Module	Description	Example	Script
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, 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, using a formatting string	print(f"PV of {payment} with T={year} and r={r} is ${p. . .	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, 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_ssq = 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	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
core	list, determining length	n = len(b_list)	g03/demo.py
core	list, extending with another list	a_list.extend(a_more)	g03/demo.py
core	list, generating a sequence	b_list = range(1,6)	g04/demo.py
core	list, joining with spaces	a_string = " ".join(a_list)	g03/demo.py
core	list, selecting an element	print(a_list[0])	g03/demo.py
core	list, selecting elements 0 to 3	print(a_list[:4])	g03/demo.py
core	list, selecting elements 1 to 2	print(a_list[1:3])	g03/demo.py
core	list, selecting elements 1 to the end	print(a_list[1:])	g03/demo.py
core	list, selecting last 3 elements	print(a_list[-3:])	g03/demo.py
core	list, selecting the last element	print(a_list[-1])	g03/demo.py
core	list, sorting	c_sort = sorted(b_list)	g03/demo.py

As of exercise g13

Module	Description	Example	Script
core	list, summing	tot_inc = sum(incomes)	g08/demo.py
core	math, raising a number to a power	a_cubes.append(n**3)	g04/demo.py
core	math, rounding a number	rounded = round(ratio,2)	g05/demo.py
core	string, concatenating	name = s1+" "+s2+" "+s3	g02/demo.py
core	string, converting to an int	values.append(int(line))	g06/demo.py
core	string, converting to title case	name = codes[key].title()	g12/demo.py
core	string, creating	filename = "demo.txt"	g02/demo.py
core	string, including a newline character	fh.write(name+"!\n")	g02/demo.py
core	string, splitting on a comma	parts = line.split(',')	g05/demo.py
core	string, splitting on whitespace	b_list = b_string.split()	g03/demo.py
core	string, stripping blank space	clean = [item.strip() for item in parts]	g05/demo.py
core	tuple, creating via split	(last,first) = name.split(',')	g12/demo.py
core	tuple, sorting	for key in sorted(codes):	g12/demo.py
core	tuple, testing equality of	if key == (29,'VA'):	g12/demo.py
csv	setting up a DictReader object	reader = csv.DictReader(fh)	g08/demo.py
io	converting a byte stream to characters	inp_handle = io.TextIOWrapper(inp_byte)	g12/demo.py
json	importing the module	import json	g05/demo.py
json	using to print an object nicely	print(json.dumps(list1,indent=4))	g05/demo.py
matplotlib	axes, setting a title	ax1.set_title('Population')	g11/demo.py
matplotlib	axis, labeling X axis	ax1.set_xlabel('Millions')	g11/demo.py
matplotlib	figure, saving	fig1.savefig('figure.png')	g11/demo.py
matplotlib	figure, tuning the layout	fig1.tight_layout()	g11/demo.py
matplotlib	importing pyplot	import matplotlib.pyplot as plt	g11/demo.py
matplotlib	using subplots to set up a figure	fig1, ax1 = plt.subplots()	g11/demo.py
pandas	columns, converting to float	pop_data['pop'] = pop_data['pop'].astype(float)	g13/demo.py
pandas	columns, dividing with explicit alignment	normed2 = 100*states.div(pa_row,axis='columns')	g09/demo.py
pandas	columns, listing names	print('\nColumns:', list(states.columns))	g09/demo.py
pandas	columns, renaming	county = county.rename(columns={'B01001_001E':'pop'})	g10/demo.py
pandas	columns, retrieving one by name	pop = states['pop']	g09/demo.py

As of exercise g13

Module	Description	Example	Script
pandas	columns, retrieving several by name	print(pop[some_states]/1e6)	g09/demo.py
pandas	dataframe, head method	print('\n', indexed.head())	g13/demo.py
pandas	dataframe, inner join	merged = name_data.merge(pop_data,left_on="State",right. . .	g13/demo.py
pandas	dataframe, sorting by a column	county = county.sort_values('pop')	g10/demo.py
pandas	dataframe, sorting by column values	sort_percent = indexed.sort_values("percent")	g13/demo.py
pandas	dataframe, writing to a CSV file	indexed.to_csv(outfile)	g13/demo.py
pandas	displaying all rows	pd.set_option('display.max_rows', None)	g09/demo.py
pandas	groupby, summing a variable	state = county.groupby('state')['pop'].sum()	g10/demo.py
pandas	groupby, summing a variable	div_pop = group_by_div['pop'].sum()	g13/demo.py
pandas	groupby, using with one grouping variable	by_reg = state_data.groupby('Region')	g11/demo.py
pandas	groupby, using with one grouping variable	group_by_div = indexed.groupby('Division')	g13/demo.py
pandas	importing the module	import pandas as pd	g09/demo.py
pandas	index, creating with two-levels	county = county.set_index(['state','county'])	g10/demo.py
pandas	index, listing names	print('\nIndex (rows):', list(states.index))	g09/demo.py
pandas	index, renaming values	div_pop = div_pop.rename(index=div_names)	g11/demo.py
pandas	index, retrieving a row by name	pa_row = states.loc['Pennsylvania']	g09/demo.py
pandas	index, retrieving first rows by location	print(low_to_high.iloc[0:10])	g09/demo.py
pandas	index, retrieving last rows by location	print(low_to_high.iloc[-5:])	g09/demo.py
pandas	index, setting to a column	new_states = states.set_index('name')	g09/demo.py
pandas	index, setting to a column in place	states.set_index('name',inplace=True)	g09/demo.py
pandas	plotting, bar plot	reg_pop.plot.bar(ax=ax1)	g11/demo.py
pandas	plotting, horizontal bar plot	div_pop.plot.barh(ax=ax1)	g11/demo.py
pandas	reading, csv data	states = pd.read_csv('state-data.csv')	g09/demo.py
pandas	reading, csv using dtype	geocodes = pd.read_csv('state-geocodes.csv',dtype=str)	g11/demo.py
pandas	reading, with dtype=str	name_data = pd.read_csv('state_name.csv',dtype=str)	g13/demo.py
pandas	series, automatic alignment by index	indexed['percent'] = 100*indexed['pop']/div_pop	g13/demo.py
pandas	series, retrieving an element	print("\nFlorida's population:", pop['Florida']/1e6)	g09/demo.py
pandas	series, sorting by value	low_to_high = normed['med_pers_inc'].sort_values()	g09/demo.py
pandas	series, summing	reg_pop = by_reg['pop'].sum()/1e6	g11/demo.py

As of exercise g13

Module	Description	Example	Script
pandas	using qcut to create deciles	dec = pd.qcut(county['pop'], 10, labels=range(1,11))	g10/demo.py
pandas	using xs to select from an index	print(county.xs('04',level='state'))	g10/demo.py
scipy	calling newton's method	cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y. . .	g07/demo.py
scipy	importing the module	import scipy.optimize as opt	g07/demo.py
zipfile	creating a ZipFile object	zip_object = zipfile.ZipFile(zipname)	g12/demo.py
zipfile	importing module	import zipfile	g12/demo.py
zipfile	opening a file in a zip in bytes mode	inp_byte = zip_object.open(csvname)	g12/demo.py

As of exercise g13