

index

Symbols

[.loc method 336](#)

[// \(floordiv\) operator 21](#)

[%%timeit command 386](#)

[%timeit magic command 386](#)

[%timeit magic method 386](#)

[~ \(tilde\) character 264](#)

A

[advanced SAT scores,
exercise 204 – 210](#)

[agg method 95, 171, 200, 300](#)

[aggfunc parameter 128 – 129](#)

[aggregation methods 13](#)

[analyzing text 256 – 261](#)

[exercise 257 – 260](#)

[apply method 157, 260](#)

[ascending parameter 163](#)

[ascending sort 163](#)

[assign method 45, 196, 336](#)

[astype 368](#)

[astype method 16, 21](#)

[axis keyword argument 209](#)

B

[best tippers 300 – 306](#)

[exercise 301 – 305](#)

[bestsellers 58 – 60](#)

[big cities exercise 97 – 98](#)

[bins parameter 35](#)

[Bitcoin values 92 – 96](#)

[boolean index 2, 24](#)

[boxplots, weather data 320 – 327](#)

C

[categories 370 – 375](#)

[exercise 371 – 373](#)

[catplot function 363](#)

[celebrity deaths exercise 143 – 148](#)

working through 144 – 147

column selector 47

columns keyword argument 205

components attribute 285

copy method 88

corr method 177, 228, 339

correlations 341 – 357

exercise 342 – 357

count method 64, 74, 136, 170, 259, 330

D

data cleaning

exercise 143 – 148

working through 144 – 147

interpolation 148 – 153

parking data 135 – 143

exercise 136 – 141

data frames 1, 37 – 69

adding products to 53 – 58

overview 54 – 56

bestsellers 58 – 60

interpolation 65 – 66

net revenue 41 – 44

outliers 60 – 64

reading and writing faster
376 – 384

exercise 376 – 380

selective updating 67 – 69

tax planning 44 – 53

updating 67 – 69

data, inconsistent 154 – 157

DataFrame class 3, 41

DataFrameGroupBy objects
170 – 171, 201, 213, 215 – 216

dataframes, pivot tables 126
– 130

date format parameter 292

date_parser keyword argument 293

dates and times 279 – 306

best tippers 300 – 306

exercise 301 – 305

[oil prices](#) 297 – 300

[reading dates](#) 291 – 297

[short, medium, and long taxi rides](#) 285 – 291

[writing dates](#) 291 – 297

[datetime](#) 279

[datetime.datetime class](#) 281

[datetime64 type](#) 282

[days attribute](#) 285

[ddof \(delta degrees of freedom\) parameter](#) 13

[deep=True keyword argument](#) 367

[default rng](#) 6

[descending sort](#) 163

[describe method](#) 27, 201

[descriptive statistics](#) 26 – 29

[df parameter](#) 215

[df variable](#) 2

[df.dropna method](#) 137

[df.dropna\(\) method](#) 88

[df.eval](#) [384, 391](#)

[df.eval method](#) [382](#)

[df.info](#) [method 373](#)

[df.interpolate](#) [method 66](#)

[df.loc](#) [287](#)

[df.loc accessor](#) [384, 391](#)

[df.query](#) [384, 391](#)

[df.query method](#) [382](#)

[dict](#) [data type 14](#)

[diff](#) [method 204](#)

[dot notation](#) [40](#)

[drop_duplicates](#) [method 218](#)

[dropna](#) [method 64, 119](#)

[dt accessor](#) [284, 297, 303](#)

[dtype](#) [20, 87 – 89, 282](#)

[dtype attribute](#) [14, 85](#)

[dtype parameter](#) [15, 85 – 86, 99](#)

[dtype series](#) [32](#)

[dtype values](#) [14](#)

[dtype backend=pyarrow](#)

[keyword argument 375](#)

[dtypes attribute 375](#)

[dtypes method 85, 286](#)

[DtypeWarning 371, 379](#)

E

[Economist 182](#)

[eval method 381, 384 – 391](#)

[exercise 385 – 389](#)

[expanding window 201 – 202](#)

[explode method 263](#)

F

[f-strings 7](#)

[fancy strings 7](#)

[fillna method 21, 64](#)

[filter function 215](#)

[filter method 191, 213, 215](#)

[float data type 14](#)

[float format option 46](#)

[float16 89](#)

[float32](#) 87

[float64](#) 89

[float64 dtype](#) 143

[for loops](#) 173, 324, 326, 378

[format strings](#) 7

[fullname df data frame](#) 185,

[187](#)

G

[g.integers](#) 5

[g.normal function](#) 30

[g.normal method](#) 13, 27

[get method](#) 21

[glob.glob function](#) 197, 378

[grading on a curve](#) 16

[GROUP BY clause](#) 159

[groupby method](#) 170, 172, 212

[groupby object](#) 328

[GroupBy objects](#) 171, 213, 217

[groupby query](#) 290

[grouping](#) 159 – 230

advanced, SAT scores 204 –
215

longest taxi rides 162 – 172

beyond exercise 169 – 170

solution 169

working out 163 – 168

multicity temperatures 194 –
204

exercise 194

taxi ride comparison
exercise 172 – 182

overview 172 – 176

tourist spending per country
182 – 190

exercise 183 – 189

weather data 215 – 222

wine scores and tourism
spending 222 – 229

grouping by time periods 280

H

has multiple readings at least
function 218, 222

head method 10

[head\(1\) item 98](#)

[head\(50\) method 168](#)

[header parameter 32, 93](#)

[hue keyword argument 350](#)

I

[idxmax 300](#)

[idxmin 300](#)

[idxmin method 28](#)

[.iloc accessor 8, 11](#)

importing and exporting data

[Bitcoin values 92 – 96](#)

[JSON \(JavaScript Object
Notation\) 97 – 98](#)

[passwd file to data frame 89
– 92](#)

[setting column types 87 – 89](#)

[taxi rides exercise 73 – 78](#)

[working out 74 – 78](#)

[include lowest=True
keyword argument 35](#)

[inconsistent data 154 – 157](#)

[index attribute 6](#)

[index keyword parameter 7](#)

[Index object 150](#)

[index_col parameter 113,](#)
[117, 149, 298](#)

[indexes 100 – 130](#)

[multi-indexes 112 – 116](#)

[Olympic games 116 – 126](#)

[parking tickets exercise 102](#)
– [112](#)

[pivot tables 126 – 130](#)

[IndexSlice method 122 – 123](#)

[info method 85](#)

[inplace parameter 103](#)

[int data type 14](#)

[INT_REC string 183](#)

[INT-EXP string 183](#)

[INT-EXP value 185](#)

[int64 columns 143](#)

[int8 87 – 88](#)

[integers 5](#)

interpolation 65 – 66, 148 – 153

interval 279

IQR (interquartile range) 60

iqr variable 62

is lexsorted method 116

is monotonic decreasing method 116

is_quarter_end attribute 299

isdigit method 144

isin method 127, 264, 266

isnan function 64

isnull().sum() trick 146

J

join method 185, 343

join operation 342

joining 159 – 230

longest taxi rides 162 – 172

beyond exercise 169 – 170

solution 169

working out 163 – 168

multicity temperatures 194 –
204

exercise 194

wine scores and tourism
spending 222 – 229

JSON (JavaScript Object
Notation) 97 – 98

K

KeyError exception 40

L

labels parameter 35

lambda expression 224

lambda function 214

lambda method 336

left join 182, 228

len method 264

level parameter 122

linear method 66

list data type 14

.loc (accessor) 8 – 9, 11, 38, 74,
113, 122, 298, 336, 386 – 387

locations df data frame 185,
190

low memory parameter 86

low memory=False keyword argument 376

lower method 258, 262

M

map function 215

mask index 2, 11, 18, 24

max function 220

max method 121

mean function 214

mean method 2, 7 – 8, 62, 184, 206, 302

memory usage method 89, 367, 371

method parameter 66

microseconds attribute 285

most common destinations series 153

multi-indexes 101, 112 – 116

Olympic games 116 – 126

[beyond exercise 123 – 126](#)

[exercise 117 – 121](#)

[solution 123](#)

[multicity temperatures 194 – 204](#)

[exercise 194](#)

N

[name attribute 215](#)

[name parameter 205](#)

[names keyword argument 90](#)

[names parameter 149, 321, 342](#)

[NaN \(not a number\) 64, 87 – 89, 148](#)

[nanoseconds attribute 285](#)

[net revenue 41 – 44](#)

[normalization 182](#)

[normalize parameter 33](#)

[np.datetime64 class 281](#)

[np.default_rng 5](#)

[np.float64 15](#)

[np.float64 objects](#) 30

[np.float64 value](#) 343

[np.int64](#) 15

[np.int8](#) 15, 20

[np.max function](#) 214

[np.mean method](#) 171

[np.random.default_rng](#) 6

[np.random.default_rng function](#) 23

[np.random.randint function](#)
19

[np.random.seed function](#) 6

[np.std method](#) 171

[numexpr backend](#) 387

[numexpr library](#) 391

[numexpr package](#) 382

O

[object](#) 87

[object data type](#) 15

[object dtype](#) 374

OECD (Organization for
Economic Cooperation and
Development) 182

oecd_df data frame 223

oecd_tourism_df data frame
223

oil prices 297 – 300

Olympic games

beyond exercise 123 – 126

exercise 117 – 121

solution 123

os.stat function 378

outer joins 182, 228

outliers 60 – 64

P

pandas

final project, problem 392 –
394

grouping 191 – 204

joining 191 – 204

sorting 191 – 204

Pandas Tutor 33, 129, 142,
148, 229, 261, 300, 306

pandemic taxi data 80 – 86

parking data, cleaning 135 –
143

exercise 136 – 141

parse dates 328

parse_dates keyword
argument 291

parse_dates list 286

parse_dates method 359

parse_dates parameter 283

passenger frequency 32 – 34

passenger count 87

passwd file 89 – 92

payment type 87

pct_change method 206 – 209

pct_change window function
203

pd namespace 35, 383

pd variable 2

pd.CategoricalDtype 370

[pd.concat 328](#)

[pd.concat function 54, 57, 301](#)

[pd.concat method 80, 197,
359](#)

[pd.cut method 35, 289](#)

[pd.eval 385](#)

[pd.eval method 383](#)

[pd.from_feather method 375](#)

[pd.NA value 64](#)

[pd.query 385](#)

[pd.read_csv function 32, 74,
205, 301](#)

[pd.read_html function 96](#)

[pd.set_option method 201](#)

[pd.StringDType type 23](#)

[pd.StringDtype type 23](#)

[pd.to_numeric function 147](#)

[pd.to_timedelta function 285](#)

[pd.to_timestamp function
285](#)

[Pearson's correlation
coefficient 339](#)

performance

[categories 370 – 375](#)

[exercise 371 – 373](#)

[eval method 384 – 391](#)

[exercise 385 – 389](#)

[query method 384 – 391](#)

[exercise 385 – 389](#)

[reading and writing faster](#)

[376 – 384](#)

[exercise 376 – 380](#)

[PerformanceWarning 118](#)

[pivot method 125](#)

[pivot tables 101, 124](#)

[plot.bar method 329, 332](#)

[plot.hist method 336](#)

[plot.scatter method 340](#)

[products df data frame 178](#)

[products, adding to data](#)

[frames 53 – 58](#)

[overview 54 – 56](#)

[proportion_of_city_precip](#)
[function 220](#)

[pyarrow engine 381](#)

Q

[quantile method 62](#)

[query method 57, 78, 381, 384,](#)
[386 – 391](#)

[exercise 385 – 389](#)

R

[randint method 5](#)

[random module 5](#)

[random seed 6](#)

[random.randint function 6](#)

[range built-in 8](#)

[read method 257](#)

[read_csv function 32, 89, 92,](#)
[283, 342](#)

[read_csv method 113, 195](#)

[read_excel method 149](#)

[read_json function 97](#)

[read_methods dictionary 379](#)

reading and writing faster

376 – 384

exercise 376 – 380

relplot function 348, 360

replace method 156

resample method 300

resampling 280, 297

right join 228

rolling window function 202

root variable 378

round method 11, 82

row selector 47

S

s.loc['Mon'].mean() 31

s series 262

s variable 2

sales df data frame 178

sample standard deviation

13

SAT scores 112 – 116, 204 – 215

[exercise 204 – 210](#)

[scaling test scores 16 – 19](#)

[scipy.stats.trimboth function](#)

[63](#)

[scipy.stats.zscore function 63](#)

[Seaborn, taxi plots 358 – 364](#)

[beyond exercise 364](#)

[exercise 358 – 363](#)

[seconds attribute 285](#)

[selective updating 67 – 69](#)

[sep keyword argument 90,](#)
[293](#)

Series

[counting tens digits 19 – 26](#)

[descriptive statistics 26 – 29](#)

[scaling test scores 16 – 19](#)

series

[exercise on test scores 5 – 16](#)

[Monday temperatures 29 – 31](#)

[creating series 30 – 31](#)

[passenger frequency 32 – 34](#)

[taxi rides 34 – 36](#)

[Series class 3, 85](#)

[set_index method 103, 113,
118](#)

[setting column types 87 – 89](#)

[SettingWithCopyWarning 64,
68, 88](#)

[shape attribute 136](#)

[shape method 146](#)

[short, medium, and long taxi
rides 285 – 291](#)

[size attribute 136](#)

[size function 125](#)

[slice function 123](#)

[slice method 143, 145](#)

[sns.displot function 356](#)

[sns.relplot function 349](#)

[sort_index 160](#)

[sort_index method 115, 118,
206](#)

[sort_values 160](#)

[sort_values method 185, 209](#)

sorting 159 – 230

longest taxi rides 162 – 172

beyond exercise 169 – 170

solution 169

working out 163 – 168

multicity temperatures 194 –
204

exercise 194

wine scores and tourism
spending 222 – 229

split method 263

split-apply-combine, longest
taxi rides 162 – 172

beyond exercise 169 – 170

solution 169

working out 163 – 168

st_size attribute 378

standard deviation 13

str accessor 21, 143 – 145, 258,
262 – 264, 367

str data type 14

str.contains method 259 –
260

str.format method 46

str.isdigit method 146

str.split method 196, 257

str.strip method 258, 263

string module 258

strings 251 – 278

analyzing text 256 – 261

beyond exercise 261

exercise 257 – 260

wine words exercise 261 –
268

beyond 268

working out 262 – 266

style parameter 351

subset parameter 141, 146

sum function 214

sum method 14, 43, 302, 332

T

T alias 206

tail method 10

target keyword argument

383

tax planning 44 – 53

taxi data, pandemic 80 – 86

taxi fare breakdown 327 –

340

taxi ride comparison

exercise 172 – 182

overview 172 – 176

taxi rides 34 – 36

exercise 73 – 78

short, medium, and long 285

– 291

taxi rides exercise

working out 74 – 78

temperatures 29 – 31

creating series 30 – 31

tens digits, counting 19 – 26

test scores 5 – 16

scaling 16 – 19

text, analyzing 256 – 261

[beyond exercise 261](#)

[exercise 257 – 260](#)

[thresh keyword argument](#)

[141](#)

[thresh parameter 64](#)

[time module 376](#)

[time series 294](#)

[time.perf_counter\(\) function](#)

[376, 378 – 379](#)

[timedelta 279](#)

[timedelta object 285](#)

[timedelta series 284](#)

[timedelta64 objects 284](#)

[timeit 366, 384 – 385](#)

[timeit module 377, 386](#)

[timestamp 279](#)

[Timestamp class 281](#)

[timestamp object 285, 287](#)

[Timestamp objects 286](#)

[Timestamp type 282](#)

[tip_amount 89](#)

Titanic data, interpolation

148 – 153

TLC (Taxi and Limousine

Commission) 73

to csv function 96

to datetime function 281

to datetime step 294

to feather method 375

to frame method 227

top 10 words function 262,
266

total amount 89

tourism df data frame 184 –
185, 189 – 190

tourist spending per country
182 – 190

exercise 183 – 189

tpep pickup datetime date
294

transform method 213, 215,
219

transpose method 206

trip distance 89

[tuple data type 14](#)

[tuple unpacking 196](#)

[TypeError exception 14, 369](#)

U

[updating data frames 67 – 69](#)

[usecols list 286](#)

[usecols parameter 74, 93,
149, 302, 376](#)

V

[value_counts method 33, 105,
120, 257](#)

[ValueError exception 15, 227](#)

[VendorID 89](#)

[visualization 307 – 364](#)

[boxplotting weather data
320 – 327](#)

[correlations 341 – 357](#)

[exercise 342 – 357](#)

[Seaborn taxi plots 358 – 364](#)

[beyond exercise 364](#)

[exercise 358 – 363](#)

[taxi fare breakdown](#) 327 –

[340](#)

W

[weather data](#) 215 – [222](#)

[boxplotting](#) 320 – [327](#)

[window functions](#) 201

[wine scores and tourism](#)

[spending](#) 222 – [229](#)

[wine words exercise](#) 261 –

[268](#)

[beyond](#) 268

[working out](#) 262 – [266](#)

[write methods dictionary](#)

[378](#)

X

[xs method](#) 122