**Python – Pandas Data Structure (series, panel & Dataframe)**

Pandas in Python deals with three data structures namely

* Series
* DataFrame
* Panel

**Dimensions and Descriptions of Pandas Datastructure:**

* **Series –**1D labeled homogeneous array, sizeimmutable
* **Data Frames –**2D labeled, size-mutable tabular structure with heterogenic columns
* **Panel –**3D labeled size mutable array.

**Series in Pandas:**

Series is a one-dimensional array with homogeneous data.  All the elements of series should be of same data type

**Example:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12 | 9 | 7 | 6 | 11 | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | D | T | E | P | V |

**Key Features of a Series:**

* Homogeneous data
* Size Immutable –size cannot be changed
* Values of Data Mutable

**DataFrame in pandas:**

DataFrame is a two-dimensional array with heterogeneous data, usually represented in the tabular format. The data is represented in rows and columns. Each column represents an attribute and each row represents a person**.**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | City | Exam date | score |
| Arindam | Bangalore | 15-Jan | 76 |
| Ally | Washington | 23-Apr | 56 |
| jackson | London | 12-Feb | 86 |
| ivy | Madrid | 8-Jul | 67 |

**Key Features of a DataFrame:**

* Heterogeneous data
* Size Mutable
* Data Mutable

**Panel in pandas:**

Panel is a three-dimensional data structure with heterogeneous data.

**Key Features of a Panel:**

* Heterogeneous data
* Size Mutable
* Data Mutable

# Create a Series in python – pandas

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). In this tutorial we will learn the different ways to create a series in python pandas (create empty series, series from array without index, series from array with index, series from dictionary and scalar value ). The axis labels are called as indexes.

#### ****Create an Empty Series:****

A basic series, which can be created is an Empty Series. Below example is for creating an empty series.

|  |  |
| --- | --- |
| 1  2  3  4  5 | # Example Create an Empty Series    import pandas as pd  s = pd.Series()  print s |

**output:**

Series([], dtype: float64)

#### ****Create a series from array without index:****

Lets see an example on how to create series from an array.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Example Create a series from array    import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data)  print s |

#### ****output:****

0  a  
1   b  
2   c  
3   d  
4   e  
5   f  
dtype: object

#### ****Create a series from array with index:****

This example depicts how to create a series in python with index, Index starting from 1000 has been added in the below example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Example Create a series from array with specified index    import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data,index=[1000,1001,1002,1003,1004,1005])  print s |

#### ****output:****

1000   a  
1001   b  
1002   c  
1003   d  
1004   e  
1005   f  
dtype: object

#### ****Create a series from Dictionary****

This example depicts how to create a series in python with dictionary. Dictionary keys are used to construct index.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # Example Create a series from dictionary    import pandas as pd  import numpy as np  data = {'a' : 0., 'b' : 1., 'c' : 2.}  s = pd.Series(data,index=['b','c','d','a'])  print s |

Index order is maintained and the missing element is filled with NaN (Not a Number). So the output will be

#### ****output:****

b   1.0  
c   2.0  
d   NaN  
a   0.0

dtype: float64

#### ****Create a series from Scalar value****

This example depicts how to create a series in python from scalar value. If data is a scalar value, an index must be provided. The value will be repeated to match the length of index

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # create a series from scalar    import pandas as pd  import numpy as np  s = pd.Series(7, index=[0, 1, 2, 3])  print s |

#### ****output:****

0   7  
1   7  
2   7  
3   7  
dtype: int64

# how to Access the elements of a Series in python – pandas

In This tutorial we will learn how to access the elements of a series in python pandas. We will be learning how to

* Accessing Data from Series with Position in python pandas
* Retrieve Data Using Label (index)  in python pandas

### **Accessing data from series with position:**

#### ****Accessing or retrieving the first element:****

Retrieve the first element. As we already know, the counting starts from zero for the array, which means the first element is stored at zeroth position and so on.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # create a series  import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data)      #retrieve the first element  print s[0] |

**output:**

a

**Access or Retrieve the first three elements in the Series:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # create a series  import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data)      # retrieve first three elements  print s[:3] |

**output:**

0    a  
1    b  
2    c

dtype: object

**Access or Retrieve the last three elements in the Series:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # create a series  import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data)      # retrieve last three elements  print s[-3:] |

**output:**

3   d  
4   e  
5   f  
dtype: object

### **Accessing data from series with Labels or index:**

A Series is like a fixed-size dictionary in that you can get and set values by index label.

#### ****Retrieve a single element using index label:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # create a series  import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data,index=[100,101,102,103,104,105])      print s[102] |

#### ****output:****

c

#### ****Retrieve multiple elements using index labels:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # create a series  import pandas as pd  import numpy as np  data = np.array(['a','b','c','d','e','f'])  s = pd.Series(data,index=[100,101,102,103,104,105])      # retrieve multiple elements with labels or index  print s[[102,103,104]] |

#### ****output:****

102    c  
103    d  
104    e

dtype: object

**Note:**If label or index is not mentioned properly an exception will be raised.

**Format integer column of Dataframe in Python pandas**

In this Tutorial we will learn how to format integer column of Dataframe in Python pandas with an example. We will learn

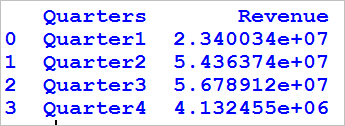
* Round off a column values of dataframe to two decimal places
* Format the column value of dataframe with commas
* Format the column value of dataframe with dollar
* Format the column value of dataframe with scientific notation

Let’s see each with an example. First lest create a dataframe.

**Create Dataframe:**

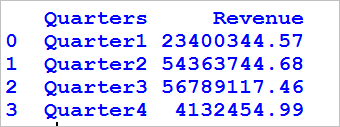
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # create dataframe    import pandas as pd  d = {'Quarters' : ['Quarter1','Quarter2','Quarter3','Quarter4'],       'Revenue':[23400344.567,54363744.678,56789117.456,4132454.987]}  df=pd.DataFrame(d)  print df |

So the resultant dataframe will be



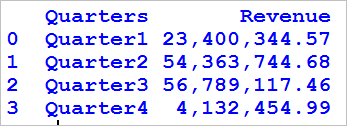
**Round off the column values to two decimal places in python pandas:**

|  |  |
| --- | --- |
| 1  2  3  4 | # round to two decimal places in python pandas    pd.options.display.float\_format = '{:.2f}'.format  print df |



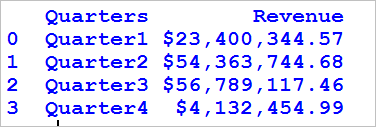
**Format with commas and round off to two decimal places in python pandas:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Format with commas and round off to two decimal places in pandas    pd.options.display.float\_format = '{:,.2f}'.format  print df |



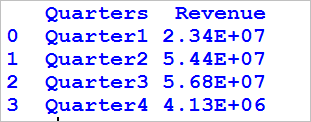
**Format with commas and Dollar sign with two decimal places in python pandas:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Format with dollars, commas and round off to two decimal places in pandas    pd.options.display.float\_format = '${:,.2f}'.format  print df |



**Format with Scientific notation in python pandas**

|  |  |
| --- | --- |
| 1  2  3  4 | # Format with Scientific notation    pd.options.display.float\_format = '{:.2E}'.format  print df |



# Head and tail function in Python pandas (Get First N Rows & Last N Rows)

In this tutorial we will learn how to get the snap shot of the data, by getting first few rows and last few rows of the data frame i.e Head and Tail function in python. Head function returns first n rows and tail function return last n rows

We will use the iris data set for demonstration of head and tail function in python

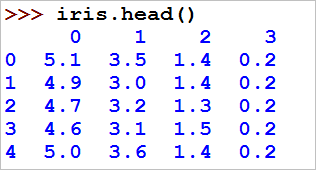
#### ****Loading the iris data set:****

|  |  |
| --- | --- |
| 1  2  3  4  5 | # load iris data set from sklearn    import pandas as pd  from sklearn import datasets  iris=pd.DataFrame(datasets.load\_iris().data) |

#### ****Head Function in Python (Get First N Rows):****

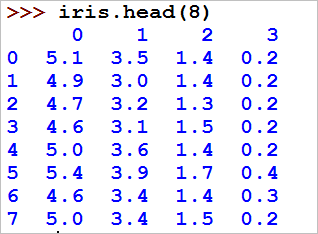
|  |  |
| --- | --- |
| 1  2  3 | # head function in python    iris.head() |

head function with no arguments gets the first five rows of data from the data frame so the output will be



|  |  |
| --- | --- |
| 1  2  3 | # head function in python with arguments    iris.head(8) |

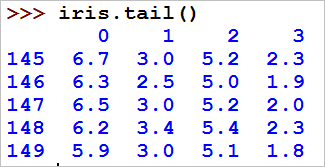
head function with specified N arguments, gets the first N rows of data from the data frame so the output will be



#### ****Tail Function in Python (Get Last N Rows):****

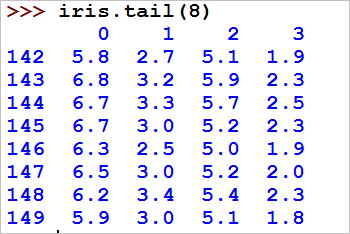
|  |  |
| --- | --- |
| 1  2  3 | # Tail function in python    iris.tail() |

tail function with no arguments gets the last five rows of data from the data frame so the output will be



|  |  |
| --- | --- |
| 1  2  3 | # tail function in python with arguments    iris.tail(8) |

tail function with specified N arguments, gets the last N rows of data from the data frame so the output will be



**Descriptive or Summary Statistics in python pandas – describe()**

Descriptive or summary statistics in python – pandas, can be obtained by using describe function – describe(). Describe Function gives the mean, std and IQR values.

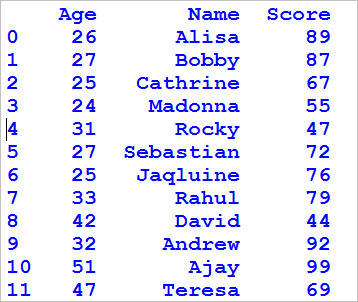
* Generally describe() function excludes the character columns and gives summary statistics of numeric columns
* We need to add a variable named include=’all’ to get the summary statistics or descriptive statistics of both numeric and character column.

**Lets see with an example**

**Example of Descriptive or Summary Statistics in python**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | # creation of DataFrame    import pandas as pd  import numpy as np    #Create a Dictionary of series  d = {'Name':pd.Series(['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa']),     'Age':pd.Series([26,27,25,24,31,27,25,33,42,32,51,47]),     'Score':pd.Series([89,87,67,55,47,72,76,79,44,92,99,69])}    #Create a DataFrame  df = pd.DataFrame(d)  print df |

So the resultant DataFrame will be

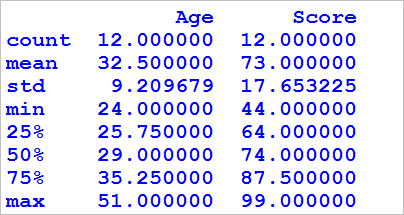


**Pandas- Descriptive or Summary Statistic of the numeric columns:**

|  |  |
| --- | --- |
| 1  2  3 | # summary statistics    print df.describe() |

* **describe()** Function gives the mean, std and IQR values. It excludes character column and calculate summary statistics only for numeric columns

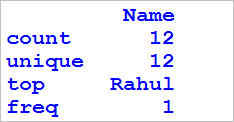
so the output will be



**Pandas – Descriptive or Summary Statistic of the character columns:**

|  |  |
| --- | --- |
| 1  2  3 | # summary statistics of character column    print df.describe(include=['object']) |

* **describe()**Function with an argument named include along with value object i.e **include=’object’** gives the summary statistics of the character columns.

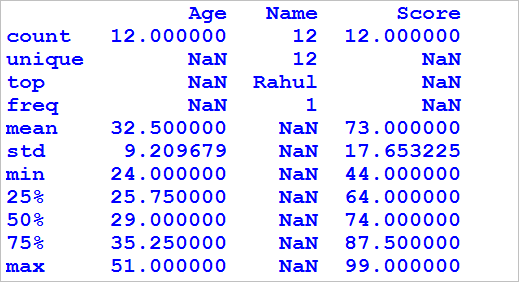


**Pandas – Descriptive or Summary Statistic of all the columns**

|  |  |
| --- | --- |
| 1  2  3 | # summary statistics of character column    print df.describe(include='all') |

**describe()**Function with **include=’all’** gives the summary statistics of all the columns.

So the output will be



**Mean Function in Python pandas (Dataframe, Row and column wise mean)**

mean() – Mean Function in python pandas is used to calculate the arithmetic mean of a given set of numbers, mean of a data frame ,mean of column and mean of rows , lets see an example of each . We need to use the package name “statistics” in calculation of mean. In this tutorial we will learn,

* How to find the mean of a given set of numbers
* How to find mean of a dataframe
* How to find the mean of a column in dataframe
* How to find row mean of a dataframe

**Mean Function in Python**

**Simple mean function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate arithmetic mean  Import statistics    print(statistics.mean([1,9,5,6,6,7]))  print(statistics.mean([4,-11,-5,16,5,7])) |

**output:**

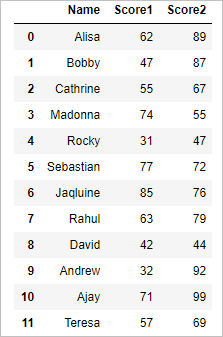
5.66666666667  
2.66666666667

**Mean of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69]}      df = pd.DataFrame(d)  df |

So the resultant dataframe will be



**Mean of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # mean of the dataframe  df.mean() |

it will calculate the mean of the dataframe across columns so the output will be

Score1  58.0  
Score2  73.0  
dtype:  float64

**Column Mean of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # column mean of the dataframe  df.mean(axis=0) |

axis=0 argument calculates the column wise mean of the dataframe so the result will be

Score1 58.0  
Score2 73.0  
dtype: float64

**Row Mean of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # Row mean of the dataframe  df.mean(axis=1) |

axis=1 argument calculates the row wise mean of the dataframe so the result will be

0   75.5  
1   67.0  
2   61.0  
3   64.5  
4   39.0  
5   74.5  
6   80.5  
7   71.0  
8   43.0  
9   62.0  
10   85.0  
11   63.0  
dtype: float64

**Calculate the mean of the specific Column**

|  |  |
| --- | --- |
| 1  2 | # mean of the specific column  df.loc[:,"Score1"].mean() |

the above code calculates the mean of the “Score1” column so the result will be

58.0

# Median Function in Python pandas (Dataframe, Row and column wise median)

median() – Median Function in python pandas is used to calculate the median or middle value of a given set of numbers, Median of a data frame, median of column and median of rows, let’s see an example of each. We need to use the package name “statistics” in calculation of median. In this tutorial we will learn,

We need to use the package name “statistics” in calculation of median. In this tutorial we will learn,

* How to find the median of a given set of numbers
* How to find median of a dataframe
* How to find the median of a column in dataframe
* How to find row median of a dataframe

**Median Function in Python**

**Simple median function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate median or middle value  Import statistics    print(statistics.median([1,9,5,6,8,7]))  print(statistics.median([4,-11,-5,16,5,7,9])) |

**output:**

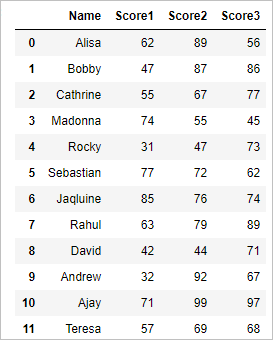
6.5  
5

**Median of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],     'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}    df = pd.DataFrame(d)  df |

So the resultant dataframe will be



**Median of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # median of the dataframe  df.median() |

will calculate the median of the dataframe across columns so the output will

Score1   59.5  
Score2   74.0  
Score3   72.0  
dtype: float64

**Column Median of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # column median of the dataframe  df.median(axis=0) |

axis=0 argument calculates the column wise median of the dataframe so the result will be

0   62.0  
1   86.0  
2   67.0  
3   55.0  
4   47.0  
5   72.0  
6   76.0  
7   79.0  
8   44.0  
9   67.0  
10   97.0  
11   68.0  
dtype: float64

**Calculate the median of the specific Column**

|  |  |
| --- | --- |
| 1  2 | # median of the specific column  df.loc[:,"Score1"].median() |

the above code calculates the median of the “Score1” column so the result will be

59.5

**Mode Function in Python pandas (Dataframe, Row and column wise mode)**

Mode Function in python pandas is used to calculate the mode or most repeated value of a given set of numbers. Mode of a data frame, mode of column and mode of rows, let’s see an example of each We need to use the package name “statistics” in calculation of mode. In this tutorial we will learn,

* How to find the mode of a given set of numbers
* How to find mode of a dataframe
* How to find the mode of a column in dataframe
* How to find row mode of a dataframe

**Mode Function in Python pandas**

**Simple mode function in python is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate mode or most repeated value  Import statistics    print(statistics.mode([1,5,5,7,5,6,8,7]))  print(statistics.mode(['lion', 'cat', 'cat','dog','tiger'])) |

**output:**

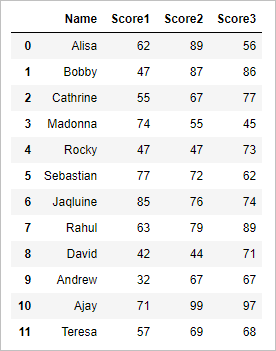
5  
cat

**Mode of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,47,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,67,99,69],     'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}        df = pd.DataFrame(d)  df |

So the resultant dataframe will be



**Mode of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # mode of the dataframe  df.mode() |

will calculate the mode of the dataframe across columns so the output will be

mode function in python 2

**Column Mode of the dataframe in python pandas :**

|  |  |
| --- | --- |
| 1  2 | # column mode of the dataframe  df.mode(axis=0) |

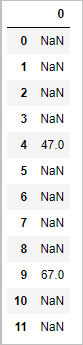
axis=0 argument calculates the column wise mode of the dataframe so the result will be

mode function in python 3

**Row Mode of the dataframe in python pandas :**

|  |  |
| --- | --- |
| 1  2 | # Row mode of the dataframe  df.mode(axis=1) |

axis=1 argument calculates the row wise mode of the dataframe so the result will be



**Calculate the mode of the specific Column – pandas**

|  |  |
| --- | --- |
| 1  2 | # mode of the specific column  df.loc[:,"Score1"].mode() |

the above code calculates the mode of the “Score1” column so the result will be

0  47  
dtype: int64

**Harmonic Mean Function in Python – pandas (Dataframe, Row and column wise harmonic mean)**

Harmonic Mean Function in python pandas is used to calculate the harmonic mean of a given set of numbers, Harmonic mean of a data frame, Harmonic mean of column and Harmonic mean of rows. let’s see an example of each we need to use the package name “stats” from scipy in calculation of harmonic mean. In this tutorial we will learn,

* How to find the harmonic mean of a given set of numbers
* How to find harmonic mean of a dataframe
* How to find the harmonic mean of a column in dataframe
* How to find row wise harmonic mean of a dataframe

**Harmonic Mean Function in Python**

**Simple harmonic mean function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate harmonic mean  from scipy import stats    print(stats.hmean([1,9,5,6,6,7]))  print(stats.hmean([4,11,15,16,5,7])) |

**output:**

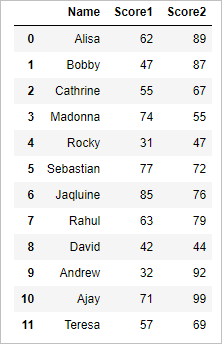
3.35701598579  
7.38068295281

**Harmonic Mean of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pandas as pd  import numpy as np  from scipy import stats    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69]}      df = pd.DataFrame(d)  df |

So the resultant dataframe will be



**Harmonic Mean of the column in dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Harmonic Mean of the  column in dataframe  from scipy import stats    scipy.stats.hmean(df.iloc[:,1:3],axis=0) |

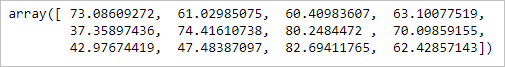
axis=0 argument calculates the column wise harmonic mean of the dataframe so the result will be

array([ 52.4769906 , 68.56766396])

**Row wise harmonic Mean of the dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Row wise harmonic mean of the dataframe  from scipy import stats    scipy.stats.hmean(df.iloc[:,1:3],axis=1) |

axis=1 argument calculates the row wise harmonic mean of the dataframe so the result will be



**Calculate the harmonic mean of the specific Column**

|  |  |
| --- | --- |
| 1  2  3  4 | # harmonic mean of the specific column  from scipy import stats    scipy.stats.hmean(df.loc[:,"Score1"]) |

the above code calculates the harmonic mean of the “Score1” column so the result will be

52.476990604079674

**Geometric Mean Function in Python – pandas (Dataframe, Row and column wise Geometric mean)**

Geometric Mean Function in python pandas is used to calculate the geometric mean of a given set of numbers, Geometric mean of a data frame, Geometric mean of column and Geometric mean of rows. let’s see an example of each we need to use the package name “stats” from scipy in calculation of geometric mean. In this tutorial we will learn,

* How to find the geometric mean of a given set of numbers
* How to find geometric mean of a dataframe
* How to find the geometric mean of a column in dataframe
* How to find row wise geometric mean of a dataframe

**Geometric Mean Function in Python**

**Simple geometric mean function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate geometric mean  from scipy import stats    print(stats.gmean([1,9,5,6,6,7]))  print(stats.gmean([4,11,15,16,5,7])) |

**output:**

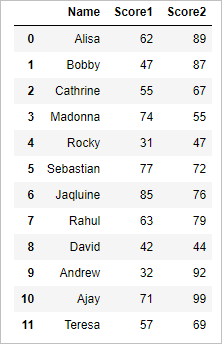
4.73989632394  
8.47140270122

**Geometric Mean of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pandas as pd  import numpy as np  from scipy import stats    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69]}      df = pd.DataFrame(d)  print df |

So the resultant dataframe will be



**Geometric Mean of the column in dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Geometric Mean of the column in dataframe  from scipy import stats    scipy.stats.gmean(df.iloc[:,1:3],axis=0) |

axis=0 argument calculates the column wise geometric mean of the dataframe so the result will be

array([ 55.33743527, 70.86175132])

**Row wise geometric Mean of the dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4 | # Row wise geometric mean of the dataframe  from scipy import stats    scipy.stats.gmean(df.iloc[:,1:3],axis=1) |

axis=1 argument calculates the row wise geometric mean of the dataframe so the result will be

Geometric Mean Function in Python - pandas 2

**Calculate the geometric mean of the specific Column**

|  |  |
| --- | --- |
| 1  2 | # geometric mean of the specific column  scipy.stats.gmean(df.loc[:,"Score1"]) |

the above code calculates the geometric mean of the “Score1” column so the result will be

55.337435272097579

**Standard deviation Function in Python pandas (Dataframe, Row and column wise standard deviation)**

Standard deviation Function in python pandas is used to calculate standard deviation of a given set of numbers, Standard deviation of a data frame, Standard deviation of column and Standard deviation of rows, let’s see an example of each. We need to use the package name “statistics” in calculation of median. In this tutorial we will learn,

* How to find the standard deviation of a given set of numbers
* How to find standard deviation of a dataframe
* How to find the standard deviation of a column in dataframe
* How to find row wise standard deviation of a dataframe

**Standard deviation Function in Python pandas**

**Simple standard deviation function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate standard deviation  import numpy as np    print(np.std([1,9,5,6,8,7]))  print(np.std([4,-11,-5,16,5,7,9])) |

**output:**

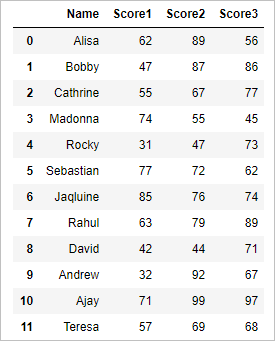
2.82842712475  
8.97881103594

**Standard deviation of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],     'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}        df = pd.DataFrame(d)  df |

So the resultant dataframe will be



**Standard deviation of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # standard deviation of the dataframe  df.std() |

will calculate the standard deviation of the dataframe across columns so the output will

Score1     17.446021  
Score2     17.653225  
Score3     14.355603  
dtype: float64

**Column Standard deviation of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # column standard deviation  of the dataframe  df.std(axis=0) |

axis=0 argument calculates the column wise standard deviation of the dataframe so the result will be

Score1     17.446021  
Score2     17.653225  
Score3     14.355603  
dtype: float64

**Row standard deviation of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # Row standard deviation of the dataframe  df.std(axis=1) |

axis=1 argument calculates the row wise standard deviation of the dataframe so the result will be

0     17.578396  
1     22.810816  
2     11.015141  
3     14.730920  
4     21.197484  
5     7.637626  
6     5.859465  
7     13.114877  
8     16.196707  
9     30.138569  
10     15.620499  
11     6.658328  
dtype: float64

**Calculate the standard deviation of the specific Column**

|  |  |
| --- | --- |
| 1  2 | # standard deviation of the specific column  df.loc[:,"Score1"].std() |

The above code calculates the standard deviation of the “Score1” column so the result will be

17.446020645512156

**Variance Function in Python pandas (Dataframe, Row and column wise Variance)**

var() – Variance Function in python pandas is used to calculate variance of a given set of numbers, Variance of a data frame, Variance of column and Variance of rows, let’s see an example of each.  We need to use the package name “statistics” in calculation of variance. In this tutorial we will learn,

* How to find the variance of a given set of numbers
* How to find variance of a dataframe
* How to find the variance of a column in dataframe
* How to find row wise variance of a dataframe

**Variance Function in Python pandas**

**Simple variance function is shown below**

|  |  |
| --- | --- |
| 1  2  3  4  5 | # calculate variance  import numpy as np    print(np.var([1,9,5,6,8,7]))  print(np.var([4,-11,-5,16,5,7,9])) |

**output:**

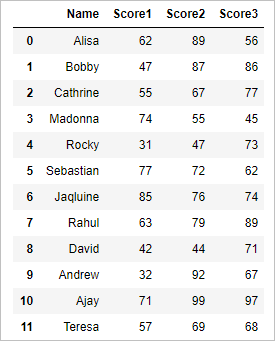
2.82842712475  
8.97881103594

**Variance of a dataframe:**

Create dataframe

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],     'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}        df = pd.DataFrame(d)  print df |

So the resultant dataframe will be



**Variance of the dataframe:**

|  |  |
| --- | --- |
| 1  2 | # variance of the dataframe  df.var() |

will calculate the variance of the dataframe across columns so the output will be

Score1   304.363636  
Score2   311.636364  
Score3   206.083333  
dtype: float64

**Column variance of the dataframe:**

|  |  |
| --- | --- |
| 1  2  3 | # column variance of the dataframe    df.var(axis=0) |

axis=0 argument calculates the column wise variance of the dataframe so the result will be

Score1   304.363636  
Score2   311.636364  
Score3   206.083333  
dtype: float64

**Row variance of the dataframe:**

|  |  |
| --- | --- |
| 1  2  3 | # Row variance of the dataframe    df.var(axis=1) |

axis=1 argument calculates the row wise variance of the dataframe so the result will be

0     309.000000  
1     520.333333  
2    121.333333  
3    217.000000  
4    449.333333  
5    58.333333  
6    34.333333  
7    172.000000  
8    262.333333  
9    908.333333  
10     244.000000  
11     44.333333  
dtype: float64

**Calculate the variance of the specific Column**

|  |  |
| --- | --- |
| 1  2 | # variance of the specific column  df.loc[:,"Score1"].var() |

the above code calculates the variance of the “Score1” column so the result will be

304.36363636363637

**How to join (merge) data frames (inner, outer, right, left join) in pandas python**

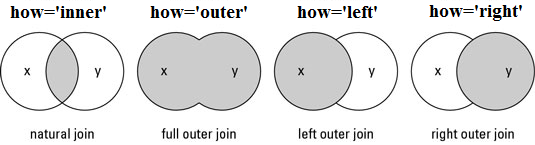
We can merge two data frames in pandas python by using the **merge()**function. The different arguments to merge() allow you to perform natural join,  left join, right join, and full outer join in pandas.

* **left**− Dataframe1.
* **right**– Dataframe2.
* **on**− Columns (names) to join on. Must be found in both the left and right DataFrame objects.
* **how** – type of join needs to be performed – ‘left’, ‘right’, ‘outer’, ‘inner’, Default is inner join

The data frames must have same column names on which the merging happens. Merge() Function in pandas is similar to database join operation in SQL.

**UNDERSTANDING THE DIFFERENT TYPES OF MERGE:**

* **Natural join:** To keep only rows that match from the data frames, specify the argument how=**‘inner’.**
* **Full outer join:**To keep all rows from both data frames, specify how=**‘outer’.**
* **Left outer join:**To include all the rows of your data frame x and only those from y that match, specify how=**‘left’.**
* **Right outer join:**To include all the rows of your data frame y and only those from x that match, specify how=**‘right’.**



**Lets try different Merge or join operation with an example:**

**Create dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    # data frame 1  d1 = {'Customer\_id':pd.Series([1,2,3,4,5,6]),    'Product':pd.Series(['Oven','Oven','Oven','Television','Television','Television'])}  df1 = pd.DataFrame(d1)      # data frame 2  d2 = {'Customer\_id':pd.Series([2,4,6]),      'State':pd.Series(['California','California','Texas'])}  df2 = pd.DataFrame(d2) |

so we will get following two data frames

**df1:**

  Customer\_id   Product  
0   1        Oven  
1   2        Oven  
2   3        Oven  
3   4       Television  
4   5       Television  
5   6       Television

**df2:**

  Customer\_id   State  
0   2        California  
1   4        California  
2   6        Texas

**Inner join pandas:** Return only the rows in which the left table have matching keys in the right table

|  |  |
| --- | --- |
| 1  2 | #inner join in python pandas  print pd.merge(df1, df2, on='Customer\_id', how='inner') |

the resultant data frame df will be

    Customer\_id       Product           State

0            2                    Oven             California

1            4                    Television     California

2            6                    Television     Texas

**Outer join pandas*:*** Returns all rows from both tables, join records from the left which have matching keys in the right table.

|  |  |
| --- | --- |
| 1  2 | # outer join in python pandas  print pd.merge(df1, df2, on='Customer\_id', how='outer') |

the resultant data frame df will be

 Customer\_id        Product           State

0            1                   Oven               NaN

1            2                    Oven              California

2            3                     Oven              NaN

3            4                 Television          California

4            5                 Television          NaN

5            6                 Television          Texas

**Left outer join pandas*:***Return all rows from the left table, and any rows with matching keys from the right table.

|  |  |
| --- | --- |
| 1  2 | # left join in python  print pd.merge(df1, df2, on='Customer\_id', how='left') |

the resultant data frame df will be

 Customer\_id      Product           State

0            1               Oven              NaN

1            2               Oven              California

2            3               Oven               NaN

3            4            Television          California

4            5            Television          NaN

5            6            Television          Texas

**Right outer join pandas*:*** Return all rows from the right table, and any rows with matching keys from the left table.

|  |  |
| --- | --- |
| 1  2  3 | # right join in python pandas    print pd.merge(df1, df2, on='Customer\_id', how='right') |

the resultant data frame df will be

    Customer\_id         Product          State

0            2                       Oven             California

1            4                   Television         California

2            6                    Television        Texas

**Apply Functions in Python pandas – Apply(), Applymap(), pipe()**

To Apply our own function or some other library’s function, pandas provide three important functions namely pipe(), apply() and applymap().  These Functions are discussed below.

* Table wise Function Application: pipe()
* Row or Column Wise Function Application: apply()
* Element wise Function Application: applymap()

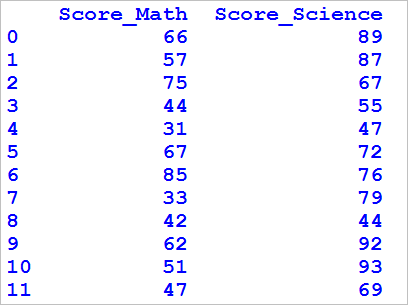
**Table wise Function Application: pipe()**

Pipe() function performs the custom operation for the entire dataframe. In below example we will using pipe() Function to add value 2 to the entire dataframe

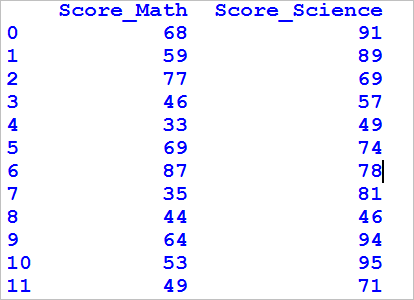
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import pandas as pd  import numpy as np  import math    # own function  def adder(adder1,adder2):     return adder1+adder2    #Create a Dictionary of series  d = {'Score\_Math':pd.Series([66,57,75,44,31,67,85,33,42,62,51,47]),     'Score\_Science':pd.Series([89,87,67,55,47,72,76,79,44,92,93,69])}    df = pd.DataFrame(d)  print df  print df.pipe(adder,2) |

output will be

**Original dataframe:**



**Dataframe with value 2 added:**



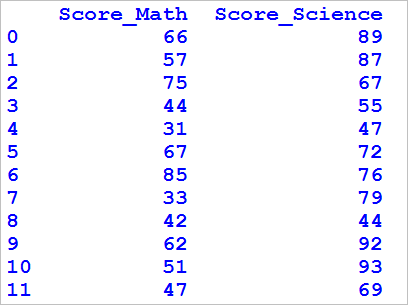
**Row or Column Wise Function Application: apply()**

apply() function performs the custom operation for either row wise or column wise . In below example we will be using apply() Function to find the mean of values across rows and mean of values across columns

**Create Dataframe**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import pandas as pd  import numpy as np  import math      #Create a DataFrame  d = {'Score\_Math':pd.Series([66,57,75,44,31,67,85,33,42,62,51,47]),     'Score\_Science':pd.Series([89,87,67,55,47,72,76,79,44,92,93,69])}    df = pd.DataFrame(d)  print df |

resultant dataframe will be

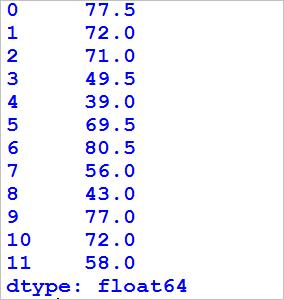


**Row wise Function in python pandas : Apply()**

apply() Function to find the mean of values across rows

|  |  |
| --- | --- |
| 1  2  3 | #row wise mean    print df.apply(np.mean,axis=1) |

so the output will be

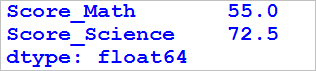


**Column wise Function in python pandas : Apply()**

apply() Function to find the mean of values across columns

|  |  |
| --- | --- |
| 1  2  3 | #column wise meanprint    df.apply(np.mean,axis=0) |

so the output will be



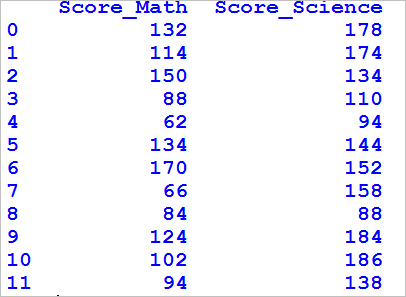
**Element wise Function Application in python pandas: applymap()**

applymap() Function performs the specified operation for all the elements the dataframe. we will be using the same dataframe to depict example of applymap() Function. We will be multiplying the all the elements of dataframe by 2 as shown below

**Example1: applymap() Function in python**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | import pandas as pd  import numpy as np  import math      # applymap() Function  print df.applymap(lambda x:x\*2) |

so the output will be

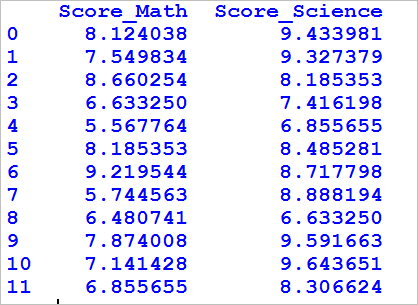


**Example2: applymap() Function in python**

We will be finding the square root of all the elements of dataframe with applymap() function as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | import math    #applymap() Function to find the sqrt  print df.applymap(lambda x:math.sqrt(x)) |

so the output will be



**Rename the column of dataframe in pandas python**

In this tutorial we will learn how to rename the column of dataframe in pandas. We will learn

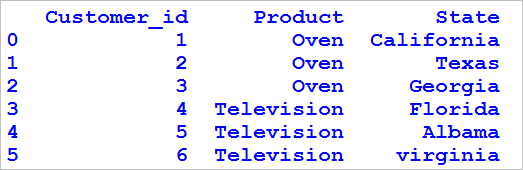
* how to rename all the column of the dataframe at once
* how to rename the specific column of our choice by column name.
* how to rename the specific column of our choice by column index.

**Let’s try with an example:**

**Create a dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import pandas as pd  import numpy as np    # data frame 1  d1 = {'Customer\_id':pd.Series([1,2,3,4,5,6]),  'Product':pd.Series(['Oven','Oven','Oven','Television','Television','Television']),        'State':pd.Series(['California','Texas','Georgia','Florida','Albama','virginia'])}  df1 = pd.DataFrame(d1)  print df1 |

so the resultant dataframe will be



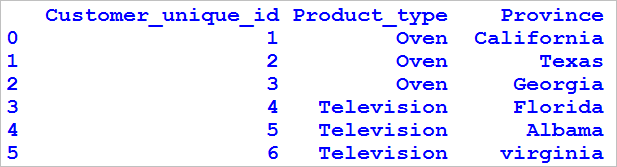
**Rename all the column names in python:**

Below code will rename all the column names in sequential order

|  |  |
| --- | --- |
| 1  2 | # rename all the columns in python  df1.columns = ['Customer\_unique\_id', 'Product\_type', 'Province'] |

* first column is renamed as **‘Customer\_unique\_id’.**
* second column is renamed as ‘**Product\_type’.**
* third column is renamed as**‘Province’.**

so the resultant dataframe will be

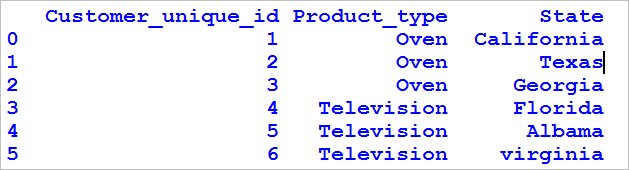


**Rename the specific column in python:**

Below code will rename the specific column.

|  |  |
| --- | --- |
| 1  2 | # rename province to state  df1.rename(columns={'Province': 'State'}, inplace=True) |

the column named **Province** is renamed to **State**with the help of rename() Function so the resultant dataframe will be

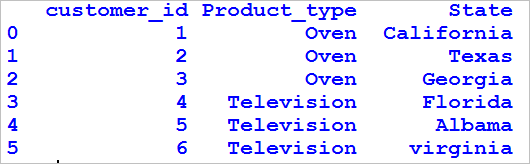


**Rename the specific column value by index in python:**

Below code will rename the specific column.

|  |  |
| --- | --- |
| 1  2 | # rename the first column  df1.columns.values[0] = "customer\_id" |

the first column is renamed to **customer\_id**so the resultant dataframe will be



**Reindex in python pandas – change order of row and column in pandas**

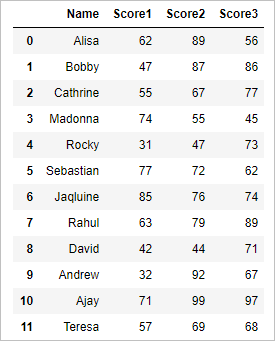
In this tutorial we will learn how to reindex in python pandas or change the order of the rows and column in python pandas with the help of reindex() function. We will discuss the example for

* Reindexing or changing the order of Rows in pandas python
* Reindexing or changing the order of columns in pandas python

**Create dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Score1':[62,47,55,74,31,77,85,63,42,32,71,57],     'Score2':[89,87,67,55,47,72,76,79,44,92,99,69],     'Score3':[56,86,77,45,73,62,74,89,71,67,97,68]}    df = pd.DataFrame(d)  df |

The resultant dataframe will be

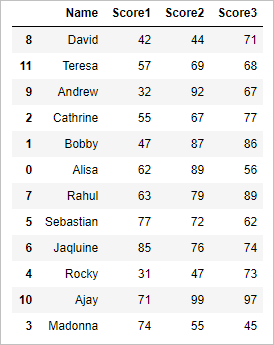


**Reindex or change the order of rows in pandas python:**

Now lets change the order of rows as shown below

|  |  |
| --- | --- |
| 1  2 | # reindex or change the order of rows  df.reindex([8,11,9,2, 1, 0,7,5,6,4,10,3]) |

so the re indexed dataframe will be

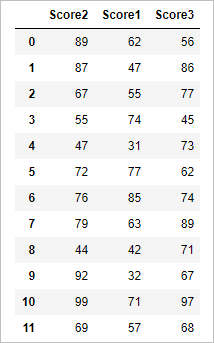


**Reindex or change the order of columns in pandas python:**

Now lets change the order of columns as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # reindex or change the order of columns    columnsTitles = ['Score2', 'Score1', 'Score3']  df.reindex(columns=columnsTitles) |

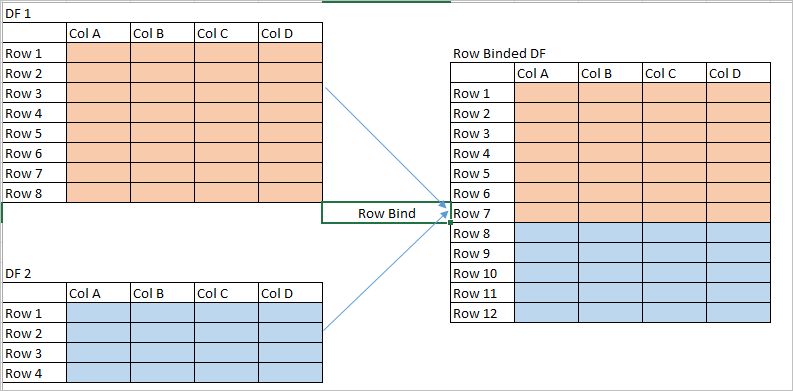
so the re indexed dataframe will be



# Row bind in python pandas – Append or concatenate rows in python pandas

Row bind in python pandas – In this tutorial we will learn how to concatenate rows to the python pandas dataframe  with append() Function and concat() Function i.e. how to row bind two data frames in python pandas with an example.

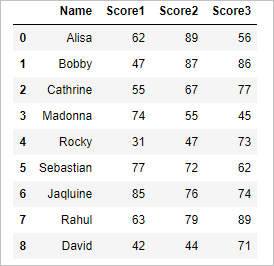
Row binding is pictographically shown below



#### ****Create dataframe 1:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David'],     'Score1':[62,47,55,74,31,77,85,63,42],     'Score2':[89,87,67,55,47,72,76,79,44],     'Score3':[56,86,77,45,73,62,74,89,71]}    df1 = pd.DataFrame(d)  df1 |

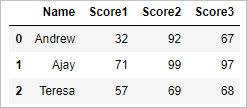
so the dataframe 1 will be



#### ****Create dataframe 2:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np    # create dataframe - df2  d = {      'Name':['Andrew','Ajay','Teresa'],     'Score1':[32,71,57],     'Score2':[92,99,69],     'Score3':[67,97,68]}    df2 = pd.DataFrame(d)  df2 |

and the dataframe 2 will be

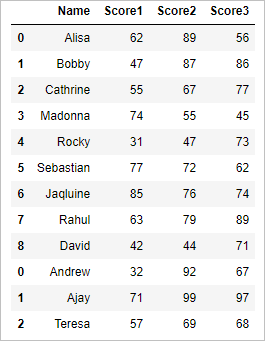


#### ****Method 1: Row bind or concatenate two dataframes in pandas :****

Now lets concatenate or row bind two dataframes df1 and df2

|  |  |
| --- | --- |
| 1 | pd.concat([df1,df2]) |

so the resultant row binded dataframe will be

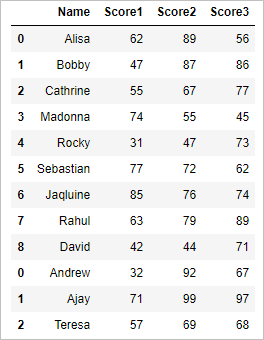


#### ****Method 2: Row bind or concatenate two dataframes in pandas:****

Now lets concatenate or row bind two dataframes df1 and df2 with append method

|  |  |
| --- | --- |
| 1 | df1.append(df2) |

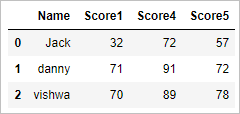
so the resultant dataframe will be



#### ****Concatenate or append rows of dataframe with different column names****

Now Lets  create dataframe 3

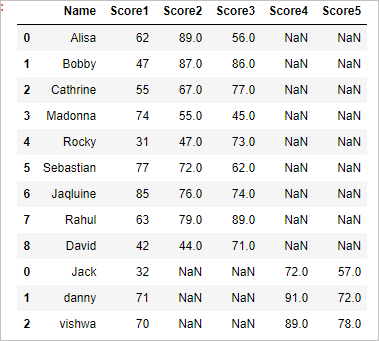
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np    # create dataframe – df3  d = {      'Name':['Jack','danny','vishwa'],     'Score1':[32,71,70],     'Score4':[72,91,89],     'Score5':[57,72,78]}    df3 = pd.DataFrame(d)  df3 |



#### ****Concatenate the df1 and df3 whose column names are different:****

|  |  |
| --- | --- |
| 1 | pd.concat([df1,df3]) |

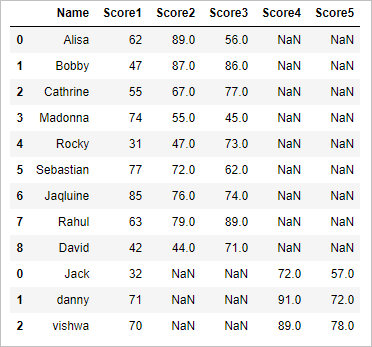
**result:**



#### ****Concatenate the df1 and df3 with append function****

|  |  |
| --- | --- |
| 1 | df1.append(df3) |

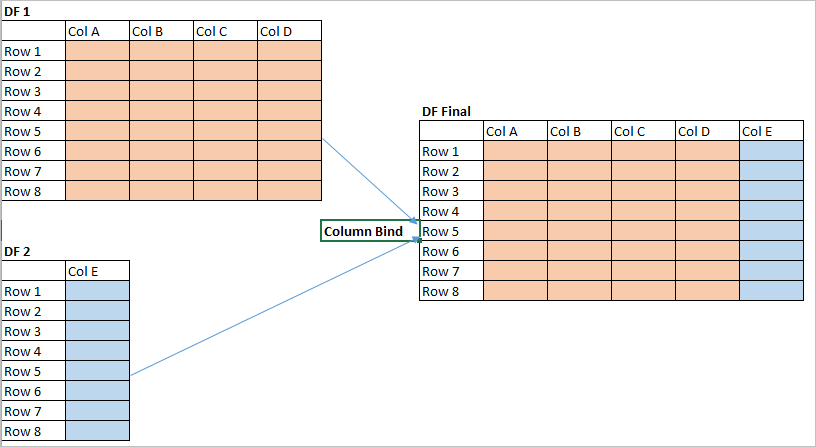
**result:**



# column bind in python pandas – concatenate columns in python pandas

Column bind in python pandas. In this tutorial we will learn how to concatenate columns to the python pandas dataframe using concat() Function with example i.e. how to column bind two data frames in python pandas.

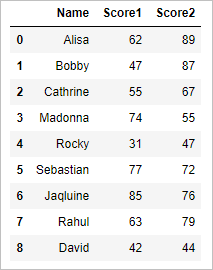
Column binding is pictographically shown below



#### ****Create dataframe 1:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pandas as pd  import numpy as np      #Create a DataFrame    d = {  'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',  'Rahul','David'],  'Score1':[62,47,55,74,31,77,85,63,42],  'Score2':[89,87,67,55,47,72,76,79,44]}    df1 = pd.DataFrame(d)  df1 |

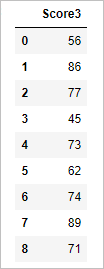
so the resultant dataframe will be



#### ****Create dataframe 2:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {  'Score3':[56,86,77,45,73,62,74,89,71]}    df2 = pd.DataFrame(d)  df2 |

and the resultant dataframe will be

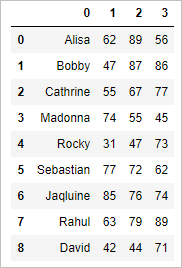


#### ****Column bind or concatenate columns of two dataframes in python pandas:****

Now lets concatenate or column bind two dataframes df1 and df2

|  |  |
| --- | --- |
| 1 | pd.concat([df1, df2], axis=1, ignore\_index=True) |

argument axis=1 binds the dataframes on column wise, so the resultant column binded dataframe will be



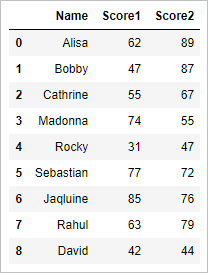
# Assign or add new column to dataframe in python pandas

In this tutorial we will learn how to assign or add new column to dataframe in python pandas.  assigning a new column the already existing dataframe in python pandas is explained with example. adding a new column the already existing dataframe in python pandas with an example

#### ****Create dataframe :****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David'],     'Score1':[62,47,55,74,31,77,85,63,42],     'Score2':[89,87,67,55,47,72,76,79,44]}    df = pd.DataFrame(d)  print df |

so the resultant dataframe will be

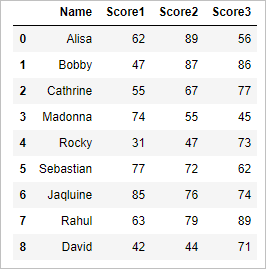


#### ****Add or assign new column to existing dataframe in python pandas****

To the above existing dataframe, lets add new column named Score3 as shown below

|  |  |
| --- | --- |
| 1  2  3 | # assign new column to existing dataframe  df2=df.assign(Score3 = [56,86,77,45,73,62,74,89,71])  print df2 |

assign() function in python, assigns the new column to existing dataframe. So the resultant dataframe will be



**groupby function in pandas – Group a dataframe in python pandas**

groupby function in pandas python: In this tutorial we will learn how to groupby in python pandas and perform aggregate functions.we will be finding the mean of a group in pandas, sum of a group in pandas python and count of a group.

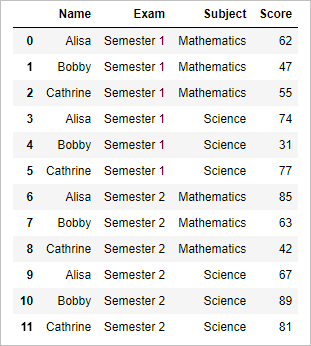
We will be working on

* getting mean score of a group using groupby function in python
* getting sum of score of a group using groupby function in python
* descriptive statistics of a group using pandas groupby function

**Create dataframe :**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Score':[62,47,55,74,31,77,85,63,42,67,89,81]}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Score'])  print df |

so the resultant dataframe will be

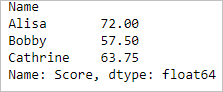


**Get mean score of a group using groupby function in pandas**

Now lets group by name of the student and find the average score of students in the following code

|  |  |
| --- | --- |
| 1  2  3 | # mean score of Students    df['Score'].groupby([df['Name']]).mean() |

result will be

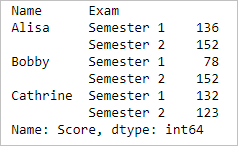


**Get sum of score of a group using groupby function in pandas**

Now lets group by name of the student and Exam and find the sum of score of students across the groups

|  |  |
| --- | --- |
| 1  2  3 | # sum of score group by Name and Exam    df['Score'].groupby([df['Name'],df['Exam']]).sum() |

so the result will be

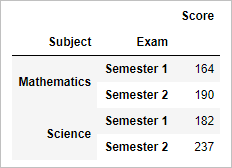


**Group the entire dataframe by Subject and Exam:**

Now lets group the entire dataframe by subject and exam and then find the sum of score of students

|  |  |
| --- | --- |
| 1  2  3 | # group the entire dataframe by Subject and  Exam    df.groupby(['Subject', 'Exam']).sum() |

so the result will be

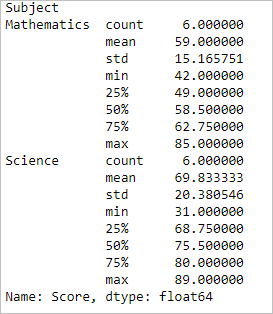


**Descriptive statistics of the group :**

Now lets group by subject and find the descriptive statistics of that group as shown below

|  |  |
| --- | --- |
| 1  2  3 | # descriptive statistics by group - subject    df['Score'].groupby(df['Subject']).describe() |

so the result will be



**Sort the List in python using sort() Function**

Sort the List in python: sort() Function in python sorts the element of given list in either ascending order or descending order. sort the list by its length. sort the list by list.sort() function and sorted() function.

**Syntax of sort() function:**

1. list.sort(key=…, reverse=…)

2. sorted(list, key=…, reverse=…)

**Sort the list in python : Ascending order:**

**Method 1:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | List1=[5,6,3,1,2,7,4]  List2=['alex','zampa','micheal','jack','milton']    # sort List1 and List2 in Ascending order  sorted(List1)  sorted(List2) |

In the above code we sorted list1 and list2 in ascending order with the help of sorted() function, so the output will be

|  |  |
| --- | --- |
| 1  2 | [1, 2, 3, 4, 5, 6, 7]  ['alex', 'jack', 'micheal', 'milton', 'zampa'] |

**Method 2:**

Sort list in ascending order with List.sort() Function

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | List1=[5,6,3,1,2,7,4]  List2=['alex','zampa','micheal','jack','milton']    # sort List1 in Ascending order  List1.sort()  print List1    # sort List2 in Ascending order  List2.sort()  print List2 |

**NOTE: List.sort() Function sorts the original list**

so the output will be

[1, 2, 3, 4, 5, 6, 7]

[‘alex’, ‘jack’, ‘micheal’, ‘milton’, ‘zampa’]

**Sort the list in python: Descending order:**

**Method 1:**

Now let’s sort list1 and list2 in descending order with the help of sorted() function, so the output will be

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | List1=[5,6,3,1,2,7,4]  List2=['alex','zampa','micheal','jack','milton']      # sort List1 and List2 in Descending order    sorted(List1,reverse=True)  sorted(List2,reverse=True) |

reverse= True argument in sorted function sorts the list in descending order

so the output will be

[7, 6, 5, 4, 3, 2, 1]

[‘zampa’, ‘milton’, ‘micheal’, ‘jack’, ‘alex’]

**Method 2:**

Sort list in Descending order with List.sort() Function

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | List1=[5,6,3,1,2,7,4]  List2=['alex','zampa','micheal','jack','milton']      # sort List1 in descending order  List1.sort(reverse=True)  print List1      # sort List2 in descending order  List2.sort(reverse=True)  print List2 |

**NOTE: List.sort() Function sorts the original list**

so the output will be

[7, 6, 5, 4, 3, 2, 1]

[‘zampa’, ‘milton’, ‘micheal’, ‘jack’, ‘alex’]

**Sort the list based on length:**

Lets sort list by length of the elements in the list

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | List2=['alex','zampa','micheal','jack','milton']      # sort the List2 by descending order of its length  List2.sort(reverse=True,key=len)  print List2 |

in the above example we sort the list by descending order of its length, so the output will be

[‘micheal’, ‘milton’, ‘zampa’, ‘alex’, ‘jack’]

**sort a dataframe in python pandas – By single & multiple column**

How to sort a dataframe in python pandas by ascending order and by descending order on multiple columns with an example for each . our focus on this exercise will be on

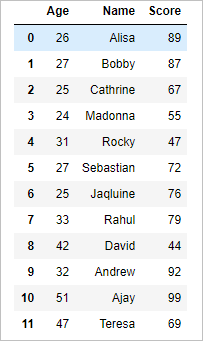
* how to sort a pandas dataframe in python by Ascending and Descending
* how to sort a python pandas dataframe by single column
* how to sort a pandas dataframe by multiple columns.

**Let’s try with an example:**

**Create a dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np    #Create a Dictionary of series  d = {'Name':pd.Series(['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa']),     'Age':pd.Series([26,27,25,24,31,27,25,33,42,32,51,47]),     'Score':pd.Series([89,87,67,55,47,72,76,79,44,92,99,69])}    #Create a DataFrame  df = pd.DataFrame(d)  print df |

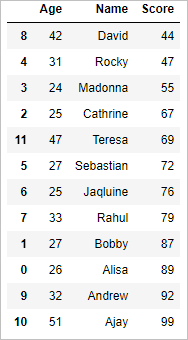
the resultant dataframe will be



**Sort the python pandas Dataframe by single column – Ascending order:**

The following code sorts the pandas dataframe by ascending values of the column **Score**

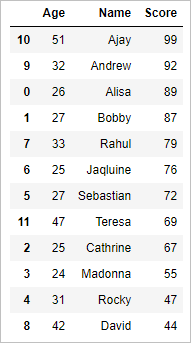
|  |  |
| --- | --- |
| 1  2  3 | # sort the pandas dataframe by ascending value of single column    df.sort\_values(by='Score') |



**Sort a Dataframe in python pandas by single Column – descending order**

The following code sorts the pandas dataframe by descending values of the column **Score**

|  |  |
| --- | --- |
| 1  2  3 | # sort the pandas dataframe by descending value of single column    df.sort\_values(by='Score',ascending=0) |

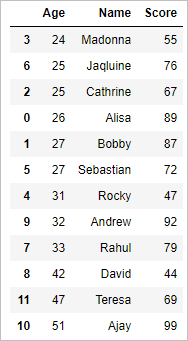


**Sort the pandas Dataframe by Multiple Columns**

In the following code, we will sort the pandas dataframe by multiple columns (Age, Score).

We will first sort with Age by ascending order and then with Score by descending order

|  |  |
| --- | --- |
| 1  2  3 | # sort the pandas dataframe by multiple columns    df.sort\_values(by=['Age', 'Score'],ascending=[True,False]) |



Also refer [sort a pandas dataframe by index](http://www.datasciencemadesimple.com/sort-the-dataframe-in-python-pandas-by-index/)

**sort the dataframe in python pandas by index**

How to sort the dataframe in python pandas by index in terms of ascending and descending order with an example for each using sort\_index() method, our focus on this exercise will be on

* how to sort a pandas dataframe in python by index in Ascending order
* how to sort a pandas dataframe in python by index in Descending order

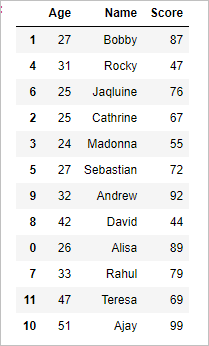
we will be using **sort\_index()** method, by passing the axis arguments and the order of sorting, DataFrame can be sorted. By default, sorting is done in ascending order.

**Let’s try with an example:**

**Create a dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Madonna','Rocky','Sebastian','Jaqluine',     'Rahul','David','Andrew','Ajay','Teresa'],     'Age':[26,27,25,24,31,27,25,33,42,32,51,47],     'Score':[89,87,67,55,47,72,76,79,44,92,99,69]}    df = pd.DataFrame(d)  df=df.reindex([1,4,6,2,3,5,9,8,0,7,11,10])  print df |

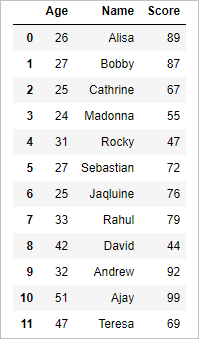
the resultant dataframe will be



**sort the dataframe in python pandas by index in ascending order:**

In the following code, we will sort the pandas dataframe by index in ascending order

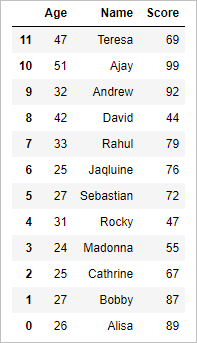
|  |  |
| --- | --- |
| 1  2  3 | # sort the pandas dataframe by index ascending    df1=df.sort\_index() |



**Sorting pandas dataframe by index in descending order:**

In the following code, we will sort the pandas dataframe by index in descending order

|  |  |
| --- | --- |
| 1  2  3 | # sort the pandas dataframe by index descending    df2=df.sort\_index(ascending=0) |



kindly refer on [sorting dataframe in python by column](http://www.datasciencemadesimple.com/sorting-pandas-dataframe-python-single-multiple-column/)

**Cross tab in python pandas (cross table)**

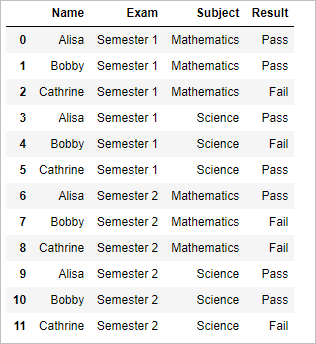
In this tutorial we will learn how to create cross tab in python pandas  ( 2 way cross table or  3 way cross table or contingency table) with example. We will learn how to create.

* 2 way cross table or contingency table in python pandas
* 3 way cross table or contingency table in python pandas

**Create dataframe :**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Result':['Pass','Pass','Fail','Pass','Fail','Pass','Pass','Fail','Fail','Pass','Pass','Fail']}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Result'])  df |

so the resultant dataframe will be

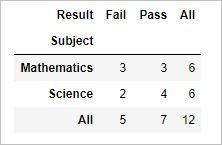


**2 Way Cross table in python pandas:**

We will calculate the cross table of subject and result as shown below

|  |  |
| --- | --- |
| 1  2  3 | # 2 way cross table    pd.crosstab(df.Subject, df.Result,margins=True) |

margin=True displays the row wise and column wise sum of the cross table  so the output will be

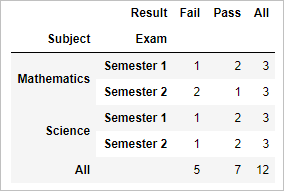


**3 Way Cross table in python pandas:**

We will calculate the cross table of subject, Exam and result as shown below

|  |  |
| --- | --- |
| 1  2  3 | # 3 way cross table    pd.crosstab([df.Subject, df.Exam],df.Result, margins=True) |

the result will be



**Rank the dataframe in python pandas – (min, max, dense & rank by group)**

In this tutorial we will learn how to rank the dataframe in python pandas by ascending and descending order with maximum rank value, minimum rank value , average rank value and dense rank . We will see an example for each.  We will be ranking the dataframe on row wise on different methods

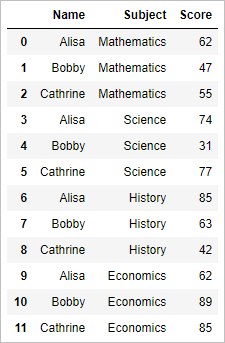
In this tutorial we will be dealing with following examples

* Rank the dataframe by ascending and descending order
* Rank the dataframe by dense rank if found 2 values are same
* Rank the dataframe by Maximum rank if found 2 values are same
* Rank the dataframe by Minimum rank if found 2 values are same
* Rank the dataframe by group

**Create data frame:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {  'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',  'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],  'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',  'History','History','History','Economics','Economics','Economics'],  'Score':[62,47,55,74,31,77,85,63,42,62,89,85]}    df = pd.DataFrame(d,columns=['Name','Subject','Score'])  df |

Resultant dataframe will be

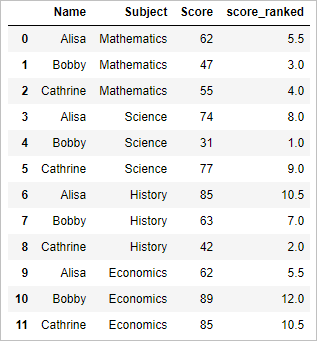


**Ranking the dataframe in python pandas on ascending order:**

Now lets rank the dataframe in ascending order of score as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # Ranking of score ascending order    df['score\_ranked']=df['Score'].rank(ascending=1)  df |

so the result will be

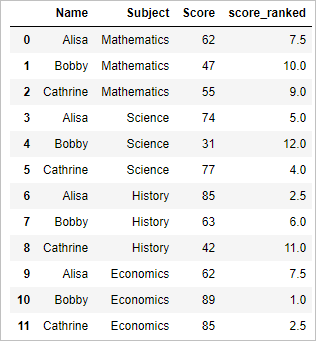


**Ranking the dataframe in python pandas on descending order:**

rank the dataframe in descending order of score as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # Ranking of score descending order    df['score\_ranked']=df['Score'].rank(ascending=0)  df |

so the result will be



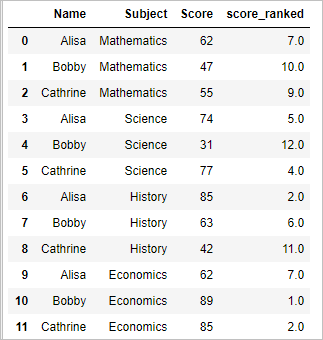
**Rank the dataframe in python pandas by minimum value of the rank**

rank the dataframe in descending order of score and if found two scores are same then assign the minimum rank to both the score as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # Ranking of score in descending order by minimum value    df['score\_ranked']=df['Score'].rank(ascending=0,method='min')  df |

in this example score 62 is found twice and is ranked by minimum value of 7

so the result will be



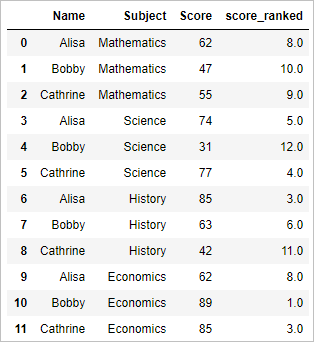
**Rank the dataframe in python pandas by maximum value of the rank**

rank the dataframe in descending order of score and if found two scores are same then assign the maximum rank to both the score as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # Ranking of score in descending order by maximum value    df['score\_ranked']=df['Score'].rank(ascending=0,method='max')  df |

In this example score 62 is found twice and is ranked by maximum value of 8

so the result will be

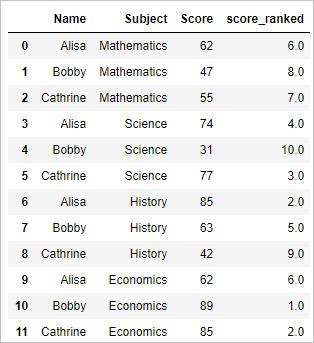


**Rank the dataframe in python pandas by dense rank**

rank the dataframe in descending order of score and if found two scores are same then assign the same rank . Dense rank does not skip any rank (in min and max ranks are skipped)

|  |  |
| --- | --- |
| 1  2  3  4 | # Ranking of score in descending order by dense    df['score\_ranked']=df['Score'].rank(ascending=0,method='dense')  df |

so the result will be

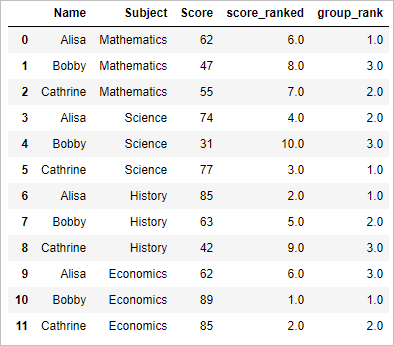


**Rank the dataframe in python pandas by Group**

rank the dataframe in descending order of score by subject . so ranking is done by subject wise

|  |  |
| --- | --- |
| 1  2  3  4 | # Rank by Group    df["group\_rank"] = df.groupby("Subject")["Score"].rank(ascending=0,method='dense')  df |

so the result will be



**Delete or Drop the duplicate row of a dataframe in python pandas**

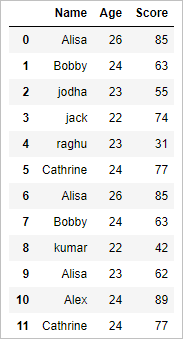
In this tutorial we will learn how to delete or drop the duplicate row of a dataframe in python pandas with example using drop\_duplicates() function. lets learn how to

* Drop the duplicate rows
* Drop the duplicate by a column name

**Create dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],        'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

so the resultant dataframe will be

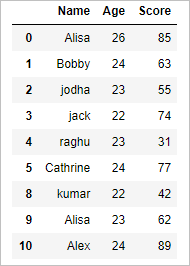


**Drop the duplicate rows:**

Now lets simply drop the duplicate rows in pandas as shown below

|  |  |
| --- | --- |
| 1  2  3 | # drop duplicate rows    df.drop\_duplicates() |

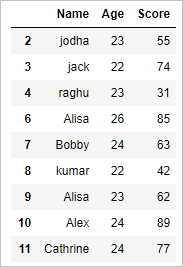
In the above example first occurrence of the duplicate row is kept and subsequent occurrence will be deleted, so the output will be



**Drop the duplicate by retaining last occurrence:**

|  |  |
| --- | --- |
| 1  2  3 | # drop duplicate rows    df.drop\_duplicates(keep='last') |

In the above example keep=’last’ argument . Keeps the last duplicate row and delete the rest duplicated rows. So the output will be



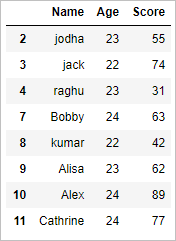
**Drop the duplicate by column:**

Now let’s drop the rows by column name. Rows are dropped in such a way that unique column value is retained for that column as shown below

|  |  |
| --- | --- |
| 1  2  3 | # drop duplicate by a column name    df.drop\_duplicates(['Name'], keep='last') |

In the above example rows are deleted in such a way that, **Name** column contains only unique values

So the result will be



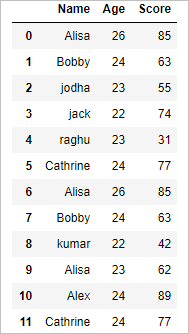
# Find the duplicate rows of the dataframe in python pandas

In this tutorial we will learn how to find the duplicate rows of the dataframe in python pandas with duplicated() Function. Lets see with an example.

We will be marking the row as TRUE if it is duplicate and FALSE if it is not duplicate. Let’s try with an example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | # import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

so the resultant dataframe will be



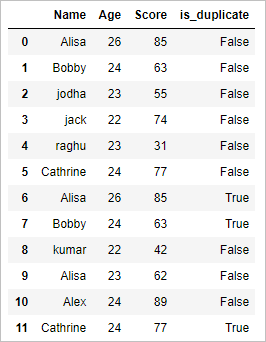
#### ****Find the duplicate row in pandas:****

duplicated() function is used for find the duplicate rows of the dataframe in python pandas

|  |  |
| --- | --- |
| 1  2  3 | df["is\_duplicate"]= df.duplicated()    df |

The above code finds whether the row is duplicate and tags TRUE if it is duplicate and tags FALSE if it is not duplicate. And assigns it to the column named “**is\_duplicate”**  of the dataframe  df.

So the resultant dataframe will be



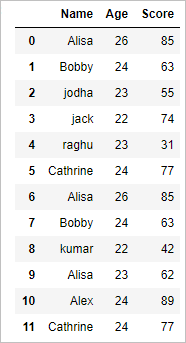
# Drop or delete the row in python pandas with conditions

In this tutorial we will learn how to drop or delete the row in python pandas by index, delete row by condition in python pandas and delete the row in python pandas  by position. Lets see example of each.

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

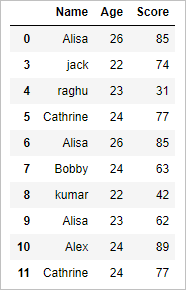
the dataframe will be



#### ****Simply drop a row or observation:****

|  |  |
| --- | --- |
| 1  2 | # Drop an observation or row  df.drop([1,2]) |

The above code will drop the second and third row. So the resultant dataframe will be

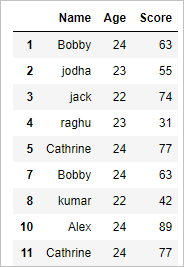


#### ****Drop a row or observation by condition:****

we can drop a row when it satisfies a specific condition

|  |  |
| --- | --- |
| 1  2 | # Drop a row by condition  df[df.Name != 'Alisa'] |

The above code takes up all the names except Alisa, thereby dropping the row with name ‘Alisa’. So the resultant dataframe will be

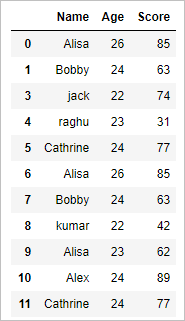


#### ****Drop a row or observation by index:****

We can drop a row by index as shown below

|  |  |
| --- | --- |
| 1  2 | # Drop a row by index  df.drop(df.index[2]) |

The above code drops the row with index number 2. So the resultant dataframe will be

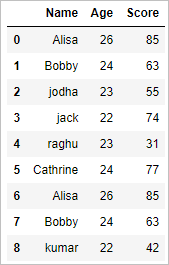


#### ****Drop the row by position:****

Now let’s drop the bottom 3 rows of a dataframe as shown below

|  |  |
| --- | --- |
| 1  2 | # Drop bottom 3 rows  df[:-3] |

The above code selects all the rows except bottom 3 rows, there by dropping bottom 3 rows, so the resultant dataframe will be



# Drop or delete column in python pandas

In this tutorial we will learn how to  drop or delete column in python pandas by index,  drop column in pandas by name and drop column in python pandas by position. Lets see example of each.

#### ****Create Dataframe****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

The resultant dataframe will be

#### ****Drop a column by name:****

Lets see an example of how to drop a column by name in python pandas

|  |  |
| --- | --- |
| 1  2  3 | # drop a column based on name    df.drop('Age',axis=1) |

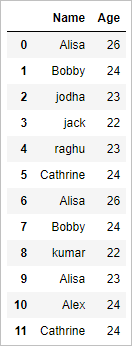
The above code drops the column named ‘Age’, the argument axis=1 denotes column, so the resultant dataframe will be

#### ****Drop a column based on column index:****

Let’s see an example on dropping the column by its index in python pandas

|  |  |
| --- | --- |
| 1  2  3 | # drop a column based on column index    df.drop(df.columns[2],axis=1) |

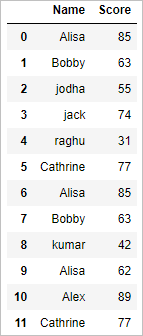
In the above example column with index 2 is dropped(3rd column). So the resultant dataframe will be



#### ****Delete a column based on column name:****

|  |  |
| --- | --- |
| 1  2  3  4 | # delete a column    del df['Age']  df |

In the above example column with the name ‘Age’ is deleted. So the resultant dataframe will be



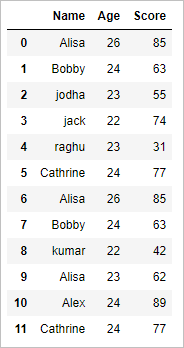
# Get the maximum value of column in python pandas

Get the maximum value of column in python pandas : In this tutorial we will learn How to get the maximum value of all the columns in dataframe of python pandas. How to get the maximum value of a specific column in python pandas using max() function**.**

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {  'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',  'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],  'Age':[26,24,23,22,23,24,26,24,22,23,24,24],    'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

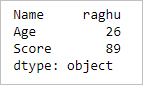
So the resultant dataframe will be



#### ****Get the maximum value of all the column in python pandas:****

|  |  |
| --- | --- |
| 1  2  3 | # get the maximum values of all the column in dataframe    df.max() |

This gives the list of all the column names and its maximum value, so the output will be



#### ****Get the maximum value of a specific column in python pandas:****

#### ****Example 1:****

|  |  |
| --- | --- |
| 1  2  3 | # get the maximum value of the column 'Age'    df['Age'].max() |

This gives the maximum value of column Age so the output will be

26

#### ****Example 2:****

|  |  |
| --- | --- |
| 1  2  3 | # get the maximum value of the column 'Name'    df['Name'].max() |

This gives the maximum value of column Name so the output will be

‘raghu’

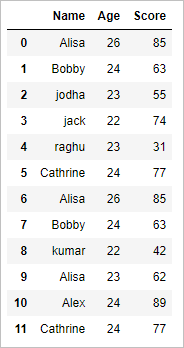
# Get the minimum value of column in python pandas

Get the minimum value of column in python pandas : In this tutorial we will learn How to get the minimum value of all the columns in dataframe of python pandas. How to get the minimum value of a specific column in python pandas using min() function**.**

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {  'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',  'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],  'Age':[26,24,23,22,23,24,26,24,22,23,24,24],    'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

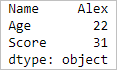
So the resultant dataframe will be



#### ****Get the minimum value of all the column in python pandas:****

|  |  |
| --- | --- |
| 1  2  3 | # get the minimum values of all the column in dataframe    df.min() |

This gives the list of all the column names and its minimum value, so the output will be



#### ****Get the minimum value of a specific column in python pandas:****

#### ****Example 1:****

|  |  |
| --- | --- |
| 1  2  3 | # get the minimum value of the column 'Age'    df['Age'].min() |

This gives the minimum value of column Age so the output will be

22

#### ****Example 2:****

|  |  |
| --- | --- |
| 1  2  3 | # get the minimum value of the column 'Name'    df['Name'].min() |

This gives the minimum value of column Name so the output will be

‘Alex’

**select row with maximum and minimum value in python pandas**

In this tutorial we will learn how to select row with maximum and minimum value in python pandas

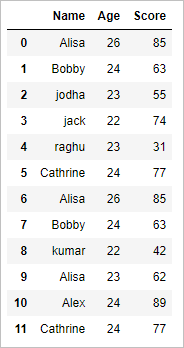
* Get the entire row which has the maximum value of a column in python pandas
* Get the entire row which has the minimum value of a column in python pandas.

Let’s see example of both

**Create dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

So the resultant dataframe will be



**Get the entire row which has the maximum value in python pandas:**

So let’s extract the entire row where score is maximum i.e. get all the details of student with maximum score as shown below

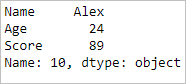
|  |  |
| --- | --- |
| 1  2 | # get the row of max value  df.loc[df['Score'].idxmax()] |

**Explanation:**

**df[‘Score’].idxmax()** **– >**returns the index of the row where column name **“Score”** has maximum value.

**df.loc[]** **->** returns the row of that index

so the output will be

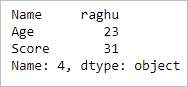


**Get the entire row which has the minimum value in python pandas:**

So let’s extract the entire row where score is minimum i.e. get all the details of student with minimum score as shown below

|  |  |
| --- | --- |
| 1  2 | # get the row of minimum value  df.loc[df['Score'].idxmin()] |

so the output will be

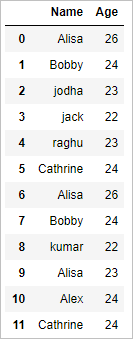


# Get the unique values (rows) of a dataframe in python Pandas

In this tutorial we will learn how to get the unique values (rows) of a dataframe in python pandas with drop\_duplicates() function.  Lets see with an example

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | Create Dataframe:  import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24]  }    df = pd.DataFrame(d,columns=['Name','Age'])  print df |

so the output will be



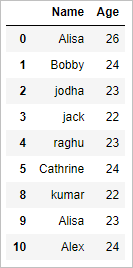
#### ****Get the unique values (rows) of the dataframe in python pandas****

drop\_duplicates() function is used to get the unique values (rows) of the dataframe in python pandas.

|  |  |
| --- | --- |
| 1  2 | # get the unique values (rows)  print df.drop\_duplicates() |

The above drop\_duplicates() function removes all the duplicate rows and returns only unique rows. Generally it retains the first row when duplicate rows are present.

So the output will be

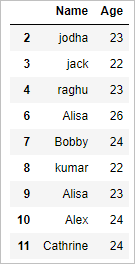


#### ****Get the unique values (rows) of the dataframe in python pandas by retaining last row:****

|  |  |
| --- | --- |
| 1  2 | # get the unique values (rows) by retaining last row  print df.drop\_duplicates(keep='last') |

The above drop\_duplicates() function with keep =’last’ argument,  removes all the duplicate rows and returns only unique rows by retaining the last row when duplicate rows are present.

So the output will be



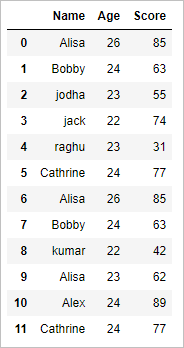
# Get the list of column headers or column name in python pandas

In this tutorial we will learn how to get the list of column headers or column name in python pandas using list() function.  Lets see with an example

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

so the output will be



#### ****Get the list of column headers or column name:****

#### ****Method 1:****

|  |  |
| --- | --- |
| 1  2 | # method 1: get list of column name  list(df.columns.values) |

The above function gets the column names and converts them to list. So the output will be

[‘Name’, ‘Age’, ‘Score’]

**Method 2:**

|  |  |
| --- | --- |
| 1  2 | # method 2: get list of column name  list(df) |

The above function gets list of column name. The output will be

[‘Name’, ‘Age’, ‘Score’]

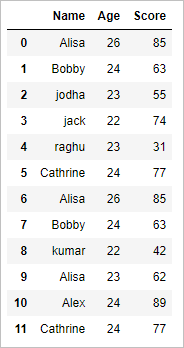
# Get unique values of a column in python pandas

In this tutorial we will learn how to get unique values of a column in python pandas using unique() function .  Lets see with an example

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','jodha','jack','raghu','Cathrine',              'Alisa','Bobby','kumar','Alisa','Alex','Cathrine'],      'Age':[26,24,23,22,23,24,26,24,22,23,24,24],           'Score':[85,63,55,74,31,77,85,63,42,62,89,77]}    df = pd.DataFrame(d,columns=['Name','Age','Score'])  df |

so the output will be



#### ****Get the unique values of a column:****

Lets get the unique values of “Name” column

|  |  |
| --- | --- |
| 1 | df.Name.unique() |

The unique() function gets the list of unique column values . So the output will be

array([‘Alisa’, ‘Bobby’, ‘jodha’, ‘jack’, ‘raghu’, ‘Cathrine’, ‘kumar’,

‘Alex’], dtype=object)

Get the unique values of “Age” column

|  |  |
| --- | --- |
| 1 | df.Age.unique() |

so the output will be

array([26, 24, 23, 22], dtype=int64)

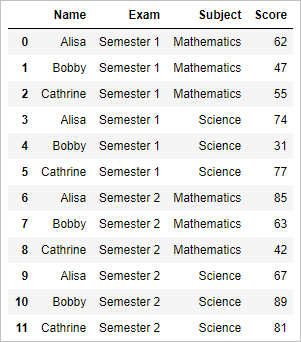
# Hierarchical indexing or multiple indexing in python pandas

Let’s see how to create Hierarchical indexing or multiple indexing in python pandas dataframe. We will be converting a normal dataframe to hierarchical dataframe. Lets see with an example

#### ****Create Dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Score':[62,47,55,74,31,77,85,63,42,67,89,81]}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Score'])  df |

so the resultant dataframe will be

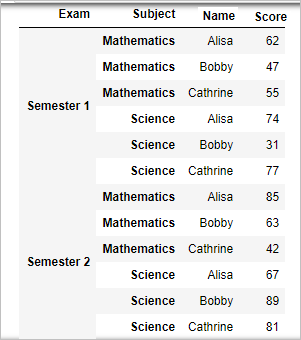


#### ****Hierarchical indexing or multiple indexing in python pandas:****

|  |  |
| --- | --- |
| 1  2  3  4 | # multiple indexing or hierarchical indexing    df1=df.set\_index(['Exam', 'Subject'])  df1 |

set\_index() Function is used for indexing , First the data is indexed on Exam and then on Subject column

So the resultant dataframe will be a hierarchical dataframe as shown below



#### ****View Index:****

One can view the details of index as shown below

|  |  |
| --- | --- |
| 1  2 | # View index  df1.index |

So the result will be

MultiIndex(levels=[[‘Semester 1’, ‘Semester 2’], [‘Mathematics’, ‘Science’]],labels=[[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1], [0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1]],

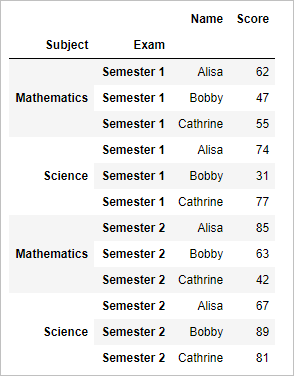
names=[‘Exam’, ‘Subject’])

#### ****Swap the column in the hierarchical index:****

Now let’s swap the “Subject” and “Exam” columns in the above hierarchical dataframe as shown below

|  |  |
| --- | --- |
| 1  2 | # Swap the column  in multiple index  df1.swaplevel('Subject','Exam') |

So the resultant swapped hierarchical dataframe will be

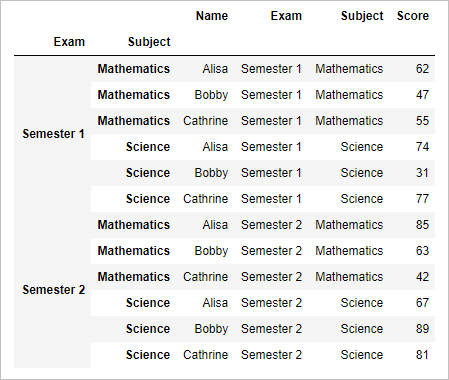


#### ****Hierarchical indexing or multiple indexing in python pandas without dropping:****

Now lets create a hierarchical dataframe by multiple indexing without dropping those columns

So all those columns will again appear

|  |  |
| --- | --- |
| 1  2  3  4 | # multiple indexing or hierarchical indexing with drop=False    df1=df.set\_index(['Exam', 'Subject'],drop=False)  df1 |



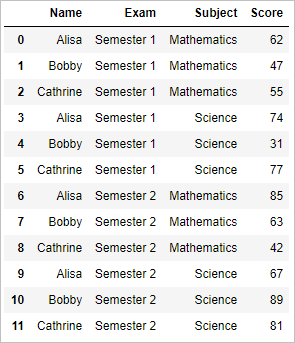
# Index, Select and Filter dataframe in pandas python

Index, Select and Filter dataframe in pandas python – In this tutorial we will learn how to index the dataframe in pandas python with example, How to select and filter the dataframe in pandas python with column name and column index using .ix(), .iloc() and .loc()

#### ****Create dataframe :****

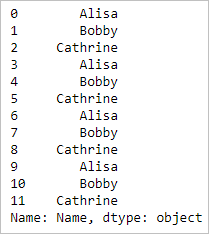
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Score':[62,47,55,74,31,77,85,63,42,67,89,81]}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Score'])  df |

so the resultant dataframe will be



#### ****View a column of the dataframe in pandas python:****

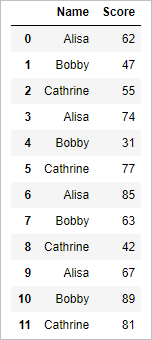
|  |  |
| --- | --- |
| 1 | df['Name'] |



#### ****View two columns of the dataframe in pandas:****

|  |  |
| --- | --- |
| 1 | df[['Name', 'Score']] |

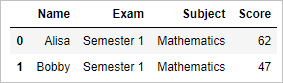
**Output:**



#### ****View first two rows of the dataframe in pandas:****

|  |  |
| --- | --- |
| 1 | df[:2] |

**Output:**

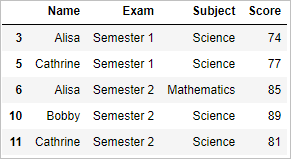


#### ****Filter in Pandas dataframe:****

#### ****View all rows where score greater than 70****

|  |  |
| --- | --- |
| 1 | df[df['Score'] > 70] |

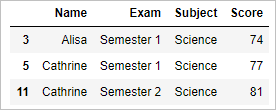
#### ****Output:****



#### ****View all the rows where score greater than 70 and less than 85****

|  |  |
| --- | --- |
| 1 | df[(df['Score'] > 70) & (df['Score'] < 85)] |

#### ****Output:****



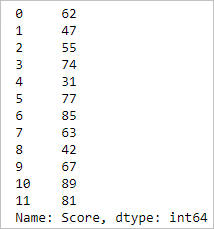
#### ****Indexing with .ix:****

.ix[] is used to index a dataframe by both name and position

#### ****View a column in pandas****

|  |  |
| --- | --- |
| 1 | df.ix[:,'Score'] |

#### ****Output:****



#### ****View the value based on row and column****

|  |  |
| --- | --- |
| 1 | df.ix[3,2] |

#### ****Output:****

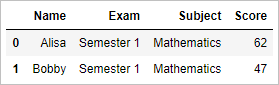
‘Science’

#### ****select row by using row number in pandas  with .iloc****

**.iloc [1:m, 1:n]**–  is used to select or index rows based on their position from 1 to m rows and 1 to n columns

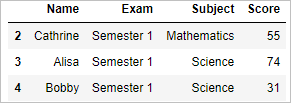
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # select first 2 rows    df.iloc[:2]    # or    df.iloc[:2,] |

#### ****output:****



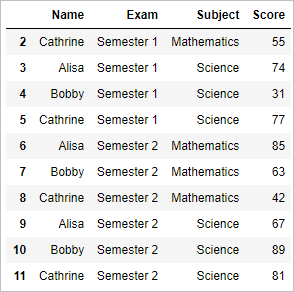
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # select 3rd to 5th rows      df.iloc[2:5]    # or    df.iloc[2:5,] |

#### ****output:****



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # select all rows starting from third row    df.iloc[2:]    # or    df.iloc[2:,] |

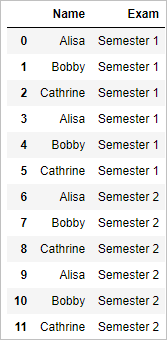
#### ****output:****



#### ****Select column by using column number in pandas with .iloc****

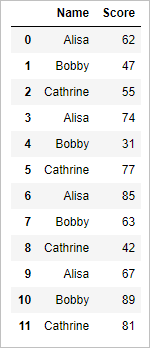
|  |  |
| --- | --- |
| 1  2 | # select first 2 columns  df.iloc[:,:2] |

#### ****output:****



|  |  |
| --- | --- |
| 1  2  3 | # select first 1st and 4th columns    df.iloc[:,[0,3]] |

#### ****output:****

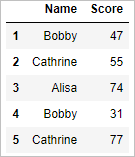


#### ****Select value by using row name and column name in pandas with .loc:****

**.loc [[Row\_names],[ column\_names]]**–  is used to select or index rows or columns  based on their name

|  |  |
| --- | --- |
| 1  2  3 | # select value by row label and column label using loc    df.loc[[1,2,3,4,5],['Name','Score']] |

#### ****output:****



# Indexing with iloc, loc and ix in pandas python

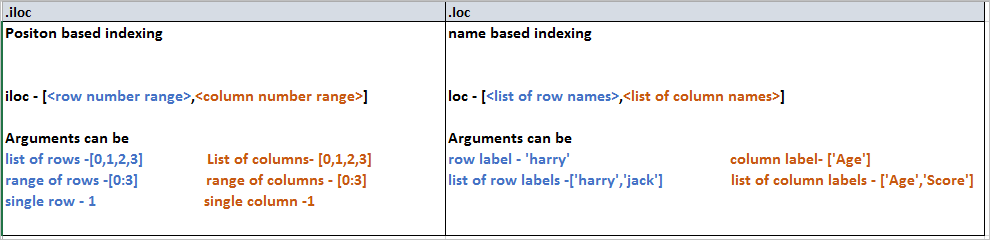
**Indexing in pandas python is done mostly with the help of iloc, loc and ix.  lets see an example of each .**

**iloc –**iloc is used for indexing or selecting based on position .i.e. by row number and column number

**loc –** loc is used for indexing or selecting based on name .i.e. by row name and column name

**ix –**indexing can be done by both position and name using ix.

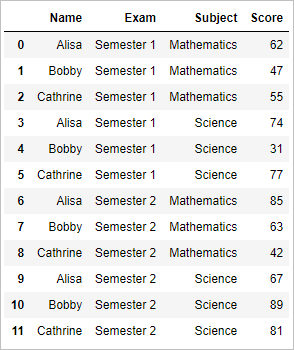
#### ****loc Vs iloc:****



**Create dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Score':[62,47,55,74,31,77,85,63,42,67,89,81]}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Score'])  df |

so the resultant dataframe will be



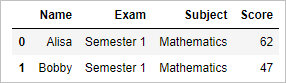
#### ****Indexing with iloc:****

**.iloc [1:m, 1:n]**–  is used to select or index rows based on their position from 1 to m rows and 1 to n columns

**Select row by using row number in pandas with .iloc**

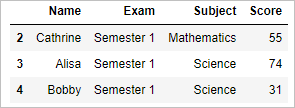
|  |  |
| --- | --- |
| 1  2  3  4 | # select first 2 rows  df.iloc[:2]  # or  df.iloc[:2,] |

**output:**



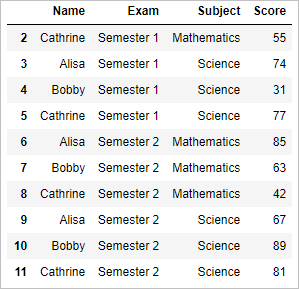
|  |  |
| --- | --- |
| 1  2  3  4 | # select 3rd to 5th rows  df.iloc[2:5]  # or  df.iloc[2:5,] |

**output:**



|  |  |
| --- | --- |
| 1  2  3  4 | # select all rows starting from third row  df.iloc[2:]  # or  df.iloc[2:,] |

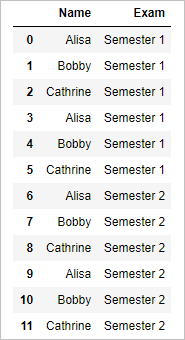
**output:**



#### ****Select column by using column number in pandas with .iloc****

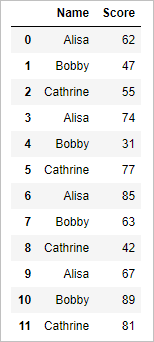
|  |  |
| --- | --- |
| 1  2  3 | # select first 2 columns    df.iloc[:,:2] |

**output:**



|  |  |
| --- | --- |
| 1  2 | # select 1st and 4thcolumn  df.iloc[:,[0,3]] |

**output:**



#### ****Select values by using .iloc****

Select 2nd row and 3rd column value

|  |  |
| --- | --- |
| 1  2  3 | # Select 2nd row and 3rd column value    df.iloc[1,2] |

**output:**

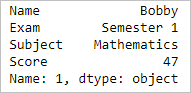
‘Mathematics’

#### ****indexing with loc :****

**loc [[Row\_names],[ column\_names]]**–  is used to select or index rows or columns  based on their name

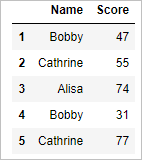
|  |  |
| --- | --- |
| 1  2  3 | # select row by now name    df.loc[1] |

In the dataframe df has default row names from 1 to 11. So **df.loc[1]** denotes selecting the row by row name 1**,**so output will be



|  |  |
| --- | --- |
| 1  2  3 | # select value by row label and column label using loc    df.loc[[1,2,3,4,5],['Name','Score']] |

#### ****output:****



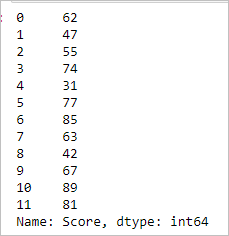
#### ****Indexing with ix****

Indexing with ix is the combination of indexing with iloc and loc. ix is used for indexing based on the position and as well by names.

#### ****View a column in pandas****

|  |  |
| --- | --- |
| 1  2  3 | # Get all values of column ‘Score’    df.ix[:,'Score'] |

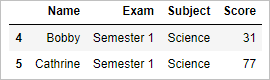
#### ****Output:****



#### ****View a row based on row numbers****

|  |  |
| --- | --- |
| 1  2  3 | # Get 3rd & 4th row    df.ix[4:5,] |

#### ****Output:****



#### ****View the value based on row and column number****

|  |  |
| --- | --- |
| 1 | df.ix[3,2] |

#### ****Output:****

‘Science’

**Reshape wide to long in pandas python with melt() function**

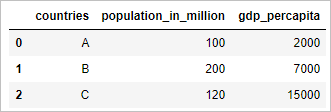
Reshaping a data from wide to long in pandas python is done with melt() function. melt function in pandas is one of the efficient function to transform the data from wide to long format. melt() Function in python pandas depicted with an example.

Let’s create a simple data frame to demonstrate our reshape example in python pandas.

**Create dataframe:**

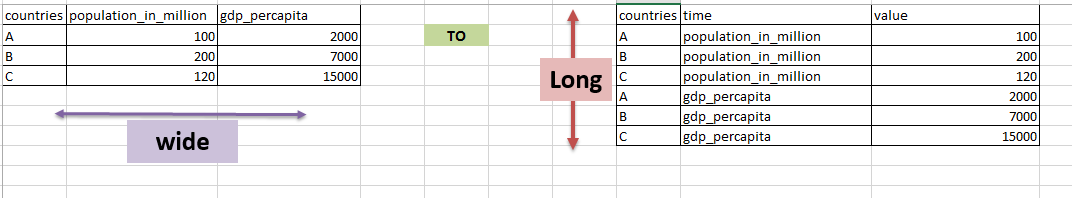
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'countries':['A','B','C'],      'population\_in\_million':[100,200,120],      'gdp\_percapita':[2000,7000,15000]      }    df = pd.DataFrame(d,columns=['countries','population\_in\_million','gdp\_percapita'])  df |

The dataframe will be like



**Reshape wide to long  in pandas python with melt() function:**

We will reshape the above data frame from wide to long format in R. The above data frame is already in wide format.

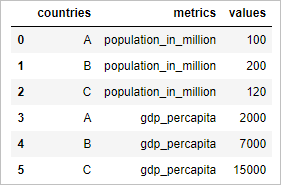


This can be accomplished with below code

|  |  |
| --- | --- |
| 1  2  3  4 | # shape from wide to long with melt function in pandas    df2=pd.melt(df,id\_vars=['countries'],var\_name='metrics', value\_name='values')  df2 |

* data frame “df” is passed to melt() function
* id\_vars is the variable which need to be left unaltered which is “countries”
* var\_name are the column names so we named it as ‘metrics’
* value\_name are its values so we named it as ‘values’

so the reshaped dataframe will be



**Reshape long to wide in pandas python with pivot function**

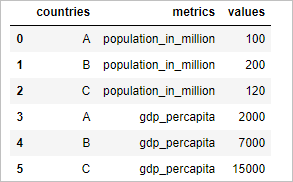
Reshaping a data from long to wide in python pandas is done with pivot() function. Pivot() function in pandas is one of the efficient function to transform the data from long to wide format. pivot() Function in python pandas depicted with an example.

Let’s create a simple data frame to demonstrate our reshape example in python pandas

**Create dataframe:**

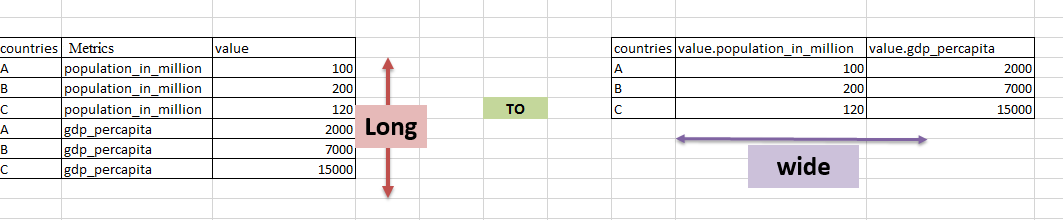
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'countries':['A','B','C','A','B','C'],      'metrics':['population\_in\_million','population\_in\_million','population\_in\_million',                               'gdp\_percapita','gdp\_percapita','gdp\_percapita'],      'values':[100,200,120,2000,7000,15000]      }    df = pd.DataFrame(d,columns=['countries','metrics','values'])  df |

The dataframe will be like



**Reshape long to wide in pandas python with pivot function:**

We will reshape the above data frame from long to wide format in R. The above data frame is already in long format.

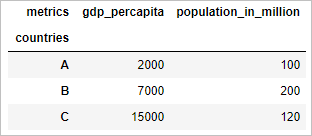


This can be accomplished with below code

|  |  |
| --- | --- |
| 1  2  3  4 | # reshape from long to wide in pandas python    df2=df.pivot(index='countries', columns='metrics', values='values')  df2 |

* Pivot function() reshapes the data from long to wide in Pandas python. Countries column is used on index.
* Values of Metrics column is used as column names and values of value column is used as its value.

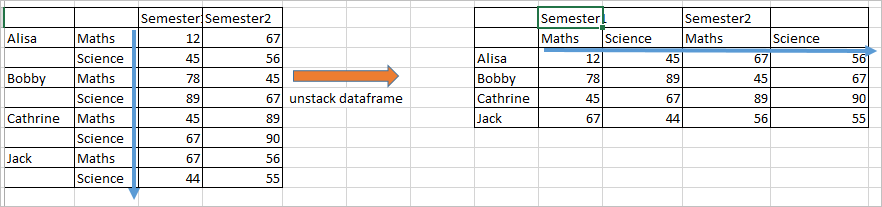
So the resultant reshaped dataframe will be



**Reshape using Stack() and unstack() function in Pandas python**

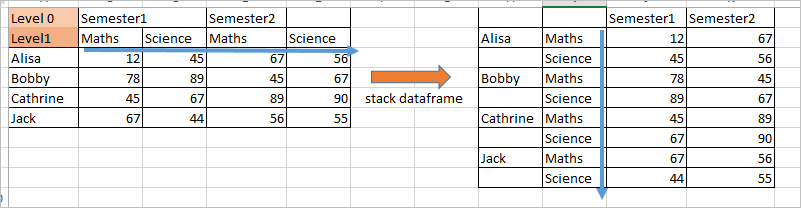
Reshape using Stack() and unstack() function in Pandas python: Reshaping the data using stack() function in pandas converts the data into stacked format .i.e. the column is stacked row wise. When more than one column header is present we can stack the specific column header by specified the level. unstack() function in pandas converts the data into unstacked format. Let’s see with an example**.**

**Stack a dataframe**



* Stacking a dataframe at level 1 will stack maths and science columns row wise
* Stacking a dataframe at level 0 will stack semester1 and semester2 columns row wise.

**Unstack a dataframe**

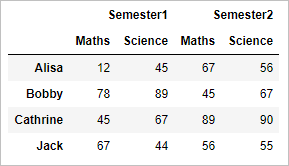


* Unstack is simply the reverse of stack

**Create multiple header dataframe:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pandas as pd  import numpy as np      header = pd.MultiIndex.from\_product([['Semester1','Semester2'],['Maths','Science']])  d=([[12,45,67,56],[78,89,45,67],[45,67,89,90],[67,44,56,55]])      df = pd.DataFrame(d,                    index=['Alisa','Bobby','Cathrine','Jack'],                    columns=header)  df |

The resultant multiple header dataframe will be

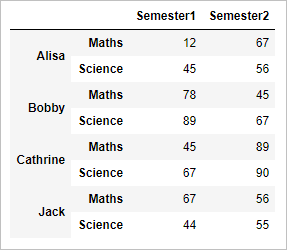


**Stack the dataframe:**

Stack() Function in dataframe stacks the column to rows at level 1 (default).

|  |  |
| --- | --- |
| 1  2  3  4  5 | # stack the dataframe      stacked\_df=df.stack()  stacked\_df |

so the stacked dataframe will be

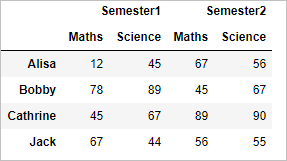


**Unstack the dataframe:**

unstack() Function in dataframe unstacks the row to columns . Basically it’s a reverse of stacking

|  |  |
| --- | --- |
| 1  2  3 | # unstack the dataframe  unstacked\_df = stacked\_df.unstack()  unstacked\_df |

so the resultant unstacked dataframe will be

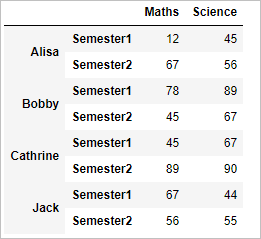


**Stack the dataframe at level 0:**

Stack() Function with level 0 argument stacks the column semester.

|  |  |
| --- | --- |
| 1  2  3  4 | # stack the dataframe of column at level 0    stacked\_df\_lvl=df.stack(level=0)  stacked\_df\_lvl |

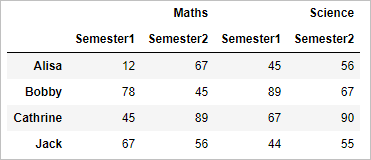
so the level 0 stacked dataframe will be



**unstack the dataframe :**

|  |  |
| --- | --- |
| 1  2  3 | # unstack the dataframe  unstacked\_df1 = stacked\_df\_lvl.unstack()  unstacked\_df1 |

so the resultant unstacked dataframe will be



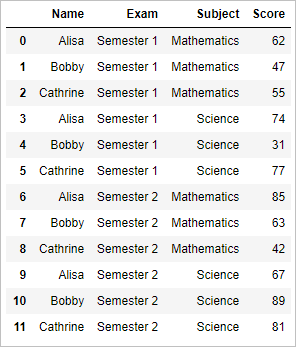
# Create Pivot table in Pandas python

In this tutorial we will be dealing on how to create pivot table from a Pandas dataframe in python with aggregate function – mean ,count and sum. Lets see how to create pivot table in pandas python with an example

#### ****Create dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pandas as pd  import numpy as np    #Create a DataFrame  d = {      'Name':['Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine',              'Alisa','Bobby','Cathrine','Alisa','Bobby','Cathrine'],      'Exam':['Semester 1','Semester 1','Semester 1','Semester 1','Semester 1','Semester 1',              'Semester 2','Semester 2','Semester 2','Semester 2','Semester 2','Semester 2'],        'Subject':['Mathematics','Mathematics','Mathematics','Science','Science','Science',                 'Mathematics','Mathematics','Mathematics','Science','Science','Science'],     'Score':[62,47,55,74,31,77,85,63,42,67,89,81]}    df = pd.DataFrame(d,columns=['Name','Exam','Subject','Score'])  df |

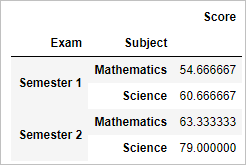
The resultant dataframe will be



#### ****Create pivot table in pandas python with aggregate function mean:****

|  |  |
| --- | --- |
| 1  2  3 | # pivot table using aggregate function mean    pd.pivot\_table(df, index=['Exam','Subject'], aggfunc='mean') |

So the pivot table with aggregate function mean will be

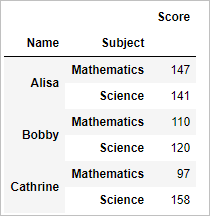


Which shows the average score of students across exams and subjects

#### ****Create pivot table in Pandas python with aggregate function sum:****

|  |  |
| --- | --- |
| 1  2  3 | # pivot table using aggregate function sum    pd.pivot\_table(df, index=['Name','Subject'], aggfunc='sum') |

So the pivot table with aggregate function sum will be

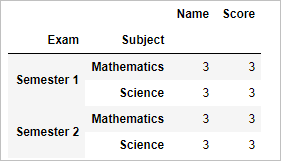


Which shows the sum of scores of students across subjects

#### ****Create pivot table in Pandas python with aggregate function count:****

|  |  |
| --- | --- |
| 1  2  3 | # pivot table using aggregate function count    pd.pivot\_table(df, index=['Exam','Subject'], aggfunc='count') |

So the pivot table with aggregate function count will be



Which shows the count of student who appeared for the exam of different subject

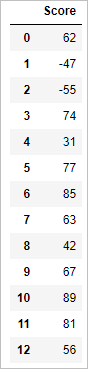
# Scaling and normalizing a column in Pandas python

Scaling and normalizing a column in pandas python is required,  to standardize the data, before we model a data. We will be using preprocessing method from scikitlearn package. Lets see an example which normalizes the column in pandas by scaling

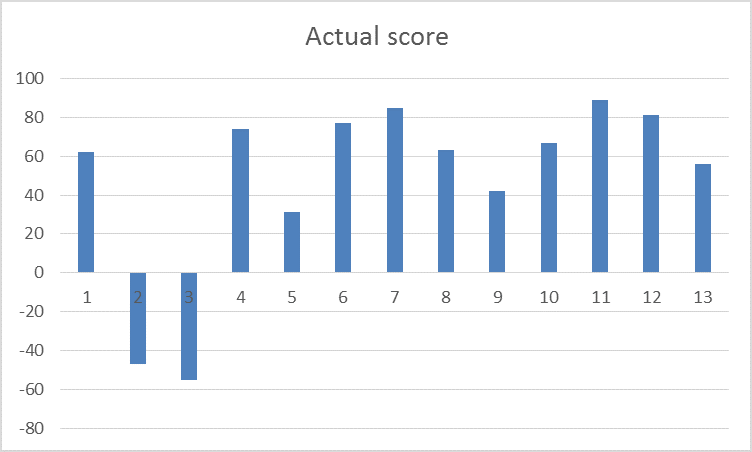
#### ****Create a single column dataframe:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | import pandas as pd  import numpy as np  from sklearn import preprocessing    # Create a DataFrame  d = {         'Score':[62,-47,-55,74,31,77,85,63,42,67,89,81,56]}    df = pd.DataFrame(d,columns=['Score'])  print df |

So the resultant dataframe will be



On plotting the score it will be



**Step 1:**  convert the column of a dataframe to float

|  |  |
| --- | --- |
| 1  2  3 | # 1.convert the column value of the dataframe as floats    float\_array = df['Score'].values.astype(float) |

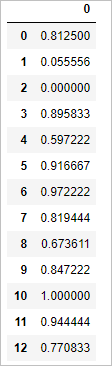
**Step 2:**  create a min max processing object. Pass the float column to the min\_max\_scaler() which scales the dataframe by processing it as shown below

|  |  |
| --- | --- |
| 1  2  3  4 | # 2. create a min max processing object    min\_max\_scaler = preprocessing.MinMaxScaler()  scaled\_array = min\_max\_scaler.fit\_transform(float\_array) |

**Step 3:**  Convert the scaled array to the dataframe.

|  |  |
| --- | --- |
| 1  2  3  4 | # 3. convert the scaled array to dataframe    df\_normalized = pd.DataFrame(scaled\_array)  df\_normalized |

so the final normalized dataframe will be



On plotting the scaled score the graph will be

