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, creating via comprehension	fips_cols = {col:str for col in fips_vars}	g13/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, grouping with commas	<pre>print(f'Total population: {tot_pop:,}')</pre>	g11/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, print without adding spaces	<pre>print('\nOuter:\n', join_o['_merge'].value_counts(), s</pre>	g14/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	for, looping through a list of tuples	for number,name in div_info:	g13/demo.py
core	function, calling	$d1_ssq = sumsq(d1)$	g06/demo.py
core	function, calling with an optional argument	sample_function(100, 10, r=0.07)	g07/demo.py
core	function, defining	def sumsq(values):	g06/demo.py
core	function, defining with optional argument	def sample_function(payment,year,r=0.05):	g07/demo.py
core	function, returning a result	return values	g06/demo.py
core	list, appending an element	a_list.append("four")	g03/demo.py
core	list, create via comprehension	$cubes = [n**3 for n in a_list]$	g04/demo.py
core	list, creating	a_list = ["zero","one","two","three"]	g03/demo.py
core	list, determining length	$n = len(b_list)$	g03/demo.py
core	list, extending with another list	a_list.extend(a_more)	g03/demo.py
core	list, generating a sequence	$b_list = range(1,6)$	g04/demo.py
core	list, joining with spaces	$a_string = "".join(a_list)$	g03/demo.py
core	list, selecting an element	print(a_list[0])	g03/demo.py
core	list, selecting elements 0 to 3	print(a_list[:4])	g03/demo.py
core	list, selecting elements 1 to 2	print(a_list[1:3])	g03/demo.py

Module	Description	Example	Script
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
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	sets, computing difference	<pre>print(name_states - pop_states)</pre>	g13/demo.py
core	sets, creating	name_states = set(name_data['State'])	g13/demo.py
core	sets, of tuples	tset1 = set([(1,2), (2,3), (1,3), (2,3)])	g13/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	reader = csv.DictReader(fh)	g08/demo.py
geopandas	extracting geometry from a geodataframe	wv_geo = wv['geometry']	g22/demo.py
geopandas	importing the module	import geopandas as gpd	g21/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	plot with categorical coloring	sel.plot('NAME',cmap='Dark2',ax=ax1)	g22/demo.py
geopandas	plotting a boundary	syr.boundary.plot(color='gray',linewidth=1,ax=ax1)	g21/demo.py
geopandas	reading a file	syr = gpd.read_file("tl_2016_36_place-syracuse.zip")	g21/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
json	importing the module	import json	g05/demo.py

Module	Description	Example	Script
json	using to print an object nicely	<pre>print(json.dumps(list1,indent=4))</pre>	g05/demo.py
matplotlib	axes, adding a horizontal line	ax21.axhline(medians['etr'], c='r', ls='-', lw=1)	g12/demo.py
matplotlib	axes, adding a vertical line	ax21.axvline(medians['inc'], c='r', ls='-', lw=1)	g12/demo.py
matplotlib	axes, labeling the X axis	ax2.set_xlabel('Millions')	g11/demo.py
matplotlib	axes, labeling the Y axis	ax1.set_ylabel("Millions")	g11/demo.py
matplotlib	axes, turning off the label	ax.set_ylabel(None)	g13/demo.py
matplotlib	colors, xkcd palette	syr.plot(color='xkcd:lightblue',ax=ax1)	g21/demo.py
matplotlib	figure, adding a title	fig2.suptitle('Pooled Data')	g12/demo.py
matplotlib	figure, four panel grid	fig3, axs = $plt.subplots(2,2,sharex=True,sharey=True)$	g12/demo.py
matplotlib	figure, left and right panels	fig2, $(ax21,ax22) = plt.subplots(1,2)$	g12/demo.py
matplotlib	figure, saving	fig2.savefig('figure.png')	g11/demo.py
matplotlib	figure, setting the size	fig, $axs = plt.subplots(1,2,figsize=(12,6))$	g20/demo.py
matplotlib	figure, tuning the layout	fig2.tight_layout()	g11/demo.py
matplotlib	figure, working with a list of axes	for ax in axs:	g20/demo.py
matplotlib	importing pyplot	import matplotlib.pyplot as plt	g11/demo.py
matplotlib	setting the default resolution	plt.rcParams['figure.dpi'] = 300	g11/demo.py
matplotlib	using subplots to set up a figure	fig1, ax1 = plt.subplots()	g11/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, renaming	county = county.rename(columns={'B01001_001E':'pop'})	g10/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, appending	gen_all = pd.concat([gen_oswego, gen_onondaga])	g15/demo.py
pandas	dataframe, boolean row selection	print(trim[has_AM], "\n")	g12/demo.py
pandas	dataframe, dropping a column	$both = both.drop(columns='_merge')$	g15/demo.py
pandas	dataframe, dropping duplicates	flood = flood.drop_duplicates(subset='TAX_ID')	g14/demo.py
pandas	dataframe, dropping missing data	trim = demo.dropna(subset="Days")	g12/demo.py
pandas	dataframe, finding duplicate records	$dups = parcels.duplicated(subset='TAX_ID', keep=False$	g14/demo.py
pandas	dataframe, getting a block of rows via index	sel = merged.loc[number]	g13/demo.py
pandas	dataframe, inner 1:1 merge	$join_{L}i = parcels.merge(flood,$	${\sf g14/demo.py}$
pandas	dataframe, inner join	$merged = name_data.merge(pop_data,left_on = ``State'',right. \ . \ .$	g13/demo.py
pandas	dataframe, left 1:1 merge	$join_l = parcels.merge(flood,$	g14/demo.py
pandas	dataframe, left m:1 merge	$both = gen_all.merge(plants,$	g15/demo.py
pandas	dataframe, making a copy	trim = trim.copy()	g12/demo.py

dataframe, reading zipped pickle format bandas dataframe, resetting the index hourly = hourly.reset_index() g17/dc	Module	Description	Example	Script
dataframe, reading zipped pickle format hourly = hourly, reset_index() g17/ds g16/ds g14/ds gandas dataframe, resetting the index hourly = hourly, reset_index() g17/ds g16/ds gandas dataframe, saving in zipped pickle format dataframe, selecting rows by list indexing pandas dataframe, selecting rows by list indexing pandas dataframe, selecting rows via boolean dataframe, selecting rows via boolean dataframe, set index keeping the column pandas dataframe, set index keeping the column pandas dataframe, sent index keeping the column pandas dataframe, sorting by a column dataframe, sorting by index summary = summary.sort_index(ascending=False) g22/ds gandas dataframe, sorting by index summary = summary.sort_index(ascending=False) g16/ds gandas dataframe, using as multilevel column index pandas dataframe, using as multilevel column index dataframe, using x sith columns dataframe, using x sith columns dataframe, using x sith columns dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building via to_datetime() datetime, building via to_datetime() datetime, extracting day attribute extracting day attribute gandas general, displaying all columns general, using copy_on_write mode groupby, terrating over groups for tild give produced groupby, terrating over groups for tild give produced groupby, terrating over groups for tild give produced groupby, state: sample2 = pd.read_polk(sample_plk.zip') g11/de g1/de g1/	pandas	dataframe, outer 1:1 merge	join_o = parcels.merge(flood,	g14/demo.py
pandas dataframe, right 1:1 merge join_r = parcels.merge(flood, pandas dataframe, saving in zipped pickle format pandas dataframe, selecting rows by list indexing pandas dataframe, selecting rows via boolean dataframe, selecting rows via boolean dataframe, selecting rows via duery trimmed = county.query("state == '04' or state == '36' ") g10/de states = states, states = states.et_index("STUSPS', drop=False) g22/de dataframe, sorting by a column dataframe, sorting by index summary = summary.sort_values("pop") g10/de states pandas dataframe, sorting by index summary = summary.sort_values("pop") g10/de states pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using x multilevel column index pandas dataframe, writing to a CSV file merged.to_csv('denomerged.csv') g10/de dataframe, writing to a CSV file merged.to_csv('denomerged.csv') g13/de datetime, building with a format ymandas datetime, extracting day attribute pandas general, displaying all columns pandas general, displaying all rows pandas general, importing the module gandas general, using copy_on_mvite mode pandas general, using copy_on_mvite mode gandas groupby, cumulative sum within group for t, g in groupby, stearting over groups for t, g in groupby, terating over groups for t, g in groupby, terating over groups for t, g in groupby, state["pop"].describe() g10/de for t, g in groupby, terating over groups	pandas	dataframe, reading zipped pickle format	sample2 = pd.read_pickle('sample_pkl.zip')	g16/demo.py
pandas dataframe, saving in zipped pickle format pandas dataframe, selecting rows by list indexing dataframe, selecting rows via operandas dataframe, selecting rows via query trimmed = county,query("state == '04' or state == '36' ") g10/ds quarter dataframe, shape attribute print('number of rows, columns', conus.shape) g22/ds quarter dataframe, sorting by a column county = county.sort_values('pop') g10/ds quarter dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using s so select a subset dataframe, writing to a CSV file datetime, building via to_datetime() datetime, building via to_datetime() datetime, extracting day attribute pandas datetime, extracting day attribute pandas general, displaying all columns pandas general, importing the module pandas general, using copy_on_write mode pandas general, using copy_on_write mode gandas groupby, cumulative sum within group pandas groupby, cumulative sum within group groups for the garden and so groupby, descriptive statistics group into the column in the county into the column index pandas groupby, tetrating over groups for trig g10/de groupby, statelipop'], describe() g10/de pandas groupby, descriptive statistics groupby statelipop'], describe() g10/de groupby statelipop'], describe() g10/de groupby, statelipop'], describe() g10/de groupby, statelipop'], describe() g10/de groupby, statelipop'], describe() g10/de grou	pandas	dataframe, resetting the index		g17/demo.py
pandas dataframe, selecting rows via boolean dup_rec = flood[dups] g14/de pandas dataframe, selecting rows via query trimmed = county.query("state == '04' or state == '36' ") g10/de pandas dataframe, selecting rows via query pandas dataframe, set index keeping the column pandas dataframe, set index keeping the column pandas dataframe, sorting by a column county = county.sort_values("pop") g22/de pandas dataframe, sorting by index pandas dataframe, sorting by index pandas dataframe, sorting by index pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, writing to a CSV file merged.to_csv("der-merged.csv") g13/de pandas dataframe, writing to a CSV file merged.to_csv("der-merged.csv") g13/de pandas datetime, building with a format datetime, extracting day attribute recs['day'] = date.dt.day general, displaying all columns pandas general, displaying all rows pandas general, displaying all rows pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas groupby, cumulative sum within group groupby, iterating over groups for time for the control of the control	pandas	dataframe, right 1:1 merge	$join_r = parcels.merge(flood,$	g14/demo.py
pandas dataframe, selecting rows via boolean dup_rec = flood[dups] g14/de pandas dataframe, selecting rows via query trimmed = county.query("state == '04' or state == '36' ") g10/de pandas dataframe, selecting rows via query pandas dataframe, set index keeping the column pandas dataframe, set index keeping the column pandas dataframe, sorting by a column county = county.sort_values("pop") g22/de pandas dataframe, sorting by index pandas dataframe, sorting by index pandas dataframe, sorting by index pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, writing to a CSV file merged.to_csv("der-merged.csv") g13/de pandas dataframe, writing to a CSV file merged.to_csv("der-merged.csv") g13/de pandas datetime, building with a format datetime, extracting day attribute recs['day'] = date.dt.day general, displaying all columns pandas general, displaying all rows pandas general, displaying all rows pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas groupby, cumulative sum within group groupby, iterating over groups for time for the control of the control	pandas	dataframe, saving in zipped pickle format	sample.to_pickle('sample_pkl.zip')	g16/demo.py
pandas dataframe, selecting rows via query trimmed = county.query("state =="04" or state == "36"") g10/ds dataframe, set index keeping the column states = states.set_index("STUSPS',drop=False) g22/ds g2/ds g22/ds	pandas		print(low_to_high[-5:])	g09/demo.py
pandas dataframe, selecting rows via query trimmed = county.query("state =="04" or state == "36"") g10/ds pandas dataframe, sapet index keeping the column states = states.set_index("STUSPS',drop=False) g22/ds pandas dataframe, sorting by a column county = county.sort_values('pop') g10/ds pandas dataframe, sorting by index summary = summary.sort_index(ascending=False) g15/ds pandas dataframe, usmming a boolean print("induplicate parcels:", dups.sum()) g14/ds pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs with columns c1 = grid.xs('c1',axis="columns',level=1) g20/ds pandas dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/ds pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) ymd = pd.to_datetime(recs['day'] = date.dt.day g14/ds pandas datetime, extracting day attribute recs['day'] = date.dt.day general, displaying all rows pandas general, displaying all rows pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas groupby, cumulative sum within group cumulative inc_stats = group_by_state("pop').escribe() g10/ds g10/ds groupby, iterating over groups for t.g in group_by_state("pop').escribe() g10/ds g10/ds g10/ds groupby, iterating over groups	pandas	dataframe, selecting rows via boolean	dup_rec = flood[dups]	g14/demo.py
pandas dataframe, set index keeping the column states = states.set_index('STUSPS',drop=False) g22/de pandas dataframe, shape attribute print('number of rows, columns:', conus.shape) g22/de pandas dataframe, sorting by a column county = county.sort_values('pop') g10/de pandas dataframe, sorting by index summary = summary.sort_index(ascending=False) g15/de pandas dataframe, unstacking an index level pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs with columns c1 = grid.xs('c1',axis='columns',level=1) g20/de pandas dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building viat to_datetime() date = pd.to_datetime(recs['ts']) g14/de datetime, extracting day attribute recs['day'] = date.dt.day g14/de datetime, extracting hour attribute recs['day'] = date.dt.day g14/de pandas general, displaying all columns pandas general, displaying all rows general, using copy_on_write mode pandas general, using copy_on_write mode general, using copy_on_write mode general, using cut to creat deciles dec pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].eumsum() g10/de pandas groupby, iterating over groups for t.g. in group_by_state: g10/de g10/de pandas groupby, iterating over groups for t.g. in group_by_state:		dataframe, selecting rows via query	trimmed = county.query("state == '04' or state == '36' ")	g10/demo.py
pandas dataframe, shape attribute print('number of rows, columns:', conus.shape) g22/de pandas dataframe, sorting by a column county = county.sort_values('pop') g15/de g10/de g10/de g16/de gandas dataframe, sorting by index summary = summary.sort_index(ascending=False) g15/de g16/de g16	pandas	dataframe, set index keeping the column		g22/demo.py
pandas dataframe, sorting by index pandas dataframe, summing a boolean print ("induplicate parcels:", dups.sum()) g15/de pandas dataframe, unstacking an index level pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs with columns c1 = grid.xs('c1',axis='columns',level=1) g20/de pandas dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) g14/de pandas datetime, extracting day attribute pandas datetime, extracting hour attribute recs['day'] = date.dt.day recs['day'] = date.dt.day g14/de pandas general, displaying all rows pandas general, displaying all rows pandas general, using copy_on_write mode pandas general, using copy_on_write	pandas	dataframe, shape attribute	print('number of rows, columns:', conus.shape)	g22/demo.py
pandas dataframe, sorting by index pandas dataframe, summing a boolean print ("induplicate parcels:", dups.sum()) g15/de pandas dataframe, unstacking an index level pandas dataframe, using a multilevel column index pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs with columns c1 = grid.xs('c1',axis='columns',level=1) g20/de pandas dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) g14/de pandas datetime, extracting day attribute pandas datetime, extracting hour attribute recs['day'] = date.dt.day recs['day'] = date.dt.day g14/de pandas general, displaying all rows pandas general, displaying all rows pandas general, using copy_on_write mode pandas general, using copy_on_write	pandas	dataframe, sorting by a column	county = county.sort_values('pop')	g10/demo.py
pandas dataframe, unstacking an index level pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs to select a subset print (county.xs('04',level='state')) g20/de pandas dataframe, writing to a CSV file pandas dateframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building with a format pandas datetime, building with a format pandas datetime, extracting day attribute pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas general, display information about object pandas general, displaying all columns pandas general, importing the module pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas groupby, cumulative sum within group cumulative sum within group pandas groupby, cumulative sum within group cumulative pandas groupby, iterating over groups for t.g in group_by_state['pop'].describe() g10/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g	pandas		summary = summary.sort_index(ascending=False)	g15/demo.py
pandas dataframe, unstacking an index level pandas dataframe, using a multilevel column index pandas dataframe, using xs to select a subset pandas dataframe, using xs to select a subset print (county.xs('04',level='state')) g20/de pandas dataframe, writing to a CSV file pandas dateframe, writing to a CSV file merged.to_csv('demo-merged.csv') g13/de pandas datetime, building with a format pandas datetime, building with a format pandas datetime, extracting day attribute pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas general, display information about object pandas general, displaying all columns pandas general, importing the module pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas groupby, cumulative sum within group cumulative sum within group pandas groupby, cumulative sum within group cumulative pandas groupby, iterating over groups for t.g in group_by_state['pop'].describe() g10/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g20/de merged.to_columns/level=1) g20/de g	pandas	dataframe, summing a boolean	<pre>print('\nduplicate parcels:', dups.sum())</pre>	g14/demo.py
pandas dataframe, using xs to select a subset print(county.xs('04',level='state')) g10/de pandas dataframe, using xs with columns c1 = grid.xs('c1',axis='columns',level=1) g20/de merged.to_csv('demo-merged.csv') g13/de pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) ymd = pd.to_datetime(sample['TRANSACTION_DT'], format= g16/de pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour g14/de pandas general, displaying all columns pd.set_option('display.max_columns',None) g16/de pandas general, displaying all rows pd.set_option('display.max_rows', None) g09/de pandas general, using copy_on_write mode pandas general, using copy_on_write mode general, using copy_on_write mode general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, iterating over groups for t,g in group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups	pandas	dataframe, unstacking an index level	bymo = bymo.unstack('month')	g17/demo.py
pandas dataframe, using xs with columns dataframe, using xs with columns dataframe, writing to a CSV file merged.to_csv('demo-merged.csv') pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) g14/de pandas datetime, building with a format ymd = pd.to_datetime(sample['TRANSACTION_DT'], format= g16/de pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour g14/de pandas general, displaying all columns pandas general, displaying all rows pandas general, importing the module pandas general, using copy_on_write mode pandas general, using copy_on_write mode general, using qut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state['pop'].describe() g10/de g1	pandas	dataframe, using a multilevel column index	means = grid['mean']	g20/demo.py
pandas dateframe, writing to a CSV file merged.to_csv('demo-merged.csv') pandas datetime, building via to_datetime() pandas datetime, building with a format ymd = pd.to_datetime(sample['TRANSACTION_DT'], format= g16/de pandas datetime, extracting day attribute recs['day'] = date.dt.day pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour pandas general, display information about object pandas general, displaying all columns pd.set_option('display.max_columns',None) g16/de pandas general, importing the module import pandas as pd general, using copy_on_write mode pandas general, using copy_on_write mode general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, iterating over groups for t,g in group_by_state:	pandas	dataframe, using xs to select a subset	<pre>print(county.xs('04',level='state'))</pre>	g10/demo.py
pandas datetime, building via to_datetime() date = pd.to_datetime(recs['ts']) g14/de pandas datetime, building with a format ymd = pd.to_datetime(sample['TRANSACTION_DT'], format= g16/de pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour g14/de pandas general, displaying all columns pandas general, displaying all columns pandas general, displaying all rows pandas general, importing the module pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas general, using copy_on_write mode pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state:	pandas	dataframe, using xs with columns		g20/demo.py
pandas datetime, building with a format ymd = pd.to_datetime(sample['TRANSACTION_DT'], format= g16/de pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas general, display information about object pandas general, displaying all columns pandas general, displaying all rows pandas general, displaying all rows pandas general, importing the module pandas general, using copy_on_write mode pandas general, using qut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group pandas groupby, iterating over groups for t,g in group_by_state: g10/de g10	pandas	dataframe, writing to a CSV file	merged.to_csv('demo-merged.csv')	g13/demo.py
pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour g14/de pandas general, display information about object pandas general, displaying all columns pd.set_option('display.max_columns',None) g16/de pandas general, displaying all rows pd.set_option('display.max_rows', None) g09/de pandas general, importing the module import pandas as pd pd.options.mode.copy_on_write = True g16/de pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state:	pandas	datetime, building via to_datetime()	date = pd.to_datetime(recs['ts'])	g14/demo.py
pandas datetime, extracting day attribute recs['day'] = date.dt.day g14/de pandas datetime, extracting hour attribute recs['hour'] = date.dt.hour g14/de pandas general, display information about object pandas general, displaying all columns pd.set_option('display.max_columns',None) g16/de pandas general, displaying all rows pd.set_option('display.max_rows', None) g09/de pandas general, importing the module import pandas as pd pd.options.mode.copy_on_write = True g16/de pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state:	pandas	datetime, building with a format	ymd = pd.to_datetime(sample['TRANSACTION_DT'], format=	g16/demo.py
pandas general, display information about object sample.info() pandas general, displaying all columns pd.set_option('display.max_columns',None) pandas general, displaying all rows pd.set_option('display.max_rows', None) pandas general, importing the module import pandas as pd pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() pandas groupby, iterating over groups for t,g in group_by_state:	pandas	datetime, extracting day attribute		g14/demo.py
pandas general, displaying all columns pd.set_option('display.max_columns',None) g16/de pandas general, displaying all rows pd.set_option('display.max_rows', None) g09/de pandas general, importing the module import pandas as pd g09/de pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state:	pandas	datetime, extracting hour attribute	recs['hour'] = date.dt.hour	g14/demo.py
pandas general, displaying all rows pd.set_option('display.max_rows', None) pandas general, importing the module import pandas as pd pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() pandas groupby, iterating over groups for t,g in group_by_state: g09/de g09/de g16/de g10/de	pandas	general, display information about object	sample.info()	g16/demo.py
pandas general, displaying all rows pd.set_option('display.max_rows', None) pandas general, importing the module import pandas as pd pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() pandas groupby, iterating over groups for t,g in group_by_state: g10/de	pandas	general, displaying all columns	pd.set_option('display.max_columns',None)	g16/demo.py
pandas general, importing the module import pandas as pd g09/de pandas general, using copy_on_write mode pd.options.mode.copy_on_write = True g16/de pandas general, using qcut to create deciles dec = pd.qcut(county['pop'], 10, labels=range(1,11)) g10/de pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() g10/de pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() g10/de pandas groupby, iterating over groups for t,g in group_by_state: g10/de	•		\cdot	g09/demo.py
pandas general, using qcut to create deciles $dec = pd.qcut(county['pop'], 10, labels=range(1,11))$ $g10/dec$ pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() $g10/dec$ pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() $g10/decc$ pandas groupby, iterating over groups for t,g in group_by_state: $g10/deccc$	pandas	general, importing the module	import pandas as pd	g09/demo.py
pandas general, using qcut to create deciles $dec = pd.qcut(county['pop'], 10, labels=range(1,11))$ $g10/dec$ pandas groupby, cumulative sum within group cumulative_inc = group_by_state['pop'].cumsum() $g10/dec$ pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() $g10/decc$ pandas groupby, iterating over groups for t,g in group_by_state: $g10/deccc$	pandas	general, using copy_on_write mode	pd.options.mode.copy_on_write = True	g16/demo.py
pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() $g10/de$ pandas groupby, iterating over groups for t,g in group_by_state: $g10/de$	•		· · · · · · · · · · · · · · · · · · ·	g10/demo.py
pandas groupby, descriptive statistics inc_stats = group_by_state['pop'].describe() $g10/de$ pandas groupby, iterating over groups for t,g in group_by_state: $g10/de$	pandas	groupby, cumulative sum within group	cumulative_inc = group_by_state['pop'].cumsum()	g10/demo.py
pandas groupby, iterating over groups for t,g in group_by_state: $g10/de$	•			g10/demo.py
	•			g10/demo.py
	-		* * '- '-	g10/demo.py
	-			g10/demo.py
	•			g10/demo.py

pandas groupby, return number within group pandas groupby, return rank within group pandas groupby, select first records first 2 = group_by_state.head(2) pandas groupby, select largest values largest = group_by_state.lead(2) pandas groupby, select largest values last2 = group_by_state.lead(2) pandas groupby, select largest values last2 = group_by_state.lead(2) pandas groupby, size of each group state = county.groupby_state.lead(2) pandas groupby, sum of each group state = county.groupby_state.lead(2) pandas index, creating with 3 levels county = county.groupby_state.jr(pop].sum() pandas index, creating with 3 levels county = county.groupby_state.jr(pop].sum() pandas index, retrieving a row by name pandas index, retrieving as trows by location pandas index, retrieving ast rows by location pandas index, retrieving ast rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot pandas plotting, bar plot pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot pandas plotting, scatter plot h_data.plot.scatter(ax=ax4, =lncome', y='ETR',c='typ h_data.plot.scatter(ax=ax4, =lncome',y='ETR',c='typ h_data.plot.scatter(ax=ax21,x='inc',y='etr',itle='ETR sel.plot.barh(x='Name',y='percent',ax=ax.el.gend=None) pandas reading, csv dat reading, from an open file handle gen_oswego = pd.read_csv('state-data.csv',index_col='na county = pd.read_csv('county_pop.csv',dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains(r''12'')	g10/demo.py g10/demo.py
pandas groupby, select first records pandas groupby, select largest values largest = group_by_state.head(2) pandas groupby, select largest values largest = group_by_state['pop'].nlargest(2) pandas groupby, size of each group num_rows = group_by_state.size() pandas groupby, sum of each group state = county.groupby('state')['pop'].sum() pandas index, creating with 3 levels county = county.set_index(['state','county', 'NAME']) pandas index, renaming values div_pop = div_pop.rename(index=div_names) pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) pandas plotting, scatter colored by 3rd var plotting, scatter colored by 3rd var plotting, scatter plot plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data reading, setting index column state_data = pd_read_csv('state-data.csv') pandas series, RE at start is_LD = trim['Number'].str.contains(r''12")	g10/demo.py
pandas groupby, select largest values pandas groupby, select last records pandas groupby, size of each group pandas groupby, size of each group pandas groupby, sum of each group state = county.groupby('state', 'county', 'NAME']) pandas index, creating with 3 levels county = county.groupby('state', 'county', 'NAME']) pandas index, renaming values pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set('pennsylvania'] print(low_to_high.iloc[0:10]) print(low_to_high.iloc[-5:]) states = raw_states.set('name') pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) plotting, scatter colored by 3rd var plotting, scatter colored by 3rd var plotting, scatter plot plotting, scatter plot plotting, turning off legend sel.pot.barh(x='Name',y='etr',title='ETR plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data raw_states = pd.read_csv('state-data.csv') gendas reading, setting index column reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips)	
pandas groupby, select last records pandas groupby, size of each group pandas groupby, size of each group pandas groupby, size of each group pandas groupby, sum of each group state = county.groupby('state')['pop'].sum() pandas index, creating with 3 levels county = county.set_index(['state','county', 'NAME']) print('\nIndex (rows):', list(raw_states.index)) print('\nIndex (rows):', list(raw_states.index (rows):', list(raw_states.index (rows):', list(raw_sta	g10/demo.py
pandas groupby, size of each group groupby, sum of each group state group groupby, sum of each group state = county.groupby('state')['pop'].sum() pandas index, creating with 3 levels county = county.set_index(['state','county', 'NAME']) pandas index, listing names print('\nlndex (rows):', list(raw_states.index)) pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, setting to a column pandas plotting, bar plot pandas plotting, horizontal bar plot pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot pandas plotting, scatter plot pandas plotting, turning off legend pandas reading, csv data reading, setting index column reading, using dtype dictionary series, RE at start is_LD = trim['Number'].str.contains(r''12")	g10/demo.py
pandas groupby, sum of each group state = county.groupby('state')['pop'].sum() pandas index, creating with 3 levels county = county.set_index(['state','county', 'NAME']) print('\nIndex (rows):', list(raw_states.index)) div_pop = div_pop.rename(index=div_names) print(low_to_high.iloc[0:10]) print(low_to_	g10/demo.py
pandas index, creating with 3 levels pandas index, listing names pandas index, renaming values pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, setting to a column pandas plotting, histogram pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot pandas plotting, turning off legend pandas reading, csv data reading, using dtype dictionary pandas series, RE at start signals index, creating with 3 levels county = county.set_index(['state','county', 'NAME']) print('\nlndex (rows):', list(raw_states.index)) print('\nlndex (rows):', list(raw_states.index)) print('\nlndex (rows):', list(raw_states.index)) pandas county = county.set_index(['state','county', 'NAME']) print('\nlndex (rows):', list(raw_states.index)) print('\nlndex (rows):', list(raw_states.index)) pandas county = palox pop.prename(index=div_names) palox pandas print(low_to_high.iloc[0:10]) pr	g10/demo.py
pandas index, listing names index, renaming values index, renaming values index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set_o['Pennsylvania'] print(low_to_high.iloc[0:10]) print(low_to_high.i	g10/demo.py
pandas index, renaming values div_pop = div_pop.rename(index=div_names) pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot pandas plotting, histogram pandas plotting, horizontal bar plot pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot pandas plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data raw_states = pd.read_csv('state-data.csv') gen_csv'.dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains(r'1 2'')	g10/demo.py
pandas index, retrieving a row by name pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot pandas plotting, histogram pandas plotting, horizontal bar plot pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot pandas plotting, scatter plot pandas plotting, turning off legend pandas reading, csv data reading, from an open file handle pandas reading, using dtype dictionary pandas series, RE at start plot pandas pandas series, RE at start plot pandas pandas pandas pandas series, RE at start plot pandas pandas pandas pandas pandas pandas series, RE at start plot pandas pandas pandas pandas pandas pandas series, RE at start plot pandas pa	g09/demo.py
pandas index, retrieving first rows by location pandas index, retrieving last rows by location pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) plotting, histogram plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) plotting, scatter colored by 3rd var plotting, scatter plot pandas plotting, scatter plot plotting, turning off legend sel.plot.barh(x='Name',y='err,itle='ETR plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data reading, setting index column reading, using dtype dictionary series, RE at start is_LD = trim['Number'].str.contains(r"1 2")	g11/demo.py
pandas index, retrieving last rows by location pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) plotting, histogram hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu pandas plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) plotting, scatter colored by 3rd var plotting, scatter plot hh_data.plot.scatter(ax=ax4,x='Income',y='ETR',c='typ pandas plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data raw_states = pd.read_csv('state-data.csv') pandas reading, setting index column reading, using dtype dictionary county = pd.read_csv('contains(r''1 2'')) pandas series, RE at start is_LD = trim['Number'].str.contains(r''1 2'')	g09/demo.py
pandas index, setting to a column states = raw_states.set_index('name') pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) pandas plotting, histogram hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu pandas plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) pandas plotting, scatter colored by 3rd var plotting, scatter plot plotting, scatter plot plotting, turning off legend sel.plot.barh(x='Name',y='etr',title='ETR	g09/demo.py
pandas plotting, bar plot reg_pop.plot.bar(title='Population',ax=ax1) pandas plotting, histogram hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu pandas plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot hh_data.plot.scatter(ax=ax4,x='Income',y='ETR',c='typ pandas plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data raw_states = pd.read_csv('state-data.csv') pandas reading, from an open file handle gen_oswego = pd.read_csv('state-data.csv',index_col='na pandas reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains(r''1 2'')	g09/demo.py
pandas plotting, histogram hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu pandas plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) pandas plotting, scatter colored by 3rd var pandas plotting, scatter plot plotting, scatter plot plotting, turning off legend plotting, tidy_data.plot.scatter(ax=ax1,bins(legend)) reading, turning off legend plotting, tidy_data.plot.scatter(ax=ax1,bins(legend)) reading, csv data plotting, tidy_data.plot.scatter(ax=ax1,bins(legend)) reading, csv data plotting, tidy_data.plot.scatter(ax=ax1,bins(legend)) reading, csv data plotting, tidy_data.plot.scatter(ax=ax21,x='inc',y='ETR'	g09/demo.py
pandas plotting, histogram hh_data['etr'].plot.hist(ax=ax1,bins=20,title='Distribu pandas plotting, horizontal bar plot div_pop.plot.barh(title='Population',ax=ax2) plotting, scatter colored by 3rd var pandas plotting, scatter plot plotting, scatter plot plotting, turning off legend plotting, turning	g11/demo.py
pandas plotting, scatter colored by 3rd var plotting, scatter plot pandas plotting, scatter plot pandas plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data reading, from an open file handle pandas reading, setting index column reading, using dtype dictionary series, RE at start tidy_data.plot.scatter(ax=ax4,x='lncome',y='ETR',c='typ hh_data.plot.scatter(ax=ax4,x='lncome',y='ETR',c='typ hh_data.plot.scatter(ax=ax21,x='inc',y='etr',title='ETR sel.plot.scatter(ax=ax21,x='inc',y='etr',title='ETR sel.plot.scatter(ax=ax4,x='lncome',y='ETR',c='typ hh_data.plot.scatter(ax=ax4,x='lncome',y='ETR',ca'typ hh_data.plot.scatter(ax=ax21,x='inc',y='etr',title='ETR sel.plot.scatter(ax=ax21,x='inc',y='etr',title='ETR	g12/demo.py
pandas plotting, scatter plot pandas plotting, turning off legend sel.plot.barh(x='Name',y='etr',title='ETR sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data reading, from an open file handle pandas reading, setting index column reading, using dtype dictionary county = pd.read_csv('state-data.csv',index_col='na	g11/demo.py
pandas plotting, turning off legend sel.plot.barh(x='Name',y='percent',ax=ax,legend=None) pandas reading, csv data raw_states = pd.read_csv('state-data.csv') pandas reading, from an open file handle gen_oswego = pd.read_csv(fh1) pandas reading, setting index column state_data = pd.read_csv('state-data.csv',index_col='na pandas reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains(r"1 2")	g12/demo.py
pandas reading, csv data raw_states = pd.read_csv('state-data.csv') pandas reading, from an open file handle gen_oswego = pd.read_csv(fh1) pandas reading, setting index column state_data = pd.read_csv('state-data.csv',index_col='na pandas reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains(r"¹ 2")	g12/demo.py
pandas reading, from an open file handle gen_oswego = pd.read_csv(fh1) state_data = pd.read_csv('state-data.csv',index_col='na pandas reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips)	g13/demo.py
pandas reading, setting index column state_data = pd.read_csv('state-data.csv',index_col='na pandas reading, using dtype dictionary county = pd.read_csv('county_pop.csv',dtype=fips) pandas series, RE at start is_LD = trim['Number'].str.contains($r^{"1}$ 2")	g09/demo.py
pandas reading, using dtype dictionary county = $pd.read_csv(`county_pop.csv',dtype=fips)$ pandas series, RE at start is_LD = $trim[`Number'].str.contains(r''^1 2'')$	g15/demo.py
pandas series, RE at start $is_LD = trim['Number'].str.contains(r"1 2")$	g11/demo.py
·	g10/demo.py
	g12/demo.py
pandas series, automatic alignment by index merged['percent'] = 100*merged['pop']/div_pop	g13/demo.py
pandas series, contains RE or RE $is_TT = trim['Days'].str.contains(r"Tu Th")$	g12/demo.py
pandas series, contains a plain string has_AM = trim['Time'].str.contains("AM")	g12/demo.py
pandas series, contains an RE has_AMPM = trim['Time'].str.contains("AM.*PM")	g12/demo.py
pandas series, converting strings to title case fixname = subset_view['NAME'].str.title()	g16/demo.py
pandas series, converting to a list print(name_data['State'].to_list())	g13/demo.py
pandas series, dropping rows using a list conus = states.drop(not_conus)	g22/demo.py
pandas series, element-by-element or is_either = is_ca is_tx	g16/demo.py

Module	Description	Example	Script
pandas	series, retrieving an element	print("\nFlorida's population:", pop['Florida']/1e6)	g09/demo.py
pandas	series, sort in decending order	div_pop = div_pop.sort_values(ascending=False)	g11/demo.py
pandas	series, sorting by value	<pre>low_to_high = normed['med_pers_inc'].sort_values()</pre>	g09/demo.py
pandas	series, splitting via RE	trim['Split'] = trim["Time"].str.split(r": - ")	g12/demo.py
pandas	series, splitting with expand	exp = trim["Time"].str.split(r": - ", expand=True)	g12/demo.py
pandas	series, summing	$reg_pop = by_reg['pop'].sum()/1e6$	g11/demo.py
pandas	series, unstacking	$tot_wide = tot_amt.unstack('PGI')$	g16/demo.py
pandas	series, using isin()	fixed = flood['TAX_ID'].isin(dup_rec['TAX_ID'])	g14/demo.py
pandas	series, using value_counts()	print('\nOuter:\n', join_o['_merge'].value_counts(), s	g14/demo.py
requests	calling the get() method	response = requests.get(api,payload)	g18/demo.py
requests	checking the URL	print('url:', response.url)	g18/demo.py
requests	checking the response text	print(response.text)	g18/demo.py
requests	checking the status code	<pre>print('status:', response.status_code)</pre>	g18/demo.py
requests	decoding a JSON response	rows = response.json()	g18/demo.py
requests	importing the module	import requests	g18/demo.py
scipy	calling newton's method	$cr = opt.newton(find_cube_root,xinit,maxiter=20,args=[y$	g07/demo.py
scipy	importing the module	import scipy.optimize as opt	g07/demo.py
seaborn	adding a title to a grid object	jg.fig.suptitle('Distribution of Hourly Load')	g17/demo.py
seaborn	barplot	hue='month',palette='deep',ax=ax1)	g17/demo.py
seaborn	basic violin plot	<pre>sns.violinplot(data=janjul,x="month",y="usage")</pre>	g17/demo.py
seaborn	boxenplot	<pre>sns.boxenplot(data=janjul,x="month",y="usage")</pre>	g17/demo.py
seaborn	calling tight_layout on a grid object	jg.fig.tight_layout()	g17/demo.py
seaborn	drawing a heatmapped grid	sns.heatmap(means,annot=True,fmt=".0f",cmap='Spectral',	g20/demo.py
seaborn	importing the module	import seaborn as sns	g17/demo.py
seaborn	joint distribution hex plot	<pre>jg = sns.jointplot(data=bymo,x=1,y=7,kind='hex')</pre>	g17/demo.py
seaborn	setting axis titles on a grid object	jg.set_axis_labels('January','July')	g17/demo.py
seaborn	setting the theme	sns.set_theme(style="white")	g17/demo.py
seaborn	split violin plot	hue="month",palette='deep',split=True)	g17/demo.py
zipfile	importing the module	import zipfile	g15/demo.py
zipfile	opening a file in an archive	fh1 = archive.open('generators-oswego.csv')	g15/demo.py
zipfile	opening an archive	archive = zipfile.ZipFile('generators.zip')	g15/demo.py
zipfile	reading the list of files	<pre>print(archive.namelist())</pre>	g15/demo.py