more_pandas

January 7, 2025

1 More Pandas

This notebook supplements pandas_tutorial.ipynb.

[1]:
```
import pandas as pd
```

1.1 Creating a Pandas DataFrame

1.1.1 Different ways of specifying the filename

In the read_csv example in pandas_tutorial.ipynb, we used forward slashes in the filename-pd.read_csv('./data/students.csv'). The following examples also load our dataset.

[2]:
```
# Notice the double backslashes. This is necessary because a single backslash␣
 ↪character is used to
# specify special sequences like \t for a tab character and \n for a new line␣
 ↪character.
students_df = pd.read_csv('.\\data\\students.csv')

students_df
```

[2]:

	Student ID	Name	Age	Subject	Year of Study	\
0	2703f3f0	Mr Clifford Watson	25.0	English Literature	1.0	
1	a8040287	Elliott Ward	25.0	Computer Science	4.0	
2	d8da5486	Miss Pauline Dunn	22.0	Engineering	4.0	
3	3ac1b74d	Mr Dominic Mason	22.0	Physics	1.0	
4	67850858	Mrs Melanie Brown	18.0	English Literature	3.0	
..	…	…	…	…	…	
96	a8be1ec3	Kelly Foster	22.0	Engineering	1.0	
97	3b69ff22	Sara Austin	19.0	Computer Science	34.0	
98	716fb45f	Miss Grace Miller	22.0	English Literature	4.0	
99	34b97db2	Miss Lydia Saunders	23.0	Physics	2.0	
100	34b97db2	Miss Lydia Saunders	23.0	Physics	2.0	

	Country of Origin
0	Saint Barthelemy
1	Guinea
2	Afghanistan
3	Palau

```
4              Algeria
..                 …
96         Netherlands
97       Liechtenstein
98             Comoros
99       Faroe Islands
100      Faroe Islands

[101 rows x 6 columns]
```

[3]:
```
# Here we're using a 'raw' string (notice the r). This removes the need for the double backslash.
students_df = pd.read_csv(r'.\data\students.csv')

students_df
```

[3]:
```
    Student ID                 Name   Age             Subject  Year of Study  \
0     2703f3f0   Mr Clifford Watson  25.0  English Literature            1.0
1     a8040287         Elliott Ward  25.0    Computer Science            4.0
2     d8da5486    Miss Pauline Dunn  22.0         Engineering            4.0
3     3ac1b74d     Mr Dominic Mason  22.0             Physics            1.0
4     67850858    Mrs Melanie Brown  18.0  English Literature            3.0
..         …                    …   …                   …             …
96    a8be1ec3         Kelly Foster  22.0         Engineering            1.0
97    3b69ff22          Sara Austin  19.0    Computer Science           34.0
98    716fb45f    Miss Grace Miller  22.0  English Literature            4.0
99    34b97db2  Miss Lydia Saunders  23.0             Physics            2.0
100   34b97db2  Miss Lydia Saunders  23.0             Physics            2.0

    Country of Origin
0     Saint Barthelemy
1               Guinea
2          Afghanistan
3                Palau
4              Algeria
..                 …
96         Netherlands
97       Liechtenstein
98             Comoros
99       Faroe Islands
100      Faroe Islands

[101 rows x 6 columns]
```

1.1.2 Creating a DataFrame from a Dictionary

You can create a DataFrame by directly passing a dictionary of lists to the pd.DataFrame() constructor, where the keys become column names and the lists become the data.

[4]:
```
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Grade': ['A', 'B', 'C']
}

students_df = pd.DataFrame(data)

students_df
```

[4]:

	Name	Age	Grade
0	Alice	25	A
1	Bob	30	B
2	Charlie	35	C

1.1.3 Creating a DataFrame from Lists of Lists

Another method is by using a list of lists, each representing a row of data. You'll need to specify the column names separately.

[5]:
```
data = [
    ['Alice', 25, 'A'],
    ['Bob', 30, 'B'],
    ['Charlie', 35, 'C']
]

columns = ['Name', 'Age', 'Grade']

students_df = pd.DataFrame(data, columns=columns)

students_df
```

[5]:

	Name	Age	Grade
0	Alice	25	A
1	Bob	30	B
2	Charlie	35	C

1.1.4 Creating a DataFrame from a List of Dictionaries

Each dictionary in the list represents a row in the DataFrame, with keys as column names and values as the data for those columns.

[6]:
```
data = [
    {'Name': 'Alice', 'Age': 25, 'Grade': 'A'},
    {'Name': 'Bob', 'Age': 30, 'Grade': 'B'},
    {'Name': 'Charlie', 'Age': 35, 'Grade': 'C'}
]
```

```
students_df = pd.DataFrame(data)

students_df
```

[6]:

	Name	Age	Grade
0	Alice	25	A
1	Bob	30	B
2	Charlie	35	C

1.1.5 Creating a DataFrame from JSON

You can load a DataFrame from a JSON string directly using pd.read_json().

[7]:
```
json_data = '''
[
    {"Name": "Alice", "Age": 25, "Grade": "A"},
    {"Name": "Bob", "Age": 30, "Grade": "B"},
    {"Name": "Charlie", "Age": 35, "Grade": "C"}
]
'''

students_df = pd.read_json(json_data)

students_df
```

```
C:\Users\jmmck\AppData\Local\Temp\ipykernel_20096\3128921394.py:9:
FutureWarning: Passing literal json to 'read_json' is deprecated and will be
removed in a future version. To read from a literal string, wrap it in a
'StringIO' object.
  students_df = pd.read_json(json_data)
```

[7]:

	Name	Age	Grade
0	Alice	25	A
1	Bob	30	B
2	Charlie	35	C

Here we 'future proof' the JSON example using StringIO:

[8]:
```
from io import StringIO

json_data = '''
[
    {"Name": "Alice", "Age": 25, "Grade": "A"},
    {"Name": "Bob", "Age": 30, "Grade": "B"},
    {"Name": "Charlie", "Age": 35, "Grade": "C"}
]
'''

students_df = pd.read_json(StringIO(json_data))
```

```
students_df
```

[8]:
```
      Name  Age Grade
0    Alice   25     A
1      Bob   30     B
2  Charlie   35     C
```

1.1.6 Creating a DataFrame from a CSV File (with different options)

Besides the basic CSV loading, you can specify additional parameters to handle different data formats and situations.

[9]:
```
# Specifying delimiters: here we're explicitly stating that the delimiter is a␣
 ↪comma.
# This is the default, but some files that we want to load might use tabs,␣
 ↪semicolons, etc.
students_df = pd.read_csv('./data/students.csv',
                          delimiter=',')

students_df
```

[9]:
```
    Student ID                 Name   Age             Subject  Year of Study  \
0     2703f3f0   Mr Clifford Watson  25.0  English Literature            1.0
1     a8040287         Elliott Ward  25.0    Computer Science            4.0
2     d8da5486    Miss Pauline Dunn  22.0         Engineering            4.0
3     3ac1b74d     Mr Dominic Mason  22.0             Physics            1.0
4     67850858    Mrs Melanie Brown  18.0  English Literature            3.0
..         …                    …   …                 …             …
96    a8be1ec3         Kelly Foster  22.0         Engineering            1.0
97    3b69ff22          Sara Austin  19.0    Computer Science           34.0
98    716fb45f    Miss Grace Miller  22.0  English Literature            4.0
99    34b97db2  Miss Lydia Saunders  23.0             Physics            2.0
100   34b97db2  Miss Lydia Saunders  23.0             Physics            2.0

    Country of Origin
0     Saint Barthelemy
1               Guinea
2          Afghanistan
3                Palau
4              Algeria
..                 …
96         Netherlands
97       Liechtenstein
98             Comoros
99       Faroe Islands
100      Faroe Islands
```

```
[101 rows x 6 columns]
```

In the example below, we assume that the first row of the file contains data values rather than headings. This doesn't apply to our original students.csv file, but it could well apply to other files that we want to load.

[10]:
```
# Loading a file without a header row, and specifying column names manually
students_df = pd.read_csv('./data/students_no_headings.csv',
                          header=None,
                          names=['Student ID', 'Name', 'Age', 'Subject', 'Year␣
 ↪of Study', 'Country of Origin'])

students_df
```

[10]:

	Student ID	Name	Age	Subject	Year of Study	\
0	2703f3f0	Mr Clifford Watson	25.0	English Literature	1.0	
1	a8040287	Elliott Ward	25.0	Computer Science	4.0	
2	d8da5486	Miss Pauline Dunn	22.0	Engineering	4.0	
3	3ac1b74d	Mr Dominic Mason	22.0	Physics	1.0	
4	67850858	Mrs Melanie Brown	18.0	English Literature	3.0	
..	…	…	…	…	…	
96	a8be1ec3	Kelly Foster	22.0	Engineering	1.0	
97	3b69ff22	Sara Austin	19.0	Computer Science	34.0	
98	716fb45f	Miss Grace Miller	22.0	English Literature	4.0	
99	34b97db2	Miss Lydia Saunders	23.0	Physics	2.0	
100	34b97db2	Miss Lydia Saunders	23.0	Physics	2.0	

	Country of Origin
0	Saint Barthelemy
1	Guinea
2	Afghanistan
3	Palau
4	Algeria
..	…
96	Netherlands
97	Liechtenstein
98	Comoros
99	Faroe Islands
100	Faroe Islands

```
[101 rows x 6 columns]
```

1.1.7 Creating a DataFrame from a SQL Database

We can also fetch data from a SQL database directly into a DataFrame. The example below connects to a SQLite database, but the read_sql_query function will work with any valid database connection.

For info, I created this SQLite database by saving the Pandas DataFrame using to_sql: students_df.to_sql(name='students', con=connection, index=False)

[11]:
```
import sqlite3
connection = sqlite3.connect('./data/students.db')
students_df = pd.read_sql_query("SELECT * from students", connection)
connection.close()

students_df
```

[11]:
```
    Student ID                 Name   Age             Subject  Year of Study  \
0     2703f3f0   Mr Clifford Watson  25.0  English Literature            1.0
1     a8040287         Elliott Ward  25.0    Computer Science            4.0
2     d8da5486    Miss Pauline Dunn  22.0         Engineering            4.0
3     3ac1b74d     Mr Dominic Mason  22.0             Physics            1.0
4     67850858    Mrs Melanie Brown  18.0  English Literature            3.0
..         …                    …   …                 …            …
96    a8be1ec3         Kelly Foster  22.0         Engineering            1.0
97    3b69ff22          Sara Austin  19.0    Computer Science           34.0
98    716fb45f    Miss Grace Miller  22.0  English Literature            4.0
99    34b97db2  Miss Lydia Saunders  23.0             Physics            2.0
100   34b97db2  Miss Lydia Saunders  23.0             Physics            2.0

    Country of Origin
0    Saint Barthelemy
1              Guinea
2         Afghanistan
3               Palau
4             Algeria
..                …
96        Netherlands
97      Liechtenstein
98             Comoros
99       Faroe Islands
100      Faroe Islands

[101 rows x 6 columns]
```

1.2 Using the index column

We saw in pandas_tutorial.ipynb that we could use index_col to set the index column when loading a DataFrame. We can also set it after the DataFrame has been loaded using set_index.

[12]:
```
students_df = students_df.set_index('Student ID')

students_df
```

[12]:
```
                            Name   Age             Subject  Year of Study  \
Student ID
2703f3f0     Mr Clifford Watson  25.0  English Literature            1.0
a8040287           Elliott Ward  25.0    Computer Science            4.0
d8da5486      Miss Pauline Dunn  22.0         Engineering            4.0
3ac1b74d       Mr Dominic Mason  22.0             Physics            1.0
67850858      Mrs Melanie Brown  18.0  English Literature            3.0
…                           …   …                 …            …
a8be1ec3           Kelly Foster  22.0         Engineering            1.0
3b69ff22            Sara Austin  19.0    Computer Science           34.0
716fb45f      Miss Grace Miller  22.0  English Literature            4.0
34b97db2    Miss Lydia Saunders  23.0             Physics            2.0
34b97db2    Miss Lydia Saunders  23.0             Physics            2.0

            Country of Origin
Student ID
2703f3f0    Saint Barthelemy
a8040287               Guinea
d8da5486          Afghanistan
3ac1b74d                Palau
67850858              Algeria
…                         …
a8be1ec3          Netherlands
3b69ff22        Liechtenstein
716fb45f                Comoros
34b97db2        Faroe Islands
34b97db2        Faroe Islands

[101 rows x 5 columns]
```

We can use the index to access a particular record using loc (which stands for location).

[13]:
```
students_df.loc['2703f3f0']
```

[13]:
```
Name                Mr Clifford Watson
Age                               25.0
Subject              English Literature
Year of Study                       1.0
Country of Origin       Saint Barthelemy
Name: 2703f3f0, dtype: object
```

Of course we really want our index column to contain only unique values, but notice that we get both rows when using the index value of the duplicated row.

[14]:
```
students_df.loc['34b97db2']
```

[14]:
```
                            Name   Age  Subject  Year of Study  \
Student ID
```

```
34b97db2    Miss Lydia Saunders  23.0  Physics            2.0
34b97db2    Miss Lydia Saunders  23.0  Physics            2.0

           Country of Origin
Student ID
34b97db2       Faroe Islands
34b97db2       Faroe Islands
```

If we want to select more than one record, we can use loc in combination with index and isin.

[15]:
```
selected_ids = ['2703f3f0', 'a8040287']
students_df.loc[students_df.index.isin(selected_ids)]
```

[15]:
```
                          Name   Age             Subject  Year of Study  \
Student ID
2703f3f0    Mr Clifford Watson  25.0  English Literature            1.0
a8040287          Elliott Ward  25.0    Computer Science            4.0

           Country of Origin
Student ID
2703f3f0    Saint Barthelemy
a8040287              Guinea
```

We can also use iloc (integer location) to select rows.

[16]:
```
students_df.iloc[0]
```

[16]:
```
Name                 Mr Clifford Watson
Age                                25.0
Subject              English Literature
Year of Study                       1.0
Country of Origin      Saint Barthelemy
Name: 2703f3f0, dtype: object
```

Here we get the first three rows. Notice that that we can use the same slicing syntax that we use for lists.

[17]:
```
students_df.iloc[0:3]
```

[17]:
```
                          Name   Age             Subject  Year of Study  \
Student ID
2703f3f0    Mr Clifford Watson  25.0  English Literature            1.0
a8040287          Elliott Ward  25.0    Computer Science            4.0
d8da5486     Miss Pauline Dunn  22.0         Engineering            4.0

           Country of Origin
Student ID
2703f3f0    Saint Barthelemy
a8040287              Guinea
```

```
d8da5486         Afghanistan
```

1.3 Joining DataFrames

Next we will load a related dataset containing some student grades and join it with the students_df DataFrame.

[18]:
```
grades_df = pd.read_csv('./data/grades.csv', index_col='Student ID')

grades_df
```

[18]:
```
           Grade
Student ID
2703f3f0       A
a8040287       C
d8da5486       A
3ac1b74d       B
67850858       C
62dd3a69       B
6b22a999       A
```

Now we join the two DataFrames where the Student IDs are equal.

[19]:
```
joined_df = students_df.join(grades_df, on='Student ID')

joined_df
```

[19]:
```
                          Name   Age             Subject  Year of Study  \
Student ID
2703f3f0     Mr Clifford Watson  25.0  English Literature            1.0
a8040287           Elliott Ward  25.0    Computer Science            4.0
d8da5486      Miss Pauline Dunn  22.0         Engineering            4.0
3ac1b74d       Mr Dominic Mason  22.0             Physics            1.0
67850858      Mrs Melanie Brown  18.0  English Literature            3.0
…                           …   …                   …            …
a8be1ec3           Kelly Foster  22.0         Engineering            1.0
3b69ff22            Sara Austin  19.0    Computer Science           34.0
716fb45f      Miss Grace Miller  22.0  English Literature            4.0
34b97db2    Miss Lydia Saunders  23.0             Physics            2.0
34b97db2    Miss Lydia Saunders  23.0             Physics            2.0

           Country of Origin Grade
Student ID
2703f3f0    Saint Barthelemy     A
a8040287              Guinea     C
d8da5486         Afghanistan     A
3ac1b74d               Palau     B
67850858             Algeria     C
```

```
…                       …   …
a8be1ec3         Netherlands   NaN
3b69ff22       Liechtenstein   NaN
716fb45f             Comoros   NaN
34b97db2       Faroe Islands   NaN
34b97db2       Faroe Islands   NaN

[101 rows x 6 columns]
```

1.4 Reseting the index

If we have a need to use the index column as a regular column, we can reset it with reset_index.

[20]:
```
students_df = students_df.reset_index()

students_df
```

[20]:
```
    Student ID                 Name   Age             Subject  Year of Study  \
0     2703f3f0   Mr Clifford Watson  25.0  English Literature            1.0
1     a8040287         Elliott Ward  25.0    Computer Science            4.0
2     d8da5486    Miss Pauline Dunn  22.0         Engineering            4.0
3     3ac1b74d     Mr Dominic Mason  22.0             Physics            1.0
4     67850858    Mrs Melanie Brown  18.0  English Literature            3.0
..         …                    …   …                   …            …
96    a8be1ec3         Kelly Foster  22.0         Engineering            1.0
97    3b69ff22          Sara Austin  19.0    Computer Science           34.0
98    716fb45f    Miss Grace Miller  22.0  English Literature            4.0
99    34b97db2  Miss Lydia Saunders  23.0             Physics            2.0
100   34b97db2  Miss Lydia Saunders  23.0             Physics            2.0

    Country of Origin
0    Saint Barthelemy
1              Guinea
2         Afghanistan
3               Palau
4             Algeria
..                …
96        Netherlands
97      Liechtenstein
98            Comoros
99      Faroe Islands
100     Faroe Islands

[101 rows x 6 columns]
```