Pandas Data Structures – Overview

Let us understand the details with respect to Pandas.

- Pandas is not a core Python module and hence we need to install using pip - `pip install pandas`.
- It has 2 types of data structures - `Series` and `DataFrame`.
- `Series` is a one dimension array while `DataFrame` is a two dimension array.
- `Series` only contains index for each row and one attribute or column.
- `DataFrame` contains index for each row and multiple columns.
- Each attribute in the DataFrame is nothing but a Series.
- We can perform all standard transformations using Pandas APIs
- We also have SQL based wrappers on top of Pandas where we can write queries.

Here are the steps to get started with Pandas Data Structures:

- Make sure Pandas library is installed using `pip`.
- Import Pandas library - `import pandas as pd`
- We need to have a collection or data in a file to create Pandas Data Structures.
- Use appropriate APIs on the data to create Pandas Data Structures.
    - `Series` for single dimension array.
    - `DataFrame` for two dimension array.

Note

Typically we use `Series` for list of regular objects or dict and `DataFrame` for list of tuples or list of dicts. Let us use list for `Series` and list of dicts for `DataFrame`.

```
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /opt/anaconda3/envs/beakerx/lib/python3.6/site-
packages (1.1.4)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/anaconda3/envs/beakerx/lib/python3.6/site-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in
/opt/anaconda3/envs/beakerx/lib/python3.6/site-packages (from pandas) (2020.4)
Requirement already satisfied: numpy>=1.15.4 in
/opt/anaconda3/envs/beakerx/lib/python3.6/site-packages (from pandas) (1.19.4)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/envs/beakerx/lib/python3.6/site-
packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

```
import pandas as pd
```

```
sals_l = [1500.0, 2000.0, 2200.00]
```

```
pd.Series?
```

```
Init signature:
pd.Series(
    data=None,
    index=None,
    dtype=None,
    name=None,
    copy=False,
    fastpath=False,
)
Docstring:
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
methods for performing operations involving the index. Statistical
methods from ndarray have been overridden to automatically exclude
missing data (currently represented as NaN).

Operations between Series (+, -, /, *, **) align values based on their
associated index values-- they need not be the same length. The result
index will be the sorted union of the two indexes.

Parameters
----------
data : array-like, Iterable, dict, or scalar value
    Contains data stored in Series.

    .. versionchanged:: 0.23.0
       If data is a dict, argument order is maintained for Python 3.6
       and later.

index : array-like or Index (1d)
    Values must be hashable and have the same length as `data`.
    Non-unique index values are allowed. Will default to
    RangeIndex (0, 1, 2, ..., n) if not provided. If both a dict and index
    sequence are used, the index will override the keys found in the
    dict.
dtype : str, numpy.dtype, or ExtensionDtype, optional
    Data type for the output Series. If not specified, this will be
    inferred from `data`.
    See the :ref:`user guide <basics.dtypes>` for more usages.
name : str, optional
    The name to give to the Series.
copy : bool, default False
    Copy input data.
File:           /opt/anaconda3/envs/beakerx/lib/python3.6/site-packages/pandas/core/series.py
Type:           type
Subclasses:     SubclassedSeries
```

```
sals_s = pd.Series(sals_l, name='sal')
```

```
sals_s
```

```
0    1500.0
1    2000.0
2    2200.0
Name: sal, dtype: float64
```

```
sals_s[:2]
```

```
0    1500.0
1    2000.0
Name: sal, dtype: float64
```

```
sals_ld = [(1, 1500.0), (2, 2000.0), (3, 2200.00)]
```

```
pd.DataFrame?
```

```
Init signature:
pd.DataFrame(
    data=None,
    index:Union[Collection, NoneType]=None,
    columns:Union[Collection, NoneType]=None,
    dtype:Union[_ForwardRef('ExtensionDtype'), str, numpy.dtype, Type[Union[str, float, int,
complex]], NoneType]=None,
    copy:bool=False,
)
Docstring:
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

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. The primary
pandas data structure.

Parameters
----------
data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
    Dict can contain Series, arrays, constants, or list-like objects.

    .. versionchanged:: 0.23.0
       If data is a dict, column order follows insertion-order for
       Python 3.6 and later.

    .. versionchanged:: 0.25.0
       If data is a list of dicts, column order follows insertion-order
       for Python 3.6 and later.

index : Index or array-like
    Index to use for resulting frame. Will default to RangeIndex if
    no indexing information part of input data and no index provided.
columns : Index or array-like
    Column labels to use for resulting frame. Will default to
    RangeIndex (0, 1, 2, ..., n) if no column labels are provided.
dtype : dtype, default None
    Data type to force. Only a single dtype is allowed. If None, infer.
copy : bool, default False
    Copy data from inputs. Only affects DataFrame / 2d ndarray input.

See Also
--------
DataFrame.from_records : Constructor from tuples, also record arrays.
DataFrame.from_dict : From dicts of Series, arrays, or dicts.
read_csv : Read a comma-separated values (csv) file into DataFrame.
read_table : Read general delimited file into DataFrame.
read_clipboard : Read text from clipboard into DataFrame.

Examples
--------
Constructing DataFrame from a dictionary.

>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4

Notice that the inferred dtype is int64.

>>> df.dtypes
col1    int64
col2    int64
dtype: object

To enforce a single dtype:

>>> df = pd.DataFrame(data=d, dtype=np.int8)
>>> df.dtypes
col1    int8
col2    int8
dtype: object

Constructing DataFrame from numpy ndarray:

>>> df2 = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),
...                    columns=['a', 'b', 'c'])
>>> df2
   a  b  c
0  1  2  3
1  4  5  6
2  7  8  9
File:           /opt/anaconda3/envs/beakerx/lib/python3.6/site-packages/pandas/core/frame.py
Type:           type
Subclasses:     SubclassedDataFrame
```

```
sals_df = pd.DataFrame(sals_ld, columns=['id', 'sal'])
```

```
sals_df
```

	id	sal
0	1	1500.0
1	2	2000.0
2	3	2200.0

```
sals_df['id']
```

```
0    1
1    2
2    3
Name: id, dtype: int64
```