Loc and iloc are two functions in Pandas that are used to slice a data set in a Pandas DataFrame. The function .loc is typically used for label indexing and can access multiple columns, while .iloc is used for integer indexing.

Loc and iloc Access a group of rows and columns by label(s) or a boolean array.The .iloc function is integer position based, but it could also be used with a boolean array. If we want to locate a cell of the data set, we can enter:

Example: df.iloc[0,0]

This command gives the element at row = 0, and column = 0. I can also extract a slice of the data set.

**>>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],**

**... index=['cobra', 'viper', 'sidewinder'],**

**... columns=['max\_speed', 'shield'])**

**>>> df**

max\_speed shield

cobra 1 2

viper 4 5

sidewinder 7 8

**>>>df.loc['viper']**

max\_speed 4

shield 5

Name: viper, dtype: int64

Note: When a Single label is specified with loc, then the result will be generated in form of row as a Series. But when we give a list of labels, then a DataFrame is returned

**>>> df.loc[['viper', 'sidewinder']]**

max\_speed shield

viper 4 5

sidewinder 7 8

**>>> df.loc['cobra', 'shield']**

2

In this example, a Slice is passed as single labels **(NO RANGE )** for row and label for column. But when a range of labels is passed with loc method then, In such case the start and stop labels of the slice are included in the output.

**Eg: >>> df.loc['cobra':'viper', 'max\_speed']**

cobra 1

viper 4

Name: max\_speed, dtype: int64

# BOOLEAN INDEXING

HERE A BOOLEAN ARRAY WILL BE USED FOR SLICING THE DATAFRAME

**>>> df.loc[[False, False, True]]**

max\_speed shield

sidewinder 7 8

#

**>>> df.loc[df['shield'] > 6]**

max\_speed shield

sidewinder 7 8

THIS SHOWS: Conditional that returns a boolean Series

**>>> df.loc[df['shield'] > 6, ['max\_speed']]**

max\_speed

sidewinder 7

THIS SHOWS: Conditional that returns a boolean Series, with column labels specified

**>>> df.loc[df['shield'] > 6, ['max\_speed']]**

max\_speed

sidewinder 7

**SETTING VALUES**

Set value for all items matching the list of labels..

**>>> df.loc[['viper', 'sidewinder'], ['shield']] = 50**

>>> df

max\_speed shield

cobra 1 2

viper 4 50

sidewinder 7 50

Set value for an entire row

**>>> df.loc['cobra'] = 10**

>>> df

max\_speed shield

cobra 10 10

viper 4 50

sidewinder 7 50

Set value for rows matching callable condition

>>> df.loc[df['shield'] > 35] = 0

>>> df

max\_speed shield

cobra 30

**Getting values on a DataFrame with an index that has integer labels**

Another example using integers for the index

>>>

>>> df **=** pd**.**DataFrame**([[1,** **2],** **[4,** **5],** **[7,** **8]],**

... index**=[7,** **8,** **9],** columns**=[**'max\_speed'**,** 'shield'**])**

>>> df

*max\_speed shield*

*7 1 2*

*8 4 5*

*9 7 8*

Slice with integer labels for rows. As mentioned above, note that both the start and stop of the slice are included.

>>>

>>> df**.**loc**[7:9]**

*max\_speed shield*

*7 1 2*

*8 4 5*

*9 7 8*

10

viper 0 0

sidewinder 0 0

***THE ILOC ATTRIBUTE***

Purely integer-location based indexing for selection by position. ….iloc[] is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.

Note: iloc will raise IndexError if a requested indexer is out-of-bounds, except *slice* indexers which allow out-of-bounds indexing (this conforms with python/numpy *slice* semantics).

**Consider a List of dictionaries**

**>>> mydict = [{'a': 1, 'b': 2, 'c': 3, 'd': 4},**

**... {'a': 100, 'b': 200, 'c': 300, 'd': 400},**

**... {'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]**

**>>> df = pd.DataFrame(mydict)**

**>>> df**

a b c d

0 1 2 3 4

1 100 200 300 400

2 1000 2000 3000 4000

# **Indexing just the rows** with a scalar integer will return a Series

**>>> type(df.iloc[0])**

<class 'pandas.core.series.Series'>

**>>> df.iloc[0]**

a 1

b 2

c 3

d 4

Name: 0, dtype: int64

**# Indexing with a list of integers will return a DataFrame**

**>>> df.iloc[[0]]**

**a b c d**

**0 1 2 3 4**

**>>> type(df.iloc[[0]])**

**<class 'pandas.core.frame.DataFrame'>**

**>>> df.iloc[[0, 1]]**

**a b c d**

**0 1 2 3 4**

**1 100 200 300 400**

**# Indexing using iloc With a slice object.**

**>>> df.iloc[:3]**

**a b c d**

**0 1 2 3 4**

**1 100 200 300 400**

**2 1000 2000 3000 4000**

**# Indexing using iloc With a boolean mask the same length as the index.**

**>>> df.iloc[[True, False, True]]**

**a b c d**

**0 1 2 3 4**

**2 1000 2000 3000 4000**

**Indexing both axes**

You can mix the indexer types for the index and columns. The value before comma is for rows and after comma is for columns.

NOTE : USE a colon symbol (:) to select the entire axis.

With scalar integers.

>>>

>>> df**.**iloc**[0,** **1]**

*2*

With lists of integers.

>>>

>>> df**.**iloc**[[0,** **2],** **[1,** **3]]**

*b d*

*0 2 4*

*2 2000 4000*

With *slice* objects.

>>>

>>> df**.**iloc**[1:3,** **0:3]**

*a b c*

*1 100 200 300*

*2 1000 2000 3000*

With a boolean array whose length matches the columns.

>>>

>>> df**.**iloc**[:,** **[True,** **False,** **True,** **False]]**

*a c*

*0 1 3*

*1 100 300*

*2 1000 3000*