Pandas Notes:

Useful Links to Learn :

USER GUIDE :

https://pandas.pydata.org/docs/user_guide/index.html#user-guide

Getting Started with Pandas :

https://pandas.pydata.org/docs/getting_started/index.html#getting-started

Input/output :

https://pandas.pydata.org/docs/reference/io.html

DataFrame : Two-dimensional, size-mutable, potentially heterogeneous tabular data.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

syntax → pandas.DataFrame(*data=None*, *index=None*, *columns=None*, *dtype=None*, *copy=None*)

Generally, we used it to create a 2-D tabular data used for Analysis.

By default, the index is Range Index(0,1,2,3…n).

Dtype*:* Data type to force. Only a single dtype is allowed. If None, infer.

Example: data = pd.DataFrame(data = cancer['data'],columns = cancer['feature_names'])

Pandas **dataframe.infer_objects()** function attempts to infer better data type for input object column. This function attempts soft conversion of object-dtyped columns, leaving non-object and unconvertible columns unchanged. The inference rules are the same as during normal Series/DataFrame construction.

Pandas dataframe.insert() :

Pandas insert method allows the user to insert a column in a dataframe or series(1-D Data frame). A column can also be inserted manually in a data frame by the following method, but there isn't much freedom here.

For example, even column location can't be decided and hence the inserted column is always inserted in the last position.

Syntax: DataFrameName.insert(loc, column, value, allow_duplicates = False)

allow_duplicates : *allow_duplicates, is a Boolean value which checks if column with same name already exists or not.*

pandas.Series

class **pandas.Series(*data=None*, *index=None*, *dtype=None*, *name=None*, *copy=False*, *fastpath=False*)**

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type.

An object is said to be hashable if it has a hash value that remains the same during its lifetime. It has a __hash__() method and it can be compared to other objects. For this, it needs the __eq__() or __cmp__()method. If hashable objects are equal when compared, then they have same hash value.

Note: All immutable built-in objects in python are hashable. Mutable containers like lists and dictionaries are not hashable while immutable container tuple is hashable