| 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 | <pre>print("It was written during",year,file=fh)</pre> | 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 = $[n**3 \text{ for n in a_list}]$ | 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 |

| 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, 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 | type, obtaining for a variable | <pre>print('\nraw_states is a DataFrame object:', type(raw</pre> | g09/demo.py |
| CSV | setting up a DictReader object | ${\sf reader} = {\sf csv.DictReader(fh)}$ | g08/demo.py |
| json | importing the module | import json | g05/demo.py |
| json | using to print an object nicely | <pre>print(json.dumps(list1,indent=4))</pre> | g05/demo.py |
| pandas | columns, dividing with explicit alignment | $normed2 = 100*states.div(pa_row,axis='columns')$ | g09/demo.py |
| pandas | columns, listing names | <pre>print('\nColumns:', list(raw_states.columns))</pre> | g09/demo.py |
| pandas | columns, retrieving one by name | pop = states['pop'] | g09/demo.py |
| pandas | columns, retrieving several by name | <pre>print(pop[some_states]/1e6)</pre> | g09/demo.py |
| pandas | dataframe, selecting rows by list indexing | <pre>print(low_to_high[-5:])</pre> | g09/demo.py |
| pandas | general, displaying all rows | pd.set_option('display.max_rows', None) | g09/demo.py |
| pandas | general, importing the module | import pandas as pd | g09/demo.py |
| pandas | index, listing names | <pre>print('\nIndex (rows):', list(raw_states.index))</pre> | g09/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 | states = raw_states.set_index('name') | g09/demo.py |
| pandas | reading, csv data | raw_states = pd.read_csv('state-data.csv') | g09/demo.py |

| Module | Description | Example | Script |
|------------------|--|--|----------------------------|
| pandas pandas | series, retrieving an element series, sorting by value | | g09/demo.py g09/demo.py |
| scipy scipy | calling newton's method importing the module | $cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y import scipy.optimize as opt$ | g07/demo.py g07/demo.py |