

Data Analysis in Base Python - Introduction



<https://github.com/learn-co-curriculum/dsc-data-analysis-python-intro>



<https://github.com/learn-co-curriculum/dsc-data-analysis-python-intro/issues/new/choose>

Introduction

In this section, you'll continue solidifying your knowledge of Python programming, descriptive statistics, and data visualization by performing key data analysis tasks. You will learn how to load data from CSV and JSON files into Python objects in memory, and wrap up with an analysis that joins together two datasets.

Base Python

When we say "base Python", this means that we are emphasizing the data types and modules that are part of Python itself, rather than using third-party libraries. For much of this course, you will use numerous Python libraries such as `pandas` and `StatsModels`, but first we want to dig into how data analysis tasks can be performed without them.

Python Data Types and Control Structures

In this section, our data is stored in data structures that are built into the Python language: numbers, strings, lists, dictionaries, etc. For example, if we have this table of data:

color	number
green	7
red	2
orange	1

A standard way we might represent that table would be a list of dictionaries, where each dictionary represents a row and has the keys `color` and `number`:

```
info_table = [  
    {"color": "green", "number": 7},  
    {"color": "red", "number": 2},  
    {"color": "orange", "number": 1}  
]
```



Help

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

list

Then if we wanted to print out all of the values associated with the `color` keys, the logic would look something like this:

```
for row in info_table:
    print(row["color"])
```

green red orange

In later sections of this course, that same task (printing the list of colors) might look something like this, using the `pandas` library:

```
import pandas as pd
df = pd.DataFrame(info_table)
type(df)
```

`pandas.core.frame.DataFrame`

```
print(df["color"])
```

0 green 1 red 2 orange Name: color, dtype: object

In the base Python example, we had a variable `info_table` which was type `list` (a built-in Python type), whereas in the pandas example, we had a variable `df` which was type `pandas.core.frame.DataFrame` (a custom type from the pandas library).

Then to print out the colors, in the base Python example we had a `for` loop and then code to extract the information from each individual dictionary. The string values got printed out without any additional markup. Then in the pandas example we did not have to use a loop (hint: this is *broadcasting* like we saw previously with NumPy) and also we had some extra markup where it printed out the index values as well as `Name: color, dtype: object`.

In this section you will practice performing data analysis with the first (base Python) format, before eventually learning how to use pandas instead.

Python Modules

In addition to the data types used, we will also use built-in Python modules. These modules must be imported, but they are part of Python itself and do not require separate installation.

The main modules we will focus on are `csv` ([documentation here](https://docs.python.org/3/library/csv.html)) and `json` ([documentation here](https://docs.python.org/3/library/json.html)).



Help

```
import csv
import json
```

Loading Data from Files

While trivial example data structures like the one above can be declared directly in Python code, more realistic data comes in the form of a file on disk.

In this section you will practice opening data files that use the CSV and JSON formats, then loading them into sensible objects using the `csv` and `json` libraries in order to perform additional analysis.

CSV Files

The first major file type we will explore is CSV (comma-separated value). For example, this code loads a dataset of 5000 apple trees in an orchard, where the column `yield` represents the total pounds of apples yielded by that tree for a given year.

You can think of the table as looking like this, except that it is thousands of lines long:

yield
39.741234289561504
39.872054738763474
44.33116416558151
46.6006230827385
40.694984210927196
40.96981882686812

In relatively few lines of code, we can perform some key data analysis tasks: counting, measuring central tendency, finding the maximum, and finding the minimum.

```
with open("data/apple_orchard.csv") as f:
    apple_orchard_data = list(csv.DictReader(f))
    apple_tree_yields = [float(x["yield"]) for x in apple_orchard_data]
    print("Total apple tree yields:", len(apple_tree_yields))
    print("Average apple tree yield:", sum(apple_tree_yields) / len(apple_tree_yields))
    print("Maximum apple tree yield:", max(apple_tree_yields))
    print("Minimum apple tree yield:", min(apple_tree_yields))
```

Total apple tree yields: 5000 Average apple tree yield: 42.40762385776258 Maximum apple tree yield: 65.54817042071103 Minimum apple tree yield: 21.93121221285836

The second major file type we will explore is JSON (JavaScript object notation). For example, this code loads a dataset of interactions between Twitter users, where each user is represented as a "node" and when one tweets at another that connection is represented as a "link".

You can think of this data as looking like this, although this version is truncated:

```
{
  'nodes': [
    {'id': '347457291'},
    {'id': '232762581'},
    {'id': '23678636'},
    {'id': '21278412'},
    {'id': '222040026'},
    {'id': '19579205'},
    {'id': '222957350'},
    {'id': '264013722'},
    ...
  ],
  'links': [
    {'source': '347457291', 'target': '232762581'},
    {'source': '347457291', 'target': '119706266'},
    {'source': '347457291', 'target': '421509544'},
    {'source': '232762581', 'target': '222957350'},
    {'source': '232762581', 'target': '21648607'},
    {'source': '232762581', 'target': '155691033'},
    {'source': '232762581', 'target': '59974294'},
    ...
  ]
  ...
}
```

In relatively few lines of code, we can count the number of users and find the users who were "sources" (the user initiating a tweet that tags someone else) most often and "targets" (the user being tagged in a tweet) most often.

```
def build_freq_table(key, links):
    table = {}
    for link in links:
        user = link[key]
        table[user] = table.get(user, 0) + 1
    return table
```

```
def print_top_n(table):
```

```
    ? Help in sorted(table.items(), key=lambda x: x[1], reverse=True)[:5]:
        print(f"User {k}\t| {v} Tweets")
```

```
with open("data/twitter_graph.json") as f:
    twitter_data = json.load(f)
    print(f"There are {len(twitter_data['nodes'])} users in this dataset")
    print()

    links = twitter_data["links"]
    sources = build_freq_table("source", links)
    targets = build_freq_table("target", links)

    print("Top 5 Sources:")
    print_top_5(sources)
    print()
    print("Top 5 Targets:")
    print_top_5(targets)
```

There are 99 users in this dataset

Top 5 Sources: User 232762581 | 23 Tweets User 49076695 | 20 Tweets User 523173553 | 19
Tweets User 24883888 | 17 Tweets User 53318310 | 16 Tweets

Top 5 Targets: User 169686021 | 13 Tweets User 23642374 | 12 Tweets User 25797630 | 11 Tweets
User 25626212 | 11 Tweets User 21648607 | 10 Tweets

What About NumPy and Matplotlib?

You can continue using NumPy and Matplotlib as you see fit. For example, if you want to convert a base Python list into a NumPy array because it will help you perform some descriptive analysis task, that is not an issue. You will also need to use Matplotlib in the lessons ahead to create the required visualizations.

The main library that you will *not* be using extensively is **pandas**, which will be introduced in a future section.

Summary


In this section you will learn how to load and manipulate CSV and JSON datasets using base Python. This means that we will mainly be using the data structures and modules built into Python, rather than third-party libraries. We also walked through some examples of the kinds of analysis you will be able to perform by the end of this section.

How do you feel about this lesson?





Have specific feedback?

[Tell us here!](https://github.com/learn-co-curriculum/dsc-data-analysis-python-intro/issues/new/choose)  **<https://github.com/learn-co-curriculum/dsc-data-analysis-python-intro/issues/new/choose>**