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 ${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, print without adding spaces	<pre>print('\nOuter:\n', join_o['_merge'].value_counts(), s</pre>	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	$d1_ssq = sumsq(d1)$	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	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 = $[n^{**3} \text{ for n in a_list}]$	g04/demo.py
core	list, creating	a_list = ["zero", "one", "two", "three"]	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, selecting an element print(a_list(1)) g03/demo.py core list, selecting elements 0 to 3 print(a_list(1)) g03/demo.py core list, selecting elements 1 to 2 print(a_list(1)) g03/demo.py core list, selecting elements 1 to 2 print(a_list(1)) g03/demo.py core list, selecting elements 1 to the end print(a_list(1)) g03/demo.py core list, selecting lements 1 to the end print(a_list(1:3)) g03/demo.py core list, selecting lements 1 to the end print(a_list(1:3)) g03/demo.py core list, selecting lements 1 to to the end print(a_list(1:3)) g03/demo.py core list, selecting the last element print(a_list(1:3)) g03/demo.py core list, summing tot_inc = sum(incomes) g03/demo.py core list, summing tot_inc = sum(incomes) g03/demo.py core math, raising a number to a power rounded = round(ratio,2) g05/demo.py core string, concettanting name = s1+" "+s2+" +s3 g02/demo.py core string, converting to an int values append(int(line)) g05/demo.py core string, converting to title case name = codes[key].title() g11/demo.py core string, creating flename = "demo.txt" g02/demo.py core string, splitting on a comma parts = line split('.') core string, splitting on a romma parts = line split('.') core string, stripping blank space clean = [item.strip() for item in parts] g05/demo.py core tuple, creating this_tuple = (med_density.state) g10/demo.py core tuple, creating for the splitting of role, state) is parts = line split('.') core tuple, creating is split (last, first) = name.split('.') core tuple, creating for the splitting of role, state) is parts = line split('.') core tuple, creating this_tuple = (med_density.state) core tuple, creating is split (last, first) = name.split('.') core tuple, lesting equality of if key == (29, VA)': core setting up a DictReader object reader = csv. DictReader(fin) csv setting up a DictReader object reader = csv. DictReader(fine) csv using DictReader with a list reader	Module	Description	Example	Script
core list, extending with another list a_list_extend(a_more) g03/demo.py core list, joining with spaces a_string = "ijoin(a_list) g04/demo.py core list, selecting an element print(a_list[0]) g03/demo.py core list, selecting elements 0 to 3 print(a_list[1]) g03/demo.py core list, selecting elements 1 to 2 print(a_list[1]) 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[1]) g03/demo.py core list, selecting the last element print(a_list[1]) g03/demo.py core list, selecting the last element print(a_list[1]) g03/demo.py core list, summing tot_inc = sum(incomes) g03/demo.py core math, raising a number to a power a_cubes.append(n**3) g04/demo.py core math, raising a number to a power a_cubes.append(int(line)) g05/demo.py core string, converting to a power a_cubes.append(int(line)) g06/demo.py core	core	list, determining length	$n = len(b_list)$	g03/demo.py
core list, joining with spaces a_string = "".join(a_list) g03/demo.py core list, selecting an element print(a_list[l)) g03/demo.py core list, selecting elements 0 to 3 print(a_list[l,3]) g03/demo.py core list, selecting elements 1 to 2 print(a_list[1,3]) g03/demo.py core list, selecting lements 1 to the end print(a_list[1,3]) g03/demo.py core list, selecting the last element print(a_list[-1]) g03/demo.py core string, sun unmer a_cubes.append(n**3) g04/demo.py core <t< td=""><td>core</td><td>list, extending with another list</td><td></td><td>g03/demo.py</td></t<>	core	list, extending with another list		g03/demo.py
core list, selecting an element print(a_list[0]) g03/demo_py g05/demo_py g05/d	core	list, generating a sequence	$b_{list} = range(1,6)$	g04/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[1:3]) g03/demo.py core list, selecting lements 1 to the end print(a_list[1:3]) g03/demo.py core list, selecting the last element print(a_list[1:3]) g03/demo.py core list, selecting the last element print(a_list[1:3]) g03/demo.py core list, sorting c_sort = sorted(b_list) g03/demo.py core list, summing tot_inc = sum(incomes) g08/demo.py core list, summing tot_inc = sum(incomes) g08/demo.py core math, raising a number to a power 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() g11/demo.py core string, coreuting filename = "demo.tx" g02/demo.py core string, splitting on a comma parts = line.split(") g02/demo.py core string, splitting on a comma parts = line.split(") g03/demo.py core string, splitting on a comma parts = line.split(") g03/demo.py core string, splitting on shritespace b_list = b_string.split() g03/demo.py core tuple, creating saplit (last,first) = name.split(") g11/demo.py core tuple, creating value in this_tuple = (med_density,state) g10/demo.py core tuple, creating value in sorted(by_density): g11/demo.py core tuple, sorting to use with DictWriter fh = open(outfile, wn,newline="") g09/demo.py core setting up a DictReader object reader = csv.DictReader(fih) g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py g09/demo.py cove writing a header with DictWriter writerwerteheader(") g09/demo.py g09/d	core	list, joining with spaces	a_string = " ".join(a_list)	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[1:1]) g03/demo.py core list, selecting last 3 elements print(a_list[1:3]) g03/demo.py core list, selecting the last element print(a_list[1:1]) g03/demo.py core list, sorting c_sort sorted(b_list) g03/demo.py core list, summing to_inc = sum(incomes) g08/demo.py core list, summing to_inc = sum(incomes) g08/demo.py core math, raising a number to a power a_ubes.append(n**3) g04/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() g11/demo.py core string, including a newline character fl.write(name+"\n") g02/demo.py core string, splitting on a comma parts = line.split(',') g05/demo.py core string, splitting on a comma parts = line.split(',') g05/demo.py core string, stripping blank space clean = [item.strip() for item in parts] g05/demo.py core tuple, creating for this_tuple = (med_density,state) g10/demo.py core tuple, creating for key in sorted(by_density): g10/demo.py core tuple, creating for key in sorted(by_density): g10/demo.py core tuple, sorting for sey with DictWriter ff h = open(outfile, w',newline=") g09/demo.py core tuple, testing equality of if key == (29, VA'): g11/demo.py core setting up a DictReader with DictWriter ff h = open(outfile, w',newline=") g09/demo.py core setting up a DictReader with DictWriter ff h = open(outfile, w',newline=") g09/demo.py core setting up a DictReader with DictWriter ff h = open(outfile, w',newline=") g09/demo.py core setting up a DictReader with DictWriter writer csv.DictReader(lines) g09/demo.py core writing a header with DictWriter writer.writeheader()	core	list, selecting an element	print(a_list[0])	g03/demo.py
core list, selecting elements 1 to the end print(a_list[1-]) g03/demo.py core list, selecting that selements print(a_list[1-]) 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 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, converting to an int values.append(int(line)) g05/demo.py core string, converting to title case name = codes[key].title() g11/demo.py core string, including a newline character fh.write(name+"\"\") 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() g05/demo.py core string, stripping blank space clean = [item.strip() for item in parts] g05/demo.py core tuple, creating this_tuple = (med_density,state) tuple, creating to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core tuple, sorting to split (last,first) = name.split(',') g11/demo.py core setting up a DictReader object reader = csv.DictReader(fih) g08/demo.py csv setting up a DictReader object reader = csv.DictReader(fih) g08/demo.py csv setting up a DictReader object writer = csv.DictReader(fines) g09/demo.py csv writing a header with DictWriter writer.writeheader()	core	list, selecting elements 0 to 3	print(a_list[:4])	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 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() g11/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 this_tuple = (med_density,state) g11/demo.py core tuple, creating this_tuple = (med_density,state) g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g11/demo.py core tuple, testing equality of if key == (29, VA'): g11/demo.py core string in a file for use with DictWriter fh = open(outfile,'w',newline=") g09/demo.py core string in a DictReader object reader = csv.DictReader(flh) g09/demo.py csv setting up a DictReader with a list reader = csv.DictReader(lines) g09/demo.py csv writing a header with a list reader = csv.DictReader(lines) writer.writeheader()	core	list, selecting elements 1 to 2	print(a_list[1:3])	g03/demo.py
core list, selecting the last element print(a_list[-1]) g03/demo.py core list, surming c_sort = sorted(b_list) g03/demo.py core list, summing tot_inc = sum(incomes) g08/demo.py core math, raising a number to a power rounded = round(ratio,2) g05/demo.py 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() g11/demo.py core string, including a newline character fh.write(name+"\n") g02/demo.py core string, splitting on a comma parts = line split(') g03/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 this_tuple = (med_density.state) g10/demo.py core tuple, creating for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py core setting up a DictReader object reader = csv.DictReader(flines) g10/demo.py core wing DictReader with DictWriter writer.eader(') g09/demo.py core writing a header with DictWriter writer.eader(') g09/demo	core	list, selecting elements 1 to the end	print(a_list[1:])	g03/demo.py
core list, selecting the last element print(a_list[-1]) g03/demo.py core list, surming c_sort = sorted(b_list) g03/demo.py core list, summing tot_inc = sum(incomes) g08/demo.py core math, raising a number to a power rounded = round(ratio,2) g05/demo.py 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() g11/demo.py core string, including a newline character fh.write(name+"\n") g02/demo.py core string, splitting on a comma parts = line split(') g03/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 this_tuple = (med_density.state) g10/demo.py core tuple, creating for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py core setting up a DictReader object reader = csv.DictReader(flines) g10/demo.py core wing DictReader with DictWriter writer.eader(') g09/demo.py core writing a header with DictWriter writer.eader(') g09/demo	core	list, selecting last 3 elements	print(a_list[-3:])	g03/demo.py
core list, summing tot_inc = sum(incomes) g08/demo.py core math, raising a number to a power 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() core string, including a newline character fl.nume+"\n") core string, splitting on a comma parts = line.split(',') core string, splitting on whitespace b_list = b_string.split() core string, stripping blank space clean = [item.strip() for item in parts] core tuple, creating to this_tuple = (med_density,state) core tuple, creating via split (last,first) = name.split(',') g10/demo.py core tuple, looping over for (den,state) in sorted(by_density): core tuple, sorting core tuple, testing equality of if key == (29,'VA'): csv opening a file for use with DictWriter flh = open(outfile,'w',newline=") csv setting up a DictReader object reader = csv.DictReader(flh) csv setting up a DictReader with DictWriter writer.writeheader() csv writing a header with DictWriter writer.writeheader()	core	list, selecting the last element		g03/demo.py
core math, raising a number to a power rounded = round(ratio,2) g05/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() g11/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 this_tuple = (med_density.state) g10/demo.py core tuple, creating for key in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g09/demo.py core writing a header with DictWriter writer-writeheader() g09/demo.py core writing a header with DictWriter writer-writeheader()	core	list, sorting	$c_sort = sorted(b_list)$	g03/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() g11/demo.py core string, including a newline character fh.write(name+"!\n") g02/demo.py core string, splitting on a comma parts = line.split(',') g02/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 this_tuple = (med_density,state) g10/demo.py core tuple, creating this_tuple = (med_density,state) g11/demo.py core tuple, sorting for key in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core string up a DictReader object reader = csv.DictReader(fh) g09/demo.py core string up a DictReader object writer = csv.DictReader(fh, fields) g10/demo.py core writing a header with DictWriter writer writer.writeheader() g09/demo.py core writing a header with DictWriter writer.writeheader()	core	list, summing	$tot_inc = sum(incomes)$	g08/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() g11/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 this_tuple = (med_density.state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictReader object writer = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader()	core	math, raising a number to a power	a_cubes.append(n**3)	g04/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() g11/demo.py core string, creating filename = "demo.tx" 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 this_tuple = (med_density,state) g10/demo.py core tuple, looping over for (den,state) in sorted(by_density): g11/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29,'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictReader with a list reader = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writen.writeheader() g09/demo.py csv writing a header with DictWriter writen.writeheader()	core	math, rounding a number	rounded = round(ratio, 2)	g05/demo.py
core string, converting to title case name = codes[key].title() g11/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 this_tuple = (med_density,state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29,'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g09/demo.py csv setting up a DictReader object writer = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writer.writeheader() g09/demo.py csv writing a header with DictWriter writer.writeheader() g09/demo.py csv writing a header with DictWriter writer.writeheader()	core	string, concatenating	name = $s1+""+s2+""+s3$	g02/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 this_tuple = (med_density,state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, sorting for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29,'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictWriter object writer = csv.DictWriter(fh,fields) g09/demo.py csv using DictReader with a list reader = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader()	core	string, converting to an int	values.append(int(line))	g06/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 this_tuple = (med_density,state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictReader with a list reader = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader()	core	string, converting to title case	name = codes[key].title()	g11/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 this_tuple = (med_density.state) g10/demo.py core tuple, creating via split (last.first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core tuple, testing up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictReader object writer = csv.DictWriter(fh,fields) g09/demo.py csv using DictReader with a list reader = csv.DictReader(lines) g09/demo.py csv writing a header with DictWriter writeheader() g09/demo.py g09/demo.py g09/demo.py csv writing a header with DictWriter writeheader()	core	string, creating	filename = "demo.txt"	g02/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 this_tuple = (med_density,state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == $(29, \text{VA'})$: g09/demo.py core tuple, testing up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictReader with a list reader = csv.DictReader(lines) g10/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader()	core	string, including a newline character	fh.write(name+"!\n")	g02/demo.py
	core	string, splitting on a comma		g05/demo.py
core tuple, creating this_tuple = (med_density,state) g10/demo.py core tuple, creating via split (last,first) = name.split(',') g11/demo.py core tuple, looping over for (den,state) in sorted(by_density): g10/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, testing equality of if key == (29, 'VA'): g11/demo.py core tuple, testing equality of the open(outfile, 'w', newline=") g09/demo.py core setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictWriter object writer = csv.DictWriter(fh,fields) g09/demo.py csv using DictReader with a list reader = csv.DictReader(lines) g10/demo.py csv writing a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader()	core	string, splitting on whitespace		g03/demo.py
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	core	string, stripping blank space	clean = [item.strip() for item in parts]	g05/demo.py
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	core	tuple, creating	${\sf this_tuple} = ({\sf med_density}, {\sf state})$	g10/demo.py
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	core	tuple, creating via split	(last,first) = name.split(`,`)	g11/demo.py
core tuple, testing equality of if key $== (29, {}^{\circ}VA')$: g11/demo.py csv opening a file for use with DictWriter fh = open(outfile, ${}^{\circ}W'$, newline=") g09/demo.py csv setting up a DictReader object reader = csv.DictReader(fh) g08/demo.py csv setting up a DictWriter object writer = csv.DictWriter(fh, fields) g09/demo.py csv using DictReader with a list reader = csv.DictReader(lines) g10/demo.py csv writing a header with DictWriter writeheader() g09/demo.py	core	tuple, looping over	for (den,state) in sorted(by_density):	g10/demo.py
csv opening a file for use with DictWriter $fh = open(outfile, 'w', newline=")$ $g09/demo.py$ csv setting up a DictReader object reader = csv.DictReader(fh) $g08/demo.py$ csv setting up a DictWriter object writer = csv.DictWriter(fh, fields) $g09/demo.py$ csv using DictReader with a list reader = csv.DictReader(lines) $g10/demo.py$ csv writing a header with DictWriter writer.writeheader() $g09/demo.py$ $g09/demo.py$	core	tuple, sorting	for key in sorted(codes):	g11/demo.py
$\begin{array}{llllllllllllllllllllllllllllllllllll$	core	tuple, testing equality of	if key $==$ (29, 'VA'):	g11/demo.py
csv setting up a DictWriter object writer = csv.DictWriter(fh,fields) g09/demo.py csv using DictReader with a list reader = csv.DictReader(lines) g10/demo.py csv writing a header with DictWriter writerheader() g09/demo.py	CSV	opening a file for use with DictWriter	fh = open(outfile, 'w', newline=")	g09/demo.py
$\begin{array}{lll} \text{csv} & \text{using DictReader with a list} & \text{reader} = \text{csv.DictReader(lines)} & \text{g10/demo.py} \\ \text{csv} & \text{writing a header with DictWriter} & \text{writer.writeheader()} & \text{g09/demo.py} \\ \end{array}$	CSV	setting up a DictReader object	reader = csv.DictReader(fh)	g08/demo.py
csv writing a header with DictWriter writer.writeheader() g09/demo.py	CSV	setting up a DictWriter object	writer = csv.DictWriter(fh,fields)	g09/demo.py
· · · · · · · · · · · · · · · · · · ·	CSV	using DictReader with a list	reader = csv.DictReader(lines)	g10/demo.py
	CSV	writing a header with DictWriter	writer.writeheader()	g09/demo.py
	CSV	writing a record with DictWriter	writer.writerow(name_rec)	g09/demo.py

Module	Description	Example	Script
geopandas	drawing a map	sel.plot('NAME',cmap='Dark2',ax=ax1)	g22/demo.py
geopandas	extracting geometry from a geodataframe	wv_geo = wv['geometry']	g22/demo.py
geopandas	importing the module	import geopandas as gpd	g22/demo.py
geopandas	merging data onto a geodataframe	conus = conus.merge(trim,on='STATEFP',how='left',valida	g22/demo.py
geopandas	obtaining coordinates	print('Number of points:', len(wv_geo.exterior.coords)	g22/demo.py
geopandas	reading a shapefile	states = gpd.read_file("cb_2019_us_state_500k.zip")	g22/demo.py
geopandas	testing if rows touch a geometry	touches_wv = conus.touches(wv_geo)	g22/demo.py
geopandas	writing a layer to a geodatabase	conus.to_file("conus.gpkg",layer="states")	g22/demo.py
io	converting a byte stream to characters	$inp_handle = io.TextIOWrapper(inp_byte)$	g11/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
matplotlib	axes, setting a title	ax1.set_title('Population')	g13/demo.py
matplotlib	axis, labeling X axis	ax1.set_xlabel('Millions')	g13/demo.py
matplotlib	figure, saving	fig1.savefig('figure.png')	g13/demo.py
matplotlib	figure, tuning the layout	fig1.tight_layout()	g13/demo.py
matplotlib	importing pyplot	import matplotlib.pyplot as plt	g13/demo.py
matplotlib	setting a figure title	fig1.suptitle('Electric Power Plants in Onondaga and Os	g16/demo.py
matplotlib	setting the default resolution	plt.rcParams['figure.dpi'] = 300	g18/demo.py
matplotlib	using subplots to set up a figure	fig1, ax1 = plt.subplots()	g13/demo.py
numpy	computing a median	med_density = round(np.median(this_list), 2)	g10/demo.py
numpy	importing	import numpy as np	g10/demo.py
pandas	RE, replacing a digit or space	$unit_part = values.str.replace(`\d \s',",regex=True)$	g23/demo.py
pandas	RE, replacing a non-digit or space	$value_part = values.str.replace(`\backslash D \backslash s',",\mathsf{regex = True)}$	g23/demo.py
pandas	RE, replacing a non-word character	units = units.str.replace(`W',",regex=True)	g23/demo.py
pandas	columns, dividing with explicit alignment	$normed2 = 100*states.div(pa_row,axis='columns')$	g12/demo.py
pandas	columns, listing names	$print('\nColumns:', list(states.columns))$	g12/demo.py
pandas	columns, renaming	$county = county.rename(columns = \{`B01001_001E':'pop'\})$	g14/demo.py
pandas	columns, retrieving one by name	pop = states['pop']	g12/demo.py
pandas	columns, retrieving several by name	<pre>print(pop[some_states]/1e6)</pre>	g12/demo.py

Module	Description	Example	Script
pandas	dataframe, appending	gen_all = pd.concat([gen_oswego, gen_onondaga])	g16/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, finding duplicate records	$dups = parcels.duplicated(subset='TAX_ID', keep=False$	g15/demo.py
pandas	dataframe, inner 1:1 merge	$join_i = parcels.merge(flood,$	g15/demo.py
pandas	dataframe, left 1:1 merge	$join_l = parcels.merge(flood,$	g15/demo.py
pandas	dataframe, left m:1 merge	$both = gen_all.merge(plants,$	g16/demo.py
pandas	dataframe, making a copy	subset_copy = sample[keepvars].copy()	g17/demo.py
pandas	dataframe, outer 1:1 merge	$join_o = parcels.merge(flood,$	g15/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	$join _r = parcels.merge(flood,$	g15/demo.py
pandas	dataframe, saving in zipped pickle format	sample.to_pickle('sample_pkl.zip')	g17/demo.py
pandas	dataframe, selecting rows via boolean	$dup_rec = flood[dups]$	g15/demo.py
pandas	dataframe, selecting rows via query	$ngcc = both.query(``Technology == `Natural\;Gas\;Fired\;Com.\;\;.$	g16/demo.py
pandas	dataframe, sorting by a column	<pre>county = county.sort_values('pop')</pre>	g14/demo.py
pandas	dataframe, sorting by index	$summary = summary.sort_index(ascending = False)$	g16/demo.py
pandas	dataframe, unstacking an index level	bymo = bymo.unstack(`month')	g18/demo.py
pandas	datetime, building via to_datetime()	$date = pd.to_datetime(recs['ts'])$	g15/demo.py
pandas	datetime, building with a format	$ymd = pd.to_datetime(sample['TRANSACTION_DT'], format=$	g17/demo.py
pandas	datetime, extracting day attribute	recs['day'] = date.dt.day	g15/demo.py
pandas	datetime, extracting hour attribute	recs['hour'] = date.dt.hour	g15/demo.py
pandas	displaying all columns	pd.set_option('display.max_columns',None)	g17/demo.py
pandas	displaying all rows	pd.set_option('display.max_rows', None)	g12/demo.py
pandas	dropping rows using a list	$conus = states.drop(not_conus)$	g22/demo.py
pandas	groupby, counting records via size	summary['units'] = tech_by_kv.size()	g16/demo.py
pandas	groupby, summing a variable	state = county.groupby('state')['pop'].sum()	g14/demo.py
pandas	groupby, using with one grouping variable	$by_reg = state_data.groupby('Region')$	g13/demo.py
pandas	importing the module	import pandas as pd	g12/demo.py
pandas	index, creating with two-levels	county = county.set_index(['state','county'])	g14/demo.py
pandas	index, listing names	<pre>print('\nIndex (rows):', list(states.index))</pre>	g12/demo.py

pandas index, retrieving a row by name pandas index, retrieving first rows by location print(low_to_high.iloc[0:10]) g12/den pandas index, retrieving last rows by location print(low_to_high.iloc[0:5]) g12/den pandas index, setting to a column in place states.set_index('name'), index, setting to g12/den pandas reading, csv data states = pd.read_csv('state-gathone) g16/den pandas series, combining via where() states = pd.read_csv('state-data.csv') g12/den pandas series, converting to float sample['dollars'] = unit_part.where(unit_part!=" '', mo g23/den pandas series, converting to float sample['dollars'] = sample['TRANSACTION_AMT'].astype(fl g17/den pandas series, element-by-element or is_elither = is_ca is_tx mod 'comb_units'].sinlina('feet') g23/den mod 'name'] = mod 'comb_units'].sinlina('feet') g23/den mod 'name'] = mod 'name'] = mod 'name'].sinlina('feet') g23/den mod 'name'] = mod 'name'] = mod 'name'].sinlina('feet') g23/den pandas series, retrieving an element print("\nFlorida's population:", pop 'Florida']/166) g12/den pandas series, using isin() fixed = flood 'TAX_ID'].sin(dup_rec['TAX_ID']) g13/den pandas series, using value counts() print("\nOuter:\n', join_o']_merge['pop'].sun().166 g13/den pandas series, using value.counts() print("\nOuter:\n', join_o']_merge['pop'].sun().166 g13/den pandas using quut to create deciles using xs to select from an index print('numbe	Module	Description	Example	Script
pandas index, retrieving first rows by location print (low_to_high.iloc[0:10]) g12/den pandas index, setting to a column new_states = states.set_index('name') g12/den g12/den pandas plotting, disabling legend pandas plotting, disabling legend plotting, horizontal bar plot div_pop.plot.barh(ax=axl) g13/den pandas plotting, horizontal bar plot div_pop.plot.barh(ax=axl) g13/den pandas reading, csv data states = pd.read_csv('state-data.csv') g12/den pandas series, converting to float series, converting to lower case units = units_plandas series, converting to lower case units = units_str.lower() g23/den pandas series, removing spaces pandas series, series, values a series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing a lement pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, units pandas series, pandas series, series, value pandas pandas series, value pandas pandas series, value pandas pandas series, value pandas p	pandas	index, renaming values	div_pop = div_pop.rename(index=div_names)	g13/demo.py
pandas index, retrieving first rows by location print (low_to_high.iloc[0:10]) g12/den pandas index, setting to a column new_states = states.set_index('name') g12/den g12/den pandas plotting, disabling legend pandas plotting, disabling legend plotting, horizontal bar plot div_pop.plot.barh(ax=axl) g13/den pandas plotting, horizontal bar plot div_pop.plot.barh(ax=axl) g13/den pandas reading, csv data states = pd.read_csv('state-data.csv') g12/den pandas series, converting to float series, converting to lower case units = units_plandas series, converting to lower case units = units_str.lower() g23/den pandas series, removing spaces pandas series, series, values a series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing a lement pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, series, value pandas series, removing spaces pandas series, series, value pandas series, removing spaces pandas series, units pandas series, pandas series, series, value pandas pandas series, value pandas pandas series, value pandas pandas series, value pandas p	pandas	index, retrieving a row by name	pa_row = states.loc['Pennsylvania']	g12/demo.py
pandas index, setting to a column new_states = states.set_index('name') g12/den g13/den g13/de	pandas	index, retrieving first rows by location		g12/demo.py
index, setting to a column in place states.set_index(name') g12/den g12/den g12/den g12/den states.set_index, setting to a column in place states.set_index(name', inplace=True) g13/den g12/den g13/den glotting, disabling legend summary.plot.bar(ax=ax1) g13/den g	pandas	index, retrieving last rows by location	print(low_to_high.iloc[-5:])	g12/demo.py
pandas plotting, bar plot summary.plot.bar(ax=ax1) summary.plot.bar(hy='mw',ax=ax1,legend=None) g13/den plotting, disabling legend div_pop.plot.barh(y='mw',ax=ax1,legend=None) g13/den plotting, horizontal bar plot div_pop.plot.barh(ax=ax1) g13/den plotting, horizontal bar plot div_pop.plot.barh(y='mw',ax=ax1,legend=None) g13/den plotting, horizontal bar plot div_pop.plot.barh(ax=ax1) g13/den plotting, horizontal bar plot div_pop.plot.barh(ax=ax1) g13/den plotting, horizontal bar plot div_pop.plot.barh(ax=ax1) g13/den plotting, horizontal bar plot div_pop.plot.bar plot.bar pl	pandas	index, setting to a column		g12/demo.py
pandas plotting, disabling legend plotting, horizontal bar plot div_pop.plot.barh(y='mw',ax=ax1,legend=None) g16/den g13/den pandas reading, csv data reading, csv using dtype geocodes = pd.read_csv('state-data.csv') g12/den gandas series, combining via where() mod['comb_units'] = unit_part.where(unit_part!=" , mo	pandas	index, setting to a column in place	states.set_index('name',inplace=True)	g12/demo.py
pandas plotting, horizontal bar plot div_pop.plot.barh(ax=ax1) g13/den pandas reading, csv data states = pd.read_csv('state-data.csv') g12/den pandas reading, csv using dtype geocodes = pd.read_csv('state-geocodes.csv',dtype=str) g13/den pandas series, combining via where() mod['comb_units'] = unit_part.where(unit_part!=" , mo g23/den pandas series, converting to filoat sample['dollars'] = sample['TRANSACTION_AMT'].astype(fl g17/den pandas series, converting to lower case units = units.rlower() g23/den pandas series, element-by-element or is_either = is_ca is_tx g17/den pandas series, filling missing values mod['comb_units'] = mod['comb_units'].fillna('feet') g23/den pandas series, replacing values using a dictionary units = mod['name] = mod['name].str.strip() g23/den pandas series, sorting by value units = mod['name].str.strip() g23/den pandas series, sorting by value pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/den pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/den pandas series, using isin() fixed = fload['TAX_ID'].sin(dup_rec 'TAX_ID') g15/den pandas series, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/den pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/den requests calling the get() method response = requests.get(api,payload)	pandas			g13/demo.py
pandas reading, csv data reading, csv data reading, csv using dtype geocodes = pd.read_csv('state-data.csv') g12/den geocodes = pd.read_csv('state-geocodes.csv',dtype=str) g13/den g17/den g	pandas	plotting, disabling legend	summary.plot.barh(y='mw',ax=ax1,legend=None)	g16/demo.py
pandas reading, csv using dtype geocodes = pd.read_csv('state-geocodes.csv',dtype=str) g13/den pandas series, combining via where() mod['comb_units'] = unit_part.where(unit_part!=" , mo	pandas	plotting, horizontal bar plot	$div_pop.plot.barh(ax=ax1)$	g13/demo.py
pandas series, combining via where() mod['comb_units'] = unit_part.where(unit_part!=" , mo	pandas			g12/demo.py
pandas series, converting strings to title case pandas series, converting to float sample['dollars'] = sample['TRANSACTION_AMT'].astype(fl. g17/dem sample[series, converting to lower case units = units.str.lower() g23/dem pandas series, element-by-element or is_either = is_ca is_tx g17/dem pandas series, filling missing values mod['comb_units'] = mod['comb_units'].fillna('feet') g23/dem pandas series, removing spaces mod['name'] = mod['name'].str.strip() g23/dem pandas series, replacing values using a dictionary pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) g12/dem pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() g13/dem pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/dem pandas series, using isin() fixed = flood['TAX_ID'].sin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print("\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index response = requests.get(api,payload)	pandas	reading, csv using dtype	${\sf geocodes} = {\sf pd.read_csv(`state-geocodes.csv',dtype=str)}$	g13/demo.py
pandas series, converting to float sample['dollars'] = sample['TRANSACTION_AMT'].astype(fl	pandas	series, combining via where()	<pre>mod['comb_units'] = unit_part.where(unit_part!=" , mo</pre>	g23/demo.py
pandas series, converting to lower case units = units.str.lower() pandas series, element-by-element or is_either = is_ca is_tx g17/den pandas series, filling missing values mod['comb_units'] = mod['comb_units'].fillna('feet') pandas series, removing spaces mod['name'] = mod['name'].str.strip() pandas series, replacing values using a dictionary pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() pandas series, unstacking reg_pop = by_reg['pop'].sum()/1e6 g13/den pandas series, unstacking tot_wide = tot_amt.unstack('PGI') pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/den pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) pandas using the shape attribute print('number of rows, columns:', conus.shape) pandas calling the get() method response = requests.get(api,payload)	pandas	series, converting strings to title case	$fixname = subset_view['NAME'].str.title()$	g17/demo.py
pandas series, element-by-element or is_either = is_ca is_tx g17/dem pandas series, filling missing values mod['comb_units'] = mod['comb_units'].fillna('feet') g23/dem pandas series, removing spaces mod['name'] = mod['name'].str.strip() g23/dem pandas series, replacing values using a dictionary pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) g12/dem pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() g12/dem pandas series, unstacking reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g19/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, converting to float	$sample['dollars'] = sample['TRANSACTION_AMT'].astype(fl.\ .\ .$	g17/demo.py
pandas series, filling missing values mod['comb_units'] = mod['comb_units'].fillna('feet') g23/dem pandas series, removing spaces mod['name'] = mod['name'].str.strip() g23/dem pandas series, replacing values using a dictionary units = units.replace(spellout) g23/dem pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) g12/dem pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() g12/dem pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, using isin() fixed = flood['TAX_ID'].sin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s. g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g19/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, converting to lower case	units = units.str.lower()	g23/demo.py
pandas series, removing spaces mod['name'] = mod['name'].str.strip() pandas series, replacing values using a dictionary units = units.replace(spellout) pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, unstacking tot_wide = tot_amt.unstack("PGI") pandas series, using isin() pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) pandas using the shape attribute print('number of rows, columns:', conus.shape) pandas using xs to select from an index response = requests.get(api,payload) g19/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, element-by-element or	is_either = is_ca is_tx	g17/demo.py
pandas series, replacing values using a dictionary pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) g12/dem g12/dem g12/dem g12/dem g13/dem g12/dem g13/dem	pandas	series, filling missing values	$mod[`comb_units'] = mod[`comb_units'].fillna(`feet')$	g23/demo.py
pandas series, retrieving an element print("\nFlorida's population:", pop['Florida']/1e6) g12/dem pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() g12/dem pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/dem pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem pandas calling the get() method response = requests.get(api,payload)	pandas			g23/demo.py
pandas series, sorting by value low_to_high = normed['med_pers_inc'].sort_values() g12/dem pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/dem pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem pandas calling the get() method response = requests.get(api,payload)	pandas	series, replacing values using a dictionary		g23/demo.py
pandas series, summing reg_pop = by_reg['pop'].sum()/1e6 g13/dem pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/dem pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, retrieving an element		g12/demo.py
pandas series, unstacking tot_wide = tot_amt.unstack('PGI') g17/dem g17/dem pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, sorting by value		g12/demo.py
pandas series, using isin() fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID']) g15/dem pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload)	pandas	series, summing	$reg_pop = by_reg['pop'].sum()/1e6$	g13/demo.py
pandas series, using value_counts() print('\nOuter:\n', join_o['_merge'].value_counts(), s g15/dem pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload) g19/dem	pandas	series, unstacking		g17/demo.py
pandas setting the index keeping the column states = states.set_index('STUSPS',drop=False) g22/dem pandas using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g14/dem pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload) g19/dem	pandas	series, using isin()	$fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID'])$	g15/demo.py
pandas using qcut to create deciles $dec = pd.qcut(county['pop'], 10, labels=range(1,11))$ $g14/dem$ pandas using the shape attribute $print('number of rows, columns:', conus.shape)$ $g22/dem$ pandas using xs to select from an index $print(county.xs('04',level='state'))$ $g14/dem$ requests $print(county.xs('payload))$ $print(county.xs('payload))$ $print(county.xs('payload))$	pandas	series, using value_counts()	<pre>print('\nOuter:\n', join_o['_merge'].value_counts(), s</pre>	g15/demo.py
pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload) g19/dem	pandas	setting the index keeping the column	${\sf states} = {\sf states.set_index(`STUSPS', drop=False)}$	g22/demo.py
pandas using the shape attribute print('number of rows, columns:', conus.shape) g22/dem pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload) g19/dem	pandas	using qcut to create deciles	dec = pd.qcut(county['pop'], 10, labels=range(1,11))	g14/demo.py
pandas using xs to select from an index print(county.xs('04',level='state')) g14/dem requests calling the get() method response = requests.get(api,payload) g19/dem	pandas	using the shape attribute		g22/demo.py
	pandas	using xs to select from an index		g14/demo.py
requests checking the URL print('url:', response,url) g19/den	requests	calling the get() method	response = requests.get(api,payload)	g19/demo.py
F()	requests	checking the URL	print('url:', response.url)	g19/demo.py

Module	Description	Example	Script
requests	checking the response text	print(response.text)	g19/demo.py
requests	checking the status code	<pre>print('status:', response.status_code)</pre>	g19/demo.py
requests	decoding a JSON response	rows = response.json()	g19/demo.py
requests	importing the module	import requests	g19/demo.py
scipy	calling newton's method	<pre>cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y</pre>	g07/demo.py
scipy	importing the module	import scipy optimize as opt	g07/demo.py
seaborn	adding a title to a grid object	jg.fig.suptitle('Distribution of Hourly Load')	g18/demo.py
seaborn	barplot	sns.barplot(data=hourly,x='hour',y='usage',hue='month',	g18/demo.py
seaborn	basic violin plot	sns.violinplot(data=janjul,x="month",y="usage")	g18/demo.py
seaborn	boxenplot	sns.boxenplot(data=janjul,x="month",y="usage")	g18/demo.py
seaborn	calling tight_layout on a grid object	jg.fig.tight_layout()	g18/demo.py
seaborn	drawing a heatmapped grid	sns.heatmap(data,annot=True,fmt=".0f",ax=ax1)	g20/demo.py
seaborn	importing the module	import seaborn as sns	g18/demo.py
seaborn	joint distribution hex plot	<pre>jg = sns.jointplot(data=bymo,x=1,y=7,kind='hex')</pre>	g18/demo.py
seaborn	setting axis titles on a grid object	jg.set_axis_labels('January','July')	g18/demo.py
seaborn	setting the theme	sns.set_theme(style="white")	g18/demo.py
seaborn	split violin plot	sns.violinplot(data=eights,x="hour",y="usage",hue="mont	g18/demo.py
zipfile	creating a ZipFile object	$zip_object = zipfile.ZipFile(zipname)$	g11/demo.py
zipfile	importing module	import zipfile	g11/demo.py
zipfile	opening a file in a zip in bytes mode	<pre>inp_byte = zip_object.open(csvname)</pre>	g11/demo.py