

Module	Description	Example	Script
collections	defaultdict, creating for lists	by_zone = defaultdict(list)	g10/demo.py
collections	defaultdict, importing	from collections import defaultdict	g10/demo.py
core	dictionary, adding a new entry	co['po'] = 'CO'	g05/demo.py
core	dictionary, checking for existing key	if fips in name_by_fips:	g09/demo.py
core	dictionary, creating	co = {'name':'Colorado', 'capital':'Denver'}	g05/demo.py
core	dictionary, deleting an entry	del name_by_fips["00"]	g09/demo.py
core	dictionary, iterating over keys	for fips in name_by_fips.keys():	g09/demo.py
core	dictionary, iterating over values	for rec in name_by_fips.values():	g09/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	dictionary, sorting keys	for tz in sorted( by_zone.keys() ):	g10/demo.py
core	f-string, using a formatting string	print( f"PV of {payment} with T={year} and r={r} is \${pv}" )	g07/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(), sep=" " )	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	function, calling	d1_ssqr = sumsq(d1)	g06/demo.py
core	function, calling with an optional argument	sample_function( 100, 10, r=0.07 )	g07/demo.py
core	function, defining	def sumsq(values):	g06/demo.py
core	function, defining with optional argument	def sample_function(payment,year,r=0.05):	g07/demo.py
core	function, returning a result	return values	g06/demo.py
core	if statement, testing for equality	if fips == "36":	g09/demo.py
core	list, appending an element	a_list.append("four")	g03/demo.py
core	list, create via comprehension	cubes = [n**3 for n in a_list]	g04/demo.py
core	list, creating	a_list = ["zero","one","two","three"]	g03/demo.py

Module	Description	Example	Script
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>tot_inc = sum(incomes)</code>	g08/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	string, concatenating	<code>name = s1+" "+s2+" "+s3</code>	g02/demo.py
core	string, converting to an int	<code>values.append( int(line) )</code>	g06/demo.py
core	string, converting to title case	<code>name = codes[key].title()</code>	g11/demo.py
core	string, creating	<code>filename = "demo.txt"</code>	g02/demo.py
core	string, including a newline character	<code>fh.write(name+"!\n")</code>	g02/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>this_tuple = (med_density,state)</code>	g10/demo.py
core	tuple, creating via split	<code>(last,first) = name.split(',')</code>	g11/demo.py
core	tuple, looping over	<code>for (den,state) in sorted(by_density):</code>	g10/demo.py
core	tuple, sorting	<code>for key in sorted(codes):</code>	g11/demo.py
core	tuple, testing equality of	<code>if key == (29,'VA'):</code>	g11/demo.py
csv	opening a file for use with DictWriter	<code>fh = open(outfile,'w',newline="")</code>	g09/demo.py
csv	setting up a DictReader object	<code>reader = csv.DictReader(fh)</code>	g08/demo.py
csv	setting up a DictWriter object	<code>writer = csv.DictWriter(fh,fields)</code>	g09/demo.py
csv	using DictReader with a list	<code>reader = csv.DictReader(lines)</code>	g10/demo.py
csv	writing a header with DictWriter	<code>writer.writeheader()</code>	g09/demo.py
csv	writing a record with DictWriter	<code>writer.writerow(name_rec)</code>	g09/demo.py

Module	Description	Example	Script
io	converting a byte stream to characters	<code>inp_handle = io.TextIOWrapper(inp_byte)</code>	g11/demo.py
json	importing the module	<code>import json</code>	g05/demo.py
json	using to print an object nicely	<code>print( json.dumps(list1,indent=4) )</code>	g05/demo.py
matplotlib	axes, setting a title	<code>ax1.set_title('Population')</code>	g13/demo.py
matplotlib	axis, labeling X axis	<code>ax1.set_xlabel('Millions')</code>	g13/demo.py
matplotlib	figure, saving	<code>fig1.savefig('figure.png')</code>	g13/demo.py
matplotlib	figure, tuning the layout	<code>fig1.tight_layout()</code>	g13/demo.py
matplotlib	importing pyplot	<code>import matplotlib.pyplot as plt</code>	g13/demo.py
matplotlib	using subplots to set up a figure	<code>fig1, ax1 = plt.subplots()</code>	g13/demo.py
numpy	computing a median	<code>med_density = round( np.median(this_list), 2 )</code>	g10/demo.py
numpy	importing	<code>import numpy as np</code>	g10/demo.py
pandas	columns, dividing with explicit alignment	<code>normed2 = 100*states.div(pa_row,axis='columns')</code>	g12/demo.py
pandas	columns, listing names	<code>print( '\nColumns:', list(states.columns) )</code>	g12/demo.py
pandas	columns, renaming	<code>county = county.rename(columns={'B01001_001E':'pop'})</code>	g14/demo.py
pandas	columns, retrieving one by name	<code>pop = states['pop']</code>	g12/demo.py
pandas	columns, retrieving several by name	<code>print( pop[some_states]/1e6 )</code>	g12/demo.py
pandas	dataframe, dropping duplicates	<code>flood = flood.drop_duplicates( subset='TAX_ID' )</code>	g15/demo.py
pandas	dataframe, finding duplicate records	<code>dups = parcels.duplicated( subset='TAX_ID', keep=False )</code>	g15/demo.py
pandas	dataframe, inner 1:1 merge	<code>join_i = parcels.merge(flood,</code>	g15/demo.py
pandas	dataframe, left 1:1 merge	<code>join_l = parcels.merge(flood,</code>	g15/demo.py
pandas	dataframe, outer 1:1 merge	<code>join_o = parcels.merge(flood,</code>	g15/demo.py
pandas	dataframe, right 1:1 merge	<code>join_r = parcels.merge(flood,</code>	g15/demo.py
pandas	dataframe, selecting rows via boolean	<code>dup_rec = flood[ dups ]</code>	g15/demo.py
pandas	dataframe, sorting by a column	<code>county = county.sort_values('pop')</code>	g14/demo.py
pandas	datetime, building via to_datetime()	<code>date = pd.to_datetime(recs['ts'])</code>	g15/demo.py
pandas	datetime, extracting day attribute	<code>recs['day'] = date.dt.day</code>	g15/demo.py
pandas	datetime, extracting hour attribute	<code>recs['hour'] = date.dt.hour</code>	g15/demo.py
pandas	displaying all rows	<code>pd.set_option('display.max_rows', None)</code>	g12/demo.py
pandas	groupby, summing a variable	<code>state = county.groupby('state')['pop'].sum()</code>	g14/demo.py
pandas	groupby, using with one grouping variable	<code>by_reg = state_data.groupby('Region')</code>	g13/demo.py
pandas	importing the module	<code>import pandas as pd</code>	g12/demo.py
pandas	index, creating with two-levels	<code>county = county.set_index(['state','county'])</code>	g14/demo.py

Module	Description	Example	Script
pandas	index, listing names	<code>print( '\nIndex (rows):', list(states.index) )</code>	<code>g12/demo.py</code>
pandas	index, renaming values	<code>div_pop = div_pop.rename(index=div_names)</code>	<code>g13/demo.py</code>
pandas	index, retrieving a row by name	<code>pa_row = states.loc['Pennsylvania']</code>	<code>g12/demo.py</code>
pandas	index, retrieving first rows by location	<code>print( low_to_high.iloc[ 0:10 ] )</code>	<code>g12/demo.py</code>
pandas	index, retrieving last rows by location	<code>print( low_to_high.iloc[ -5: ] )</code>	<code>g12/demo.py</code>
pandas	index, setting to a column	<code>new_states = states.set_index('name')</code>	<code>g12/demo.py</code>
pandas	index, setting to a column in place	<code>states.set_index('name',inplace=True)</code>	<code>g12/demo.py</code>
pandas	plotting, bar plot	<code>reg_pop.plot.bar(ax=ax1)</code>	<code>g13/demo.py</code>
pandas	plotting, horizontal bar plot	<code>div_pop.plot.barh(ax=ax1)</code>	<code>g13/demo.py</code>
pandas	reading, csv data	<code>states = pd.read_csv('state-data.csv')</code>	<code>g12/demo.py</code>
pandas	reading, csv using dtype	<code>geocodes = pd.read_csv('state-geocodes.csv',dtype=str)</code>	<code>g13/demo.py</code>
pandas	series, retrieving an element	<code>print( "\nFlorida's population:", pop['Florida']/1e6 )</code>	<code>g12/demo.py</code>
pandas	series, sorting by value	<code>low_to_high = normed['med_pers_inc'].sort_values()</code>	<code>g12/demo.py</code>
pandas	series, summing	<code>reg_pop = by_reg['pop'].sum())/1e6</code>	<code>g13/demo.py</code>
pandas	series, using <code>isin()</code>	<code>fixed = flood['TAX_ID'].isin( dup_rec['TAX_ID'] )</code>	<code>g15/demo.py</code>
pandas	series, using <code>value_counts()</code>	<code>print( '\nOuter:\n', join_o['_merge'].value_counts(), sep=" " )</code>	<code>g15/demo.py</code>
pandas	using <code>qcut</code> to create deciles	<code>dec = pd.qcut( county['pop'], 10, labels=range(1,11) )</code>	<code>g14/demo.py</code>
pandas	using <code>xs</code> to select from an index	<code>print( county.xs('04',level='state') )</code>	<code>g14/demo.py</code>
scipy	calling newton's method	<code>cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y])</code>	<code>g07/demo.py</code>
scipy	importing the module	<code>import scipy.optimize as opt</code>	<code>g07/demo.py</code>
zipfile	creating a <code>ZipFile</code> object	<code>zip_object = zipfile.ZipFile(zipname)</code>	<code>g11/demo.py</code>
zipfile	importing module	<code>import zipfile</code>	<code>g11/demo.py</code>
zipfile	opening a file in a zip in bytes mode	<code>inp_byte = zip_object.open(csvname)</code>	<code>g11/demo.py</code>