**Basic**

import pandas as pd

s = pd.Series([1, 3, 5, np.nan, 6, 8])

dates = pd.date\_range("20130101", periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list("ABCD"))

df2 = pd.DataFrame(

{

"A": 1.0,

"B": pd.Timestamp("20130102"),

"C": pd.Series(1, index=list(range(4)), dtype="float32"),

"D": np.array([3] \* 4, dtype="int32"),

"E": pd.Categorical(["test", "train", "test", "train"]),

"F": "foo",

}

)

df2.dtypes

df2.<TAB>

**View Data**

df.head()

df.tail(3)

df.index

df.columns

df.shape

df.to\_numpy()

df.describe()

df.T

df.sort\_index(axis=1, ascending=False)

df.sort\_values(by="B")

**Getting data**

df["A"]

df[0:3]

df["20130102":"20130104"]

**Selecting by label**

df.loc[dates[0]]

df.loc[:, ["A", "B"]]

df.loc["20130102":"20130104", ["A", "B"]]

df.loc["20130102", ["A", "B"]]

df.loc[dates[0], "A"] or df.at[dates[0], "A"]

**Select by position**

df.iloc[3]

df.iloc[3:5, 0:2]

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

df.iloc[ : , 1:3]

df.iloc[1, 1]

df.iat[1, 1]

**Boolean indexing**

df[df['A'] > 0]

df[df > 0]

df2 = df.copy()

df2["E"] = ["one", "one", "two", "three", "four", "three"]

df2[df2["E"].isin(["two", "four"])] --> Filtering

**Setting data**

s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date\_range("20130102", periods=6))

df["F"] = s1

df.at[dates[0], "A"] = 0 --> set value by label

df.iat[0, 1] = 0 --> set value by position

df.loc[:, "D"] = np.array([5] \* len(df)) --> setting by assigning with numpy array

**Missing Data**

df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ["E"])

df1.loc[dates[0] : dates[1], "E"] = 1

df1.dropna(how="any") --> drop missing values

df1.fillna(value=5) --> fill missing values

pd.isna(df1) --> for boolean mask where values are not present

**Stat ops**

df.mean()

df.mean(1)

s = pd.Series([1, 3, 5, np.nan, 6, 8], index=dates).shift(2) --> multidimentional alignment

df.sub(s, axis="index")

**Apply functions**

df.apply(np.cumsum)

df.apply(lambda x: x.max() - x.min())

**Histograms**

s = pd.Series(np.random.randint(0, 7, size=10))

s.value\_counts()

**String methods**

s = pd.Series(["A", "B", "C", "Aaba", "Baca", np.nan, "CABA", "dog", "cat"])

s.str.lower()

**Merge ops**

1. Concat

df = pd.DataFrame(np.random.randn(10, 4))

pieces = [df[:3], df[3:7], df[7:]] --> breaking the data set

pd.concat(pieces) --> joining them back

**Grouping**

df = pd.DataFrame(

{

"A": ["foo", "bar", "foo", "bar", "foo", "bar", "foo", "foo"],

"B": ["one", "one", "two", "three", "two", "two", "one", "three"],

"C": np.random.randn(8),

"D": np.random.randn(8),

}

)

df.groupby("A").sum()

df.groupby(["A", "B"]).sum()

**Reshaping :**

tuples = list(

zip(

\*[

["bar", "bar", "baz", "baz", "foo", "foo", "qux", "qux"],

["one", "two", "one", "two", "one", "two", "one", "two"],

] ) )

index = pd.MultiIndex.from\_tuples(tuples, names=["first", "second"])

df = pd.DataFrame(np.random.randn(8, 2), index=index, columns=["A", "B"])

df2 = df[:4]

df2

stacked = df2.stack() --> stacking data

stacked.unstack() --> unstacking data

stacked.unstack(1)

stacked.unstack(0)

**Time series**

rng = pd.date\_range("1/1/2012", periods=100, freq="S")

ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)

ts.resample("5Min").sum()

rng = pd.date\_range("3/6/2012 00:00", periods=5, freq="D")

ts = pd.Series(np.random.randn(len(rng)), rng)

ts\_utc = ts.tz\_localize("UTC")

ts\_utc

ts\_utc.tz\_convert("US/Eastern")

rng = pd.date\_range("1/1/2012", periods=5, freq="M")

ts = pd.Series(np.random.randn(len(rng)), index=rng)

ps = ts.to\_period()

ps.to\_timestamp()

**Categorical vars**

df = pd.DataFrame(

{"id": [1, 2, 3, 4, 5, 6], "raw\_grade": ["a", "b", "b", "a", "a", "e"]}

)

df["grade"] = df["raw\_grade"].astype("category")

df['grade']

df["grade"].cat.categories = ["very good", "good", "very bad"] -->Series.cat.catergories()

df["grade"] = df["grade"].cat.set\_categories(

["very bad", "bad", "medium", "good", "very good"]

)

df["grade"]

df.sort\_values(by="grade")

df.groupby("grade").size()

**Plotting data**

import matplotlib.pyplot as plt

plt.close() --> used to close a figure window

ts = pd.Series(np.random.randn(1000), index=pd.date\_range("1/1/2000", periods=1000))

ts = ts.cumsum()

ts.plot();

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=["A", "B", "C", "D"] )

df = df.cumsum()

plt.figure();

df.plot();

plt.legend(loc='best')