Installation and Loading Data

Install pandas globally using pip:

$ sudo -H pip3 install pandas Or set up a virtual environment and do install there:

$python3 -m venv venv$ source venv/bin/activate (venv) $ pip install pandas

In [24]:

```
import pandas as pd
```

Pandas.DataFrame

syntax:-

class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects.

In [4]:

```
df = pd.DataFrame([[0, 2, 3], [0, 4, 1], [10, 20, 30]],
                  index=[4, 5, 6], columns=['A', 'B', 'C'])
df
```

Out[4]:

	A	B	C
4	0	2	3
5	0	4	1
6	10	20	30

Attributes:-

ATTRIBUTE	DESCRIPTION
at	Access a single value for a row/column label pair.
iat	Access a single value for a row/column pair by integer position.
loc	Access a group of rows and columns by label(s) or a boolean array.
iloc	Purely integer-location based indexing for selection by position.
axes	Return a list representing the axes of the DataFrame.
columns	The column labels of the DataFrame.
dtypes	Return the dtypes in the DataFrame.

ATTRIBUTE	DESCRIPTION
empty	Indicator whether Series/DataFrame is empty.
values	Return a Numpy representation of the DataFrame.
size	Return an int representing the number of elements in this object.
shape	Return a tuple representing the dimensionality of the DataFrame.
ndim	Return an int representing the number of axes / array dimensions.
index	The index (row labels) of the DataFrame

1. pandas.DataFrame.at

Access a single value for a row/column label pair. Similar to loc, in that both provide label-based lookups. Use at if you only need to get or set a single value in a DataFrame or Series.

- **Raises:**
  - KeyError (If 'label' does not exist in DataFrame.)

In [5]:

```
df.at[4, 'B']
```

Out[5]:

```
2
```

2. pandas.DataFrame.iat

Access a single value for a row/column pair by integer position. Similar to iloc, in that both provide integer-based lookups. Use iat if you only need to get or set a single value in a DataFrame or Series.

- **Raises:**
  - IndexError(When integer position is out of bounds.)

In [6]:

```
df.iat[1, 2]
```

Out[6]:

```
1
```

3. pandas.DataFrame.loc

Access a group of rows and columns by label(s) or a boolean array. .loc[] is primarily label based, but may also be used with a boolean array. Allowed inputs are:

1. A single label, e.g. 5 or 'a', (note that 5 is interpreted as a label of the index, and never as an integer position along the index).
2. A list or array of labels, e.g. ['a', 'b', 'c'].
3. A slice object with labels, e.g. 'a':'f'.
4. A boolean array of the same length as the axis being sliced, e.g. [True, False, True].
5. An alignable boolean Series. The index of the key will be aligned before masking.
6. An alignable Index. The Index of the returned selection will be the input.
7. A callable function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above)

- **Raises:**
  - KeyError - (If any items are not found.)
  - IndexingError - (If an indexed key is passed and its index is unalignable to the frame index.)

In [8]:

```
df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
     index=['cobra', 'viper', 'sidewinder'],
     columns=['max_speed', 'shield'])
df.loc['viper']
```

Out[8]:

```
max_speed    4
shield       5
Name: viper, dtype: int64
```

4. pandas.DataFrame.iloc

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. Allowed inputs are:

1. An integer, e.g. 5.
2. A list or array of integers, e.g. [4, 3, 0].
3. A slice object with ints, e.g. 1:7.
4. A boolean array.

A callable function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above). This is useful in method chains, when you don’t have a reference to the calling object, but would like to base your selection on some value.

- **Raises:**
  - IndexError (if a requested indexer is out-of-bounds, except slice indexers which allow out-of-bounds indexing)

In [12]:

```
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)
print(type(df.iloc[0]))
```

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

In [13]:

```
df.iloc[0]
```

Out[13]:

```
a    1
b    2
c    3
d    4
Name: 0, dtype: int64
```

5. pandas.DataFrame.axes

Syntax:- property DataFrame.axes

Return a list representing the axes of the DataFrame. It has the row axis labels and column axis labels as the only members. They are returned in that order.

In [14]:

```
df.axes
```

Out[14]:

```
[RangeIndex(start=0, stop=3, step=1),
 Index(['a', 'b', 'c', 'd'], dtype='object')]
```

6. pandas.DataFrame.columns

Syntax:- DataFrame.columns

```
The column labels of the DataFrame.
It has the row axis labels and column axis labels as the only members. They are re
turned in that order.
```

In [15]:

```
df.columns
```

Out[15]:

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

7. pandas.DataFrame.dtypes

Syntax:- property DataFrame.dtypes

Return the dtypes in the DataFrame.This returns a Series with the data type of each column. The result's index is the original DataFrame's columns. Columns with mixed types are stored with the object dtype. See the User Guide for more.

In [17]:

```
df.dtypes
```

Out[17]:

```
a    int64
b    int64
c    int64
d    int64
dtype: object
```

8. pandas.DataFrame.empty

Syntax:- property DataFrame.empty

Indicator whether Series/DataFrame is empty. True if Series/DataFrame is entirely empty (no items), meaning any of the axes are of length 0.

In [18]:

```
df.empty
```

Out[18]:

```
False
```

9. pandas.DataFrame.values

Syntax:- property DataFrame.values

Return a Numpy representation of the DataFrame.

In [19]:

```
df.values
```

Out[19]:

```
array([[   1,    2,    3,    4],
       [ 100,  200,  300,  400],
       [1000, 2000, 3000, 4000]], dtype=int64)
```

10. pandas.DataFrame.size

Syntax:- property DataFrame.size

Return an int representing the number of elements in this object. Return the number of rows if Series. Otherwise return the number of rows times number of columns if DataFrame.

In [20]:

```
df.size
```

Out[20]:

```
12
```

11. pandas.DataFrame.shape

Syntax:- property DataFrame.shape

Return a tuple representing the dimensionality of the DataFrame.

In [21]:

```
df.shape
```

Out[21]:

```
(3, 4)
```

12. pandas.DataFrame.ndim

Syntax:- property DataFrame.ndim

Return an int representing the number of axes / array dimensions. Return 1 if Series. Otherwise return 2 if DataFrame.

In [22]:

```
df.ndim
```

Out[22]:

```
2
```

13. pandas.DataFrame.index

Syntax:- DataFrame.index

The index (row labels) of the DataFrame.

In [23]:

```
df.index
```

Out[23]:

```
RangeIndex(start=0, stop=3, step=1)
```