References
http://www.danielmsullivan.com/pages/tutorial_stata_to_python.html
http://pandas.pydata.org/pandas-docs/stable/getting_started/comparison/comparison_with_stata.html

Displaying / logging output
log using <file>

Using .dta file with only specific columns
use <varlist> using <dtafile>
➔ df = pd.read_stata('<dtafile>', columns=<varlist>) in python

Read in .xlsx or .csv or .txt files
import excel using <excelfile>
➔ df = pd.read_excel('<excelfile>') in python

import delimited using <csvfile>
➔df = pd.read_csv('<csvfile>') in python

Save as .dta file to certain directory
save <filename>, replace
➔df.to_stata('<filename>') OR
➔df.to_pickle('<filename>') for Python-native file type.

Save as .csv or excel file to certain directory
outsheet using <csv_name>, comma
➔ df.to_csv('<csv_name>')

export excel using <excel_name>
➔ df.to_excel('<excel_name>')

Basic Description and Summary Stats
Tab var ➔ Look at summary statistic and breakdown of values

describe
➔ df.info( ) OR df.dtypes just to get data types. Note that Python does not have value labels like Stata does.
describe <var> df[<var>].dtype

count
➔ df.shape[0] OR len(df). Here df.shape returns a tuple with the length and width of the DataFrame.

count if <condition>
➔ df[<condition>].shape[0] OR (<condition>).sum() if the condition involves a DataFrame, e.g., (df['age'] > 2).sum()

summ <var> df['<var>'].describe()

summ <var> if <condition> df[<condition>][<var>].describe() OR df.loc[<condition>, <var>].describe()

summ <var> [aw = <weight>]
➔ Right now you have to calculate weighted summary stats manually. There are also some tools available in the Statsmodels package.

summ <var>, d
➔ df[<var>].describe() plus df[<var>].quantile([.1, .25, .5, .75, .9]) or whatever other statistics you want.

Keep and drop
keep if <condition> df = df[<condition>]
drop if <condition> df = df[~(<condition>)]
keep <var> df = df[<var>]
keep varstem* df = df.filter(like='varstem*')
drop <var> del df[<var>] OR df = df.drop(<var>, axis=1)

drop varstem* df = df.drop(df.filter(like='varstem*').columns, axis=1)

Create new variables and replace

gen <newvar> = <expression>
gen <newvar> = <expression> if <condition>
replace <var> = <expression> if <condition>

egen <newvar> = count(<var>)
➔ <newvar> = df[<var>].notnull().sum(). NOTE: For these egen commands, <newvar> is a full (constant) column in Stata, while it is a scalar in Python.

egen <newvar> = group(<varlist>)
➔ <newvar> = econtools.group_id(df, cols=<varlist>)

egen <newvar> = max(<var>)
➔ <newvar> = df[<var>].max()

egen <newvar> = mean(<var>)
➔ <newvar> = df[<var>].mean()

egen <newvar> = total(<var>)
➔ <newvar> = df[<var>].sum()

egen <newvar> = <stat>(<var>), by(<groupvars>)
➔ df[<newvar>] = df.groupby(<groupvars>)[<var>].transform('<stat>').

Other manipulations
rename (var1) (var2) Rename Variables; rename var1 to var2

inlist(<var>, <val1>, <val2>) ➔ df[<var>].isin((<val1>, <val2>))
inrange(<var>, <val1>, <val2>) ➔ df[<var>].between((<val1>, <val2>))

collapse (sd) <var> (median) <var> ///
(max) <var> (min) <var>, ///
by(<groupvars>)
➔ df.groupby(<groupvars>)[<var>].agg(['std', 'median', 'min', 'max', 'sum'])

collapse (<stat>) <var> [iw = <weight>]

collapse (<stat>) <stat_vars>, by(<groupvars>)
➔ df.groupby(<groupvars>)[<stat_vars>].<stat>()

recode <var> (1/5 = 1)

recode <var> (1/5 = 1), gen(<newvar>)

label var <var> <label>

label define <labelname> 1 <valuelabel>

label values <var> <labelname>

label list <labelname>

Parsing subset of strings
subinstr(<str>, " ", "_", .) ➔ df[<var>].str.replace(' ', '_')
Gen dob_yr = substr(DOB, -4, .) # parse 4th from last to last string
Substr(raw_name, 1, 3) #parse 1st string to 3rd string from raw_name

Substituting strings with some other value/string

Subinstr(var, “x”, “”, .); #replace “x” with empty string(“”) for all values in column var

Sort observations using var or multiple vars
Sort var1 var2;

Fill in missing values using values from the same group
bysort id (number): replace number=number[_N]
That would sort the highest value observed in each block (which could be missing) to the end of each block and use that to overwrite all values in the same block, regardless of whether values were missing.

bysort id: replace number=number[1]
bysort id : replace number = number[_n-1] if missing(number) & _n > 1

Certain utility functions not working … then
global codedir "D:/Users/{uniqname}/Desktop/code”

merge and join
use “[PATH]”, clear;
merge 1:1 CASE_NUMBER record_id using [file name (without .dta)]

append using <filename>	df_joint = df1.append(df2)
merge 1:1 <vars> using <filename>	df_joint = df1.join(df2) if <vars> are the DataFrames' indexes, or df_joint = pd.merge(df1, df2, on=<vars>) otherwise. Beware that pd.merge will not keep the index of either DataFrame. NOTE: Merging in Python is like R, SQL, etc. Needs more robust explanation.

Pandas how	*Stata , keep()*	*Intuition*
how='left'	keep(1, 3)	Keeps all observations in the "left" DataFrame.
how='right'	keep(2, 3)	Keeps all observations in the "right" DataFrame.
how='inner'	keep(3)	Keeps observations that are in both DataFrames.
how='outer'	keep(1 2 3)	Keeps all observations.

Reshape

reshape <wide/long> <stubs>, i(<vars>) j(<var>) |||||	wide: df.unstack(<level>) long: df.stack(<column_level>) see also df.pivot

Extracting year,month,day info from date-like formatted string
year(date(adj_sent_dt, “MDY”));
month
day

remove (or more like change them to empty string) if a value does not start with “$” (usually for amount vars)
replace rest_amt = “” if regex(rest_amt, “(^[\$])”)

remove (or more like change them to empty string) if a value does not have xx/xx/xx date format (usually for date vars)
replace sent_dt = “” if !regexm(sent_dt, “(^[0-9]+/[0-9]+/[0-9]+$)”)

Getting position index of a certain character in string
strops(string, “__”) : get position index of first occurrence of “__” in string
strrpos(string, “__”) : get position index of last occurrence of “__” in string

similar to try except
capture
_rc != 0 {

}

Getting unique number of observations (in this example, unique # of ids)
egen x = group(id)
sum x
>> Variable | Obs Mean Std. Dev. Min Max
---+---
x | 5 1.4 .5477226 1 2
The value for Max is the # of unique ids.

Changing DOB or some date variable that has mm/dd/yyyy 00:00:00 format with DOUBLE/FLOAT data type to string

Gen newvar = String(DOB, "%tcDDmonCCYY")

Setting index of panel data
tsset <panelvar> <timevar>
➔ df = df.set_index([<panelvar>, <timevar>])

Lags for time series
L.<var>
➔ df.shift() NOTE: The index must be correctly sorted for shift to work the way you want it to.

L2.<var>
➔ df.shift(2)

F.<var>
➔ df.shift(-1)

Econometrics

Stata	Python
ttest <var>, by(<var>)	from scipy.stats import ttest_ind ttest_ind(<array1>, <array2>)
xi: i.<var>	pd.get_dummies(df[<var>])
i.<var2>#c.<var1>	pd.get_dummies(df[<var2>]).multiply(df[<var1>])
reg <yvar> <xvar> if <condition>, r	import econtools.metrics as mt results = mt.reg(df[<condition>], <yvar>, <xvar>, robust=True)
reg <yvar> <xvar> if <condition>, vce(cluster <clustervar>)	results = mt.reg(df[<condition>], <yvar>, <xvar>, cluster=<clustervar>)
areg <yvar> <xvar>, absorb(<fe_var>)	results = mt.reg(df, <yvar>, <xvar>, a_name=<fe_var>)
predict <newvar>, resid	<newvar> = results.resid
predict <newvar>, xb	<newvar> = results.yhat
_b[<var>], _se[<var>]	results.beta[<var>], results.se[<var>]
test <varlist>	results.Ftest(<varlist>)
test <varlist>, equal	results.Ftest(<varlist>, equal=True)

Stata	Python
lincom <var1> + <var2>	econtools.metrics.f_test with appropriate parameters.
ivreg2	econtools.metrics.ivreg
outreg2	econtools.outreg
reghdfe	None (hoping to add it to Econtools soon).

Plotting

Stata	*Python*
binscatter	econtools.binscatter
maptile	No quick tool, but easy to do with Cartopy.
coefplot	ax.scatter(results.beta.index, results.beta) often works. Depends on context.
twoway scatter <var1> <var2>	df.scatter(<var2>, <var1>)
twoway scatter <var1> <var2> if <condition>	df[<condition>].scatter(<var2>, <var1>)
twoway <connected/line/area/bar/rarea>	As above, though ax.plot(<var1>, <var2>) is better. Like merge, it's a different paradigm, needs more explanation.