Module	Description	Example	Script
collections	creating a defaultdict of lists	by_zone = defaultdict(list)	g10/demo.py
collections	importing defaultdict	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, 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 = $[n^{**3}]$ 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 g03/demo.py g03/demo.py g03/demo.py g03/demo.py g03/demo.py core list, generating a sequence b_list = range(1.6) g04/demo.py g03/demo.py g03/demo.py g03/demo.py g03/demo.py ported.alist] g03/demo.py	Module	Description	Example	Script
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 1 to 2 print(a_list[1:4]) 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[1:1]) g03/demo.py core list, selecting the last element print(a_list[1:1]) 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, sorting states = ",'join(sorted(by_zone[tz])) g10/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() g11/demo.py <td>core</td> <td>list, extending with another list</td> <td>a_list.extend(a_more)</td> <td>g03/demo.py</td>	core	list, extending with another list	a_list.extend(a_more)	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[1:3]) 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 last 3 elements print(a_list[1:1]) 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, sorting c_sort = sorted(b_list) g03/demo_py core list, sorting states = '.'.join(sorted(by_zone[tz])) g10/demo_py core list, sorting states = '.'.join(sorted(by_zone[tz])) g10/demo_py core list, summing tot_inc = sum(incomes) g08/demo_py core math, raising a number rounded = round(ratio_2) g05/demo_py core string, concatenating name = s1+" "+s2+" "+s3 g02/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 for title case name = codes[kgv], title() g11/demo_py core string, including a newline character fluxmite(name+"\n"\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, stripting blank space clean = [item-strip() for item in parts] g05/demo_py core tuple, creating the split (last,first) = name.split(',') g11/demo_py core tuple, looping over for (den,state) in sorted(by_density): g11/demo_py core tuple, looping over for (den,state) in sorted(by_density): g11/demo_py core tuple, sorting a blic for use with DictWriter flate over the case cost. DictReader(flh) g03/demo_py core setting up a DictReader object reader = csv. DictReader(flh) g03/demo_py core setting up a DictReader with a list reader csv. DictReader(lines) g10/demo_py core writing a header with DictWriter writeheader()	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 the end 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 helments 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, sorting c_sort = sorted(b_list) g03/demo_py core list, sorting c_sort = sorted(b_list) 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, including a newline character fl.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, splitting on whitespace b_list = b_string.split() g03/demo_py core string, splitting on whitespace 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 use with DictWriter flavored flow, winewline=") g09/demo_py core tuple, sorting for use with DictWriter flavored flow, winewline=") g09/demo_py core setting up a DictReader object reader = csv. DictReader(flh) g09/demo_py cov writing a header with DictWriter writereader(lines) g10/demo_py cov writing a header with	core	list, joining with spaces	a_string = " ".join(a_list)	g03/demo.py
core list, selecting elements 0 to 3 print(a_list[:4]) g03/demo_py core list, selecting elements 1 to the end 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 helments 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, sorting c_sort = sorted(b_list) g03/demo_py core list, sorting c_sort = sorted(b_list) 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, including a newline character fl.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, splitting on whitespace b_list = b_string.split() g03/demo_py core string, splitting on whitespace 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 use with DictWriter flavored flow, winewline=") g09/demo_py core tuple, sorting for use with DictWriter flavored flow, winewline=") g09/demo_py core setting up a DictReader object reader = csv. DictReader(flh) g09/demo_py cov writing a header with DictWriter writereader(lines) g10/demo_py cov writing a header with	core	list, selecting an element		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, sorting c_sort = sorted(b_list) g03/demo.py core list, sorting c_sort = sorted(b_list) g03/demo.py core list, summing tot_ince = sum(incomes) g08/demo.py core list, summing tot_ince = sum(incomes) g08/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 filename = "demo.txt" 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 = [tem.strip() for item in parts] g05/demo.py core tuple, creating for this_tuple = (med_density.state) g10/demo.py core tuple, creating for well clast.first) aname.split(',') g11/demo.py core tuple, looping over for (den.state) in sorted(by_density): g10/demo.py core tuple, testing equality of if key == (29, VA): g11/demo.py core tuple, testing equality of if key == (29, VA): g11/demo.py core setting up a DictReader object reader = csv.DictReader(fine) g03/demo.py core setting up a DictReader with DictWriter writer esv.DictReader(fines) g09/demo.py core writing a header with DictWriter writer.writeheader()	core			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, sorting states = '.'.join(sorted(by_zone[tz])) g10/demo.py core list, summing to to_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 mounded = 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 fliename = "demo.tx" 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, splitting on whitespace b_list = b_string.split() g03/demo.py core tuple, creating this_space clean = [tem.strip() for item in parts] g05/demo.py core tuple, creating this_split (last,first) = name.split(',') g11/demo.py core tuple, creating for key in sorted(codes): g11/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, sorting for use with DictWriter for key in sorted(codes): g11/demo.py core tuple, sorting to title case fire g09/demo.py core setting up a DictReader object reader = csv.DictReader(flh) g08/demo.py csv setting up a DictReader object reader = csv.DictReader(flh, fields) g09/demo.py csv writing a header with DictWriter writer.eader(flines) 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, sorting c_sort = sorted(b_list) g03/demo.py core list, sorting states = '.';oin(sorted(by_zone[tz])) g10/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) g10/demo.py core tuple, looping over for (den,state) in sorted(by_density): g11/demo.py core tuple, looping over for key in sorted(codes): 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(fh) 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) g09/demo.py csv writing a header with DictWriter writer.writeheader() writer.writeheader()	core	list, selecting elements 1 to the end	print(a_list[1:])	g03/demo.py
core list, sorting c_sort = sorted(b_list) g03/demo.py core list, sorting states = ','.join(sorted(by_zone[tz])) g10/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, 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(codes): g11/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 tuple, testing up a DictReader object reader = csv.DictReader(fh) g09/demo.py csv setting up a DictReader with DictWriter write-writeheader() g09/demo.py csv writing a header with DictWriter write-writeheader()	core	list, selecting last 3 elements	print(a_list[-3:])	g03/demo.py
core list, sorting states = ','join(sorted(by_zone[tz]) g10/demo.py g08/demo.py tot_inc = sum(incomes) g08/demo.py g08/demo.py g08/demo.py core math, raising a number to a power math, rounding a number rounded = round(ratio.2) g05/demo.py g05/demo.py core string, concatenating name = s1+" "+s2+" "+s3 g02/demo.py g05/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 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 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 this_sorting from sorted(by_density): g11/demo.py core tuple, sorting for key in sorted(codes): g11/demo.py core tuple, sorting a file for use with DictWriter fh = open(outfile, 'w',newline=") g09/demo.py core setting up a DictReader object reader = csv.DictReader(fih) g08/demo.py csv setting up a DictReader with DictWriter 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	list, selecting the last element	print(a_list[-1])	g03/demo.py
core list, summing tot_inc = sum(incomes) core math, raising a number to a power math, rounding a number math, rounded = round(ratio,2) core string, concatenating name = s1+" "+s2+" "+s3 core string, converting to an int values.append(int(line)) core string, converting to title case name = codes[key].title() core string, including a newline character fh.write(name+"!\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 wis split (last, first) = name.split(',') core tuple, creating for key in sorted(by_density): core tuple, sorting for use with DictWriter fh = open(outfile,'w',newline=") cove setting up a DictReader object reader = csv. DictReader(fh) csv setting up a DictReader with DictWriter writer-average cov.	core	list, sorting	$c_sort = sorted(b_list)$	g03/demo.py
core list, summing tot_inc = sum(incomes) core math, raising a number to a power math, rounding a number math, rounded = round(ratio,2) core string, concatenating name = s1+" "+s2+" "+s3 core string, converting to an int values.append(int(line)) core string, converting to title case name = codes[key].title() core string, including a newline character fh.write(name+"!\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 wis split (last, first) = name.split(',') core tuple, creating for key in sorted(by_density): core tuple, sorting for use with DictWriter fh = open(outfile,'w',newline=") cove setting up a DictReader object reader = csv. DictReader(fh) csv setting up a DictReader with DictWriter writer-average cov.	core	list, sorting	states = ','.join(sorted(by_zone[tz]))	g10/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, creating fllename = "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 this_tuple = (med_density,state) 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 string up a DictReader object reader = csv.DictReader(fh) csv setting up a DictReader object writer = csv.DictReader(fh, fields) g09/demo.py csv writing a header with DictWriter writer-writeheader() g09/demo.py csv writing a header with DictWriter writer-makes	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 reader = csv.DictReader(flines) g09/demo.py csv writing a header with DictWriter writen.ear() g09/demo.py csv	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.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, 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 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) g08/demo.py csv setting up a DictReader object writer = csv.DictReader(lines) g09/demo.py csv using DictReader with a list reader = 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()	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, 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 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()	core	string, converting to an int	<pre>values.append(int(line))</pre>	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 tuple, testing 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 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 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.DictReader(lines) 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() g09/demo.py	core	string, creating	filename = "demo.txt"	g02/demo.py
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	core	string, including a newline character	fh.write(name+"!\n")	g02/demo.py
core tuple, creating tuple, creating tuple, creating variety (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 cov setting 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() g09/demo.py csv writer = csv.DictReader(lines) g09/demo.py csv writer a header with DictWriter writeheader() g09/demo.py csv writing a header with DictWriter writeheader() g09/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 cov opening a file for use with DictWriter fh = open(outfile, 'w',newline=") cov 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	$b_list = b_string.split()$	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	$this_tuple = (med_density,state)$	g10/demo.py
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	core	tuple, creating via split		g11/demo.py
core tuple, testing equality of if key $==$ (29, 'VA'): g11/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	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 writeheader() $g09/demo.py$	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
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	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
io	converting a byte stream to characters	$inp_handle = io.TextIOWrapper(inp_byte)$	g11/demo.py
json json	importing the module using to print an object nicely	<pre>import json print(json.dumps(list1,indent=4))</pre>	g05/demo.py g05/demo.py
numpy numpy	computing a median importing	$\label{eq:med_density} \mbox{med_density} = \mbox{round(np.median(this_list), 2)} \\ \mbox{import numpy as np}$	$\begin{array}{c} \text{g10/demo.py} \\ \text{g10/demo.py} \end{array}$
scipy scipy	calling newton's method importing the module	$\label{eq:cr} \begin{split} &\text{cr} = \text{opt.newton}(\text{find_cube_root,xinit,maxiter} = 20, \text{args} = [y]) \\ &\text{import scipy.optimize as opt} \end{split}$	g07/demo.py g07/demo.py
zipfile zipfile zipfile	creating a ZipFile object importing module opening a file in a zip in bytes mode	<pre>zip_object = zipfile.ZipFile(zipname) import zipfile inp_byte = zip_object.open(csvname)</pre>	$\begin{array}{c} {\rm g11/demo.py} \\ {\rm g11/demo.py} \\ {\rm g11/demo.py} \end{array}$