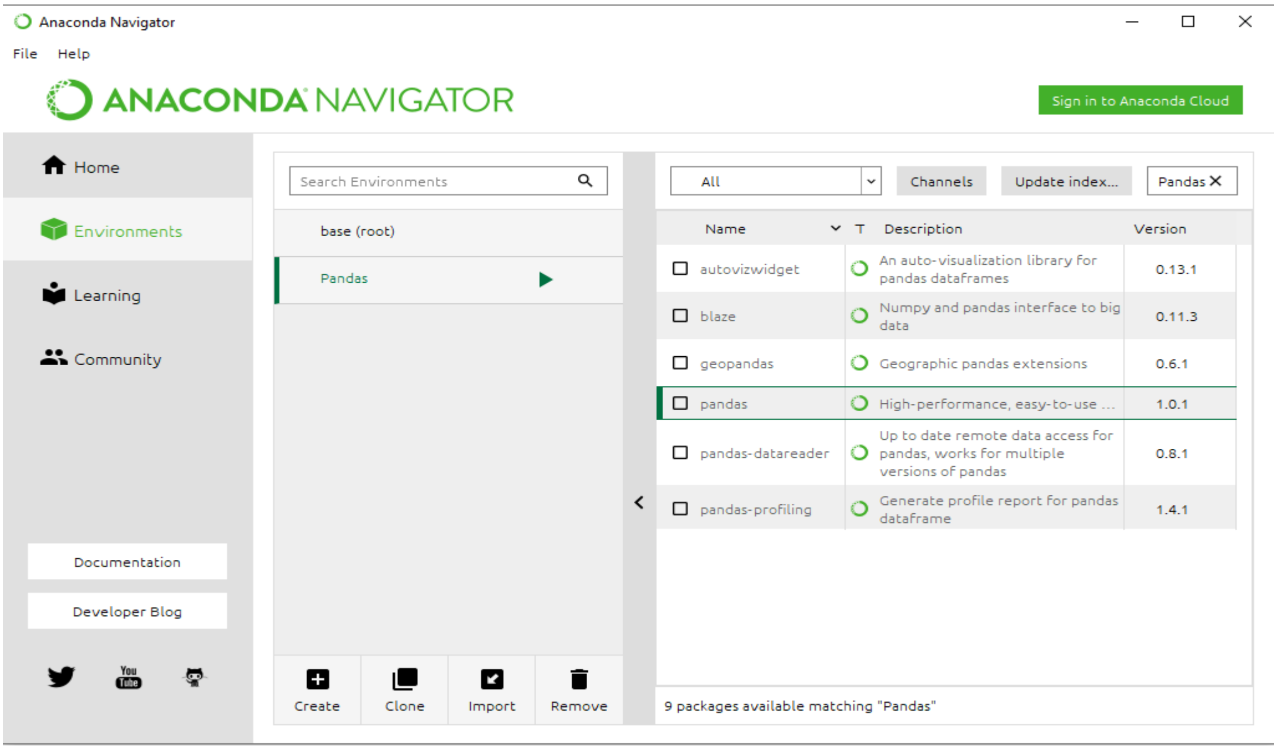
Step 5: In the list above package names, select **All** to filter all the packages.



Name	Description	Version
		2019.10
		0.8.3
		1.0.1
□ astroid	○	2.3.1
□ attrs	○	19.2.0
□ backports.weakref	○	1.0.post1
☑ ca-certificates	○	2020.1.1
☑ certifi	○	2019.11.28
□ cloudpickle	○	1.2.2

102 packages available

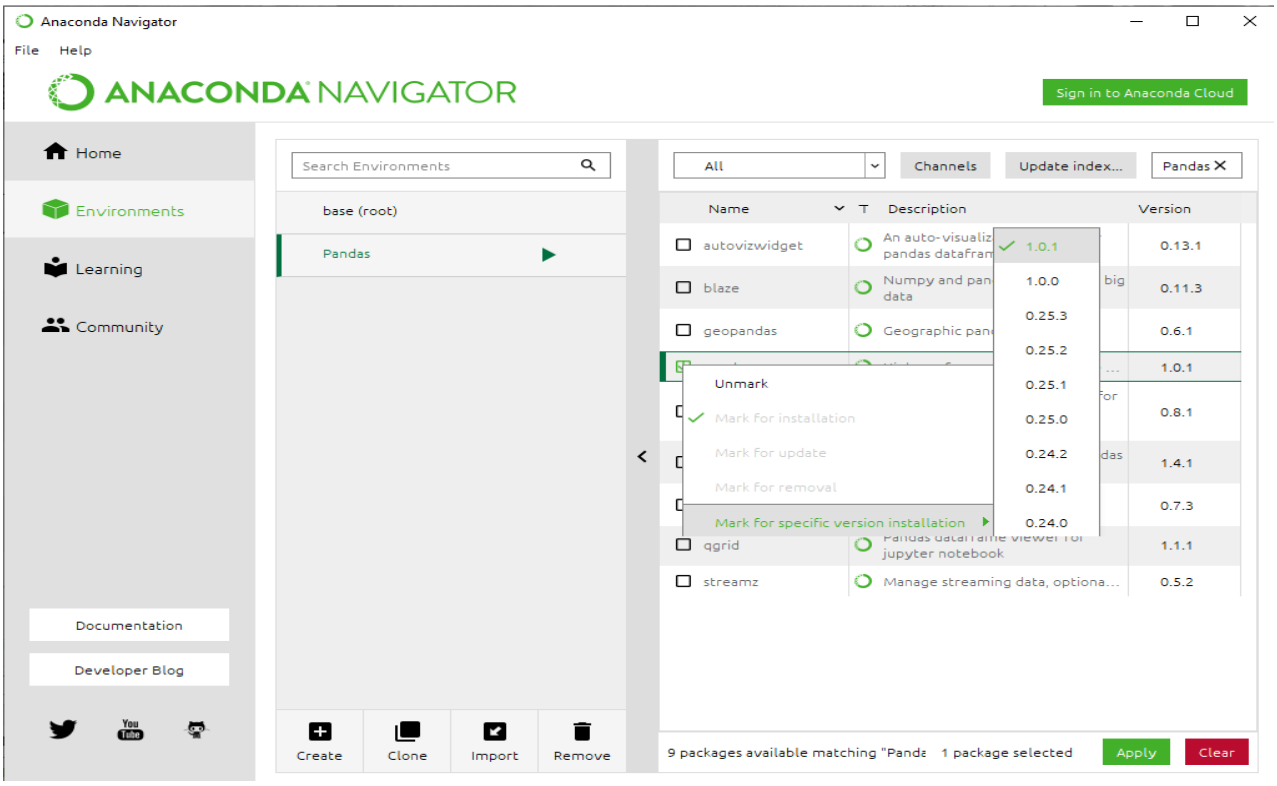
Step 6: Now in the Search Bar, look for '**Pandas**'. Select the **Pandas package** for Installation.



Name	T	Description	Version
□ autovizwidget	○	An auto-visualization library for pandas dataframes	0.13.1
□ blaze	○	Numpy and pandas interface to big data	0.11.3
□ geopandas	○	Geographic pandas extensions	0.6.1
□ pandas	○	High-performance, easy-to-use ...	1.0.1
□ pandas-datareader	○	Up to date remote data access for pandas, works for multiple versions of pandas	0.8.1
□ pandas-profiling	○	Generate profile report for pandas dataframe	1.4.1

9 packages available matching "Pandas"

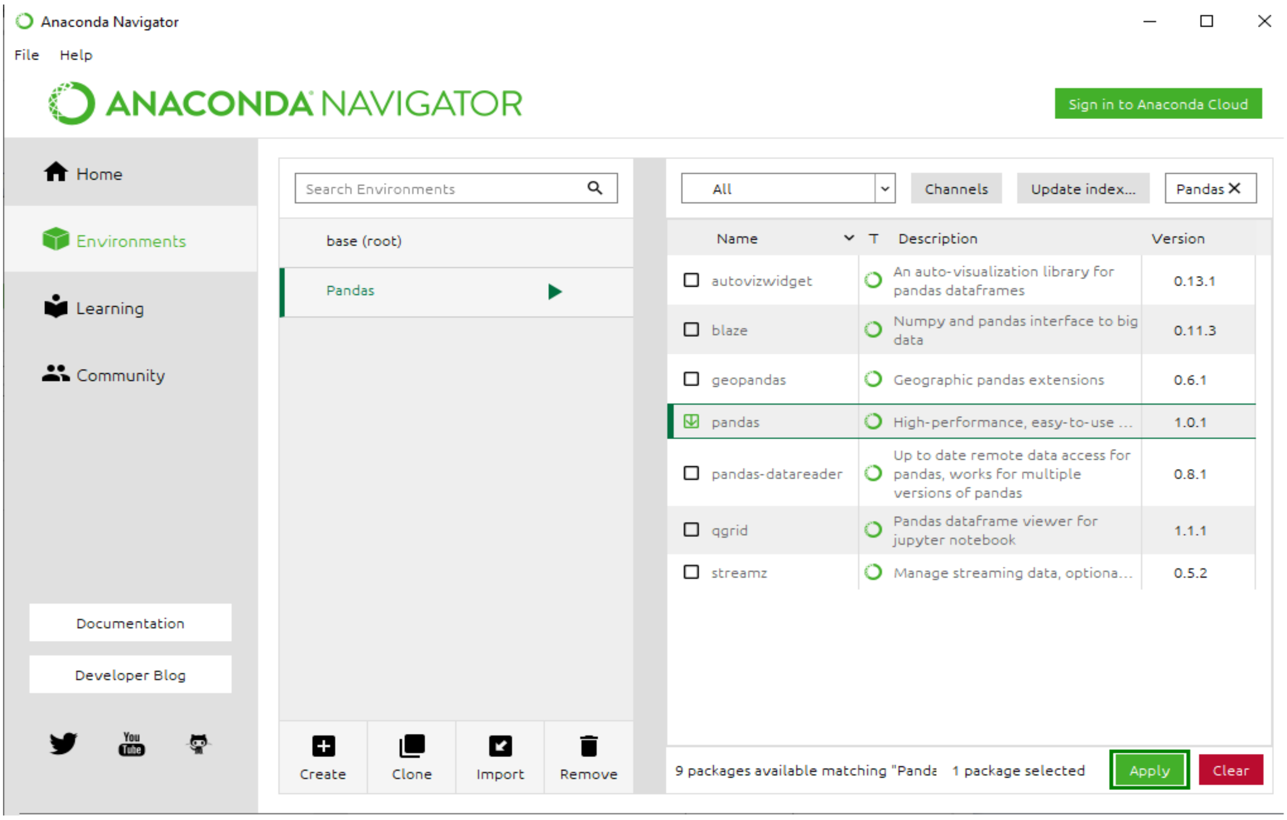
Step 7: Now Right Click on the checkbox given before the name of the package and then go to '**Mark for specific version installation**'. Now select the version that you want to install.



Name	T	Description	Version
□ autovizwidget	○	An auto-visualiz… pandas datafram…	0.13.1
□ blaze	○	Numpy and pan… big data	0.11.3
□ geopandas	○	Geographic pan…	0.6.1
☑		…	1.0.1
□		…for	0.8.1
□		…das	1.4.1
□			0.7.3
□ qgrid	○	Pandas dataframe viewer for jupyter notebook	1.1.1
□ streamz	○	Manage streaming data, optiona…	0.5.2

9 packages available matching "Panda 1 package selected

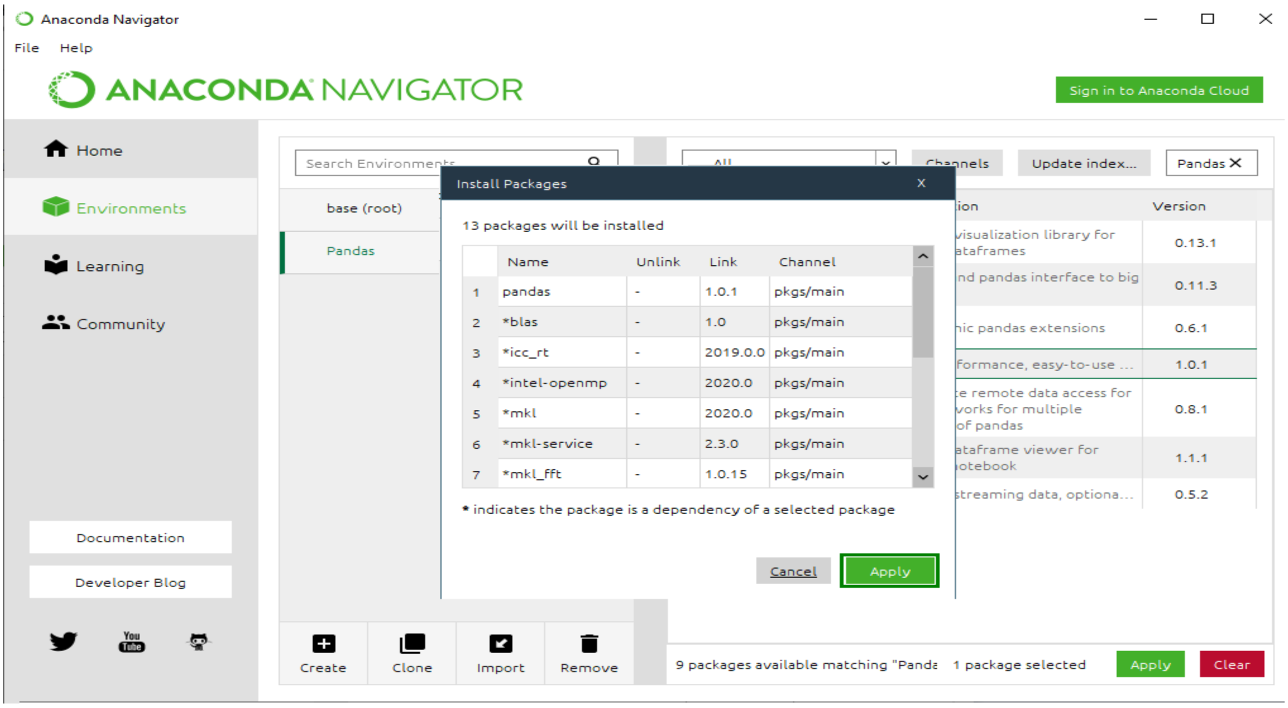
Step 8: Click on the **Apply** button to install the Pandas Package.



Name	T	Description	Version
□ autovizwidget	○	An auto-visualization library for pandas dataframes	0.13.1
□ blaze	○	Numpy and pandas interface to big data	0.11.3
□ geopandas	○	Geographic pandas extensions	0.6.1
☑ pandas	○	High-performance, easy-to-use …	1.0.1
□ pandas-datareader	○	Up to date remote data access for pandas, works for multiple versions of pandas	0.8.1
□ qgrid	○	Pandas dataframe viewer for jupyter notebook	1.1.1
□ streamz	○	Manage streaming data, optiona…	0.5.2

9 packages available matching "Panda 1 package selected

Step 9: Finish the Installation process by clicking on the **Apply** button.



Anaconda Navigator

File   Help

ANACONDA NAVIGATOR   Sign in to Anaconda Cloud

Home | Environments | Learning | Community | Documentation | Developer Blog

Search Environments

base (root)

Pandas

Install Packages

13 packages will be installed

	Name	Unlink	Link	Channel
1	pandas	-	1.0.1	pkgs/main
2	*blas	-	1.0	pkgs/main
3	*icc_rt	-	2019.0.0	pkgs/main
4	*intel-openmp	-	2020.0	pkgs/main
5	*mkl	-	2020.0	pkgs/main
6	*mkl-service	-	2.3.0	pkgs/main
7	*mkl_fft	-	1.0.15	pkgs/main

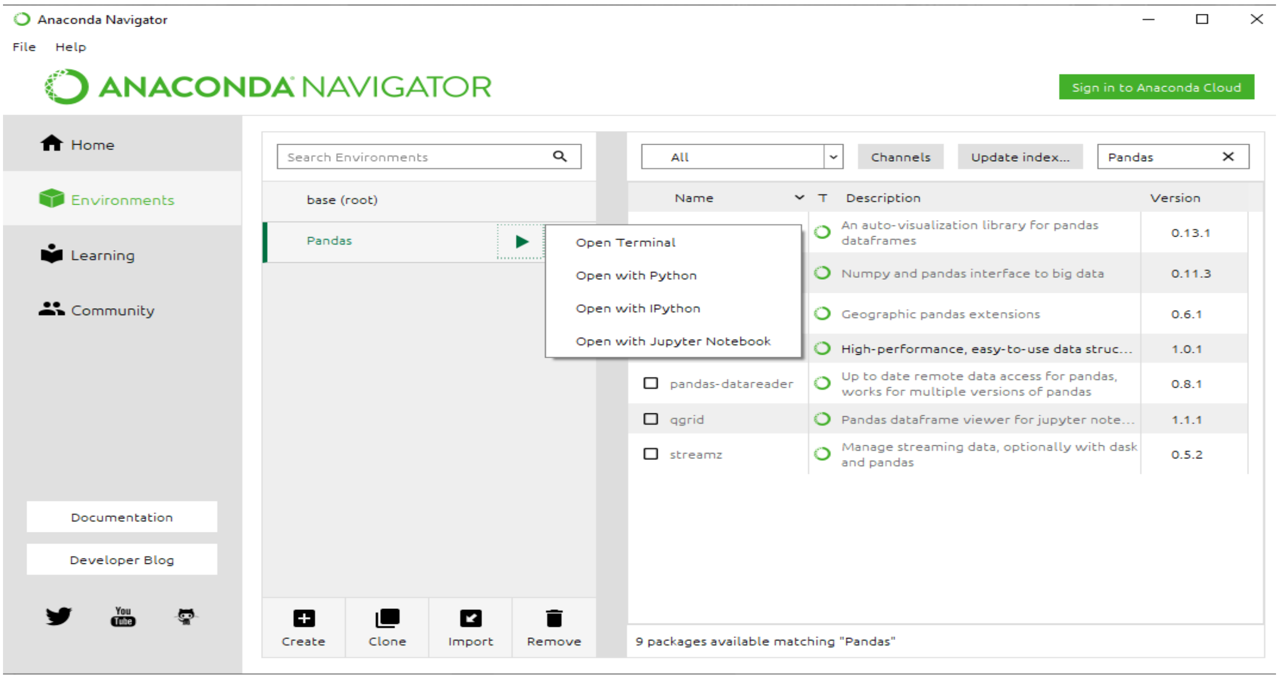
* indicates the package is a dependency of a selected package

Cancel   Apply

Create | Clone | Import | Remove

9 packages available matching "Pandas   1 package selected   Apply   Clear

Step 10: Now to open the Pandas Environment, click on the **Green Arrow** on the right of package name and select the Console with which you want to begin your Pandas programming.



Anaconda Navigator

File   Help

ANACONDA NAVIGATOR   Sign in to Anaconda Cloud

Home | Environments | Learning | Community | Documentation | Developer Blog

Search Environments

base (root)

Pandas ▶

- Open Terminal
- Open with Python
- Open with IPython
- Open with Jupyter Notebook

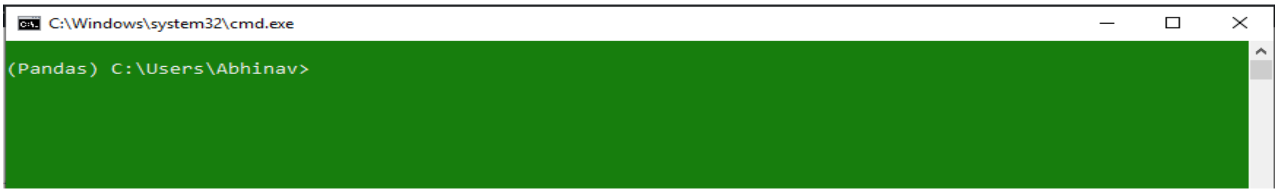
All   Channels   Update index...   Pandas ✕

Name	T	Description	Version
	○	An auto-visualization library for pandas dataframes	0.13.1
	○	Numpy and pandas interface to big data	0.11.3
	○	Geographic pandas extensions	0.6.1
	○	High-performance, easy-to-use data struc...	1.0.1
□ pandas-datareader	○	Up to date remote data access for pandas, works for multiple versions of pandas	0.8.1
□ qgrid	○	Pandas dataframe viewer for jupyter note...	1.1.1
□ streamz	○	Manage streaming data, optionally with dask and pandas	0.5.2

Create | Clone | Import | Remove

9 packages available matching "Pandas"

Pandas Terminal Window:



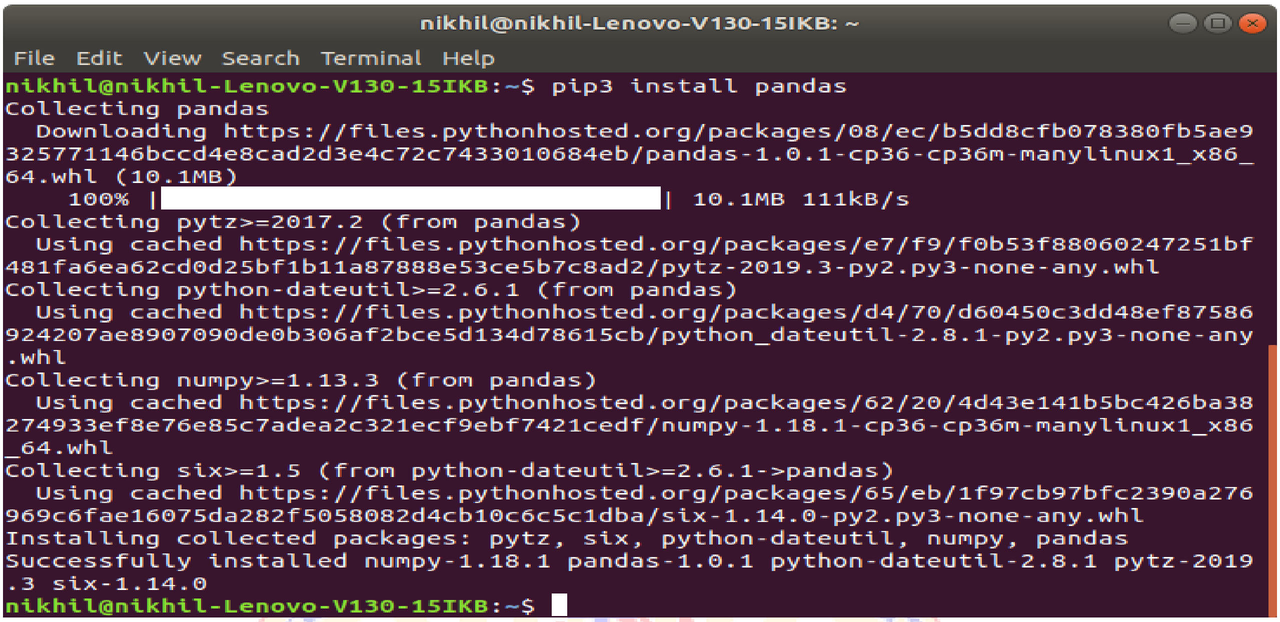
C:\Windows\system32\cmd.exe

```
(Pandas) C:\Users\Abhinav>
```

Linux

To install Pandas on Linux, just type the following command in the Terminal Window and press Enter. Linux will automatically download and install the packages and files required to run Pandas Environment in Python:

pip3 install pandas


nikhil@nikhil-Lenovo-V130-15IKB: ~
File Edit View Search Terminal Help

```
nikhil@nikhil-Lenovo-V130-15IKB:~$ pip3 install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/08/ec/b5dd8cfb078380fb5ae9
325771146bccd4e8cad2d3e4c72c7433010684eb/pandas-1.0.1-cp36-cp36m-manylinux1_x86_
64.whl (10.1MB)
    100% |████████████████████████████████| 10.1MB 111kB/s
Collecting pytz>=2017.2 (from pandas)
  Using cached https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf
481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl
Collecting python-dateutil>=2.6.1 (from pandas)
  Using cached https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586
924207ae8907090de0b306af2bce5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any
.whl
Collecting numpy>=1.13.3 (from pandas)
  Using cached https://files.pythonhosted.org/packages/62/20/4d43e141b5bc426ba38
274933ef8e76e85c7adea2c321ecf9ebf7421cedf/numpy-1.18.1-cp36-cp36m-manylinux1_x86
_64.whl
Collecting six>=1.5 (from python-dateutil>=2.6.1->pandas)
  Using cached https://files.pythonhosted.org/packages/65/eb/1f97cb97bfc2390a276
969c6fae16075da282f5058082d4cb10c6c5c1dba/six-1.14.0-py2.py3-none-any.whl
Installing collected packages: pytz, six, python-dateutil, numpy, pandas
Successfully installed numpy-1.18.1 pandas-1.0.1 python-dateutil-2.8.1 pytz-2019
.3 six-1.14.0
nikhil@nikhil-Lenovo-V130-15IKB:~$ 
```

Experiment No: 2 Creating DataFrames.

Aim: To create DataFrames using python pandas.

Description :

A Data Frame is a two-dimension collection of data. It is a data structure where data is stored in tabular form. Datasets are arranged in rows and columns; we can store multiple datasets in the data frame. We can perform various arithmetic operations, such as adding column/row selection and columns/rows in the data frame.We can import the DataFrames from the external storage; these storages can be referred to as the SQL . Database, CSV file, and an Excel file. We can also use the lists, dictionary, and from a list of dictionary, etc.

First, we need to install the pandas library into the Python environment.

An empty dataframe

We can create a basic empty Dataframe. The dataframe constructor needs to be called to create the DataFrame.

Program :

```
import pandas as pd

df = pd.DataFrame()

print(df)
```

output :

```
Empty DataFrame
Columns: []
Index: []
```

Method - 2: Create a dataframe using List

We can create dataframe using a single list or list of lists.

Program :

```
import pandas as pd

print("Method - 2: Create a dataframe using List")

list = ['Java', 'Python', 'C', 'C++',

        'JavaScript', 'Swift', 'Go']

df = pd.DataFrame(list)

print(dframe)
```

output :

```
Method - 2: Create a dataframe using List
          0
0       Java
1     Python
2          C
3        C++
4  JavaScript
5       Swift
6          Go
```

Method - 3: Create Dataframe from dict of ndarray/lists

The dict of ndarray/lists can be used to create a dataframe, all the **ndarray** must be of the same length. The index will be a range(n) by default; where n denotes the array length.

Program :

```
import pandas as pd

print("Method - 3: Create Dataframe from dict of ndarray/lists")

data = {'Name': ['Sanjana', 'Jahnavi', 'Ambica', 'Pavani'], 'Age': [20, 21, 19, 18]}

df = pd.DataFrame(data)

print(df)
```

output :

```
Method - 3: Create Dataframe from dict of ndarray/lists
      Name  Age
0  Sanjana   20
1  Jahnavi   21
2   Ambica   19
3   Pavani   18
```

Method - 4: Create a indexes Dataframe using arrays

In the above code, we have defined the column name with the various car names and their ratings. We used the array to create indexes.

Program:

```
import pandas as pd

print('Method - 4: Create a indexes Dataframe using arrays')

data = {'Name':['Honda','Tvs','Jupiter','Mahendhra'], 'Ratings':[9.0, 8.0, 5.0, 3.0]}

df = pd.DataFrame(data, index =['position1', 'position2', 'position3', 'position4'])

print(df)
```

output:

```
Method - 4: Create a indexes Dataframe using arrays
                Name  Ratings
position1      Honda      9.0
position2        Tvs      8.0
position3    Jupiter      5.0
position4  Mahendhra      3.0
```

Method - 5: Create Dataframe from list of dicts:

We can pass the lists of dictionaries as input data to create the Pandas dataframe. The column names are taken as keys by default.

Program:

```
import pandas as pd

print('Method - 5: Create Dataframe from list of dicts')

data = [{'A': 10, 'B': 20, 'C':30}, {'d':100, 'e': 200, 'f': 300}]

df = pd.DataFrame(data)

print(df)
```

output:

```
Method - 5: Create Dataframe from list of dicts
      A     B     C      d      e      f
0  10.0  20.0  30.0    NaN    NaN    NaN
1   NaN   NaN   NaN  100.0  200.0  300.0
```

Method - 6: Create Dataframe from Dicts of series

The dictionary can be passed to create a dataframe. We can use the Dicts of series where the subsequent index is the union of all the series of passed index value.

Program:

```
import pandas as pd

print('Method - 6: Create Dataframe from Dicts of series')

d = {'Electronics' : pd.Series([97, 56, 87, 45], index =['Ajay', 'Raghu', 'Vijay', 'Arjun']),

  'Civil' : pd.Series([97, 88, 44, 96], index =['Ajay', 'Raghu', 'Vijay', 'Arjun'])}

df=pd.DataFrame(d)

print(df)
```

output:

```
Method - 6: Create Dataframe from Dicts of series
       Electronics  Civil
Ajay            97     97
Raghu           56     88
Vijay           87     44
Arjun           45     96
```

Experiment No: 3 Pandas DataSeries:

3. i)Write a Pandas program to create and display a one-dimensional array-like object containing an array of data using Pandas module.

Aim: To write a Pandas program to create and display a one-dimensional array-like object containing an array of data using Pandas module.

Description:

Series() is a function present in the Pandas library that creates a one-dimensional array and can hold any type of objects or data in it. Syntax : pandas.Series(parameters)

Parameters :

data : Contains data stored in Series.

index : Values must be hashable and have the same length as data.

dtype : Data type for the output Series.

name : The name to give to the Series.

copy : Copy input data.

Returns : An object of class Series

Program:

```
import pandas as pd

ds=pd.Series([1,2,3,4,5])

print(ds)
```

Output:

```
0    1

1    2

2    3

3    4

4    5

dtype: int64
```

3. ii) Write a Pandas program to convert a Panda module Series to Python list and it's type.

Aim: To write a Pandas program to convert a Panda module Series to Python list and it's type.

Description:

Pandas tolist() is used to convert a series to list. Initially the series is of type pandas.core.series.Series and applying tolist() method, it is converted to list data type.

Syntax: Series.tolist()

Return type: Converted series into List

Program:

```
import pandas as pd
ds=pd.Series([1,2,3,4,5])
print("Pandas series and its type")
print(ds)
print(type(ds))
print("converting pandas series into list")
list=ds.tolist()
print(list)
print(type(list))
```

Output:

```
Pandas series and its type
0    1
1    2
2    3
3    4
4    5
dtype: int64
<class 'pandas.core.series.Series'>
converting pandas series into list
[1, 2, 3, 4, 5]
<class 'list'>
```

3. iii)Write a Pandas program to add, subtract, multiple and divide two Pandas Series.

Aim: To write a Pandas program to add, subtract, multiple and divide two Pandas Series.

Description: To perform basic arithmetic operations like addition, subtraction, multiplication, and division on 2 Pandas Series.

For all the 4 operations we will follow the basic algorithm :

1. Import the Pandas module.
2. Create 2 Pandas Series objects.
3. Perform the required arithmetic operation using the respective arithmetic operator between the 2 Series and assign the result to another Series.
4. Display the resultant Series.

Program:

```
import pandas as pd

ds1=pd.Series([2,3,4,5])

ds2=pd.Series([1,2,3,4])

print("Data Series 1")

print(ds1)

print("Data Series 2")

print(ds2)

print("Addition of two Series")

print(ds1+ds2)

print("Subtraction of two Series")

print(ds1-ds2)

print("Multiplication of two Series")

print(ds1*ds2)

print("Division of two Series")

print(ds1/ds2)
```

Output:

```
Data Series 1

0    2

1    3

2    4

3    5

dtype: int64

Data Series 2
```

```
0    1
1    2
2    3
3    4
dtype: int64
Addition of two Series
0    3
1    5
2    7
3    9
dtype: int64
Subtraction of two Series
0    1
1    1
2    1
3    1
dtype: int64
Multiplication of two Series
0     2
1     6
2    12
3    20
dtype: int64
Division of two Series
0    2.000000
1    1.500000
2    1.333333
3    1.250000
dtype: float64
```

3. iv) Write a Pandas program to convert a NumPy array to a Pandas series.

Aim: To write a Pandas program to convert a NumPy array to a Pandas series.

Description:

Series() is a function present in the Pandas library that creates a one-dimensional array and can hold any type of objects or data in it. In this article, let us learn the syntax, create and display one-dimensional array-like object containing an array of data using Pandas library.

Syntax : pandas.Series(parameters)

Program:

```
import numpy as np

import pandas as pd

np_array=np.array([1,2,3,4,5])

print("Numpy Array")

print(np_array)

new_series=pd.Series(np_array)

print("Converted Pandas Series")

print(new_series)
```

Output:

```
Numpy Array

[1 2 3 4 5]

Converted Pandas Series

0    1

1    2

2    3

3    4

4    5

dtype: int64
```

Experiment No : 4 Pandas DataFrames

Consider Sample Python dictionary data and list labels:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

4. i)Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

Aim : To write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

Description : Creating a DataFrame from Dictionary with user-defined indexes.

Program:

```
import pandas as pd

import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df=pd.DataFrame(exam_data,index=labels)

print(df)
```

Output:

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

4.ii)Write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame.

Aim : To write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame.

Description:

Pandas dataframe.replace() function is used to replace a string, regex, list, dictionary, series, number etc. from a dataframe. This is a very rich function as it has many variations. The most powerful thing about this function is that it can work with Python regex (regular expressions).

Syntax: DataFrame.replace(to_replace=None, value=None, inplace=False, limit=None, regex=False, method=’pad’, axis=None)

Program:

```
import pandas as pd

import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df=pd.DataFrame(exam_data,index=labels)

print("\n Changing name James to Suresh")

df['name']=df['name'].replace('James','Suresh')

print(df)
```

Output:

```
 Changing name James to Suresh
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d     Suresh    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

4.iii)Write a Pandas program to insert a new column in existing DataFrame.

Aim : To write a Pandas program to insert a new column in existing DataFrame.

Description : By declaring a new list as a column.

Program:

```
import pandas as pd

import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df=pd.DataFrame(exam_data,index=labels)

color = ['Red','Blue','Orange','Red','White','White','Blue','Green','Green','Red']

print("Method 1: By declaring a new list as a column. ")

df['color']=color

print(df)
```

output :

```
Method 1: By declaring a new list as a column.
        name  score  attempts qualify   color
a  Anastasia   12.5         1     yes     Red
b       Dima    9.0         3      no    Blue
c  Katherine   16.5         2     yes  Orange
d      James    NaN         3      no     Red
e      Emily    9.0         2      no   White
f    Michael   20.0         3     yes   White
g    Matthew   14.5         1     yes    Blue
h      Laura    NaN         1      no   Green
i      Kevin    8.0         2      no   Green
j      Jonas   19.0         1     yes     Red
```

4.iv)Write a Pandas program to get list from DataFrame column headers.

Aim : To write a Pandas program to get list from DataFrame column headers.

Description : column.values method returs an array of index.

Program:

```
import pandas as pd

import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',

'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df=pd.DataFrame(exam_data,index=labels)

print(df)

print(list(df.columns.values))
```

Output:

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
['name', 'score', 'attempts', 'qualify']
```

Experimnet No: 5. Pandas Index:

5. i)Write a Pandas program to display the default index and set a column as an Index in a given dataframe.

Aim: To write a Pandas program to display the default index and set a column as an Index in a given dataframe.

Description : Pandas reset_index() is a method to reset index of a Data Frame. reset_index() method sets a list of integer ranging from 0 to length of data as index.

Syntax:

DataFrame.reset_index(level=None, drop=False, inplace=False, col_level=0, col_fill=”)

Parameters:

level: int, string or a list to select and remove passed column from index.

drop: Boolean value, Adds the replaced index column to the data if False.

inplace: Boolean value, make changes in the original data frame itself if True.

col_level: Select in which column level to insert the labels.

col_fill: Object, to determine how the other levels are named.

Return type: DataFrame

Program :

```
import pandas as pd
df=pd.DataFrame({
    'rollno':['5D1','5D2','5D3','5D4','5D5'],
    'name':['Sanjana','Vijay','Madhuri','Ramya','Aisha'],
    'section':['A','A','B','B','C'],
    'date_of_birth':['22/02/2002','15/03/2003','08/12/2001','23/07/2002','14/06/2001'],
    'weight':[60,56,49,52,48],
    'id':['I1','I2','I3','I4','I5']})
print("Default Index :")
print(df)
print("\n id as new index :")
df1=df.set_index('id')
print(df1)
print("\n Reseting the index :")
df2=df1.reset_index(inplace=False)
print(df2)
```

Output :

```
Default Index :
  rollno    name section date_of_birth  weight  id
0    5D1  Sanjana       A    22/02/2002      60  I1
1    5D2    Vijay       A    15/03/2003      56  I2
2    5D3  Madhuri       B    08/12/2001      49  I3
3    5D4    Ramya       B    23/07/2002      52  I4
4    5D5    Aisha       C    14/06/2001      48  I5

 id as new index :
   rollno    name section date_of_birth  weight
id
```

```
I1    5D1  Sanjana       A    22/02/2002      60
I2    5D2    Vijay       A    15/03/2003      56
I3    5D3  Madhuri       B    08/12/2001      49
I4    5D4    Ramya       B    23/07/2002      52
I5    5D5    Aisha       C    14/06/2001      48

 Reseting the index :
   id rollno     name section date_of_birth  weight
0  I1    5D1  Sanjana       A    22/02/2002      60
1  I2    5D2    Vijay       A    15/03/2003      56
2  I3    5D3  Madhuri       B    08/12/2001      49
3  I4    5D4    Ramya       B    23/07/2002      52
4  I5    5D5    Aisha       C    14/06/2001      48
```

5. ii)Write a Pandas program to create an index labels by using 64-bit integers, using floating-point numbers in a given dataframe.

Aim: To write a Pandas program to create an index labels by using 64-bit integers, using floating-point numbers in a given dataframe.

Description:

class pandas.Index(*data=None*, *dtype=None*, *copy=False*, *name=None*, *tupleize_cols=True*, ***kwargs*)[source] . Immutable sequence used for indexing and alignment. The basic object storing axis labels for all pandas objects.

Parameters

data*array-like (1-dimensional)*

dtype*NumPy dtype (default: object)*: If dtype is None, we find the dtype that best fits the data. If an actual dtype is provided, we coerce to that dtype if it’s safe. Otherwise, an error will be raised.

Copy*bool*: Make a copy of input ndarray.

Name*object* : Name to be stored in the index.

tupleize_cols*bool (default: True)*: When True, attempt to create a MultiIndex if possible.

Program :

```
import pandas as pd
data={'rollno':['5D1','5D2','5D3','5D4','5D5'],
    'name':['Sanjana','Vijay','Madhuri','Ramya','Aisha'],
    'section':['A','A','B','B','C'],
    'date_of_birth':['22/02/2002','15/03/2003','08/12/2001','23/07/2002','14/06/2001'],
    'weight':[60,56,49,52,48],
    'id':['I1','I2','I3','I4','I5']}
index_int=[1,2,3,4,5]
df1=pd.DataFrame(data,index_int)
print(df1)
print(df1.index)
print("Floating point index")
index_float=[.1,.2,.3,.4,.5]
df2=pd.DataFrame(data,index_float)
print(df2)
print(df2.index)
```

Output :

```
  rollno     name section date_of_birth  weight  id
1    5D1  Sanjana       A    22/02/2002      60  I1
2    5D2    Vijay       A    15/03/2003      56  I2
3    5D3  Madhuri       B    08/12/2001      49  I3
4    5D4    Ramya       B    23/07/2002      52  I4
5    5D5    Aisha       C    14/06/2001      48  I5
Int64Index([1, 2, 3, 4, 5], dtype='int64')
Floating point index
    rollno     name section date_of_birth  weight  id
0.1    5D1  Sanjana       A    22/02/2002      60  I1
0.2    5D2    Vijay       A    15/03/2003      56  I2
0.3    5D3  Madhuri       B    08/12/2001      49  I3
0.4    5D4    Ramya       B    23/07/2002      52  I4
0.5    5D5    Aisha       C    14/06/2001      48  I5
Float64Index([0.1, 0.2, 0.3, 0.4, 0.5], dtype='float64')
```

Experiment No: 6. Pandas Joining and merging DataFrame:

6.i) Write a Pandas program to join the two given dataframes along rows and assign all data.

Aim: To write a Pandas program to join the two given dataframes along rows and assign all data.

Description: pandas.concat() function does all the heavy lifting of performing concatenation operations along with an axis od Pandas objects while performing optional set logic (union or intersection) of the indexes (if any) on the other axes.
Syntax: concat(objs, axis, join, ignore_index, keys, levels, names, verify_integrity, sort, copy)
Parameters:

- objs: Series or DataFrame objects
- axis: axis to concatenate along; default = 0
- ignore_index: if True, do not use the index values along the concatenation axis; default = False
- keys: sequence to add an identifier to the result indexes; default = None

Returns: type of objs (Series of DataFrame)

Program:

```
import pandas as pd

df1=pd.DataFrame({'name':['Sanjana','Jahnavi','Ambica','Pavani'],

              'course':['Pyhton','C++','Java','Pyhton'],

              'marks':[96,97,94,97]})

print("Student data 1")

print(df1)

df2=pd.DataFrame({'name':['Arun','Vijay','Rishi','Ravi'],

              'course':['C','Python','Java','C++'],

              'marks':[92,94,98,97]})

print("Student data 2")

print(df2)

print("After joining the two dataframes")

print(pd.concat([df1,df2],ignore_index=True))
```

Output:

```
Student data 1

      name  course  marks

0  Sanjana  Pyhton     96
```

```
1  Jahnavi     C++     97

2   Ambica    Java     94

3   Pavani  Pyhton     97

Student data 2

    name  course  marks

0   Arun       C     92

1  Vijay  Python     94

2  Rishi    Java     98

3   Ravi     C++     97

After joining the two dataframes

      name  course  marks

0  Sanjana  Pyhton     96

1  Jahnavi     C++     97

2   Ambica    Java     94

3   Pavani  Pyhton     97

4     Arun       C     92

5    Vijay  Python     94

6    Rishi    Java     98

7     Ravi     C++     97
```

6.ii)Write a Pandas program to append a list of dictioneries or series to a existing DataFrame and display the combined data.

Aim: To write a Pandas program to append a list of dictioneries or series to a existing DataFrame and display the combined data.

Description:

DataFrame.append(*other*, *ignore_index=False*, *verify_integrity=False*, *sort=False*)[source]

Append rows of *other* to the end of caller, returning a new object. Columns in *other* that are not in the caller are added as new columns.

Parameters:

other*DataFrame or Series/dict-like object, or list of these*: The data to append.

ignore_index*bool, default False*: If True, the resulting axis will be labeled 0, 1, …, n - 1.

verify_integrity*bool, default False*: If True, raise ValueError on creating index with duplicates.

sort*bool, default False*: Sort columns if the columns of *self* and *other* are not aligned.

Returns: A new DataFrame consisting of the rows of caller and the rows of *other*.

Program:

```
import pandas as pd

df1=pd.DataFrame({'name':['Sanjana','Jahnavi','Ambica','Pavani'],

               'course':['Pyhton','C++','Java','Pyhton'],

               'marks':[96,97,94,97]})

print("Student data frame")

print(df1)

series=pd.Series(['Ravi','Python',99],index=['name','course','marks'])

print("Series")

print(series)

dictionary=[{'name':'Arun','course':'C','marks':93},{'name':'Vijay','course':'Java','marks':91}]

print("dictionary")

print(dictionary)

print("After appending dictionary to dataframe")

print(df1.append(dictionary,ignore_index=True,sort=False))

print("After appending series to dataframe")

print(df1.append(series,ignore_index=True,sort=False))
```

Output:

```
Student data frame

     name  course  marks
```

```
0  Sanjana  Pyhton     96
1  Jahnavi     C++     97
2   Ambica    Java     94
3   Pavani  Pyhton     97
Series
name        Ravi
course    Python
marks         99
dtype: object
dictionary
[{'name': 'Arun', 'course': 'C', 'marks': 93}, {'name': 'Vijay', 'course': 'Java', 'marks': 91}]
After appending dictionary to dataframe
      name  course  marks
0  Sanjana  Pyhton     96
1  Jahnavi     C++     97
2   Ambica    Java     94
3   Pavani  Pyhton     97
4     Arun       C     93
5    Vijay    Java     91
After appending series to dataframe
      name  course  marks
0  Sanjana  Pyhton     96
1  Jahnavi     C++     97
2   Ambica    Java     94
3   Pavani  Pyhton     97
4     Ravi  Python     99
```

6.iii)Write a Pandas program to join the two dataframes with matching records from both sides where available.

Aim: To write a Pandas program to join the two dataframes with matching records from both sides where available.

Description:

pandas.merge(*left*, *right*, *how='inner'*, *on=None*, *left_on=None*, *right_on=None*, *left_index=False*, *right_index=False*, *sort=False*, *suffixes=('_x', '_y')*, *copy=True*, *indicator=False*, *validate=None*)[source]

Merge DataFrame or named Series objects with a database-style join. A named Series object is treated as a DataFrame with a single named column. The join is done on columns or indexes. If joining columns on columns, the DataFrame indexes *will be ignored*. Otherwise if joining indexes on indexes or indexes on a column or columns, the index will be passed on. When performing a cross merge, no column specifications to merge on are allowed.

Parameters :

left*DataFrame*
right*DataFrame or named Series*
Object to merge with.
how*{'left', 'right', 'outer', 'inner', 'cross'}, default 'inner'*
Type of merge to be performed.

- left: use only keys from left frame, similar to a SQL left outer join; preserve key order.
- right: use only keys from right frame, similar to a SQL right outer join; preserve key order.
- outer: use union of keys from both frames, similar to a SQL full outer join; sort keys lexicographically.
- inner: use intersection of keys from both frames, similar to a SQL inner join; preserve the order of the left keys.
- cross: creates the cartesian product from both frames, preserves the order of the left keys.

Returns: A DataFrame of the two merged objects.

Program:

```
import pandas as pd

student_data1 = pd.DataFrame({

        'id': ['5D1','5D2','5D3','5D4','5D5'],

         'name': ['Ramya','Tulasi','Lakshmi','Vennala','Satya'],

        'marks': [ 89,92,95,86,92]})



student_data2 = pd.DataFrame({

        'id': ['5D6','5D7','5D8','5D9'],

        'name': ['Ramya','Ravali','Lasya','Arya'],
        'marks': [99,89,97,94]})


print("Original DataFrames:")

print(student_data1)

print(student_data2)

merged_data = pd.merge(student_data1, student_data2, on='id', how='outer')

print("Merged data (outer join):")

print(merged_data)
```

Output:

```
Original DataFrames:

   id    name  marks

0  5D1    Ramya     89

1  5D2   Tulasi     92

2  5D3  Lakshmi     95

3  5D4  Vennala     86

4  5D5    Satya     92

   id   name  marks

0  5D6   Ramya     99

1  5D7  Ravali     89

2  5D8   Lasya     97

3  5D9    Arya     94

Merged data (outer join):

   id   name_x  marks_x  name_y  marks_y

0  5D1    Ramya     89.0     NaN      NaN

1  5D2   Tulasi     92.0     NaN      NaN

2  5D3  Lakshmi     95.0     NaN      NaN
```

```
3  5D4  Vennala    86.0    NaN     NaN

4  5D5    Satya    92.0    NaN     NaN

5  5D6      NaN     NaN  Ramya    99.0

6  5D7      NaN     NaN Ravali    89.0

7  5D8      NaN     NaN  Lasya    97.0

8  5D9      NaN     NaN   Arya    94.0
```