KEY_Lesson12_Pandas-Subsetting

July 18, 2019

1 Subsetting Pandas DataFrames

You now know how to read external datasets into pandas. Let's put those skills to use and read in the tips dataset again:

[0]:
```
# mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
path = '/content/gdrive/My Drive/SummerExperience-master/'
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

[0]:
```
# import the pandas package
import pandas as pd
# load tips
tips = pd.read_csv(path + 'SampleData/tips.csv')
```

Take a look again at the beginning of the tips DataFrame:

[0]:
```
# view the beginning of tips
tips.head()
```

[0]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

What if we decided we didn't want to keep all of the data recorded in this dataset? To do that, we need to learn how to subset DataFrames. Subsetting means taking a dataset and pulling out a small portion of it that we're interested in.

First, we'll look at a single column (you can use head to keep the printed result short):

[0]:
```
tips['day'].head(10)
```

[0]:
```
0    Sun
1    Sun
2    Sun
3    Sun
```

```
4    Sun
5    Sun
6    Sun
7    Sun
8    Sun
9    Sun
Name: day, dtype: object
```

We use the square brackets [ ] after the name of the `DataFrame` to tell `pandas` that we want to look at one of the columns. We put the name of the column in quotes to tell `pandas` exactly which column we want to look at. Try subsetting the `total_bill` column:

[0]:
```
# subset the total_bill column
tips['total_bill'].head(10)
```

[0]:
```
0    16.99
1    10.34
2    21.01
3    23.68
4    24.59
5    25.29
6     8.77
7    26.88
8    15.04
9    14.78
Name: total_bill, dtype: float64
```

`pandas` simply showed us the result of subsetting the column, but it didn't save the result anywhere. Try saving the `total_bill` column to a new variable, `bills`:

[0]:
```
# save the total_bill column to a variable
bills = tips['total_bill']
```

We can also pull out multiple columns at a time to create a new `DataFrame`. If we were only interested in the `total_bill` and `tip`, we can subset them like this:

[0]:
```
tips[['total_bill', 'tip']].head(10)
```

[0]:
```
   total_bill   tip
0       16.99  1.01
1       10.34  1.66
2       21.01  3.50
3       23.68  3.31
4       24.59  3.61
5       25.29  4.71
6        8.77  2.00
7       26.88  3.12
8       15.04  1.96
9       14.78  3.23
```

Does that look familiar? Instead of putting a single string between the square brackets, we put a whole list of strings -- you can tell it's a list by the second set of square brackets. Now you try: subset the columns `total_bill`, `tip`, and `time` and save the result to a variable called

tips_subset:

[0]:
```
# subset three columns and save to a new variable
tips_subset = tips[['total_bill', 'tip', 'time']]

# take a look at the beginning of the new DataFrame
tips_subset.head()
```

[0]:
```
   total_bill   tip    time
0       16.99  1.01  Dinner
1       10.34  1.66  Dinner
2       21.01  3.50  Dinner
3       23.68  3.31  Dinner
4       24.59  3.61  Dinner
```

Now we've learned how to subset columns. How do we subset rows? We use a `method` of `DataFrame` called `iloc`. When you see `iloc`, think "index location" -- because we want to get the location where the row is a certain index. Let's try it:

[0]:
```
tips.iloc[1]
```

[0]:
```
total_bill     10.34
tip             1.66
sex             Male
smoker            No
day              Sun
time          Dinner
size               3
Name: 1, dtype: object
```

That showed us the row with an index of 1. Similarly to subsetting columns, we can also subset multiple rows:

[0]:
```
tips.iloc[[0,1,2]]
```

[0]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
```

That gave us a smaller `DataFrame` where the rows have an index of 0, 1, or 2. We can do the same thing with slicing syntax:

[0]:
```
tips.iloc[0:3]
```

[0]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
```

Notice that this does the same thing as calling `head` with a value of 3:

[0]:
```
tips.head(3)
```

[0]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
```

What if we want to grab some rows in the middle of the `DataFrame`? Try subsetting rows 100 through 105:

[0]:
```
# subset rows 100 through 105
tips.iloc[100:106]
```

[0]:
```
     total_bill   tip     sex smoker  day    time  size
100       11.35  2.50  Female    Yes  Fri  Dinner     2
101       15.38  3.00  Female    Yes  Fri  Dinner     2
102       44.30  2.50  Female    Yes  Sat  Dinner     3
103       22.42  3.48  Female    Yes  Sat  Dinner     2
104       20.92  4.08  Female     No  Sat  Dinner     2
105       15.36  1.64    Male    Yes  Sat  Dinner     2
```

We can even subset rows and columns in the same line of code. What do you think the following cell will do?

[0]:
```
tips.iloc[5:10][['total_bill', 'day', 'time']]
```

[0]:
```
    total_bill  day    time
5        25.29  Sun  Dinner
6         8.77  Sun  Dinner
7        26.88  Sun  Dinner
8        15.04  Sun  Dinner
9        14.78  Sun  Dinner
10       10.27  Sun  Dinner
```

Now you try! Subset rows 11 and 12 and columns `total_bill` and `tip`:

[0]:
```
# subset rows and columns
tips.iloc[5:10][['total_bill', 'day', 'time']]
```

[0]:
```
    total_bill  day    time
5        25.29  Sun  Dinner
6         8.77  Sun  Dinner
7        26.88  Sun  Dinner
8        15.04  Sun  Dinner
9        14.78  Sun  Dinner
10       10.27  Sun  Dinner
```

Sometimes we don't know exactly which row(s) we want to subset ahead of time. What if we want to subset rows that have a certain value in the `time` column? We don't want to scroll through hundreds of rows to find them. The good news is: we don't have to! Let's use the `method` called `query`. Inside the parentheses of `query` we'll enclose a statement in quotes with the name of the column and an expression.

[0]:
```
tips.query('time == "Lunch"')
```

[0]:

	total_bill	tip	sex	smoker	day	time	size
77	27.20	4.00	Male	No	Thur	Lunch	4
78	22.76	3.00	Male	No	Thur	Lunch	2
79	17.29	2.71	Male	No	Thur	Lunch	2
80	19.44	3.00	Male	Yes	Thur	Lunch	2
81	16.66	3.40	Male	No	Thur	Lunch	2
82	10.07	1.83	Female	No	Thur	Lunch	1
83	32.68	5.00	Male	Yes	Thur	Lunch	2
84	15.98	2.03	Male	No	Thur	Lunch	2
85	34.83	5.17	Female	No	Thur	Lunch	4
86	13.03	2.00	Male	No	Thur	Lunch	2
87	18.28	4.00	Male	No	Thur	Lunch	2
88	24.71	5.85	Male	No	Thur	Lunch	2
89	21.16	3.00	Male	No	Thur	Lunch	2
117	10.65	1.50	Female	No	Thur	Lunch	2
118	12.43	1.80	Female	No	Thur	Lunch	2
119	24.08	2.92	Female	No	Thur	Lunch	4
120	11.69	2.31	Male	No	Thur	Lunch	2
121	13.42	1.68	Female	No	Thur	Lunch	2
122	14.26	2.50	Male	No	Thur	Lunch	2
123	15.95	2.00	Male	No	Thur	Lunch	2
124	12.48	2.52	Female	No	Thur	Lunch	2
125	29.80	4.20	Female	No	Thur	Lunch	6
126	8.52	1.48	Male	No	Thur	Lunch	2
127	14.52	2.00	Female	No	Thur	Lunch	2
128	11.38	2.00	Female	No	Thur	Lunch	2
129	22.82	2.18	Male	No	Thur	Lunch	3
130	19.08	1.50	Male	No	Thur	Lunch	2
131	20.27	2.83	Female	No	Thur	Lunch	2
132	11.17	1.50	Female	No	Thur	Lunch	2
133	12.26	2.00	Female	No	Thur	Lunch	2
..	...	...	...	...	...	...	...
142	41.19	5.00	Male	No	Thur	Lunch	5
143	27.05	5.00	Female	No	Thur	Lunch	6
144	16.43	2.30	Female	No	Thur	Lunch	2
145	8.35	1.50	Female	No	Thur	Lunch	2
146	18.64	1.36	Female	No	Thur	Lunch	3
147	11.87	1.63	Female	No	Thur	Lunch	2
148	9.78	1.73	Male	No	Thur	Lunch	2
149	7.51	2.00	Male	No	Thur	Lunch	2
191	19.81	4.19	Female	Yes	Thur	Lunch	2
192	28.44	2.56	Male	Yes	Thur	Lunch	2
193	15.48	2.02	Male	Yes	Thur	Lunch	2
194	16.58	4.00	Male	Yes	Thur	Lunch	2
195	7.56	1.44	Male	No	Thur	Lunch	2
196	10.34	2.00	Male	Yes	Thur	Lunch	2
197	43.11	5.00	Female	Yes	Thur	Lunch	4

```
198        13.00  2.00  Female    Yes  Thur  Lunch     2
199        13.51  2.00    Male    Yes  Thur  Lunch     2
200        18.71  4.00    Male    Yes  Thur  Lunch     3
201        12.74  2.01  Female    Yes  Thur  Lunch     2
202        13.00  2.00  Female    Yes  Thur  Lunch     2
203        16.40  2.50  Female    Yes  Thur  Lunch     2
204        20.53  4.00    Male    Yes  Thur  Lunch     4
205        16.47  3.23  Female    Yes  Thur  Lunch     3
220        12.16  2.20    Male    Yes   Fri  Lunch     2
221        13.42  3.48  Female    Yes   Fri  Lunch     2
222         8.58  1.92    Male    Yes   Fri  Lunch     1
223        15.98  3.00  Female     No   Fri  Lunch     3
224        13.42  1.58    Male    Yes   Fri  Lunch     2
225        16.27  2.50  Female    Yes   Fri  Lunch     2
226        10.09  2.00  Female    Yes   Fri  Lunch     2

[68 rows x 7 columns]
```

The above cell showed us all the rows where `time` is equal to "Lunch". We had to enclose "Lunch" in quotes above because it's not the name of a column, but a value within the `time` column.

Now you try: subset the rows where the waitress is female and save it to a variable, `female`:

[0]:
```
# subset rows with a female waitress and save it to a variable
female = tips.query('sex == "Female"')

# take a look at the beginning
female.head()
```

[0]:
```
    total_bill   tip     sex smoker  day    time  size
0        16.99  1.01  Female     No  Sun  Dinner     2
4        24.59  3.61  Female     No  Sun  Dinner     4
11       35.26  5.00  Female     No  Sun  Dinner     4
14       14.83  3.02  Female     No  Sun  Dinner     2
16       10.33  1.67  Female     No  Sun  Dinner     3
```

Now lets do the same for males. Subset the male waiter data and save it to a variable, `male`:

[0]:
```
# subset the male waiters and save it
male = tips.query('sex == "Male"')

# look at the beginning
male.head()
```

[0]:
```
   total_bill   tip   sex smoker  day    time  size
1       10.34  1.66  Male     No  Sun  Dinner     3
2       21.01  3.50  Male     No  Sun  Dinner     3
3       23.68  3.31  Male     No  Sun  Dinner     2
5       25.29  4.71  Male     No  Sun  Dinner     4
6        8.77  2.00  Male     No  Sun  Dinner     2
```

How would you determine the number of male waiters in this `DataFrame`? Think back to the

last lesson when we used the `len` function.

[0]:
```
# number of males
len(male)
```

[0]:
```
157
```

How about the number of female waitreses?

[0]:
```
# number of females
len(female)
```

[0]:
```
87
```

We can use `query` on multiple columns at a time. Let's find out how many tables were served by a female waitress on a Sunday.

[0]:
```
tips.query('sex == "Female" and day == "Sun"')
```

[0]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
11	35.26	5.00	Female	No	Sun	Dinner	4
14	14.83	3.02	Female	No	Sun	Dinner	2
16	10.33	1.67	Female	No	Sun	Dinner	3
18	16.97	3.50	Female	No	Sun	Dinner	3
51	10.29	2.60	Female	No	Sun	Dinner	2
52	34.81	5.20	Female	No	Sun	Dinner	4
114	25.71	4.00	Female	No	Sun	Dinner	3
115	17.31	3.50	Female	No	Sun	Dinner	2
155	29.85	5.14	Female	No	Sun	Dinner	5
157	25.00	3.75	Female	No	Sun	Dinner	4
158	13.39	2.61	Female	No	Sun	Dinner	2
162	16.21	2.00	Female	No	Sun	Dinner	3
164	17.51	3.00	Female	Yes	Sun	Dinner	2
178	9.60	4.00	Female	Yes	Sun	Dinner	2
186	20.90	3.50	Female	Yes	Sun	Dinner	3
188	18.15	3.50	Female	Yes	Sun	Dinner	3

We used the ampersand symbol (&) or the keyword `and` to chain together two statements inside the `query` function. Both statements have to be true for a row to be included.

Besides checking whether values are equal using ==, we can also use greater than, less than, greater than or equal, etc. Try subsetting the rows where the bill is greater than $15 and the tip is less than $2:

[0]:
```
# subset by bill and tip
tips.query('total_bill > 15 & tip < 2')
```

[0]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
8	15.04	1.96	Male	No	Sun	Dinner	2
12	15.42	1.57	Male	No	Sun	Dinner	2
57	26.41	1.50	Female	No	Sat	Dinner	2
105	15.36	1.64	Male	Yes	Sat	Dinner	2

```
130         19.08  1.50    Male     No  Thur   Lunch     2
146         18.64  1.36  Female     No  Thur   Lunch     3
190         15.69  1.50    Male    Yes   Sun  Dinner     2
237         32.83  1.17    Male    Yes   Sat  Dinner     2
242         17.82  1.75    Male     No   Sat  Dinner     2
```

Instead of the ampersand (&) we can use the pipe (|) or the keyword `or` to represent a query where *one* of the two conditions must be fulfilled. Try subsetting where the bill is greater than $15 or the tip is greater than $5:

[0]:
```
# subset by bill or tip
tips.query('total_bill > 15 | tip > 5')
```

[0]:
```
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
5         25.29  4.71    Male     No   Sun  Dinner     4
7         26.88  3.12    Male     No   Sun  Dinner     4
8         15.04  1.96    Male     No   Sun  Dinner     2
11        35.26  5.00  Female     No   Sun  Dinner     4
12        15.42  1.57    Male     No   Sun  Dinner     2
13        18.43  3.00    Male     No   Sun  Dinner     4
15        21.58  3.92    Male     No   Sun  Dinner     2
17        16.29  3.71    Male     No   Sun  Dinner     3
18        16.97  3.50  Female     No   Sun  Dinner     3
19        20.65  3.35    Male     No   Sat  Dinner     3
20        17.92  4.08    Male     No   Sat  Dinner     2
21        20.29  2.75  Female     No   Sat  Dinner     2
22        15.77  2.23  Female     No   Sat  Dinner     2
23        39.42  7.58    Male     No   Sat  Dinner     4
24        19.82  3.18    Male     No   Sat  Dinner     2
25        17.81  2.34    Male     No   Sat  Dinner     4
28        21.70  4.30    Male     No   Sat  Dinner     2
29        19.65  3.00  Female     No   Sat  Dinner     2
31        18.35  2.50    Male     No   Sat  Dinner     4
32        15.06  3.00  Female     No   Sat  Dinner     2
33        20.69  2.45  Female     No   Sat  Dinner     4
34        17.78  3.27    Male     No   Sat  Dinner     2
35        24.06  3.60    Male     No   Sat  Dinner     3
36        16.31  2.00    Male     No   Sat  Dinner     3
37        16.93  3.07  Female     No   Sat  Dinner     3
38        18.69  2.31    Male     No   Sat  Dinner     3
..          ...   ...     ...    ...   ...     ...   ...
193       15.48  2.02    Male    Yes  Thur   Lunch     2
194       16.58  4.00    Male    Yes  Thur   Lunch     2
197       43.11  5.00  Female    Yes  Thur   Lunch     4
200       18.71  4.00    Male    Yes  Thur   Lunch     3
```

```
203       16.40  2.50  Female    Yes  Thur   Lunch     2
204       20.53  4.00    Male    Yes  Thur   Lunch     4
205       16.47  3.23  Female    Yes  Thur   Lunch     3
206       26.59  3.41    Male    Yes   Sat  Dinner     3
207       38.73  3.00    Male    Yes   Sat  Dinner     4
208       24.27  2.03    Male    Yes   Sat  Dinner     2
210       30.06  2.00    Male    Yes   Sat  Dinner     3
211       25.89  5.16    Male    Yes   Sat  Dinner     4
212       48.33  9.00    Male     No   Sat  Dinner     4
214       28.17  6.50  Female    Yes   Sat  Dinner     3
216       28.15  3.00    Male    Yes   Sat  Dinner     5
219       30.14  3.09  Female    Yes   Sat  Dinner     4
223       15.98  3.00  Female     No   Fri   Lunch     3
225       16.27  2.50  Female    Yes   Fri   Lunch     2
227       20.45  3.00    Male     No   Sat  Dinner     4
229       22.12  2.88  Female    Yes   Sat  Dinner     2
230       24.01  2.00    Male    Yes   Sat  Dinner     4
231       15.69  3.00    Male    Yes   Sat  Dinner     3
234       15.53  3.00    Male    Yes   Sat  Dinner     2
237       32.83  1.17    Male    Yes   Sat  Dinner     2
238       35.83  4.67  Female     No   Sat  Dinner     3
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[165 rows x 7 columns]
```

Congrats on making it to the end of this lesson -- we learned a lot!

- How to use square brackets to subset columns.
- How to use `iloc` to subset rows.
- How to use `iloc` and square brackets at the same time.
- How to use `query` to find rows where the column has a certain value.

[0]: