AN7914 Week 4 Python

March 1, 2023

1 Week 4 Python

1.1 Introduction to Pandas

Pandas is a package built on top of **NumPy**, and provides an efficient implementation of a `DataFrame`. `DataFrame`s are essentially multidimensional *arrays* with attached row and column labels, and often with heterogeneous types and/or missing data. ### Installing Pandas Installation of Pandas on your system requires NumPy to be installed. Details on this installation can be found in the Pandas documentation. Once Pandas is installed, you can import it and check the version:

```
[1]: import pandas
```

```
[2]: pandas.__version__
```

```
[2]: '1.5.2'
```

We will however use an alias to call pandas. So when importing we do the following

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

In the code above we imported `pandas` under the alias `pd`. Now let's check the version again, but this we will use the alias.

```
[4]: pd.__version__
```

```
[4]: '1.5.2'
```

1.2 Creating data

There are two core objects in pandas: the `DataFrame` and the `Series`. ### `DataFrame` A `DataFrame` is a table. It contains an array of individual entries, each of which has a certain value. Each entry corresponds to a *row* (or record) and a *column*.

For example, consider the following simple DataFrame:

```
[5]: pd.DataFrame({'Yup': [50, 21,32], 'Nope': [131, 2,200]})
```

```
[5]:    Yup  Nope
     0   50   131
     1   21     2
```

```
2   32   200
```

In this example, the “0, No” entry has the value of 131. The “0, Yes” entry has a value of 50, and so on.

DataFrame entries are not limited to integers. For instance, here’s a DataFrame whose values are strings:

```
[6]: pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'], 'Sue': ['Pretty good.',␣
       ↪'Bland.']})
```

```
[6]:              Bob           Sue
     0    I liked it.  Pretty good.
     1  It was awful.        Bland.
```

We are using the pd.DataFrame() constructor to generate these DataFrame objects. The syntax for declaring a new one is a **dictionary** whose keys are the *column* names (Bob and Sue in this example), and whose *values* are a list of entries.

The dictionary-list constructor assigns values to the column labels, but just uses an ascending count from 0 (0, 1, 2, 3, ...) for the row labels. Sometimes this is OK, but oftentimes we will want to assign these labels ourselves.

```
[7]: pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'],
                   'Sue': ['Pretty good.', 'Bland.']},
                  index=['Product A', 'Product B'])
```

```
[7]:                      Bob           Sue
     Product A    I liked it.  Pretty good.
     Product B  It was awful.        Bland.
```

1.2.1 Series

A Series, by contrast, is a sequence of data values. If a DataFrame is a table, a Series is a **list**. And in fact you can create one with nothing more than a list:

```
[8]: pd.Series([1, 2, 3, 4, 5])
```

```
[8]: 0    1
     1    2
     2    3
     3    4
     4    5
     dtype: int64
```

A Series is, in essence, a single **column** of a DataFrame. So you can assign row labels to the Series the same way as before, using an index parameter. However, a Series does not have a column name, it only has one overall name:

```
[9]: pd.Series([30, 35, 40], index=['2015 Sales', '2016 Sales', '2017 Sales'],␣
      ↪name='Product A')
```

```
[9]: 2015 Sales    30
     2016 Sales    35
     2017 Sales    40
     Name: Product A, dtype: int64
```

The Series and the DataFrame are intimately related. It's helpful to think of a DataFrame as actually being just a bunch of Series "glued together".

1.3 Importing data sets

You will need to use and import data sets from the internet or from your hard-drive. So if you want to import a **csv** file you will need to use pd.read_csv() command. The argument in the command could be the location where the file is stored in your computer or it could be a file store in the internet as show below.

```
[10]: df_tips = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/
       ↪master/tips.csv')
```

We use pd.read_csv() to read a file stored in 'https://raw.githubusercontent.com/mwaskom/seaborn-data/m
Then we store this dataset as a dataframe in df_tips.

Now lets take a look at the dataset. We can simply type df_tips the name of the dataframe. It is not going to show all columns and rows.

```
[11]: df_tips
```

```
[11]:      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
      ..          …     …       …      …     …       …     …
      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

      [244 rows x 7 columns]
```

If you want to know exactly how many rows and columns the dataframe has we can simply type df_tips.shape

```
[12]: df_tips.shape
```

```
[12]: (244, 7)
```

We see that the output is `(244, 7)`– this means we have 244 rows and 7 columns.

Now let's take a look at the first 15 rows.

```
[13]: df_tips.head(15)
```

```
[13]:     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
      5        25.29  4.71    Male     No  Sun  Dinner     4
      6         8.77  2.00    Male     No  Sun  Dinner     2
      7        26.88  3.12    Male     No  Sun  Dinner     4
      8        15.04  1.96    Male     No  Sun  Dinner     2
      9        14.78  3.23    Male     No  Sun  Dinner     2
      10       10.27  1.71    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
      14       14.83  3.02  Female     No  Sun  Dinner     2
```

`df_tips.head(15)` gives us the first 15 rows of the dataframe. If we typed `df_tips.head(25)` it would show us the first 25 rows.

To see the last 15 rows we can use `df_tips.tail(15)`

```
[14]: df_tips.tail(15)
```

```
[14]:      total_bill   tip     sex smoker   day    time  size
      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
      232       11.61  3.39    Male     No   Sat  Dinner     2
      233       10.77  1.47    Male     No   Sat  Dinner     2
      234       15.53  3.00    Male    Yes   Sat  Dinner     2
      235       10.07  1.25    Male     No   Sat  Dinner     2
      236       12.60  1.00    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
```

If we want to see the names of the columns we use `df_tips.columns`

[15]:

```
df_tips.columns
```

[15]:

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'],
dtype='object')
```

If we want to see a specific columns we can pass in a list – `df_tips[['total_bill','tip']]`. We passed in the list `['total_bill','tip']`. This list contains the list of column names. The output is going to look a `dataframe` and not a `series`.

[16]:

```
df_tips[['total_bill','tip']]
```

[16]:

```
     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
..          …     …
239       29.03  5.92
240       27.18  2.00
241       22.67  2.00
242       17.82  1.75
243       18.78  3.00

[244 rows x 2 columns]
```

[ ]:

```

```