

Module	Description	Example	Script
bs4	extracting the text from a tag	pagelist = [int(a.text) for a in alist]	g31/demo.py
bs4	finding a tag with a given class	paging = soup.find('div',{'class':'paging'})	g31/demo.py
bs4	finding all A tags	alist = pages.findAll('a')	g31/demo.py
bs4	finding an HTML title tag	title = soup.find('title').text	g31/demo.py
bs4	importing the main function	from bs4 import BeautifulSoup	g31/demo.py
bs4	parsing a web page	soup = BeautifulSoup(response.text,'html.parser')	g31/demo.py
core	continue, going on to next loop item	continue	g06/demo.py
core	dictionary, adding a new entry	co['po'] = 'CO'	g05/demo.py
core	dictionary, creating	co = {'name':'Colorado', 'capital':'Denver'}	g05/demo.py
core	dictionary, creating via comprehension	word_lengths = { w:len(w) for w in wordlist }	g06/demo.py
core	dictionary, deleting an element	del races[year]	g31/demo.py
core	dictionary, iterating through key-value pairs	for w,l in word_lengths.items():	g06/demo.py
core	dictionary, looking up a value	name = ny['name']	g05/demo.py
core	dictionary, making a list of	list1 = [co,ny]	g05/demo.py
core	dictionary, obtaining a list of keys	names = super_dict.keys()	g05/demo.py
core	f-string, grouping with commas	print(f'Total population: {tot_pop:,}')	g12/demo.py
core	f-string, using a formatting string	print(f"PV of {payment} with T={year} and r={r} is \${p...")	g08/demo.py
core	file, closing	fh.close()	g02/demo.py
core	file, opening for reading	fh = open('states.csv')	g05/demo.py
core	file, opening for writing	fh = open(filename,"w")	g02/demo.py
core	file, output using print	print("It was written during",year,file=fh)	g02/demo.py
core	file, output using write	fh.write("Where was this file was written?\n")	g02/demo.py
core	file, print without adding spaces	print('\nOuter:\n', join_o['_merge'].value_counts(), s...)	g15/demo.py
core	file, reading one line at a time	for line in fh:	g05/demo.py
core	for, looping through a list	for n in a_list:	g04/demo.py
core	for, looping through a list of tuples	for number,name in div_info:	g14/demo.py
core	function, calling	d1_ssqr = sumsq(d1)	g07/demo.py
core	function, calling with an optional argument	sample_function(100, 10, r=0.07)	g08/demo.py
core	function, defining	def sumsq(values: list) -> float:	g07/demo.py
core	function, defining with optional argument	def sample_function(payment:float,year:int,r:float=0.05...)	g08/demo.py
core	function, returning a result	return values	g07/demo.py

Module	Description	Example	Script
core	function, using type hinting	<code>def readlist(filename: str) -> list:</code>	g07/demo.py
core	if, starting a conditional block	<code>if l == 5:</code>	g06/demo.py
core	if, using an elif statement	<code>elif s.isalpha():</code>	g06/demo.py
core	if, using an else statement	<code>else:</code>	g06/demo.py
core	list, appending an element	<code>a_list.append("four")</code>	g03/demo.py
core	list, create via comprehension	<code>cubes = [n**3 for n in a_list]</code>	g04/demo.py
core	list, creating	<code>a_list = ["zero", "one", "two", "three"]</code>	g03/demo.py
core	list, determining length	<code>n = len(b_list)</code>	g03/demo.py
core	list, extending with another list	<code>a_list.extend(a_more)</code>	g03/demo.py
core	list, generating a sequence	<code>b_list = range(1,6)</code>	g04/demo.py
core	list, joining with spaces	<code>a_string = " ".join(a_list)</code>	g03/demo.py
core	list, selecting an element	<code>print(a_list[0])</code>	g03/demo.py
core	list, selecting elements 0 to 3	<code>print(a_list[:4])</code>	g03/demo.py
core	list, selecting elements 1 to 2	<code>print(a_list[1:3])</code>	g03/demo.py
core	list, selecting elements 1 to the end	<code>print(a_list[1:])</code>	g03/demo.py
core	list, selecting last 3 elements	<code>print(a_list[-3:])</code>	g03/demo.py
core	list, selecting the last element	<code>print(a_list[-1])</code>	g03/demo.py
core	list, sorting	<code>c_sort = sorted(b_list)</code>	g03/demo.py
core	list, summing	<code>total = sum(numbers)</code>	g06/demo.py
core	math, raising a number to a power	<code>a_cubes.append(n**3)</code>	g04/demo.py
core	math, rounding a number	<code>rounded = round(ratio,2)</code>	g05/demo.py
core	sets, computing difference	<code>print(name_states - pop_states)</code>	g14/demo.py
core	sets, creating	<code>name_states = set(name_data['State'])</code>	g14/demo.py
core	sets, of tuples	<code>tset1 = set([(1,2), (2,3), (1,3), (2,3)])</code>	g14/demo.py
core	string, concatenating	<code>name = s1+" "+s2+" "+s3</code>	g02/demo.py
core	string, convert to lower case	<code>lower = [s.lower() for s in wordlist]</code>	g06/demo.py
core	string, convert to title case	<code>new_s = s.title()</code>	g06/demo.py
core	string, converting to an int	<code>value = int(s)</code>	g06/demo.py
core	string, creating	<code>filename = "demo.txt"</code>	g02/demo.py
core	string, finding starting index	<code>mm_start = long_string.find("mm")</code>	g06/demo.py
core	string, including a newline character	<code>fh.write(name+"!\n")</code>	g02/demo.py
core	string, is entirely numeric	<code>if s.isnumeric():</code>	g06/demo.py

Module	Description	Example	Script
core	string, matching a substring	<code>has_ñ = [s for s in lower if "ñ" in s]</code>	g06/demo.py
core	string, matching end	<code>a_end = [s for s in lower if s.endswith("a")]</code>	g06/demo.py
core	string, matching multiple starts	<code>ab_start = [s for s in lower if s.startswith(starters)]</code>	g06/demo.py
core	string, matching partial string	<code>is_gas = trim['DBA Name'].str.contains('XPRESS')</code>	g29/demo.py
core	string, matching start	<code>a_start = [s for s in lower if s.startswith("a")]</code>	g06/demo.py
core	string, replacing a substring	<code>words = s.replace(","," ").split()</code>	g06/demo.py
core	string, splitting on a comma	<code>parts = line.split(',')</code>	g05/demo.py
core	string, splitting on whitespace	<code>b_list = b_string.split()</code>	g03/demo.py
core	string, stripping blank space	<code>clean = [item.strip() for item in parts]</code>	g05/demo.py
core	tuple, creating	<code>starters = ("a","b","0")</code>	g06/demo.py
core	type, obtaining for a variable	<code>print('\nraw_states is a DataFrame object:', type(raw_...</code>	g10/demo.py
core	while: looping while a condition is True	<code>while numpage <= pagemax:</code>	g31/demo.py
csv	setting up a DictReader object	<code>reader = csv.DictReader(fh)</code>	g09/demo.py
fiona	importing the module	<code>import fiona</code>	g25/demo.py
fiona	list layers in a geopackage	<code>layers = fiona.listlayers(demo_file)</code>	g25/demo.py
geopandas	adding a heatmap legend	<code>slices.plot('s_pop',edgecolor='yellow',linewidth=0.2,le...</code>	g27/demo.py
geopandas	building points from lat, lon	<code>geom = gpd.points_from_xy(adds['lon'], adds['lat'])</code>	g30/demo.py
geopandas	clip a layer	<code>zips_clip = zips.clip(county,keep_geom_type=True)</code>	g25/demo.py
geopandas	combine all geographies in a layer	<code>water_dis = water_by_name.dissolve()</code>	g25/demo.py
geopandas	combine geographies by attribute	<code>water_by_name = water.dissolve('FULLNAME')</code>	g25/demo.py
geopandas	computing areas	<code>zips['z_area'] = zips.area</code>	g27/demo.py
geopandas	construct a buffer	<code>near_water = water_dis.buffer(1600)</code>	g25/demo.py
geopandas	constructing centroids	<code>centroids['geometry'] = tracts.centroid</code>	g29/demo.py
geopandas	drawing a heatmap	<code>near_wv.plot("mil",cmap='Blues',legend=True,ax=ax)</code>	g23/demo.py
geopandas	extracting geometry from a geodataframe	<code>wv_geo = wv['geometry']</code>	g23/demo.py
geopandas	importing the module	<code>import geopandas as gpd</code>	g22/demo.py
geopandas	join to nearest object	<code>served_by = centroids.sjoin_nearest(geo,how='left',dist...</code>	g29/demo.py
geopandas	merging data onto a geodataframe	<code>conus = conus.merge(trim,on='STATEFP',how='left',valida...</code>	g23/demo.py
geopandas	obtaining coordinates	<code>print('Number of points:', len(wv_geo.exterior.coords)...</code>	g23/demo.py
geopandas	overlaying a layer using union	<code>slices = zips.overlay(county,how='union',keep_geom_type...</code>	g27/demo.py
geopandas	plot with categorical coloring	<code>sel.plot('NAME',cmap='Dark2',ax=ax1)</code>	g23/demo.py

Module	Description	Example	Script
geopandas	plotting a boundary	<code>syr.boundary.plot(color='gray',linewidth=1,ax=ax1)</code>	<code>g22/demo.py</code>
geopandas	project a layer	<code>county = county.to_crs(epsg=utm18n)</code>	<code>g25/demo.py</code>
geopandas	reading a file	<code>syr = gpd.read_file("tl_2016_36_place-syracuse.zip")</code>	<code>g22/demo.py</code>
geopandas	reading a shapefile	<code>states = gpd.read_file("cb_2019_us_state_500k.zip")</code>	<code>g23/demo.py</code>
geopandas	reading a zip with a subdirectory	<code>stores = gpd.read_file(zip2+'!'+subdir)</code>	<code>g32/demo.py</code>
geopandas	reading data in WKT format	<code>coords = gpd.GeoSeries.from_wkt(big['Georeference'])</code>	<code>g29/demo.py</code>
geopandas	reading one layer from a zip	<code>county = gpd.read_file(zip1+'!'+layer)</code>	<code>g32/demo.py</code>
geopandas	setting the color of a plot	<code>county.plot(color='tan',ax=ax1)</code>	<code>g25/demo.py</code>
geopandas	setting transparency via alpha	<code>near_clip.plot(alpha=0.25,ax=ax1)</code>	<code>g25/demo.py</code>
geopandas	spatial join, contains	<code>c_contains_z = county.sjoin(zips,how='right',predicate=...</code>	<code>g26/demo.py</code>
geopandas	spatial join, crosses	<code>i_crosses_z = inter.sjoin(zips,how='right',predicate='c...</code>	<code>g26/demo.py</code>
geopandas	spatial join, intersects	<code>z_intersect_c = zips.sjoin(county,how='left',predicate=...</code>	<code>g26/demo.py</code>
geopandas	spatial join, overlaps	<code>z_overlaps_c = zips.sjoin(county,how='left',predicate='...</code>	<code>g26/demo.py</code>
geopandas	spatial join, touches	<code>z_touch_c = zips.sjoin(county,how='left',predicate='tou...</code>	<code>g26/demo.py</code>
geopandas	spatial join, within	<code>z_within_c = zips.sjoin(county,how='left',predicate='wi...</code>	<code>g26/demo.py</code>
geopandas	testing if rows touch a geometry	<code>touches_wv = conus.touches(wv_geo)</code>	<code>g23/demo.py</code>
geopandas	writing a layer to a geodatabase	<code>conus.to_file("conus.gpkg",layer="states")</code>	<code>g23/demo.py</code>
glob	importing the module	<code>import glob</code>	<code>g32/demo.py</code>
glob	listing files via wildcards	<code>nyiso = glob.glob('raw/20??01*')</code>	<code>g32/demo.py</code>
json	importing the module	<code>import json</code>	<code>g05/demo.py</code>
json	using to print an object nicely	<code>print(json.dumps(list1,indent=4))</code>	<code>g05/demo.py</code>
matplotlib	axes, adding a horizontal line	<code>ax21.axhline(medians['etr'], c='r', ls='-', lw=1)</code>	<code>g13/demo.py</code>
matplotlib	axes, adding a vertical line	<code>ax21.axvline(medians['inc'], c='r', ls='-', lw=1)</code>	<code>g13/demo.py</code>
matplotlib	axes, labeling the X axis	<code>ax2.set_xlabel('Millions')</code>	<code>g12/demo.py</code>
matplotlib	axes, labeling the Y axis	<code>ax1.set_ylabel('Millions')</code>	<code>g12/demo.py</code>
matplotlib	axes, turning off a label	<code>ax.set_ylabel(None)</code>	<code>g14/demo.py</code>
matplotlib	axis, turning off	<code>ax1.axis('off')</code>	<code>g27/demo.py</code>
matplotlib	changing marker shape	<code>geo.plot(color='red', marker='D', markersize=20, ax=ax)</code>	<code>g30/demo.py</code>
matplotlib	changing marker size	<code>geo.plot(color='blue',markersize=1,ax=ax1)</code>	<code>g29/demo.py</code>
matplotlib	colors, xkcd palette	<code>syr.plot(color='xkcd:lightblue',ax=ax1)</code>	<code>g22/demo.py</code>
matplotlib	figure, adding a title	<code>fig2.suptitle('Pooled Data')</code>	<code>g13/demo.py</code>
matplotlib	figure, four panel grid	<code>fig3, axs = plt.subplots(2,2,sharex=True,sharey=True)</code>	<code>g13/demo.py</code>
matplotlib	figure, left and right panels	<code>fig2, (ax21,ax22) = plt.subplots(1,2)</code>	<code>g13/demo.py</code>
matplotlib	figure, saving	<code>fig2.savefig('figure.png')</code>	<code>g12/demo.py</code>

Module	Description	Example	Script
matplotlib	figure, setting the size	fig, axs = plt.subplots(1,2,figsize=(12,6))	g21/demo.py
matplotlib	figure, tuning the layout	fig2.tight_layout()	g12/demo.py
matplotlib	figure, working with a list of axes	for ax in axs:	g21/demo.py
matplotlib	importing pyplot	import matplotlib.pyplot as plt	g12/demo.py
matplotlib	setting a linewidth	us.boundary.plot(color='black', linewidth=0.4, ax=ax)	g30/demo.py
matplotlib	setting an edge color	slices.plot('COUNTYFP', edgecolor='yellow', linewidth=0.2...	g27/demo.py
matplotlib	setting the default resolution	plt.rcParams['figure.dpi'] = 300	g12/demo.py
matplotlib	using subplots to set up a figure	fig1, ax1 = plt.subplots()	g12/demo.py
os	delete a file	os.remove(out_file)	g25/demo.py
os	importing the module	import os	g25/demo.py
os	listing files in a directory	files = os.listdir('raw')	g32/demo.py
os	path, base file name	print(' Basename:', os.path.basename(fname))	g32/demo.py
os	path, directory name	print(' Dirname: ', os.path.dirname(fname))	g32/demo.py
os	path, split directory and filename	print(' Split: ', os.path.split(fname))	g32/demo.py
os	path, split filename and extension	print(' Splitext:', os.path.splitext(fname))	g32/demo.py
os	path, test for directory	print(' Dir? ', os.path.isdir(fname))	g32/demo.py
os	path, test for regular file	print(' File? ', os.path.isfile(fname))	g32/demo.py
os	test if a file or directory exists	if os.path.exists(out_file):	g25/demo.py
pandas	RE, replacing a digit or space	unit_part = values.str.replace(r'\d\s', "", regex=True)	g24/demo.py
pandas	RE, replacing a non-digit or space	value_part = values.str.replace(r'\D\s', "", regex=True)	g24/demo.py
pandas	RE, replacing a non-word character	units = units.str.replace(r'\W', "", regex=True)	g24/demo.py
pandas	columns, dividing along index	by_day_pct = 100*by_day_use.div(by_day_tot, axis='index'...	g18/demo.py
pandas	columns, dividing with explicit alignment	normed2 = 100*states.div(pa_row, axis='columns')	g10/demo.py
pandas	columns, listing names	print('\nColumns:', list(raw_states.columns))	g10/demo.py
pandas	columns, renaming	county = county.rename(columns={'B01001_001E': 'pop'})	g11/demo.py
pandas	columns, retrieving one by name	pop = states['pop']	g10/demo.py
pandas	columns, retrieving several by name	print(pop[some_states]/1e6)	g10/demo.py
pandas	dataframe, appending	gen_all = pd.concat([gen_oswego, gen_onondaga])	g16/demo.py
pandas	dataframe, appending via dictionary	bg_all = pd.concat(bg_data)	g32/demo.py
pandas	dataframe, boolean row selection	print(trim[has_AM], "\n")	g13/demo.py
pandas	dataframe, dropping a column	both = both.drop(columns='_merge')	g16/demo.py
pandas	dataframe, dropping duplicates	flood = flood.drop_duplicates(subset='TAX_ID')	g15/demo.py
pandas	dataframe, dropping missing data	merged = geocodes.dropna()	g12/demo.py

Module	Description	Example	Script
pandas	dataframe, finding duplicate records	dups = parcels.duplicated(subset='TAX_ID', keep=False. . .	g15/demo.py
pandas	dataframe, getting a block of rows via index	sel = merged.loc[number]	g14/demo.py
pandas	dataframe, inner 1:1 merge	join_i = parcels.merge(flood, how='inner', on="TAX_ID",. . .	g15/demo.py
pandas	dataframe, inner join	merged = name_data.merge(pop_data,left_on="State",right. . .	g14/demo.py
pandas	dataframe, left 1:1 merge	join_l = parcels.merge(flood, how='left', on="TAX_ID",. . .	g15/demo.py
pandas	dataframe, left m:1 merge	both = gen_all.merge(plants, how='left', on='Plant Code. . .	g16/demo.py
pandas	dataframe, making a copy	trim = trim.copy()	g13/demo.py
pandas	dataframe, melting	long_form = means.reset_index().melt(id_vars='month')	g18/demo.py
pandas	dataframe, outer 1:1 merge	join_o = parcels.merge(flood, how='outer', on="TAX_ID",. . .	g15/demo.py
pandas	dataframe, pivoting	by_day_use = usage.pivot(index=['month','day'],columns=. . .	g18/demo.py
pandas	dataframe, reading Excel file	wb1 = pd.read_excel(case1)	g32/demo.py
pandas	dataframe, reading Stata DTA file	wbx = pd.read_stata('bg_single.dta')	g32/demo.py
pandas	dataframe, reading several Excel sheets	wb2 = pd.read_excel(case2,sheet_name=None)	g32/demo.py
pandas	dataframe, reading zipped pickle format	sample2 = pd.read_pickle('sample_pkl.zip')	g17/demo.py
pandas	dataframe, resetting the index	hourly = hourly.reset_index()	g18/demo.py
pandas	dataframe, right 1:1 merge	join_r = parcels.merge(flood, how='right', on="TAX_ID",. . .	g15/demo.py
pandas	dataframe, saving in zipped pickle format	sample.to_pickle('sample_pkl.zip')	g17/demo.py
pandas	dataframe, selecting a sample	print(bg_all.sample(10))	g32/demo.py
pandas	dataframe, selecting rows by list indexing	print(low_to_high[-5:])	g10/demo.py
pandas	dataframe, selecting rows via boolean	dup_rec = flood[dups]	g15/demo.py
pandas	dataframe, selecting rows via query	trimmed = county.query("state == '04' or state == '36' ")	g11/demo.py
pandas	dataframe, selective drop of missing data	trim = demo.dropna(subset="Days")	g13/demo.py
pandas	dataframe, set index keeping the column	states = states.set_index('STUSPS',drop=False)	g23/demo.py
pandas	dataframe, shape attribute	print('number of rows, columns:', conus.shape)	g23/demo.py
pandas	dataframe, sorting by a column	county = county.sort_values('pop')	g11/demo.py
pandas	dataframe, sorting by index	summary = summary.sort_index(ascending=False)	g16/demo.py
pandas	dataframe, summing a boolean	print('\nduplicate parcels:', dups.sum())	g15/demo.py
pandas	dataframe, summing across columns	by_day_tot = by_day_use.sum(axis='columns')	g18/demo.py
pandas	dataframe, unstacking an index level	bymo = bymo.unstack('month')	g18/demo.py
pandas	dataframe, using a multilevel column index	means = grid['mean']	g21/demo.py
pandas	dataframe, using compare	cmp = wb_a.compare(wb_b,result_names=('wb_a','wb_b'))	g32/demo.py
pandas	dataframe, using xs to select a subset	print(county.xs('04',level='state'))	g11/demo.py
pandas	dataframe, using xs with columns	c1 = grid.xs('c1',axis='columns',level=1)	g21/demo.py
pandas	dataframe, writing Stata DTA file	wb1.to_stata('bg_single.dta',write_index=False)	g32/demo.py
pandas	dataframe, writing to a CSV file	merged.to_csv('demo-merged.csv')	g14/demo.py
pandas	datetime, building via to_datetime()	date = pd.to_datetime(recs['ts'])	g15/demo.py

Module	Description	Example	Script
pandas	datetime, building with a format	<code>ymd = pd.to_datetime(sample['TRANSACTION_DT'], format=...</code>	<code>g17/demo.py</code>
pandas	datetime, extracting day attribute	<code>recs['day'] = date.dt.day</code>	<code>g15/demo.py</code>
pandas	datetime, extracting hour attribute	<code>recs['hour'] = date.dt.hour</code>	<code>g15/demo.py</code>
pandas	general, display information about object	<code>sample.info()</code>	<code>g17/demo.py</code>
pandas	general, displaying all columns	<code>pd.set_option('display.max_columns',None)</code>	<code>g17/demo.py</code>
pandas	general, displaying all rows	<code>pd.set_option('display.max_rows', None)</code>	<code>g10/demo.py</code>
pandas	general, importing the module	<code>import pandas as pd</code>	<code>g10/demo.py</code>
pandas	general, using <code>copy_on_write</code> mode	<code>pd.options.mode.copy_on_write = True</code>	<code>g17/demo.py</code>
pandas	general, using <code>qcut</code> to create deciles	<code>dec = pd.qcut(county['pop'], 10, labels=range(1,11))</code>	<code>g11/demo.py</code>
pandas	groupby, cumulative sum within group	<code>cumulative_inc = group_by_state['pop'].cumsum()</code>	<code>g11/demo.py</code>
pandas	groupby, descriptive statistics	<code>inc_stats = group_by_state['pop'].describe()</code>	<code>g11/demo.py</code>
pandas	groupby, iterating over groups	<code>for t,g in group_by_state:</code>	<code>g11/demo.py</code>
pandas	groupby, median of each group	<code>pop_med = group_by_state['pop'].median()</code>	<code>g11/demo.py</code>
pandas	groupby, quantile of each group	<code>pop_25th = group_by_state['pop'].quantile(0.25)</code>	<code>g11/demo.py</code>
pandas	groupby, return group number	<code>groups = group_by_state.ngroup()</code>	<code>g11/demo.py</code>
pandas	groupby, return number within group	<code>seqnum = group_by_state.cumcount()</code>	<code>g11/demo.py</code>
pandas	groupby, return rank within group	<code>rank_age = group_by_state['pop'].rank()</code>	<code>g11/demo.py</code>
pandas	groupby, select first records	<code>first2 = group_by_state.head(2)</code>	<code>g11/demo.py</code>
pandas	groupby, select largest values	<code>largest = group_by_state['pop'].nlargest(2)</code>	<code>g11/demo.py</code>
pandas	groupby, select last records	<code>last2 = group_by_state.tail(2)</code>	<code>g11/demo.py</code>
pandas	groupby, size of each group	<code>num_rows = group_by_state.size()</code>	<code>g11/demo.py</code>
pandas	groupby, sum of each group	<code>state = county.groupby('state')['pop'].sum()</code>	<code>g11/demo.py</code>
pandas	index, creating with 3 levels	<code>county = county.set_index(['state','county', 'NAME'])</code>	<code>g11/demo.py</code>
pandas	index, dropping a level	<code>rates.index = rates.index.droplevel('title')</code>	<code>g31/demo.py</code>
pandas	index, listing names	<code>print('\nIndex (rows):', list(raw_states.index))</code>	<code>g10/demo.py</code>
pandas	index, renaming values	<code>div_pop = div_pop.rename(index=div_names)</code>	<code>g12/demo.py</code>
pandas	index, retrieving a row by name	<code>pa_row = states.loc['Pennsylvania']</code>	<code>g10/demo.py</code>
pandas	index, retrieving first rows by location	<code>print(low_to_high.iloc[0:10])</code>	<code>g10/demo.py</code>
pandas	index, retrieving last rows by location	<code>print(low_to_high.iloc[-5:])</code>	<code>g10/demo.py</code>
pandas	index, setting to a column	<code>states = raw_states.set_index('name')</code>	<code>g10/demo.py</code>
pandas	plotting, bar plot	<code>reg_pop.plot.bar(title='Population',ax=ax1)</code>	<code>g12/demo.py</code>
pandas	plotting, histogram	<code>hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu...</code>	<code>g13/demo.py</code>
pandas	plotting, horizontal bar plot	<code>div_pop.plot.barh(title='Population',ax=ax2)</code>	<code>g12/demo.py</code>

Module	Description	Example	Script
pandas	plotting, scatter colored by 3rd var	<code>tidy_data.plot.scatter(ax=ax4,x='Income',y='ETR',c='typ. . .</code>	<code>g13/demo.py</code>
pandas	plotting, scatter plot	<code>hh_data.plot.scatter(ax=ax21,x='inc',y='etr',title='ETR. . .</code>	<code>g13/demo.py</code>
pandas	plotting, stacked bar plot	<code>rates.plot.bar(stacked=True,ax=ax)</code>	<code>g31/demo.py</code>
pandas	plotting, turning off legend	<code>sel.plot.barh(x='Name',y='percent',ax=ax,legend=None)</code>	<code>g14/demo.py</code>
pandas	reading, csv data	<code>raw_states = pd.read_csv('state-data.csv')</code>	<code>g10/demo.py</code>
pandas	reading, from an open file handle	<code>gen_oswego = pd.read_csv(fh1)</code>	<code>g16/demo.py</code>
pandas	reading, setting index column	<code>state_data = pd.read_csv('state-data.csv',index_col='na. . .</code>	<code>g12/demo.py</code>
pandas	reading, using dtype dictionary	<code>county = pd.read_csv('county_pop.csv',dtype=fips)</code>	<code>g11/demo.py</code>
pandas	series, RE at start	<code>is_LD = trim['Number'].str.contains(r"1 2")</code>	<code>g13/demo.py</code>
pandas	series, applying a function to each element	<code>name_clean = name_parts.apply(' '.join)</code>	<code>g24/demo.py</code>
pandas	series, automatic alignment by index	<code>merged['percent'] = 100*merged['pop']/div_pop</code>	<code>g14/demo.py</code>
pandas	series, check if all elements are true	<code>print((wb1 == wbx).all())</code>	<code>g32/demo.py</code>
pandas	series, combining via where()	<code>mod['comb_units'] = unit_part.where(unit_part!="", mo. . .</code>	<code>g24/demo.py</code>
pandas	series, contains RE or RE	<code>is_TT = trim['Days'].str.contains(r"Tu Th")</code>	<code>g13/demo.py</code>
pandas	series, contains a plain string	<code>has_AM = trim['Time'].str.contains("AM")</code>	<code>g13/demo.py</code>
pandas	series, contains an RE	<code>has_AMPM = trim['Time'].str.contains("AM.*PM")</code>	<code>g13/demo.py</code>
pandas	series, converting strings to title case	<code>fixname = subset_view['NAME'].str.title()</code>	<code>g17/demo.py</code>
pandas	series, converting to a list	<code>print(name_data['State'].tolist())</code>	<code>g14/demo.py</code>
pandas	series, converting to lower case	<code>name = mod['name'].str.lower()</code>	<code>g24/demo.py</code>
pandas	series, dropping rows using a list	<code>conus = states.drop(not_conus)</code>	<code>g23/demo.py</code>
pandas	series, element-by-element or	<code>is_either = is_ca is_tx</code>	<code>g17/demo.py</code>
pandas	series, filling missing values	<code>mod['comb_units'] = mod['comb_units'].fillna('feet')</code>	<code>g24/demo.py</code>
pandas	series, removing spaces	<code>units = units.str.strip()</code>	<code>g24/demo.py</code>
pandas	series, replacing values using a dictionary	<code>units = units.replace(spellout)</code>	<code>g24/demo.py</code>
pandas	series, retrieving an element	<code>print("\nFlorida's population:", pop['Florida']/1e6)</code>	<code>g10/demo.py</code>
pandas	series, sort in decending order	<code>div_pop = div_pop.sort_values(ascending=False)</code>	<code>g12/demo.py</code>
pandas	series, sorting by value	<code>low_to_high = normed['med_pers_inc'].sort_values()</code>	<code>g10/demo.py</code>
pandas	series, splitting strings on whitespace	<code>name_parts = name.str.split()</code>	<code>g24/demo.py</code>
pandas	series, splitting via RE	<code>trim['Split'] = trim["Time"].str.split(r": - ")</code>	<code>g13/demo.py</code>
pandas	series, splitting with expand	<code>exp = trim["Time"].str.split(r": - ", expand=True)</code>	<code>g13/demo.py</code>
pandas	series, summing	<code>reg_pop = by_reg['pop'].sum()/1e6</code>	<code>g12/demo.py</code>
pandas	series, unstacking	<code>tot_wide = tot_amt.unstack('PGI')</code>	<code>g17/demo.py</code>
pandas	series, using isin()	<code>fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID'])</code>	<code>g15/demo.py</code>
pandas	series, using value_counts()	<code>print('\nOuter:\n', join_o['_merge'].value_counts(), s. . .</code>	<code>g15/demo.py</code>

Module	Description	Example	Script
requests	adding headers	<code>response = requests.get(api,payload,headers=headers)</code>	<code>g30/demo.py</code>
requests	calling the <code>get()</code> method	<code>response = requests.get(api,payload)</code>	<code>g19/demo.py</code>
requests	checking the URL	<code>print('url:', response.url)</code>	<code>g19/demo.py</code>
requests	checking the response text	<code>print(response.text)</code>	<code>g19/demo.py</code>
requests	checking the status code	<code>print('status:', response.status_code)</code>	<code>g19/demo.py</code>
requests	decoding a JSON response	<code>rows = response.json()</code>	<code>g19/demo.py</code>
requests	geocoding via nominatim	<code>api = "https://nominatim.openstreetmap.org/search"</code>	<code>g30/demo.py</code>
requests	getting a web page	<code>response = requests.get(base_url,payload)</code>	<code>g31/demo.py</code>
requests	importing the module	<code>import requests</code>	<code>g19/demo.py</code>
scipy	calling newton's method	<code>cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y. .</code>	<code>g08/demo.py</code>
scipy	importing the module	<code>import scipy.optimize as opt</code>	<code>g08/demo.py</code>
seaborn	adding a title to a grid object	<code>jg.fig.suptitle('Distribution of Hourly Load')</code>	<code>g18/demo.py</code>
seaborn	barplot	<code>hue='month',palette='deep',ax=ax1)</code>	<code>g18/demo.py</code>
seaborn	basic violin plot	<code>sns.violinplot(data=janjul,x="month",y="usage")</code>	<code>g18/demo.py</code>
seaborn	boxenplot	<code>sns.boxenplot(data=janjul,x="month",y="usage")</code>	<code>g18/demo.py</code>
seaborn	calling <code>tight_layout</code> on a grid object	<code>jg.fig.tight_layout()</code>	<code>g18/demo.py</code>
seaborn	drawing a heatmapped grid	<code>sns.heatmap(means,annot=True,fmt="0f",cmap='Spectral',. . .</code>	<code>g21/demo.py</code>
seaborn	importing the module	<code>import seaborn as sns</code>	<code>g18/demo.py</code>
seaborn	joint distribution hex plot	<code>jg = sns.jointplot(data=bymo,x=1,y=7,kind='hex')</code>	<code>g18/demo.py</code>
seaborn	line plot	<code>sns.lineplot(data=long_form,x='hour',y='value',hue='mon. . .</code>	<code>g18/demo.py</code>
seaborn	setting axis titles on a grid object	<code>jg.set_axis_labels('January','July')</code>	<code>g18/demo.py</code>
seaborn	setting the theme	<code>sns.set_theme(style="white")</code>	<code>g18/demo.py</code>
seaborn	split violin plot	<code>hue="month",palette='deep',split=True)</code>	<code>g18/demo.py</code>
sql	appending to a table via pandas	<code>n = df.to_sql('enrollment',con,if_exists='append',index. . .</code>	<code>g28/demo.py</code>
sql	connecting to a SQLite database	<code>con = sqlite3.connect(demo_name)</code>	<code>g28/demo.py</code>
sql	create table with primary key	<code>cur = con.execute(create_table)</code>	<code>g28/demo.py</code>
sql	creating a table with a unique constraint	<code>cur = con.executescript(create_enrollment)</code>	<code>g28/demo.py</code>
sql	creating a view by joining tables	<code>cur = con.executescript(create_summary)</code>	<code>g28/demo.py</code>
sql	dropping a table	<code>cur = con.execute("DROP TABLE IF EXISTS courses;")</code>	<code>g28/demo.py</code>
sql	executing a SQL script	<code>cur = con.executescript(sql_cmds)</code>	<code>g28/demo.py</code>
sql	grouping and counting records	<code>cur = con.execute(count_recs)</code>	<code>g28/demo.py</code>
sql	handling column names with spaces	<code>data = pd.read_sql(ny_gen,con)</code>	<code>g28/demo.py</code>
sql	inserting a single record	<code>cur = con.execute(insert_one)</code>	<code>g28/demo.py</code>
sql	inserting multiple rows via <code>executemany</code>	<code>cur = con.executemany("INSERT INTO courses VALUES (?,?. . .</code>	<code>g28/demo.py</code>

Module	Description	Example	Script
sql	obtaining a list of tables	<code>cur = con.execute("SELECT name,sql FROM sqlite_master;". . .</code>	<code>g28/demo.py</code>
sql	reading a table via pandas	<code>summary = pd.read_sql("SELECT * FROM summary",con)</code>	<code>g28/demo.py</code>
sql	retrieving column names from cursor	<code>cur_info = cur.description</code>	<code>g28/demo.py</code>
sql	retrieving count of rows affected	<code>print("\nRows affected",cur.rowcount)</code>	<code>g28/demo.py</code>
sql	retrieving rows via fetchall	<code>rows = cur.fetchall()</code>	<code>g28/demo.py</code>
sql	select with order by clause	<code>cur = con.execute("SELECT * FROM courses ORDER BY pref. . .</code>	<code>g28/demo.py</code>
sql	selecting data using like	<code>cur = con.execute(select_cmd)</code>	<code>g28/demo.py</code>
sql	selecting using the IN clause	<code>cur = con.execute(select_some)</code>	<code>g28/demo.py</code>
sql	simple select of all columns	<code>cur = con.execute("SELECT * FROM courses;")</code>	<code>g28/demo.py</code>
sql	starting a with block	<code>with con:</code>	<code>g28/demo.py</code>
sql	updating fields for selected records	<code>cur = con.execute(update_cmd)</code>	<code>g28/demo.py</code>
sql	using DISTINCT	<code>cur = con.execute("SELECT DISTINCT year FROM races;")</code>	<code>g31/demo.py</code>
sql	using fetchone()	<code>row = cur.fetchone()</code>	<code>g31/demo.py</code>
sql	using the sum function	<code>cur = con.execute(count_cmd)</code>	<code>g28/demo.py</code>
stringio	accessing a string as a file	<code>tables = pd.read_html(StringIO(response.text))</code>	<code>g31/demo.py</code>
stringio	importing the function	<code>from io import StringIO</code>	<code>g31/demo.py</code>
sys	exiting a script	<code>sys.exit()</code>	<code>g28/demo.py</code>
sys	loading the module	<code>import sys</code>	<code>g28/demo.py</code>
zipfile	importing the module	<code>import zipfile</code>	<code>g16/demo.py</code>
zipfile	opening a file in an archive	<code>fh1 = archive.open('generators-oswego.csv')</code>	<code>g16/demo.py</code>
zipfile	opening an archive	<code>archive = zipfile.ZipFile('generators.zip')</code>	<code>g16/demo.py</code>
zipfile	reading the list of files	<code>print(archive.namelist())</code>	<code>g16/demo.py</code>