Pandas

- **Pandas is a built in library using for data analysis. You'll be using Pandas heavily for data manipulation, visualisation, building machine learning models, etc.**
- **Pandas implements a number of powerful data operations familiar to users of both database frameworks and spreadsheet programs.**
- **There are two main data structures in Pandas - Series and Dataframes. The default way to store data is dataframes, and thus manipulating dataframes quickly is probably the most important skill set for data analysis.**

  Source: https://pandas.pydata.org/pandas-docs/stable/overview.html

In [ ]:

```
pip install pandas
```

Pandas Series

- **A series is similar to a 1-D numpy array, and contains values of the same type (numeric, character, datetime etc.). A dataframe is simply a table where each column is a pandas series.**
- **creating series**
  - **List**
  - **Tuple**
  - **Dictionary**
  - **Numpy**
  - **Date_Range**
- **Series Indexing**

In [1]:

```
# importing library
import pandas as pd
```

In [3]:

```
len(dir(pd))   # 142 methods inside pandas library
```

Out[3]:

```
142
```

In [5]:

```
# version check
pd.__version__
```

Out[5]:

```
'1.0.5'
```

1. Creating Pandas Series

In [6]:

```
# using list
li = [12,34,56,87,67,78,89]
s1 = pd.Series(li)
s1 # s1 series object
# Series having index values
```

Out[6]:

```
0    12
1    34
2    56
3    87
4    67
5    78
6    89
dtype: int64
```

In [51]:

```
# using tuple
t = (12,34,5665,34,23,67,78.789)
s2 = pd.Series(t)
s2
# default panas Series index starts from 0
```

Out[51]:

```
0      12.000
1      34.000
2    5665.000
3      34.000
4      23.000
5      67.000
6      78.789
dtype: float64
```

In [9]:

```
# using dict
di = {"a":234,"b":"Lavanya",123:3224.56}
s3 = pd.Series(di)
s3
# keys are index values
```

Out[9]:

```
a          234
b      Lavanya
123    3224.56
dtype: object
```

In [52]:

```
s2.index = ["a","f","t","y",23,"7",456]   # explict indexing
s2
```

Out[52]:

```
a        12.000
f        34.000
t      5665.000
y        34.000
23       23.000
7        67.000
456      78.789
dtype: float64
```

In [14]:

```
s2.dtype
```

Out[14]:

```
dtype('float64')
```

In [15]:

```
s3.dtype  # object data
```

Out[15]:

```
dtype('O')
```

In [16]:

```
s1.dtype
```

Out[16]:

```
dtype('int64')
```

2. Series Indexing

In [53]:

```
s2
```

Out[53]:

```
a        12.000
f        34.000
t      5665.000
y        34.000
23       23.000
7        67.000
456      78.789
dtype: float64
```

In [18]:

```
s2["t"]
```

Out[18]:

```
5665.0
```

In [20]:

```
# access from a to y
s2["a":23]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-0eb9bfb613b8> in <module>
      1 # access from a to y
----> 2 s2["a":23]

~\anaconda3\lib\site-packages\pandas\core\series.py in __getitem__(self, key)
    908             key = check_bool_indexer(self.index, key)
    909
--> 910         return self._get_with(key)
    911
    912     def _get_with(self, key):

~\anaconda3\lib\site-packages\pandas\core\series.py in _get_with(self, key)
    913         # other: fancy integer or otherwise
    914         if isinstance(key, slice):
--> 915             return self._slice(key)
    916         elif isinstance(key, ABCDataFrame):
    917             raise TypeError(

~\anaconda3\lib\site-packages\pandas\core\series.py in _slice(self, slobj, axis, kind)
    863
    864     def _slice(self, slobj: slice, axis: int = 0, kind=None):
--> 865         slobj = self.index._convert_slice_indexer(slobj, kind=kind or "getitem")
    866         return self._get_values(slobj)
    867

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in _convert_slice_indexer(self,
key, kind)
   2961             indexer = key
```

```
   2962         else:
-> 2963             indexer = self.slice_indexer(start, stop, step, kind=kind)
   2964
   2965         return indexer

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in slice_indexer(self, start, e
nd, step, kind)
   4711         slice(1, 3)
   4712         """
-> 4713         start_slice, end_slice = self.slice_locs(start, end, step=step, kind=kind
)
   4714
   4715         # return a slice

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in slice_locs(self, start, end,
step, kind)
   4930         end_slice = None
   4931         if end is not None:
-> 4932             end_slice = self.get_slice_bound(end, "right", kind)
   4933         if end_slice is None:
   4934             end_slice = len(self)

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_slice_bound(self, label,
side, kind)
   4836         # For datetime indices label may be a string that has to be converted
   4837         # to datetime boundary according to its resolution.
-> 4838         label = self._maybe_cast_slice_bound(label, side, kind)
   4839
   4840         # we need to look up the label

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in _maybe_cast_slice_bound(self
, label, side, kind)
   4788         # this is rejected (generally .loc gets you here)
   4789         elif is_integer(label):
-> 4790             self._invalid_indexer("slice", label)
   4791
   4792         return label

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in _invalid_indexer(self, form,
key)
   3074         Consistent invalid indexer message.
   3075         """
-> 3076         raise TypeError(
   3077             f"cannot do {form} indexing on {type(self)} with these "
   3078             f"indexers [{key}] of {type(key)}"

TypeError: cannot do slice indexing on <class 'pandas.core.indexes.base.Index'> with thes
e indexers [23] of <class 'int'>
```

In [21]:

```
# explict slicing
s2["a":"y"] # start, end
```

Out[21]:

```
a      12.0
f      34.0
t    5665.0
y      34.0
dtype: float64
```

In [23]:

```
# implict slicing/ indexing
s2[0:4]
```

Out[23]:

```
a      12.0
f      34.0
t    5665.0
y      34.0
```

```
y      34.0
dtype: float64
```

In [24]:

```
s2[:4]
```

Out[24]:

```
a      12.0
f      34.0
t    5665.0
y      34.0
dtype: float64
```

In [54]:

```
# fancy slicing
s2
```

Out[54]:

```
a        12.000
f        34.000
t      5665.000
y        34.000
23       23.000
7        67.000
456      78.789
dtype: float64
```

In [57]:

```
s2[["a","7",456]]
```

Out[57]:

```
a      12.000
7      67.000
456    78.789
dtype: float64
```

In [28]:

```
# create Series object by using dict with nan value
import numpy as np
s4 = pd.Series({1:np.nan, 2:np.nan,3:245,4:356})
s4
```

Out[28]:

```
1      NaN
2      NaN
3    245.0
4    356.0
dtype: float64
```

In [33]:

```
di2 =  {1:np.nan, 2:np.nan,3:245,4:356}
s5  =  pd.Series(di2,index = [3,77])
s5
```

Out[33]:

```
3     245.0
77      NaN
dtype: float64
```

In [34]:

```
s1={'a':35,'b':67,'c':"nan",'d':"nan"}
s2=pd.Series(s1)
```

```
s2
```

Out[34]:

```
a     35
b     67
c    nan
d    nan
dtype: object
```

In [35]:

```
s5 = pd.Series("APSSDC",index = ["ap","Te","Mi"])
s5
```

Out[35]:

```
ap    APSSDC
Te    APSSDC
Mi    APSSDC
dtype: object
```

In [36]:

```
s5.index = np.arange(10,13)
s5
```

Out[36]:

```
10    APSSDC
11    APSSDC
12    APSSDC
dtype: object
```

In [38]:

```
# create series object power of index  having index values starts from 10 - 25
s2 = pd.Series(range(1,16))

s2.index = np.arange(10,25)
s2
```

Out[38]:

```
10     1
11     2
12     3
13     4
14     5
15     6
16     7
17     8
18     9
19    10
20    11
21    12
22    13
23    14
24    15
dtype: int64
```

1 1 2 4 3 9 4 16 5 25

In [42]:

```
s6 = pd.Series(np.arange(10,26)**2, index = np.arange(10,26))
s6
```

Out[42]:

```
10    100
11    121
12    144
13    169
14    196
```

```
15    225
16    256
17    289
18    324
19    361
20    400
21    441
22    484
23    529
24    576
25    625
dtype: int32
```

In [ ]:

```
```

NOTE: always data values is equal to the index values in series

3. Date range/ series

In [43]:

```
s7 = pd.date_range(start = "2020-10-26", end = "2020-10-31" )
s7
```

Out[43]:

```
DatetimeIndex(['2020-10-26', '2020-10-27', '2020-10-28', '2020-10-29',
               '2020-10-30', '2020-10-31'],
              dtype='datetime64[ns]', freq='D')
```

In [44]:

```
help(pd.date_range)
```

```
Help on function date_range in module pandas.core.indexes.datetimes:

date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=
None, closed=None, **kwargs) -> pandas.core.indexes.datetimes.DatetimeIndex
    Return a fixed frequency DatetimeIndex.

    Parameters
    ----------
    start : str or datetime-like, optional
        Left bound for generating dates.
    end : str or datetime-like, optional
        Right bound for generating dates.
    periods : int, optional
        Number of periods to generate.
    freq : str or DateOffset, default 'D'
        Frequency strings can have multiples, e.g. '5H'. See
        :ref:`here <timeseries.offset_aliases>` for a list of
        frequency aliases.
    tz : str or tzinfo, optional
        Time zone name for returning localized DatetimeIndex, for example
        'Asia/Hong_Kong'. By default, the resulting DatetimeIndex is
        timezone-naive.
    normalize : bool, default False
        Normalize start/end dates to midnight before generating date range.
    name : str, default None
        Name of the resulting DatetimeIndex.
    closed : {None, 'left', 'right'}, optional
        Make the interval closed with respect to the given frequency to
        the 'left', 'right', or both sides (None, the default).
    **kwargs
        For compatibility. Has no effect on the result.

    Returns
```

```
    -------
    rng : DatetimeIndex

    See Also
    --------
    DatetimeIndex : An immutable container for datetimes.
    timedelta_range : Return a fixed frequency TimedeltaIndex.
    period_range : Return a fixed frequency PeriodIndex.
    interval_range : Return a fixed frequency IntervalIndex.

    Notes
    -----
    Of the four parameters ``start``, ``end``, ``periods``, and ``freq``,
    exactly three must be specified. If ``freq`` is omitted, the resulting
    ``DatetimeIndex`` will have ``periods`` linearly spaced elements between
    ``start`` and ``end`` (closed on both sides).

    To learn more about the frequency strings, please see `this link
    <https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-alias
es>`__.

    Examples
    --------
    **Specifying the values**

    The next four examples generate the same `DatetimeIndex`, but vary
    the combination of `start`, `end` and `periods`.

    Specify `start` and `end`, with the default daily frequency.

    >>> pd.date_range(start='1/1/2018', end='1/08/2018')
    DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
                   '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08'],
                  dtype='datetime64[ns]', freq='D')

    Specify `start` and `periods`, the number of periods (days).

    >>> pd.date_range(start='1/1/2018', periods=8)
    DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
                   '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08'],
                  dtype='datetime64[ns]', freq='D')

    Specify `end` and `periods`, the number of periods (days).

    >>> pd.date_range(end='1/1/2018', periods=8)
    DatetimeIndex(['2017-12-25', '2017-12-26', '2017-12-27', '2017-12-28',
                   '2017-12-29', '2017-12-30', '2017-12-31', '2018-01-01'],
                  dtype='datetime64[ns]', freq='D')

    Specify `start`, `end`, and `periods`; the frequency is generated
    automatically (linearly spaced).

    >>> pd.date_range(start='2018-04-24', end='2018-04-27', periods=3)
    DatetimeIndex(['2018-04-24 00:00:00', '2018-04-25 12:00:00',
                   '2018-04-27 00:00:00'],
                  dtype='datetime64[ns]', freq=None)

    **Other Parameters**

    Changed the `freq` (frequency) to ``'M'`` (month end frequency).

    >>> pd.date_range(start='1/1/2018', periods=5, freq='M')
    DatetimeIndex(['2018-01-31', '2018-02-28', '2018-03-31', '2018-04-30',
                   '2018-05-31'],
                  dtype='datetime64[ns]', freq='M')

    Multiples are allowed

    >>> pd.date_range(start='1/1/2018', periods=5, freq='3M')
    DatetimeIndex(['2018-01-31', '2018-04-30', '2018-07-31', '2018-10-31',
                   '2019-01-31'],
                  dtype='datetime64[ns]', freq='3M')
```

```
    `freq` can also be specified as an Offset object.

    >>> pd.date_range(start='1/1/2018', periods=5, freq=pd.offsets.MonthEnd(3))
    DatetimeIndex(['2018-01-31', '2018-04-30', '2018-07-31', '2018-10-31',
                   '2019-01-31'],
                  dtype='datetime64[ns]', freq='3M')

    Specify `tz` to set the timezone.

    >>> pd.date_range(start='1/1/2018', periods=5, tz='Asia/Tokyo')
    DatetimeIndex(['2018-01-01 00:00:00+09:00', '2018-01-02 00:00:00+09:00',
                   '2018-01-03 00:00:00+09:00', '2018-01-04 00:00:00+09:00',
                   '2018-01-05 00:00:00+09:00'],
                  dtype='datetime64[ns, Asia/Tokyo]', freq='D')

    `closed` controls whether to include `start` and `end` that are on the
    boundary. The default includes boundary points on either end.

    >>> pd.date_range(start='2017-01-01', end='2017-01-04', closed=None)
    DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03', '2017-01-04'],
                  dtype='datetime64[ns]', freq='D')

    Use ``closed='left'`` to exclude `end` if it falls on the boundary.

    >>> pd.date_range(start='2017-01-01', end='2017-01-04', closed='left')
    DatetimeIndex(['2017-01-01', '2017-01-02', '2017-01-03'],
                  dtype='datetime64[ns]', freq='D')

    Use ``closed='right'`` to exclude `start` if it falls on the boundary.

    >>> pd.date_range(start='2017-01-01', end='2017-01-04', closed='right')
    DatetimeIndex(['2017-01-02', '2017-01-03', '2017-01-04'],
                  dtype='datetime64[ns]', freq='D')
```

In [48]:

```
import calendar
import datetime
import time
# modules
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Data Analysis with Pandas

* Pandas DataFrame
* Combining & Merging
* File I/O
* Indexing
* Grouping
* Features
* Filtering

* Sorting
* Statistical
* Plotting
* Saving

id	col1	col2
1	678	xyz
2	123	sdf
3	454	jhg

1. Pandas DataFrame

In [58]:

```
# using List
li  = [[12,34],[45,67],[67,89]]
df1 = pd.DataFrame(li)
df1
# defualt index and columns are starts from 0
```

Out[58]:

	0	1
0	12	34
1	45	67
2	67	89

In [60]:

```
df1.index = ["a","f","y"] # indexing
df1
```

Out[60]:

	0	1
a	12	34
f	45	67
y	67	89

In [61]:

```
# columns/ labels/ features
df1.columns = ["One","Two"]
df1
```

Out[61]:

	One	Two
a	12	34
f	45	67
y	67	89

In [62]:

```
df1.columns = list("AB")
df1
```

Out[62]:

	A	B
a	12	34
f	45	67
y	67	89

In [63]:

```
df2 =  pd.DataFrame(li,dtype = "float64")
df2
 # int to float values
```

Out[63]:

	0	1
0	12.0	34.0
1	45.0	67.0
2	67.0	89.0

In [66]:

```
# using Dict
di3  = {
    "Name" : ["Chaitanya","Rajyalakshmi","Harsha"],
    "Branch" : ["ec","ee","cse"],
    "Gender" : ["M","M","F"]

}
df3 = pd.DataFrame(di3)
df3.index = [195,408,123]
df3
```

Out[66]:

	Name	Branch	Gender
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F

In [67]:

```
# using Pandas series
di4  = {
    "Name" : pd.Series(["Chaitanya","Rajyalakshmi","Harsha"]),
    "Branch" : pd.Series(["ec","ee","cse"]),
    "Gender" : pd.Series(["M","M","F"])

}
df4 = pd.DataFrame(di3)
df4.index = [195,408,123]
df4
```

Out[67]:

	Name	Branch	Gender
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F

In [70]:

```
di5 = [{"a":234,"b":657,"c":870},{"a":567,"c":70}]
```

```
di5 = [{"a":234,"b":657,"c":879},{"a":567,"c":79}]
df5 = pd.DataFrame(di5)
df5
# NaN -- Not  a number -- special type float value
```

Out[70]:

	a	b	c
0	234	657.0	879
1	567	NaN	79

In [71]:

```
del df5["b"]  # delete a particular column
```

In [72]:

```
df5
```

Out[72]:

	a	c
0	234	879
1	567	79

2. Combining & Merging

In [73]:

```
df4
```

Out[73]:

	Name	Branch	Gender
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F

In [74]:

```
pd.concat([df4,df4])  # concat at rows
```

Out[74]:

	Name	Branch	Gender
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F

In [76]:

```
pd.concat([df4,df4], axis = 1)
# axis 1 represents columns
# axis 0 rows
# default row concat
```

Out[76]:

Out[76]:

	Name	Branch	Gender	Name	Branch	Gender
195	Chaitanya	ec	M	Chaitanya	ec	M
408	Rajyalakshmi	ee	M	Rajyalakshmi	ee	M
123	Harsha	cse	F	Harsha	cse	F

In [77]:

```
df4.append(df4) # only rows
```

Out[77]:

	Name	Branch	Gender
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F
195	Chaitanya	ec	M
408	Rajyalakshmi	ee	M
123	Harsha	cse	F

In [78]:

```
pd.merge(df4,df4)
```

Out[78]:

	Name	Branch	Gender
0	Chaitanya	ec	M
1	Rajyalakshmi	ee	M
2	Harsha	cse	F

In [130]:

```
d1 = pd.DataFrame({
    "std_Name" : ["a","b","v","t"],
    "fvt_fruit" : ["mango","apple","orange","lemon"]

})
d1
```

Out[130]:

	std_Name	fvt_fruit
0	a	mango
1	b	apple
2	v	orange
3	t	lemon

In [133]:

```
d2 = pd.DataFrame({
    "std_Name" : ["r","b","v","t"],
    "age" : [23,45,34,55]

})
d2
```

Out[133]:

	std_Name	age
0	r	23
1	b	45
2	v	34
3	t	55

In [81]:

```
pd.concat([d1,d2])
```

Out[81]:

	std_Name	fvt_fruit	age
0	a	mango	NaN
1	b	apple	NaN
2	v	orange	NaN
3	t	lemon	NaN
0	r	NaN	23.0
1	b	NaN	45.0
2	v	NaN	34.0
3	t	NaN	55.0

In [134]:

```
pd.merge(d1,d2)   # common data in both df's
```

Out[134]:

	std_Name	fvt_fruit	age
0	b	apple	45
1	v	orange	34
2	t	lemon	55

In [85]:

```
pd.merge(d1,d2, how = "left")
```

Out[85]:

	std_Name	fvt_fruit	age
0	a	mango	NaN
1	b	apple	45.0
2	v	orange	34.0
3	t	lemon	55.0

In [86]:

```
pd.merge(d1,d2, how = "right")
```

Out[86]:

	std_Name	fvt_fruit	age
0	b	apple	45
1	v	orange	34

	std_Name	fvt_fruit	age
2	t	lemon	55
3	r	NaN	23

In [88]:

```
pd.merge(d1,d2, how = "outer")  # it returns all elements in both df -- union
```

Out[88]:

	std_Name	fvt_fruit	age
0	a	mango	NaN
1	b	apple	45.0
2	v	orange	34.0
3	t	lemon	55.0
4	r	NaN	23.0

In [89]:

```
pd.merge(d1,d2, how = "inner")  # common data - intersection of keys from both frames
```

Out[89]:

	std_Name	fvt_fruit	age
0	b	apple	45
1	v	orange	34
2	t	lemon	55

In [90]:

```
help(pd.merge)
```

```
Help on function merge in module pandas.core.reshape.merge:

merge(left, right, how: str = 'inner', on=None, left_on=None, right_on=None, left_index:
bool = False, right_index: bool = False, sort: bool = False, suffixes=('_x', '_y'), copy:
bool = True, indicator: bool = False, validate=None) -> 'DataFrame'
    Merge DataFrame or named Series objects with a database-style join.

    The join is done on columns or indexes. If joining columns on
    columns, the DataFrame indexes *will be ignored*. Otherwise if joining indexes
    on indexes or indexes on a column or columns, the index will be passed on.

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

        * left: use only keys from left frame, similar to a SQL left outer join;
          preserve key order.
        * right: use only keys from right frame, similar to a SQL right outer join;
          preserve key order.
        * outer: use union of keys from both frames, similar to a SQL full outer
          join; sort keys lexicographically.
        * inner: use intersection of keys from both frames, similar to a SQL inner
          join; preserve the order of the left keys.
    on : label or list
        Column or index level names to join on. These must be found in both
        DataFrames. If `on` is None and not merging on indexes then this defaults
        to the intersection of the columns in both DataFrames.
    left_on : label or list, or array-like
        Column or index level names to join on in the left DataFrame. Can also
        be an array or list of arrays of the length of the left DataFrame
```

```
    be an array or list of arrays of the length of the left DataFrame.
    These arrays are treated as if they are columns.
right_on : label or list, or array-like
    Column or index level names to join on in the right DataFrame. Can also
    be an array or list of arrays of the length of the right DataFrame.
    These arrays are treated as if they are columns.
left_index : bool, default False
    Use the index from the left DataFrame as the join key(s). If it is a
    MultiIndex, the number of keys in the other DataFrame (either the index
    or a number of columns) must match the number of levels.
right_index : bool, default False
    Use the index from the right DataFrame as the join key. Same caveats as
    left_index.
sort : bool, default False
    Sort the join keys lexicographically in the result DataFrame. If False,
    the order of the join keys depends on the join type (how keyword).
suffixes : tuple of (str, str), default ('_x', '_y')
    Suffix to apply to overlapping column names in the left and right
    side, respectively. To raise an exception on overlapping columns use
    (False, False).
copy : bool, default True
    If False, avoid copy if possible.
indicator : bool or str, default False
    If True, adds a column to output DataFrame called "_merge" with
    information on the source of each row.
    If string, column with information on source of each row will be added to
    output DataFrame, and column will be named value of string.
    Information column is Categorical-type and takes on a value of "left_only"
    for observations whose merge key only appears in 'left' DataFrame,
    "right_only" for observations whose merge key only appears in 'right'
    DataFrame, and "both" if the observation's merge key is found in both.

validate : str, optional
    If specified, checks if merge is of specified type.

    * "one_to_one" or "1:1": check if merge keys are unique in both
      left and right datasets.
    * "one_to_many" or "1:m": check if merge keys are unique in left
      dataset.
    * "many_to_one" or "m:1": check if merge keys are unique in right
      dataset.
    * "many_to_many" or "m:m": allowed, but does not result in checks.

    .. versionadded:: 0.21.0

Returns
-------
DataFrame
    A DataFrame of the two merged objects.

See Also
--------
merge_ordered : Merge with optional filling/interpolation.
merge_asof : Merge on nearest keys.
DataFrame.join : Similar method using indices.

Notes
-----
Support for specifying index levels as the `on`, `left_on`, and
`right_on` parameters was added in version 0.23.0
Support for merging named Series objects was added in version 0.24.0

Examples
--------

>>> df1 = pd.DataFrame({'lkey': ['foo', 'bar', 'baz', 'foo'],
...                     'value': [1, 2, 3, 5]})
>>> df2 = pd.DataFrame({'rkey': ['foo', 'bar', 'baz', 'foo'],
...                     'value': [5, 6, 7, 8]})
>>> df1
    lkey value
0   foo      1
1   bar      2
```

```
1   bar       2
2   baz       3
3   foo       5
>>> df2
    rkey value
0   foo      5
1   bar      6
2   baz      7
3   foo      8

Merge df1 and df2 on the lkey and rkey columns. The value columns have
the default suffixes, _x and _y, appended.

>>> df1.merge(df2, left_on='lkey', right_on='rkey')
  lkey  value_x rkey  value_y
0  foo        1  foo        5
1  foo        1  foo        8
2  foo        5  foo        5
3  foo        5  foo        8
4  bar        2  bar        6
5  baz        3  baz        7

Merge DataFrames df1 and df2 with specified left and right suffixes
appended to any overlapping columns.

>>> df1.merge(df2, left_on='lkey', right_on='rkey',
...           suffixes=('_left', '_right'))
  lkey  value_left rkey  value_right
0  foo           1  foo            5
1  foo           1  foo            8
2  foo           5  foo            5
3  foo           5  foo            8
4  bar           2  bar            6
5  baz           3  baz            7

Merge DataFrames df1 and df2, but raise an exception if the DataFrames have
any overlapping columns.

>>> df1.merge(df2, left_on='lkey', right_on='rkey', suffixes=(False, False))
Traceback (most recent call last):
...
ValueError: columns overlap but no suffix specified:
    Index(['value'], dtype='object')
```

3. File I/O

In [91]:

```
data = pd.read_csv("https://raw.githubusercontent.com/APSSDC-Data-Analysis/DataAnalysis-7/main/Datasets/birds.csv")
data
```

Out[91]:

	id	huml	humw	ulnal	ulnaw	feml	femw	tibl	tibw	tarl	tarw	type
0	0	80.78	6.68	72.01	4.88	41.81	3.70	5.50	4.03	38.70	3.84	SW
1	1	88.91	6.63	80.53	5.59	47.04	4.30	80.22	4.51	41.50	4.01	SW
2	2	79.97	6.37	69.26	5.28	43.07	3.90	75.35	4.04	38.31	3.34	SW
3	3	77.65	5.70	65.76	4.77	40.04	3.52	69.17	3.40	35.78	3.41	SW
4	4	62.80	4.84	52.09	3.73	33.95	2.72	56.27	2.96	31.88	3.13	SW
...	...	...	...	...	...	...	...	...	...	...	...	...
415	415	17.96	1.63	19.25	1.33	18.36	1.54	31.25	1.33	21.99	1.15	SO
416	416	19.21	1.64	20.76	1.49	19.24	1.45	33.21	1.28	23.60	1.15	SO
417	417	18.79	1.63	19.83	1.53	20.96	1.43	34.45	1.41	22.86	1.21	SO

	id	huml	humw	ulnal	ulnaw	feml	femw	tibl	tibw	tarl	tarw	type
418	418	20.38	1.78	22.53	1.50	21.35	1.48	36.09	1.53	25.98	1.24	SO
419	419	17.89	1.44	19.26	1.10	17.62	1.34	29.81	1.24	21.69	1.05	SO

420 rows × 12 columns

In [94]:

```
data = pd.read_csv("employe.csv")
data
```

Out[94]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance
3	Jerry	Male	3/4/2005	1:00 PM	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 PM	101004	1.389	True	Client Services
...	...	...	...	...	...	...	...	...
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

1000 rows × 8 columns

In [95]:

```
data_ex = pd.read_excel("B4.xlsx")
data_ex
```

Out[95]:

	Roll number	Name	Email
0	1210316262	Ravuri Sai Ram Nikhil	nikhilravuri13@gmail.com
1	14KT5A0429	srinivasa rao	sspalle07@gmail.com
2	178A1A0204	Battula. Ramya	bathularamya26@gmail.com
3	17f21a0348	KADAPALA RAKESH REDDY	kadapalarakeshreddy@gmail.com
4	Y17EC2681	Varikuntla shashikala	varikuntla.shashi@gmail.com
...	...	...	...
132	R141465	Vemannagari Nandini	r141465@rguktrkv.ac.in
133	1710126	Kade Vandana	kadevandana3@gmail.com
134	Y17EC067	KARAMSETTY PRASAD	kkprasad740@gmail.com
135	170040246	G.Venkata sai kumar	gvenkatasaikumar@gmail.com
136	Y17EC125	PATCHAVA BRAHMENDRA	brahmendrapatchava765@gmail.com

137 rows × 3 columns

In [98]:

```
data.head(3)   # accessing first 5 records/ rows / observation
```

Out[98]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	8/6/1993	12:42 PM	97308	6.945	True	Marketing
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
2	Maria	Female	4/23/1993	11:17 AM	130590	11.858	False	Finance

In [99]:

```
data.tail() # last 5 records
```

Out[99]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
995	Henry	NaN	11/23/2014	6:09 AM	132483	16.655	False	Distribution
996	Phillip	Male	1/31/1984	6:30 AM	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 PM	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 PM	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 PM	129949	10.169	True	Sales

In [102]:

```
data_ex.sample(2) # random selection
```

Out[102]:

	Roll number	Name	Email
12	17A81A05B3	Satish adabala	thecodersatish@gmail.com
117	Y18CS162	Tanneeru Ashish	Ashishtannee@gmail.com

In [103]:

```
data.shape  # it returns rows, columns
```

Out[103]:

```
(1000, 8)
```

In [104]:

```
data.columns
```

Out[104]:

```
Index(['First Name', 'Gender', 'Start Date', 'Last Login Time', 'Salary',
       'Bonus %', 'Senior Management', 'Team'],
      dtype='object')
```

In [105]:

```
data.index
```

Out[105]:

```
RangeIndex(start=0, stop=1000, step=1)
```

4. Indexing

In [106]:

```
data.index
```

Out[106]:

```
RangeIndex(start=0, stop=1000, step=1)
```

In [109]:

```
data[11]  # start at 0, end value excusive
# Key error -- dataset is 2D
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

KeyError: 11

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-109-42b293e2267f> in <module>
----> 1 data[11]  # start at 0, end value excusive

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2798             if self.columns.nlevels > 1:
   2799                 return self._getitem_multilevel(key)
-> 2800             indexer = self.columns.get_loc(key)
   2801             if is_integer(indexer):
   2802                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   2646                 return self._engine.get_loc(key)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2649         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2650         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

KeyError: 11
```

In [112]:

```
data[1:200:10] # start, end , step
```

Out[112]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
1	Thomas	Male	3/31/1996	6:53 AM	61933	4.170	True	NaN
11	Julie	Female	10/26/1997	3:19 PM	102508	12.637	True	Legal
21	Matthew	Male	9/5/1995	2:12 AM	100612	13.645	False	Marketing

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
31	Joyce	NaN	2/20/2005	2:40 PM	88657	12.752	False	Product
41	Christine	NaN	6/28/2015	1:08 AM	66582	11.308	True	Business Development
51	NaN	NaN	12/17/2011	8:29 AM	41126	14.009	NaN	Sales
61	Denise	Female	11/6/2001	12:03 PM	106862	3.699	False	Business Development
71	Johnny	Male	11/6/2009	4:23 PM	118172	16.194	True	Sales
81	Christopher	Male	3/30/2008	10:52 AM	47369	14.822	False	Legal
91	James	NaN	1/26/2005	11:00 PM	128771	8.309	False	NaN
101	Aaron	Male	2/17/2012	10:20 AM	61602	11.849	True	Marketing
111	Bonnie	Female	12/17/1999	3:12 PM	42153	8.454	True	Business Development
121	Kathleen	NaN	5/9/2016	8:55 AM	119735	18.740	False	Product
131	Rebecca	Female	7/10/1992	12:23 AM	94231	17.517	False	Product
141	Adam	Male	12/24/1990	8:57 PM	110194	14.727	True	Product
151	Brandon	NaN	11/3/1997	8:17 PM	121333	15.295	False	Business Development
161	Marilyn	NaN	8/22/1999	9:09 AM	103386	11.451	False	Distribution
171	Patrick	Male	8/17/2007	3:16 AM	143499	17.495	True	Engineering
181	Randy	Male	11/14/1999	12:12 PM	58129	1.952	True	Distribution
191	Lois	Female	10/18/2013	4:51 PM	36946	6.652	False	Engineering

In [125]:

```
data.columns
```

Out[125]:

```
Index(['First Name', 'Gender', 'Start Date', 'Last Login Time', 'Salary',
       'Bonus %', 'Senior Management', 'Team'],
      dtype='object')
```

In [116]:

```
type(data["First Name"])  # accessing single column
# one column inside df is called pandas series
```

Out[116]:

```
pandas.core.series.Series
```

In [118]:

```
type(data[["First Name"]] ) # data as sub df
```

Out[118]:

```
pandas.core.frame.DataFrame
```

In [119]:

```
data["First Name","Gender"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()
```

```
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('First Name', 'Gender')

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-119-a2ad82f27c02> in <module>
----> 1 data["First Name","Gender"]

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2798             if self.columns.nlevels > 1:
   2799                 return self._getitem_multilevel(key)
-> 2800             indexer = self.columns.get_loc(key)
   2801             if is_integer(indexer):
   2802                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   2646                 return self._engine.get_loc(key)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2649         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2650         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('First Name', 'Gender')
```

In [120]:

```
data[["First Name","Gender"]]  # sub df
```

Out[120]:

	First Name	Gender
0	Douglas	Male
1	Thomas	Male
2	Maria	Female
3	Jerry	Male
4	Larry	Male
...	...	...
995	Henry	NaN
996	Phillip	Male
997	Russell	Male
998	Larry	Male
999	Albert	Male

1000 rows × 2 columns

In [122]:

```
data["First Name"][5]  # indexing
```

Out[122]:

```
'Dennis'
```

In [124]:

```
data[100:101]["Gender"]
```

Out[124]:

```
100    Female
Name: Gender, dtype: object
```

In [129]:

```
type(data[120:127]["Team"])
```

Out[129]:

```
pandas.core.series.Series
```

In [128]:

```
data[120:127]["Team", "Start Date"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   2645             try:
-> 2646                 return self._engine.get_loc(key)
   2647             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

KeyError: ('Team', 'Start Date')

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-128-6f9aebe1efd3> in <module>
----> 1 data[120:127]["Team", "Start Date"]

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2798             if self.columns.nlevels > 1:
   2799                 return self._getitem_multilevel(key)
-> 2800             indexer = self.columns.get_loc(key)
   2801             if is_integer(indexer):
   2802                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, t
olerance)
   2646                 return self._engine.get_loc(key)
   2647             except KeyError:
-> 2648                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2649         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
   2650         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
tem()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_i
```

```
tem()
```

```
KeyError: ('Team', 'Start Date')
```

LOC -

ILOC

In [ ]:

```
```