1.1 Introduction to Tabular Data

May 8, 2019

1 Chapter 1. Tables, Observations, and Variables

2 1.1 Introduction to Tabular Data

What does data look like? For most people, the first image that comes to mind is a spreadsheet, with numbers neatly arranged in a table of rows and columns. One goal of this book is to get you to think beyond tables of numbers—to recognize that the words in a book and the markers on a map are also data to be collected, processed, and analyzed. But a lot of data is still organized into tables, so it is important to know how to work with **tabular data**.

Let's look at a tabular data set. Shown below are the first 5 rows of a data set about the passengers on the Titanic. This data set contains information about each passenger (e.g., name, sex, age), their journey (e.g., the fare they paid, their destination), and their ultimate fate (e.g., whether they survived or not, the lifeboat they were on).

In a tabular data set, each row represents a distinct observation and each column a distinct variable. Each **observation** is an entity being measured, and **variables** are the attributes we measure. In the Titanic data set above, each row represents a passenger on the Titanic. For each passenger, 14 variables have been recorded, including `pclass` (their ticket class: 1, 2, or 3) and `boat` (which lifeboat they were on, if they survived).

2.1 Storing Data on Disk and in Memory

How do we represent tabular data on disk so that it can be saved for later or shared with someone else? The Titanic data set above is saved in a file called `titanic.csv`. Let's peek inside this file using the shell command `head`.

Jupyter Tip: To run a shell command inside a Jupyter notebook, simply prefix the shell command by the `!` character.

Jupyter Tip: To run a cell, click on it and press the "play" button in the toolbar above. (Alternatively, you can press `Shift+Enter` on the keyboard.)

```
In [1]: !head /data301/data/titanic.csv
```

```
pclass,survived,name,sex,age,sibsp,parch,ticket,fare,cabin,embarked,boat,body,home.dest
1,1,"Allen, Miss. Elisabeth Walton",female,29,0,0,24160,211.3375,B5,S,2,,"St Louis, MO"
1,1,"Allison, Master. Hudson Trevor",male,0.9167,1,2,113781,151.5500,C22 C26,S,11,,"Montreal, P
1,0,"Allison, Miss. Helen Loraine",female,2,1,2,113781,151.5500,C22 C26,S,,,"Montreal, PQ / Che
1,0,"Allison, Mr. Hudson Joshua Creighton",male,30,1,2,113781,151.5500,C22 C26,S,,135,"Montreal
1,0,"Allison, Mrs. Hudson J C (Bessie Waldo Daniels)",female,25,1,2,113781,151.5500,C22 C26,S,
```

```
1,1,"Anderson, Mr. Harry",male,48,0,0,19952,26.5500,E12,S,3,,"New York, NY"
1,1,"Andrews, Miss. Kornelia Theodosia",female,63,1,0,13502,77.9583,D7,S,10,,"Hudson, NY"
1,0,"Andrews, Mr. Thomas Jr",male,39,0,0,112050,0.0000,A36,S,,,"Belfast, NI"
1,1,"Appleton, Mrs. Edward Dale (Charlotte Lamson)",female,53,2,0,11769,51.4792,C101,S,D,,"Bays
```

The first line of this file contains the names of the variables, separated by commas. Each subsequent line contains the values of those variables for a passenger. The values appear in the same order as the variable names in the first line and are also separated by commas. Because the values in this file are separated (or *delimited*) by commas, this file is called a **comma-separated values** file, or **CSV** for short. CSV files typically have a `.csv` file extension, but not always.

Although commas are by far the most common delimiter, you may encounter tabular data files that use tabs, semicolons (;), or pipes (|) as delimiters.

How do we represent this information in memory so that it can be manipulated efficiently? In Python, the `pandas` library provides a convenient data structure for storing tabular data, called the `DataFrame`.

```
In [2]: import pandas as pd
        pd.DataFrame

Out[2]: pandas.core.frame.DataFrame
```

To read a file from disk into a `pandas` `DataFrame`, we can use the `read_csv` function in `pandas`. The first line of code below reads the Titanic dataset into a `DataFrame` called `df`. The second line calls the `.head()` method of `DataFrame`, which returns a new `DataFrame` consisting of just the first few rows (or "head") of the original.

```
In [4]:
        df.head();
```

Jupyter Tip: When you execute a cell in a Jupyter notebook, the result of the last line is automatically printed. To suppress this output, you can do one of two things:

- Assign the result to a variable, e.g., `df_head = df.head()`.
- Add a semicolon to the end of the line, e.g., `df.head();`.

I encourage you to try these out by modifying the code above and re-running the cell!

Now that the tabular data is in memory as a `DataFrame`, we can manipulate it by writing Python code.

2.2 Observations

Recall that **observations** are the rows in a tabular data set. It is important to think about what each row represents, or the **unit of observation**, before starting a data analysis. In the Titanic `DataFrame`, the unit of observation is a passenger. This makes it easy to answer questions about passengers (e.g., "What percentage of passengers survived?") but harder to answer questions about families (e.g., "What percentage of families had at least one surviving member?")

What if we instead had one row per *family*, instead of one row per *passenger*? We could still store information about *how many* members of each family survived, but this representation would make it difficult to store information about *which* members survived.

There is no single "best" representation of the data. The right representation depends on the question you are trying to answer: if you are studying families on the Titanic, then you might want the unit of observation to be a family, but if you need to know which passengers survived, then you might prefer that it be a passenger. No matter which representation you choose, it is important to be conscious of the unit of observation.

2.2.1 The Row Index

In a `DataFrame`, each observation is identified by an index. You can determine the index of a `DataFrame` by looking for the **bolded** values at the beginning of each row when you print the `DataFrame`. For example, notice how the numbers **0**, **1**, **2**, **3**, **4**, ... above are bolded, which means that this `DataFrame` is indexed by integers starting from 0. This is the default index when you read in a data set from disk into `pandas`, unless you explicitly specify otherwise.

Since each row represents one passenger, it might be useful to re-index the rows by the name of the passenger. To do this, we call the `.set_index()` method of `DataFrame`, passing in the name of the column we want to use as the index. Notice how `name` now appears at the very left, and the passengers' names are all bolded. This is how you know that `name` is the index of this `DataFrame`.

```
In [5]: df.set_index("name").head()
```

```
Out[5]:                                                pclass  survived     sex  \
        name
        Allen, Miss. Elisabeth Walton                       1         1  female
        Allison, Master. Hudson Trevor                      1         1    male
        Allison, Miss. Helen Loraine                        1         0  female
        Allison, Mr. Hudson Joshua Creighton                1         0    male
        Allison, Mrs. Hudson J C (Bessie Waldo Daniels)     1         0  female

                                                           age  sibsp  parch  \
        name
        Allen, Miss. Elisabeth Walton                  29.0000      0      0
        Allison, Master. Hudson Trevor                  0.9167      1      2
        Allison, Miss. Helen Loraine                    2.0000      1      2
        Allison, Mr. Hudson Joshua Creighton           30.0000      1      2
        Allison, Mrs. Hudson J C (Bessie Waldo Daniels) 25.0000      1      2

                                                       ticket      fare    cabin  \
        name
        Allen, Miss. Elisabeth Walton                   24160  211.3375       B5
        Allison, Master. Hudson Trevor                 113781  151.5500  C22 C26
        Allison, Miss. Helen Loraine                   113781  151.5500  C22 C26
        Allison, Mr. Hudson Joshua Creighton           113781  151.5500  C22 C26
        Allison, Mrs. Hudson J C (Bessie Waldo Daniels) 113781  151.5500  C22 C26

                                                      embarked boat   body  \
        name
        Allen, Miss. Elisabeth Walton                        S    2    NaN
        Allison, Master. Hudson Trevor                       S   11    NaN
```

```
Allison, Miss. Helen Loraine                              S  NaN    NaN
Allison, Mr. Hudson Joshua Creighton                      S  NaN  135.0
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)           S  NaN    NaN

                                                                 home.dest
name
Allen, Miss. Elisabeth Walton                                   St Louis, MO
Allison, Master. Hudson Trevor                 Montreal, PQ / Chesterville, ON
Allison, Miss. Helen Loraine                   Montreal, PQ / Chesterville, ON
Allison, Mr. Hudson Joshua Creighton           Montreal, PQ / Chesterville, ON
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  Montreal, PQ / Chesterville, ON
```

Warning: The `.set_index()` method does *not* modify the original `DataFrame`. It returns a *new* `DataFrame` with the specified index. To verify this, let's look at `df` again after running the above code.

```
In [6]: df.head()
```

```
Out[6]:    pclass  survived                                             name     sex  \
        0       1         1                    Allen, Miss. Elisabeth Walton  female
        1       1         1                   Allison, Master. Hudson Trevor    male
        2       1         0                     Allison, Miss. Helen Loraine  female
        3       1         0               Allison, Mr. Hudson Joshua Creighton    male
        4       1         0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  female

              age  sibsp  parch  ticket      fare    cabin embarked boat   body  \
        0  29.0000      0      0   24160  211.3375       B5        S    2    NaN
        1   0.9167      1      2  113781  151.5500  C22 C26        S   11    NaN
        2   2.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN
        3  30.0000      1      2  113781  151.5500  C22 C26        S  NaN  135.0
        4  25.0000      1      2  113781  151.5500  C22 C26        S  NaN    NaN

                              home.dest
        0                    St Louis, MO
        1  Montreal, PQ / Chesterville, ON
        2  Montreal, PQ / Chesterville, ON
        3  Montreal, PQ / Chesterville, ON
        4  Montreal, PQ / Chesterville, ON
```

Nothing has changed! If you want to save the `DataFrame` with the new index, you have to explicitly assign it to a variable.

```
In [7]: df_by_name = df.set_index("name")
        df_by_name.head()
```

```
Out[7]:                                              pclass  survived     sex  \
        name
        Allen, Miss. Elisabeth Walton                     1         1  female
        Allison, Master. Hudson Trevor                    1         1    male
```

```
Allison, Miss. Helen Loraine                          1         0  female
Allison, Mr. Hudson Joshua Creighton                  1         0    male
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)       1         0  female

                                                    age  sibsp  parch  \
name
Allen, Miss. Elisabeth Walton                   29.0000      0      0
Allison, Master. Hudson Trevor                   0.9167      1      2
Allison, Miss. Helen Loraine                     2.0000      1      2
Allison, Mr. Hudson Joshua Creighton            30.0000      1      2
Allison, Mrs. Hudson J C (Bessie Waldo Daniels) 25.0000      1      2

                                                ticket      fare    cabin  \
name
Allen, Miss. Elisabeth Walton                    24160  211.3375       B5
Allison, Master. Hudson Trevor                  113781  151.5500  C22 C26
Allison, Miss. Helen Loraine                    113781  151.5500  C22 C26
Allison, Mr. Hudson Joshua Creighton            113781  151.5500  C22 C26
Allison, Mrs. Hudson J C (Bessie Waldo Daniels) 113781  151.5500  C22 C26

                                               embarked boat   body  \
name
Allen, Miss. Elisabeth Walton                         S    2    NaN
Allison, Master. Hudson Trevor                        S   11    NaN
Allison, Miss. Helen Loraine                          S  NaN    NaN
Allison, Mr. Hudson Joshua Creighton                  S  NaN  135.0
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)       S  NaN    NaN

                                                                home.dest
name
Allen, Miss. Elisabeth Walton                                   St Louis, MO
Allison, Master. Hudson Trevor                 Montreal, PQ / Chesterville, ON
Allison, Miss. Helen Loraine                   Montreal, PQ / Chesterville, ON
Allison, Mr. Hudson Joshua Creighton           Montreal, PQ / Chesterville, ON
Allison, Mrs. Hudson J C (Bessie Waldo Daniels) Montreal, PQ / Chesterville, ON
```

If you do not want the modified `DataFrame` to be stored in a new variable, you can either assign the result back to itself:

`df = df.set_index("name")`

or use the `inplace=True` argument, which will modify the `DataFrame` in place:

`df.set_index("name", inplace=True)`.

These two commands should only be run once. If you try to run them a second time, you will get an error. Don't just take my word for it—create a cell below and try it! The reason for the error is: after the command is executed the first time, `name` is no longer a column in `df`, since it is now in the index. When the command is run again, `pandas` will try (and fail) to find a column called `name`.

Thus, the interactivity of Jupyter notebooks is both a blessing and a curse. It allows us to see the results of our code immediately, but it makes it easy to lose track of the state, especially if you

run a cell twice or out of order. Remember that Jupyter notebooks are designed to be run from beginning to end. Keep this in mind as you run other people's notebooks and as you organize your own notebooks.

```
In [10]: df1 = df.set_index("name")
         df.set_index("name", inplace=True)
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

/opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key
   2524             try:
-> 2525                 return self._engine.get_loc(key)
   2526             except KeyError:


pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get


pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get


KeyError: 'name'


During handling of the above exception, another exception occurred:


KeyError                                  Traceback (most recent call last)

<ipython-input-10-a9809bb1d66b> in <module>()
----> 1 df1 = df.set_index("name")
      2 df.set_index("name", inplace=True)


/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in set_index(self, keys, dr
   3144                 names.append(None)
   3145             else:
-> 3146                 level = frame[col]._values
   3147                 names.append(col)
```

```
   3148                 if drop:


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2137             return self._getitem_multilevel(key)
   2138         else:
-> 2139             return self._getitem_column(key)
   2140
   2141     def _getitem_column(self, key):


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in _getitem_column(self, ke
   2144         # get column
   2145         if self.columns.is_unique:
-> 2146             return self._get_item_cache(key)
   2147
   2148         # duplicate columns & possible reduce dimensionality


    /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in _get_item_cache(self,
   1840         res = cache.get(item)
   1841         if res is None:
-> 1842             values = self._data.get(item)
   1843             res = self._box_item_values(item, values)
   1844             cache[item] = res


    /opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in get(self, item, fast
   3841
   3842             if not isna(item):
-> 3843                 loc = self.items.get_loc(item)
   3844             else:
   3845                 indexer = np.arange(len(self.items))[isna(self.items)]


    /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key
   2525                 return self._engine.get_loc(key)
   2526             except KeyError:
-> 2527                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2528
   2529         indexer = self.get_indexer([key], method=method, tolerance=tolerance)


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get

KeyError: 'name'
```

2.2.2 Selecting Rows

Now that we have set the (row) index of the `DataFrame` to be the passengers' names, we can use the index to select specific passengers. To do this, we use the `.loc` selector. The `.loc` selector takes in a label and returns the row(s) corresponding to that index label.

For example, if we wanted to find the data for the father of the Allison family, we would pass in the label "Allison, Master. Hudson Trevor" to `.loc`. Notice the square brackets.

```
In [11]: df_by_name.loc["Allison, Master. Hudson Trevor"]

Out[11]: pclass                                    1
         survived                                  1
         sex                                    male
         age                                  0.9167
         sibsp                                     1
         parch                                     2
         ticket                               113781
         fare                                 151.55
         cabin                               C22 C26
         embarked                                  S
         boat                                     11
         body                                    NaN
         home.dest    Montreal, PQ / Chesterville, ON
         Name: Allison, Master. Hudson Trevor, dtype: object
```

Notice that the data for a single row is printed differently. This is no accident. If we inspect the type of this data structure:

```
In [12]: type(df_by_name.loc["Allison, Master. Hudson Trevor"])

Out[12]: pandas.core.series.Series
```

we see that it is not a `DataFrame`, but a different data structure called a `Series`.

`.loc` also accepts a *list* of labels, in which case it returns multiple rows, one row for each label in the list. So, for example, if we wanted to select all 4 members of the Allison family from `df_by_name`, we would pass in a list with each of their names.

```
In [13]: df_by_name.loc[[
             "Allison, Master. Hudson Trevor",
```

```
    "Allison, Miss. Helen Loraine",
    "Allison, Mr. Hudson Joshua Creighton",
    "Allison, Mrs. Hudson J C (Bessie Waldo Daniels)"
]]
```

Out[13]:

name	pclass	survived	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
Allison, Master. Hudson Trevor	1	1	male	0.9167	1	2	113781	151.55	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
Allison, Miss. Helen Loraine	1	0	female	2.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
Allison, Mr. Hudson Joshua Creighton	1	0	male	30.0000	1	2	113781	151.55	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	1	0	female	25.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

Notice that when there are multiple rows, the resulting data is stored in a `DataFrame`.

The members of the Allison family happen to be consecutive rows of the `DataFrame`. If you want to select a consecutive set of rows, you do not need to type the index of every row that you want. Instead, you can use **slice notation**. The slice notation `a:b` allows you to select all rows from `a` to `b`. So another way we could have selected all four members of the Allison family is to write:

```
In [14]: df_by_name.loc["Allison, Master. Hudson Trevor":"Allison, Mrs. Hudson J C (Bessie Wal
```

Out[14]:

name	pclass	survived	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
Allison, Master. Hudson Trevor	1	1	male	0.9167	1	2	113781	151.55	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
Allison, Miss. Helen Loraine	1	0	female	2.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
Allison, Mr. Hudson Joshua Creighton	1	0	male	30.0000	1	2	113781	151.55	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	1	0	female	25.0000	1	2	113781	151.55	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON

```
Allison, Master. Hudson Trevor                          1         1    male
Allison, Miss. Helen Loraine                            1         0  female
Allison, Mr. Hudson Joshua Creighton                    1         0    male
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)         1         0  female

                                                      age  sibsp  parch  \
name
Allison, Master. Hudson Trevor                     0.9167      1      2
Allison, Miss. Helen Loraine                       2.0000      1      2
Allison, Mr. Hudson Joshua Creighton              30.0000      1      2
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)   25.0000      1      2

                                                  ticket    fare    cabin  \
name
Allison, Master. Hudson Trevor                    113781  151.55  C22 C26
Allison, Miss. Helen Loraine                      113781  151.55  C22 C26
Allison, Mr. Hudson Joshua Creighton              113781  151.55  C22 C26
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)   113781  151.55  C22 C26

                                                 embarked boat   body  \
name
Allison, Master. Hudson Trevor                          S   11    NaN
Allison, Miss. Helen Loraine                            S  NaN    NaN
Allison, Mr. Hudson Joshua Creighton                    S  NaN  135.0
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)         S  NaN    NaN

                                                                 home.dest
name
Allison, Master. Hudson Trevor                    Montreal, PQ / Chesterville, ON
Allison, Miss. Helen Loraine                      Montreal, PQ / Chesterville, ON
Allison, Mr. Hudson Joshua Creighton              Montreal, PQ / Chesterville, ON
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)   Montreal, PQ / Chesterville, ON
```

This behavior of the slice may be surprising to you if you are a Python veteran. We will say more about this in a second.

What if you wanted to inspect the 100th row of the `DataFrame`, but didn't know the index label for that row? You can use `.iloc` to **select by position** (in contrast to `.loc`, which **selects by label**).

Remember that `pandas` (and Python in general) uses zero-based indexing, so the position index of the 100th row is 99.

```
In [15]: df_by_name.iloc[99]
```

```
Out[15]: pclass                    1
         survived                  1
         sex                  female
         age                      48
         sibsp                     1
         parch                     0
```

```
ticket               11755
fare                  39.6
cabin                  A16
embarked                 C
boat                     1
body                   NaN
home.dest    London / Paris
Name: Duff Gordon, Lady. (Lucille Christiana Sutherland) ("Mrs Morgan"), dtype: object
```

You can also select multiple rows by position, either by passing in a list:

```
In [16]: df_by_name.iloc[[99, 100]]
```

Out[16]:

name	pclass	survived	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
Duff Gordon, Lady. (Lucille Christiana Sutherla...	1	1	female	48.0	1	0	11755	39.6000	A16	C	1	NaN	London / Paris
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")	1	1	male	49.0	1	0	PC 17485	56.9292	A20	C	1	NaN	London / Paris

or by using slice notation:

```
In [17]: df_by_name.iloc[99:101]
```

Out[17]:

name	pclass	survived	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
Duff Gordon, Lady. (Lucille Christiana Sutherla...	1	1	female	48.0	1	0	11755	39.6000	A16	C	1	NaN	London / Paris
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")	1	1	male	49.0	1	0	PC 17485	56.9292	A20	C	1	NaN	London / Paris

```
Duff Gordon, Lady. (Lucille Christiana Sutherla...  48.0      1      0
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")        49.0      1      0

                                                      ticket     fare cabin  \
name
Duff Gordon, Lady. (Lucille Christiana Sutherla...     11755  39.6000   A16
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")        PC 17485  56.9292   A20

                                                    embarked boat  body  \
name
Duff Gordon, Lady. (Lucille Christiana Sutherla...        C    1   NaN
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")              C    1   NaN

                                                         home.dest
name
Duff Gordon, Lady. (Lucille Christiana Sutherla...  London / Paris
Duff Gordon, Sir. Cosmo Edmund ("Mr Morgan")        London / Paris
```

Notice the difference between how slice notation works for `.loc` and `.iloc`.

- `.loc[a:b]` returns the rows from `a` up to `b`, *including* `b`.
- `.iloc[a:b]` returns the rows from `a` up to `b`, *not including* `b`.

So to select the rows in positions 99 and 100, we do `.iloc[99:101]` because we want the rows from position 99 up to 101, *not including 101*. This is consistent with the behavior of slices elsewhere in Python. For example, the slice `1:2` applied to a list returns one element, not two.

```
In [18]: test = ["a", "b", "c", "d"]
         test[1:2]

Out[18]: ['b']
```

2.2.3 What Makes a Good Index?

Something odd happens if we look for "Mr. James Kelly" in this `DataFrame`. Although we only ask for one label, we get two rows back.

```
In [19]: df_by_name.loc["Kelly, Mr. James"]

Out[19]:                  pclass  survived   sex   age  sibsp  parch  ticket    fare  \
         name
         Kelly, Mr. James       3         0  male  34.5      0      0  330911  7.8292
         Kelly, Mr. James       3         0  male  44.0      0      0  363592  8.0500

                         cabin embarked boat  body home.dest
         name
         Kelly, Mr. James   NaN        Q  NaN  70.0       NaN
         Kelly, Mr. James   NaN        S  NaN   NaN       NaN
```

This happened because there were two passengers on the Titanic named "James Kelly". In general, a good row index should uniquely identify observations in the data set. Names are often, but not always, unique. The best row indexes are usually IDs that are guaranteed to be unique.

Another common row index is time. If each row represents a measurement in time, then it makes sense to have the date or the timestamp be the index.

2.3 Variables

Recall that **variables** are the columns in a tabular data set. They are the measurements that we make on each observation.

2.3.1 Selecting Variables

Suppose we want to select the `age` column from the `DataFrame` above. There are three ways to do this.

1. Use `.loc`, specifying both the rows and columns. (*Note:* The colon `:` is Python shorthand for "all".)

```
In [20]: df.loc[:, "age"]
```

```
Out[20]: name
         Allen, Miss. Elisabeth Walton                      29.0000
         Allison, Master. Hudson Trevor                      0.9167
         Allison, Miss. Helen Loraine                        2.0000
         Allison, Mr. Hudson Joshua Creighton               30.0000
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)    25.0000
         Anderson, Mr. Harry                                48.0000
         Andrews, Miss. Kornelia Theodosia                  63.0000
         Andrews, Mr. Thomas Jr                             39.0000
         Appleton, Mrs. Edward Dale (Charlotte Lamson)      53.0000
         Artagaveytia, Mr. Ramon                            71.0000
         Astor, Col. John Jacob                             47.0000
         Astor, Mrs. John Jacob (Madeleine Talmadge Force)  18.0000
         Aubart, Mme. Leontine Pauline                      24.0000
         Barber, Miss. Ellen "Nellie"                       26.0000
         Barkworth, Mr. Algernon Henry Wilson               80.0000
         Baumann, Mr. John D                                    NaN
         Baxter, Mr. Quigg Edmond                           24.0000
         Baxter, Mrs. James (Helene DeLaudeniere Chaput)    50.0000
         Bazzani, Miss. Albina                              32.0000
         Beattie, Mr. Thomson                               36.0000
         Beckwith, Mr. Richard Leonard                      37.0000
         Beckwith, Mrs. Richard Leonard (Sallie Monypeny)   47.0000
         Behr, Mr. Karl Howell                              26.0000
         Bidois, Miss. Rosalie                              42.0000
         Bird, Miss. Ellen                                  29.0000
         Birnbaum, Mr. Jakob                                25.0000
         Bishop, Mr. Dickinson H                            25.0000
```

```
Bishop, Mrs. Dickinson H (Helen Walton)             19.0000
Bissette, Miss. Amelia                              35.0000
Bjornstrom-Steffansson, Mr. Mauritz Hakan           28.0000
                                                       ...
Vestrom, Miss. Hulda Amanda Adolfina                14.0000
Vovk, Mr. Janko                                     22.0000
Waelens, Mr. Achille                                22.0000
Ware, Mr. Frederick                                     NaN
Warren, Mr. Charles William                             NaN
Webber, Mr. James                                       NaN
Wenzel, Mr. Linhart                                 32.5000
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)     38.0000
Widegren, Mr. Carl/Charles Peter                    51.0000
Wiklund, Mr. Jakob Alfred                           18.0000
Wiklund, Mr. Karl Johan                             21.0000
Wilkes, Mrs. James (Ellen Needs)                    47.0000
Willer, Mr. Aaron ("Abi Weller")                        NaN
Willey, Mr. Edward                                      NaN
Williams, Mr. Howard Hugh "Harry"                       NaN
Williams, Mr. Leslie                                28.5000
Windelov, Mr. Einar                                 21.0000
Wirz, Mr. Albert                                    27.0000
Wiseman, Mr. Phillippe                                  NaN
Wittevrongel, Mr. Camille                           36.0000
Yasbeck, Mr. Antoni                                 27.0000
Yasbeck, Mrs. Antoni (Selini Alexander)             15.0000
Youseff, Mr. Gerious                                45.5000
Yousif, Mr. Wazli                                       NaN
Yousseff, Mr. Gerious                                   NaN
Zabour, Miss. Hileni                                14.5000
Zabour, Miss. Thamine                                   NaN
Zakarian, Mr. Mapriededer                           26.5000
Zakarian, Mr. Ortin                                 27.0000
Zimmerman, Mr. Leo                                  29.0000
Name: age, Length: 1309, dtype: float64
```

2. Access the column as you would a key in a `dict`.

```
In [21]: df["age"]
```

```
Out[21]: name
         Allen, Miss. Elisabeth Walton                29.0000
         Allison, Master. Hudson Trevor                0.9167
         Allison, Miss. Helen Loraine                  2.0000
         Allison, Mr. Hudson Joshua Creighton         30.0000
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)  25.0000
         Anderson, Mr. Harry                          48.0000
         Andrews, Miss. Kornelia Theodosia            63.0000
```

Andrews, Mr. Thomas Jr	39.0000
Appleton, Mrs. Edward Dale (Charlotte Lamson)	53.0000
Artagaveytia, Mr. Ramon	71.0000
Astor, Col. John Jacob	47.0000
Astor, Mrs. John Jacob (Madeleine Talmadge Force)	18.0000
Aubart, Mme. Leontine Pauline	24.0000
Barber, Miss. Ellen "Nellie"	26.0000
Barkworth, Mr. Algernon Henry Wilson	80.0000
Baumann, Mr. John D	NaN
Baxter, Mr. Quigg Edmond	24.0000
Baxter, Mrs. James (Helene DeLaudeniere Chaput)	50.0000
Bazzani, Miss. Albina	32.0000
Beattie, Mr. Thomson	36.0000
Beckwith, Mr. Richard Leonard	37.0000
Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	47.0000
Behr, Mr. Karl Howell	26.0000
Bidois, Miss. Rosalie	42.0000
Bird, Miss. Ellen	29.0000
Birnbaum, Mr. Jakob	25.0000
Bishop, Mr. Dickinson H	25.0000
Bishop, Mrs. Dickinson H (Helen Walton)	19.0000
Bissette, Miss. Amelia	35.0000
Bjornstrom-Steffansson, Mr. Mauritz Hakan	28.0000
	...
Vestrom, Miss. Hulda Amanda Adolfina	14.0000
Vovk, Mr. Janko	22.0000
Waelens, Mr. Achille	22.0000
Ware, Mr. Frederick	NaN
Warren, Mr. Charles William	NaN
Webber, Mr. James	NaN
Wenzel, Mr. Linhart	32.5000
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)	38.0000
Widegren, Mr. Carl/Charles Peter	51.0000
Wiklund, Mr. Jakob Alfred	18.0000
Wiklund, Mr. Karl Johan	21.0000
Wilkes, Mrs. James (Ellen Needs)	47.0000
Willer, Mr. Aaron ("Abi Weller")	NaN
Willey, Mr. Edward	NaN
Williams, Mr. Howard Hugh "Harry"	NaN
Williams, Mr. Leslie	28.5000
Windelov, Mr. Einar	21.0000
Wirz, Mr. Albert	27.0000
Wiseman, Mr. Phillippe	NaN
Wittevrongel, Mr. Camille	36.0000
Yasbeck, Mr. Antoni	27.0000
Yasbeck, Mrs. Antoni (Selini Alexander)	15.0000
Youseff, Mr. Gerious	45.5000
Yousif, Mr. Wazli	NaN

```
Youssef, Mr. Gerious                                    NaN
Zabour, Miss. Hileni                                14.5000
Zabour, Miss. Thamine                                   NaN
Zakarian, Mr. Mapriededer                           26.5000
Zakarian, Mr. Ortin                                 27.0000
Zimmerman, Mr. Leo                                  29.0000
Name: age, Length: 1309, dtype: float64
```

3. Access the column as an attribute of the `DataFrame`.

```
In [22]: df.age
```

```
Out[22]: name
         Allen, Miss. Elisabeth Walton                      29.0000
         Allison, Master. Hudson Trevor                      0.9167
         Allison, Miss. Helen Loraine                        2.0000
         Allison, Mr. Hudson Joshua Creighton               30.0000
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)    25.0000
         Anderson, Mr. Harry                                48.0000
         Andrews, Miss. Kornelia Theodosia                  63.0000
         Andrews, Mr. Thomas Jr                             39.0000
         Appleton, Mrs. Edward Dale (Charlotte Lamson)      53.0000
         Artagaveytia, Mr. Ramon                            71.0000
         Astor, Col. John Jacob                             47.0000
         Astor, Mrs. John Jacob (Madeleine Talmadge Force)  18.0000
         Aubart, Mme. Leontine Pauline                      24.0000
         Barber, Miss. Ellen "Nellie"                       26.0000
         Barkworth, Mr. Algernon Henry Wilson               80.0000
         Baumann, Mr. John D                                    NaN
         Baxter, Mr. Quigg Edmond                           24.0000
         Baxter, Mrs. James (Helene DeLaudeniere Chaput)    50.0000
         Bazzani, Miss. Albina                              32.0000
         Beattie, Mr. Thomson                               36.0000
         Beckwith, Mr. Richard Leonard                      37.0000
         Beckwith, Mrs. Richard Leonard (Sallie Monypeny)   47.0000
         Behr, Mr. Karl Howell                              26.0000
         Bidois, Miss. Rosalie                              42.0000
         Bird, Miss. Ellen                                  29.0000
         Birnbaum, Mr. Jakob                                25.0000
         Bishop, Mr. Dickinson H                            25.0000
         Bishop, Mrs. Dickinson H (Helen Walton)            19.0000
         Bissette, Miss. Amelia                             35.0000
         Bjornstrom-Steffansson, Mr. Mauritz Hakan          28.0000
                                                             ...
         Vestrom, Miss. Hulda Amanda Adolfina               14.0000
         Vovk, Mr. Janko                                    22.0000
         Waelens, Mr. Achille                               22.0000
         Ware, Mr. Frederick                                    NaN
```

```
Warren, Mr. Charles William                              NaN
Webber, Mr. James                                        NaN
Wenzel, Mr. Linhart                                  32.5000
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)      38.0000
Widegren, Mr. Carl/Charles Peter                     51.0000
Wiklund, Mr. Jakob Alfred                            18.0000
Wiklund, Mr. Karl Johan                              21.0000
Wilkes, Mrs. James (Ellen Needs)                     47.0000
Willer, Mr. Aaron ("Abi Weller")                         NaN
Willey, Mr. Edward                                       NaN
Williams, Mr. Howard Hugh "Harry"                        NaN
Williams, Mr. Leslie                                 28.5000
Windelov, Mr. Einar                                  21.0000
Wirz, Mr. Albert                                     27.0000
Wiseman, Mr. Phillippe                                   NaN
Wittevrongel, Mr. Camille                            36.0000
Yasbeck, Mr. Antoni                                  27.0000
Yasbeck, Mrs. Antoni (Selini Alexander)              15.0000
Youseff, Mr. Gerious                                 45.5000
Yousif, Mr. Wazli                                        NaN
Yousseff, Mr. Gerious                                    NaN
Zabour, Miss. Hileni                                 14.5000
Zabour, Miss. Thamine                                    NaN
Zakarian, Mr. Mapriededer                            26.5000
Zakarian, Mr. Ortin                                  27.0000
Zimmerman, Mr. Leo                                   29.0000
Name: age, Length: 1309, dtype: float64
```

Method 3 (attribute access) is the most concise. However, it does not work if the variable name contains spaces or special characters, begins with a number, or matches an existing attribute of `DataFrame`. For example, if `df` had a column called `head`, `df.head` would not return the column because `df.head` already means something else, as we have seen.

Notice that the data structure used to store a single column is again a `Series`, not a `DataFrame`. So single rows and columns are stored in `Series`.

To select multiple columns, you would pass in a *list* of variable names, instead of a single variable name. For example, to select both the `age` and `sex` variables, we could do one of the following:

```
In [23]: # METHOD 1
         df.loc[:, ["age", "sex"]].head()

         # METHOD 2
         df[["age", "sex"]].head()
```

```
Out[23]:                                                 age     sex
         name
         Allen, Miss. Elisabeth Walton                29.0000  female
         Allison, Master. Hudson Trevor                0.9167    male
```

```
Allison, Miss. Helen Loraine                        2.0000  female
Allison, Mr. Hudson Joshua Creighton               30.0000    male
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)    25.0000  female
```

Note that there is no way to generalize attribute access (Method 3 above) to select multiple columns.

2.3.2 The Different Types of Variables

There is a fundamental difference between variables like `age` and `fare`, which can be measured on a numeric scale, and variables like `sex` and `home.dest`, which cannot.

Variables that can be measured on a numeric scale are called **quantitative variables**. Just because a variable happens to contain numbers does not necessarily make it "quantitative". For example, consider the variable `survived` in the Titanic data set. Each passenger either survived or didn't. This data set happens to use 1 for "survived" and 0 for "died", but these numbers do not reflect an underlying numeric scale.

Variables that are not quantitative but take on a limited set of values are called **categorical variables**. For example, the variable `sex` takes on one of two possible values ("female" or "male"), so it is a categorical variable. So is the variable `home.dest`, which takes on a larger, but still limited, set of values. We call each possible value of a categorical variable a "category". Although categories are usually non-numeric (as in the case of `sex` and `home.dest`), they are sometimes numeric. For example, the variable `survived` in the Titanic data set is a categorical variable with two categories (1 if the passenger survived, 0 if they didn't), even though those values are numbers. With a categorical variable, one common analysis question is, "How many observations are there in each category?".

Some variables do not fit neatly into either category. For example, the variable `name` in the Titanic data set is obviously not quantitative, but it is not categorical either because it does not take on a limited set of values. Generally speaking, every passenger will have a different name (the two James Kellys notwithstanding), so it does not make sense to analyze the frequencies of different names, as one might do with a categorical variable. We will group variables like `name`, that are neither quantitative nor categorical, into an "other" category.

Every variable can be classified into one of these three **types**: quantitative, categorical, or other. The type of the variable often dictates the kind of analysis we do and the kind of visualizations we make, as we will see later in this chapter.

`pandas` tries to infer the type of each variable automatically. If every value in a column (except for missing values) can be cast to a number, then `pandas` will treat that variable as quantitative. Otherwise, the variable is treated as categorical. To see the type that Pandas inferred for a variable, simply select that variable using the methods above and look for its `dtype`. A `dtype` of `float64` or `int64` indicates that the variable is quantitative. For example, the `age` variable above had a `dtype` of `float64`, so it is quantitative. On the other hand, if we look at the `sex` variable,

```
In [24]: df.sex

Out[24]: name
         Allen, Miss. Elisabeth Walton                    female
         Allison, Master. Hudson Trevor                     male
         Allison, Miss. Helen Loraine                     female
         Allison, Mr. Hudson Joshua Creighton               male
```

```
Allison, Mrs. Hudson J C (Bessie Waldo Daniels)      female
Anderson, Mr. Harry                                    male
Andrews, Miss. Kornelia Theodosia                    female
Andrews, Mr. Thomas Jr                                 male
Appleton, Mrs. Edward Dale (Charlotte Lamson)        female
Artagaveytia, Mr. Ramon                                male
Astor, Col. John Jacob                                 male
Astor, Mrs. John Jacob (Madeleine Talmadge Force)    female
Aubart, Mme. Leontine Pauline                        female
Barber, Miss. Ellen "Nellie"                         female
Barkworth, Mr. Algernon Henry Wilson                   male
Baumann, Mr. John D                                    male
Baxter, Mr. Quigg Edmond                               male
Baxter, Mrs. James (Helene DeLaudeniere Chaput)      female
Bazzani, Miss. Albina                                female
Beattie, Mr. Thomson                                   male
Beckwith, Mr. Richard Leonard                          male
Beckwith, Mrs. Richard Leonard (Sallie Monypeny)     female
Behr, Mr. Karl Howell                                  male
Bidois, Miss. Rosalie                                female
Bird, Miss. Ellen                                    female
Birnbaum, Mr. Jakob                                    male
Bishop, Mr. Dickinson H                                male
Bishop, Mrs. Dickinson H (Helen Walton)              female
Bissette, Miss. Amelia                               female
Bjornstrom-Steffansson, Mr. Mauritz Hakan              male
                                                        ...
Vestrom, Miss. Hulda Amanda Adolfina                 female
Vovk, Mr. Janko                                        male
Waelens, Mr. Achille                                   male
Ware, Mr. Frederick                                    male
Warren, Mr. Charles William                            male
Webber, Mr. James                                      male
Wenzel, Mr. Linhart                                    male
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)      female
Widegren, Mr. Carl/Charles Peter                       male
Wiklund, Mr. Jakob Alfred                              male
Wiklund, Mr. Karl Johan                                male
Wilkes, Mrs. James (Ellen Needs)                     female
Willer, Mr. Aaron ("Abi Weller")                       male
Willey, Mr. Edward                                     male
Williams, Mr. Howard Hugh "Harry"                      male
Williams, Mr. Leslie                                   male
Windelov, Mr. Einar                                    male
Wirz, Mr. Albert                                       male
Wiseman, Mr. Phillippe                                 male
Wittevrongel, Mr. Camille                              male
Yasbeck, Mr. Antoni                                    male
```

```
Yasbeck, Mrs. Antoni (Selini Alexander)              female
Youseff, Mr. Gerious                                   male
Yousif, Mr. Wazli                                      male
Yousseff, Mr. Gerious                                  male
Zabour, Miss. Hileni                                 female
Zabour, Miss. Thamine                                female
Zakarian, Mr. Mapriededer                              male
Zakarian, Mr. Ortin                                    male
Zimmerman, Mr. Leo                                     male
Name: sex, Length: 1309, dtype: object
```

its `dtype` is `object`, so `pandas` will treat it as a categorical variable. Sometimes, this check can yield surprises. For example, if you only looked the first few rows of `df`, you might expect `ticket` to be a quantitative variable. But if we actually look at its `dtype`:

```
In [25]: df.ticket
```

```
Out[25]: name
         Allen, Miss. Elisabeth Walton                        24160
         Allison, Master. Hudson Trevor                      113781
         Allison, Miss. Helen Loraine                        113781
         Allison, Mr. Hudson Joshua Creighton                113781
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)     113781
         Anderson, Mr. Harry                                  19952
         Andrews, Miss. Kornelia Theodosia                    13502
         Andrews, Mr. Thomas Jr                              112050
         Appleton, Mrs. Edward Dale (Charlotte Lamson)        11769
         Artagaveytia, Mr. Ramon                           PC 17609
         Astor, Col. John Jacob                            PC 17757
         Astor, Mrs. John Jacob (Madeleine Talmadge Force) PC 17757
         Aubart, Mme. Leontine Pauline                     PC 17477
         Barber, Miss. Ellen "Nellie"                         19877
         Barkworth, Mr. Algernon Henry Wilson                 27042
         Baumann, Mr. John D                               PC 17318
         Baxter, Mr. Quigg Edmond                          PC 17558
         Baxter, Mrs. James (Helene DeLaudeniere Chaput)   PC 17558
         Bazzani, Miss. Albina                                11813
         Beattie, Mr. Thomson                                 13050
         Beckwith, Mr. Richard Leonard                        11751
         Beckwith, Mrs. Richard Leonard (Sallie Monypeny)     11751
         Behr, Mr. Karl Howell                               111369
         Bidois, Miss. Rosalie                             PC 17757
         Bird, Miss. Ellen                                 PC 17483
         Birnbaum, Mr. Jakob                                  13905
         Bishop, Mr. Dickinson H                              11967
         Bishop, Mrs. Dickinson H (Helen Walton)              11967
         Bissette, Miss. Amelia                            PC 17760
         Bjornstrom-Steffansson, Mr. Mauritz Hakan           110564
```

```
                                                      ...
Vestrom, Miss. Hulda Amanda Adolfina                  350406
Vovk, Mr. Janko                                       349252
Waelens, Mr. Achille                                  345767
Ware, Mr. Frederick                                   359309
Warren, Mr. Charles William                         C.A. 49867
Webber, Mr. James                                SOTON/OQ 3101316
Wenzel, Mr. Linhart                                   345775
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)         2688
Widegren, Mr. Carl/Charles Peter                      347064
Wiklund, Mr. Jakob Alfred                            3101267
Wiklund, Mr. Karl Johan                              3101266
Wilkes, Mrs. James (Ellen Needs)                      363272
Willer, Mr. Aaron ("Abi Weller")                        3410
Willey, Mr. Edward                                S.O./P.P. 751
Williams, Mr. Howard Hugh "Harry"                     A/5 2466
Williams, Mr. Leslie                                    54636
Windelov, Mr. Einar                              SOTON/OQ 3101317
Wirz, Mr. Albert                                      315154
Wiseman, Mr. Phillippe                              A/4. 34244
Wittevrongel, Mr. Camille                             345771
Yasbeck, Mr. Antoni                                     2659
Yasbeck, Mrs. Antoni (Selini Alexander)                 2659
Youseff, Mr. Gerious                                    2628
Yousif, Mr. Wazli                                       2647
Yousseff, Mr. Gerious                                   2627
Zabour, Miss. Hileni                                    2665
Zabour, Miss. Thamine                                   2665
Zakarian, Mr. Mapriededer                               2656
Zakarian, Mr. Ortin                                     2670
Zimmerman, Mr. Leo                                    315082
Name: ticket, Length: 1309, dtype: object
```

it appears to be an `object`. That is because there are some values in this column that contain non-numeric characters. For example:

```
In [26]: df.ticket[9]

Out[26]: 'PC 17609'
```

As long as there is one value in the column that cannot be cast to a numeric type, the entire column will be treated as categorical, and the individual values will be strings (notice the quotes around even a number like 24160, indicating that `pandas` is treating it as a string).

```
In [27]: df.ticket[0]

Out[27]: '24160'
```

If you wanted `pandas` to treat this variable as quantitative, you can use the `to_numeric()` function. However, you have to specify what to do for values like `'PC 17609'` that cannot be

converted to a number. The `errors="coerce"` option tells pandas to treat these values as missing (`NaN`).

```
In [28]: pd.to_numeric(df.ticket, errors="coerce")
```

```
Out[28]: name
         Allen, Miss. Elisabeth Walton                          24160.0
         Allison, Master. Hudson Trevor                        113781.0
         Allison, Miss. Helen Loraine                          113781.0
         Allison, Mr. Hudson Joshua Creighton                  113781.0
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)       113781.0
         Anderson, Mr. Harry                                    19952.0
         Andrews, Miss. Kornelia Theodosia                      13502.0
         Andrews, Mr. Thomas Jr                                112050.0
         Appleton, Mrs. Edward Dale (Charlotte Lamson)          11769.0
         Artagaveytia, Mr. Ramon                                    NaN
         Astor, Col. John Jacob                                     NaN
         Astor, Mrs. John Jacob (Madeleine Talmadge Force)          NaN
         Aubart, Mme. Leontine Pauline                              NaN
         Barber, Miss. Ellen "Nellie"                           19877.0
         Barkworth, Mr. Algernon Henry Wilson                   27042.0
         Baumann, Mr. John D                                        NaN
         Baxter, Mr. Quigg Edmond                                   NaN
         Baxter, Mrs. James (Helene DeLaudeniere Chaput)            NaN
         Bazzani, Miss. Albina                                  11813.0
         Beattie, Mr. Thomson                                   13050.0
         Beckwith, Mr. Richard Leonard                          11751.0
         Beckwith, Mrs. Richard Leonard (Sallie Monypeny)       11751.0
         Behr, Mr. Karl Howell                                 111369.0
         Bidois, Miss. Rosalie                                      NaN
         Bird, Miss. Ellen                                          NaN
         Birnbaum, Mr. Jakob                                    13905.0
         Bishop, Mr. Dickinson H                                11967.0
         Bishop, Mrs. Dickinson H (Helen Walton)                11967.0
         Bissette, Miss. Amelia                                     NaN
         Bjornstrom-Steffansson, Mr. Mauritz Hakan             110564.0
                                                                  ...
         Vestrom, Miss. Hulda Amanda Adolfina                  350406.0
         Vovk, Mr. Janko                                       349252.0
         Waelens, Mr. Achille                                  345767.0
         Ware, Mr. Frederick                                   359309.0
         Warren, Mr. Charles William                                NaN
         Webber, Mr. James                                          NaN
         Wenzel, Mr. Linhart                                   345775.0
         Whabee, Mrs. George Joseph (Shawneene Abi-Saab)         2688.0
         Widegren, Mr. Carl/Charles Peter                      347064.0
         Wiklund, Mr. Jakob Alfred                            3101267.0
         Wiklund, Mr. Karl Johan                              3101266.0
```

```
Wilkes, Mrs. James (Ellen Needs)                    363272.0
Willer, Mr. Aaron ("Abi Weller")                      3410.0
Willey, Mr. Edward                                       NaN
Williams, Mr. Howard Hugh "Harry"                        NaN
Williams, Mr. Leslie                                 54636.0
Windelov, Mr. Einar                                      NaN
Wirz, Mr. Albert                                    315154.0
Wiseman, Mr. Phillippe                                   NaN
Wittevrongel, Mr. Camille                           345771.0
Yasbeck, Mr. Antoni                                   2659.0
Yasbeck, Mrs. Antoni (Selini Alexander)               2659.0
Youseff, Mr. Gerious                                  2628.0
Yousif, Mr. Wazli                                     2647.0
Yousseff, Mr. Gerious                                 2627.0
Zabour, Miss. Hileni                                  2665.0
Zabour, Miss. Thamine                                 2665.0
Zakarian, Mr. Mapriededer                             2656.0
Zakarian, Mr. Ortin                                   2670.0
Zimmerman, Mr. Leo                                  315082.0
Name: ticket, Length: 1309, dtype: float64
```

If we wanted to keep this change, we would assign this column back to the original `DataFrame`, as follows:

`df.ticket = pd.to_numeric(df.ticket, errors="coerce")`.

But since `ticket` does not appear to be a quantitative variable, this is not actually a change we want to make.

There are also categorical variables that `pandas` infers as quantitative because the values happen to be numbers. As we discussed earlier, the `survived` variable is categorical, but the values happen to be coded as 1 or 0. To force `pandas` to treat this as a categorical variable, you can cast the values to strings. Notice how the `dtype` changes:

```
In [29]: df.survived.astype(str)

Out[29]: name
         Allen, Miss. Elisabeth Walton                      1
         Allison, Master. Hudson Trevor                     1
         Allison, Miss. Helen Loraine                       0
         Allison, Mr. Hudson Joshua Creighton               0
         Allison, Mrs. Hudson J C (Bessie Waldo Daniels)    0
         Anderson, Mr. Harry                                1
         Andrews, Miss. Kornelia Theodosia                  1
         Andrews, Mr. Thomas Jr                             0
         Appleton, Mrs. Edward Dale (Charlotte Lamson)      1
         Artagaveytia, Mr. Ramon                            0
         Astor, Col. John Jacob                             0
         Astor, Mrs. John Jacob (Madeleine Talmadge Force)  1
         Aubart, Mme. Leontine Pauline                      1
         Barber, Miss. Ellen "Nellie"                       1
```

```
Barkworth, Mr. Algernon Henry Wilson                1
Baumann, Mr. John D                                 0
Baxter, Mr. Quigg Edmond                            0
Baxter, Mrs. James (Helene DeLaudeniere Chaput)     1
Bazzani, Miss. Albina                               1
Beattie, Mr. Thomson                                0
Beckwith, Mr. Richard Leonard                       1
Beckwith, Mrs. Richard Leonard (Sallie Monypeny)    1
Behr, Mr. Karl Howell                               1
Bidois, Miss. Rosalie                               1
Bird, Miss. Ellen                                   1
Birnbaum, Mr. Jakob                                 0
Bishop, Mr. Dickinson H                             1
Bishop, Mrs. Dickinson H (Helen Walton)             1
Bissette, Miss. Amelia                              1
Bjornstrom-Steffansson, Mr. Mauritz Hakan           1
                                                   ..
Vestrom, Miss. Hulda Amanda Adolfina                0
Vovk, Mr. Janko                                     0
Waelens, Mr. Achille                                0
Ware, Mr. Frederick                                 0
Warren, Mr. Charles William                         0
Webber, Mr. James                                   0
Wenzel, Mr. Linhart                                 0
Whabee, Mrs. George Joseph (Shawneene Abi-Saab)     1
Widegren, Mr. Carl/Charles Peter                    0
Wiklund, Mr. Jakob Alfred                           0
Wiklund, Mr. Karl Johan                             0
Wilkes, Mrs. James (Ellen Needs)                    1
Willer, Mr. Aaron ("Abi Weller")                    0
Willey, Mr. Edward                                  0
Williams, Mr. Howard Hugh "Harry"                   0
Williams, Mr. Leslie                                0
Windelov, Mr. Einar                                 0
Wirz, Mr. Albert                                    0
Wiseman, Mr. Phillippe                              0
Wittevrongel, Mr. Camille                           0
Yasbeck, Mr. Antoni                                 0
Yasbeck, Mrs. Antoni (Selini Alexander)             1
Youseff, Mr. Gerious                                0
Yousif, Mr. Wazli                                   0
Yousseff, Mr. Gerious                               0
Zabour, Miss. Hileni                                0
Zabour, Miss. Thamine                               0
Zakarian, Mr. Mapriededer                           0
Zakarian, Mr. Ortin                                 0
Zimmerman, Mr. Leo                                  0
Name: survived, Length: 1309, dtype: object
```

In this case, this is a change that we actually want to keep, so we assign the modified column back to the `DataFrame`.

```
In [30]: df.survived = df.survived.astype(str)
```

2.4 Summary

- Tabular data is stored in a data structure called a `DataFrame`.
- Rows represent observations; columns represent variables.
- Single rows and columns are stored in a data structure called a `Series`.
- The row index should be a set of labels that uniquely identify observations.
- To select rows by label, we use `.loc[]`. To select rows by (0-based) position, we use `.iloc[]`.
- To select columns, we can use `.loc` notation (specifying both the rows and columns we want, separated by a comma), key access, or attribute access.
- Variables can be quantitative, categorical, or other.
- Pandas will try to infer the type, and you can check the type that Pandas inferred by looking at the `dtype`.

3 Exercises

Exercise 1. Consider the variable `pclass` in the Titanic data set, which is 1, 2, or 3, depending on whether the passenger was in 1st, 2nd, or 3rd class.

- What type of variable is this: quantitative, categorical, or other? (*Hint:* One useful test is to ask yourself, "Does it make sense to add up values of this variable?" If the variable can be measured on a numeric scale, then it should make sense to add up values of that variable.)
- Did `pandas` correctly infer the type of this variable? If not, convert this variable to the appropriate type.

```
In [35]: #The pclass variable is categorical
         df.pclass
         #pandas did not correctly infer the type of this variable
         df.pclass = df.pclass.astype(str)
```

Exercises 2-7 deal with the Tips data set (`/data301/data/tips.csv`). You can learn more about this data set on the first page of this reference.

Exercise 2. Read in the Tips data set into a `pandas` `DataFrame` called `tips`.

- What is the unit of observation in this data set?
- For each variable in the data set, identify it as quantitative, categorical, or other, based on your understanding of each variable. Did `pandas` correctly infer the type of each variable?

```
In [54]: tips = pd.read_csv("/data301/data/tips.csv")
         tips.head()
         #total_bill - quantitative
         tips.total_bill # quantitative
         #tip - quantitative
         tips.tip #quantitative
```

```
#sex - categorical
tips.sex #categorical
#smoker - categorical
tips.smoker #categorical
#day - categorical
tips.day #categorical
#time - categorical
tips.time #categorical
#size - categorical
tips.size = tips.size.astype(str)
```

```
In [76]: tips.dtypes
```

```
Out[76]: total_bill    float64
         tip           float64
         sex            object
         smoker         object
         day            object
         time           object
         size           object
         dtype: object
```

```
In [78]: for colname in tips.columns:
             print(colname, tips[colname].dtype)
```

```
total_bill float64
tip float64
sex object
smoker object
day object
time object
size object
```

Exercise 3. Make the day of the week the index of the `DataFrame`.

- What do you think will happen when you call `tips.loc["Thur"]`? Try it. What happens?
- Is this a good variable to use as the index? Explain why or why not.

```
In [81]: tips.set_index("day").head()
         tips.set_index("day").loc["Thur"].head()
         #This is a good variable to use as the index in specific cases, however we usually wa
         #index to be unique
```

```
Out[81]:       total_bill   tip   sex smoker   time  size
         day
         Thur       27.20  4.00  Male     No  Lunch  1708
         Thur       22.76  3.00  Male     No  Lunch  1708
         Thur       17.29  2.71  Male     No  Lunch  1708
         Thur       19.44  3.00  Male    Yes  Lunch  1708
         Thur       16.66  3.40  Male     No  Lunch  1708
```

Exercise 4. Make sure the index of the `DataFrame` is the default (i.e., 0, 1, 2, ...). If you changed it away from the default in the previous exercise, you can use `.reset_index()` to reset it.

- How do you think `tips.loc[50]` and `tips.iloc[50]` will compare? Now try it. Was your prediction correct?
- How do you think `tips.loc[50:55]` and `tips.iloc[50:55]` will compare? Now try it. Was your prediction correct?

```
In [86]: tips.reset_index()
         tips.loc[50]
         tips.iloc[50]

         tips.loc[50:55]
         tips.iloc[50:55]
```

```
Out[86]:     total_bill   tip     sex smoker  day    time  size
         50       12.54  2.50    Male     No  Sun  Dinner  1708
         51       10.29  2.60  Female     No  Sun  Dinner  1708
         52       34.81  5.20  Female     No  Sun  Dinner  1708
         53        9.94  1.56    Male     No  Sun  Dinner  1708
         54       25.56  4.34    Male     No  Sun  Dinner  1708
```

Exercise 5. How do you think `tips.loc[50]` and `tips.loc[[50]]` will compare? Now try it. Was your prediction correct?

```
In [92]: tips.loc[50] #returns as series
         tips.loc[[50]] #returns as data frame because we are passing in a list because python
                        #expecting the list to be greater than 1
```

```
Out[92]:     total_bill  tip   sex smoker  day    time  size
         50       12.54  2.5  Male     No  Sun  Dinner  1708
```

Exercise 6. What data structure is used to represent a single column, such as `tips["total_bill"]`? How could you modify this code to obtain a `DataFrame` consisting of just one column, `total_bill`?

```
In [93]: tips["total_bill"]
         tips.loc[:, "total_bill"]
         tips[["total_bill"]]
```

```
Out[93]:      total_bill
         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
10       10.27
11       35.26
12       15.42
13       18.43
14       14.83
15       21.58
16       10.33
17       16.29
18       16.97
19       20.65
20       17.92
21       20.29
22       15.77
23       39.42
24       19.82
25       17.81
26       13.37
27       12.69
28       21.70
29       19.65
..         ...
214      28.17
215      12.90
216      28.15
217      11.59
218       7.74
219      30.14
220      12.16
221      13.42
222       8.58
223      15.98
224      13.42
225      16.27
226      10.09
227      20.45
228      13.28
229      22.12
230      24.01
231      15.69
232      11.61
233      10.77
234      15.53
235      10.07
236      12.60
237      32.83
238      35.83
```

```
239       29.03
240       27.18
241       22.67
242       17.82
243       18.78

[244 rows x 1 columns]
```

Exercise 7. Create a new `DataFrame` from the Tips data that consists of just information about the table (i.e., whether or not there was a smoker, the day and time they visited the restaurant, and the size of the party), without information about the check or who paid.

(There are many ways to do this. How many ways can you find?)

```
In [74]: tips.loc[:,["smoker", "day", "time", "size"]] #have to have the colon then comma to i
         #row and column
```

```
Out[74]:     smoker   day    time  size
         0       No   Sun  Dinner  1708
         1       No   Sun  Dinner  1708
         2       No   Sun  Dinner  1708
         3       No   Sun  Dinner  1708
         4       No   Sun  Dinner  1708
         5       No   Sun  Dinner  1708
         6       No   Sun  Dinner  1708
         7       No   Sun  Dinner  1708
         8       No   Sun  Dinner  1708
         9       No   Sun  Dinner  1708
         10      No   Sun  Dinner  1708
         11      No   Sun  Dinner  1708
         12      No   Sun  Dinner  1708
         13      No   Sun  Dinner  1708
         14      No   Sun  Dinner  1708
         15      No   Sun  Dinner  1708
         16      No   Sun  Dinner  1708
         17      No   Sun  Dinner  1708
         18      No   Sun  Dinner  1708
         19      No   Sat  Dinner  1708
         20      No   Sat  Dinner  1708
         21      No   Sat  Dinner  1708
         22      No   Sat  Dinner  1708
         23      No   Sat  Dinner  1708
         24      No   Sat  Dinner  1708
         25      No   Sat  Dinner  1708
         26      No   Sat  Dinner  1708
         27      No   Sat  Dinner  1708
         28      No   Sat  Dinner  1708
         29      No   Sat  Dinner  1708
         ..     ...   ...     ...   ...
```

```
214    Yes   Sat  Dinner  1708
215    Yes   Sat  Dinner  1708
216    Yes   Sat  Dinner  1708
217    Yes   Sat  Dinner  1708
218    Yes   Sat  Dinner  1708
219    Yes   Sat  Dinner  1708
220    Yes   Fri   Lunch  1708
221    Yes   Fri   Lunch  1708
222    Yes   Fri   Lunch  1708
223     No   Fri   Lunch  1708
224    Yes   Fri   Lunch  1708
225    Yes   Fri   Lunch  1708
226    Yes   Fri   Lunch  1708
227     No   Sat  Dinner  1708
228     No   Sat  Dinner  1708
229    Yes   Sat  Dinner  1708
230    Yes   Sat  Dinner  1708
231    Yes   Sat  Dinner  1708
232     No   Sat  Dinner  1708
233     No   Sat  Dinner  1708
234    Yes   Sat  Dinner  1708
235     No   Sat  Dinner  1708
236    Yes   Sat  Dinner  1708
237    Yes   Sat  Dinner  1708
238     No   Sat  Dinner  1708
239     No   Sat  Dinner  1708
240    Yes   Sat  Dinner  1708
241    Yes   Sat  Dinner  1708
242     No   Sat  Dinner  1708
243     No  Thur  Dinner  1708

[244 rows x 4 columns]
```

```
In [95]: tips[["smoker", "day", "time", "size"]]
```

Out[95]:

	smoker	day	time	size
0	No	Sun	Dinner	1708
1	No	Sun	Dinner	1708
2	No	Sun	Dinner	1708
3	No	Sun	Dinner	1708
4	No	Sun	Dinner	1708
5	No	Sun	Dinner	1708
6	No	Sun	Dinner	1708
7	No	Sun	Dinner	1708
8	No	Sun	Dinner	1708
9	No	Sun	Dinner	1708
10	No	Sun	Dinner	1708
11	No	Sun	Dinner	1708

```
12      No   Sun  Dinner  1708
13      No   Sun  Dinner  1708
14      No   Sun  Dinner  1708
15      No   Sun  Dinner  1708
16      No   Sun  Dinner  1708
17      No   Sun  Dinner  1708
18      No   Sun  Dinner  1708
19      No   Sat  Dinner  1708
20      No   Sat  Dinner  1708
21      No   Sat  Dinner  1708
22      No   Sat  Dinner  1708
23      No   Sat  Dinner  1708
24      No   Sat  Dinner  1708
25      No   Sat  Dinner  1708
26      No   Sat  Dinner  1708
27      No   Sat  Dinner  1708
28      No   Sat  Dinner  1708
29      No   Sat  Dinner  1708
..     ...   ...     ...   ...
214    Yes   Sat  Dinner  1708
215    Yes   Sat  Dinner  1708
216    Yes   Sat  Dinner  1708
217    Yes   Sat  Dinner  1708
218    Yes   Sat  Dinner  1708
219    Yes   Sat  Dinner  1708
220    Yes   Fri   Lunch  1708
221    Yes   Fri   Lunch  1708
222    Yes   Fri   Lunch  1708
223     No   Fri   Lunch  1708
224    Yes   Fri   Lunch  1708
225    Yes   Fri   Lunch  1708
226    Yes   Fri   Lunch  1708
227     No   Sat  Dinner  1708
228     No   Sat  Dinner  1708
229    Yes   Sat  Dinner  1708
230    Yes   Sat  Dinner  1708
231    Yes   Sat  Dinner  1708
232     No   Sat  Dinner  1708
233     No   Sat  Dinner  1708
234    Yes   Sat  Dinner  1708
235     No   Sat  Dinner  1708
236    Yes   Sat  Dinner  1708
237    Yes   Sat  Dinner  1708
238     No   Sat  Dinner  1708
239     No   Sat  Dinner  1708
240    Yes   Sat  Dinner  1708
241    Yes   Sat  Dinner  1708
242     No   Sat  Dinner  1708
```

```
          243     No  Thur  Dinner  1708

          [244 rows x 4 columns]
```

```
In [96]: tips.loc[:, "smoker":"size"]
```

```
Out[96]:     smoker   day    time  size
          0       No   Sun  Dinner  1708
          1       No   Sun  Dinner  1708
          2       No   Sun  Dinner  1708
          3       No   Sun  Dinner  1708
          4       No   Sun  Dinner  1708
          5       No   Sun  Dinner  1708
          6       No   Sun  Dinner  1708
          7       No   Sun  Dinner  1708
          8       No   Sun  Dinner  1708
          9       No   Sun  Dinner  1708
          10      No   Sun  Dinner  1708
          11      No   Sun  Dinner  1708
          12      No   Sun  Dinner  1708
          13      No   Sun  Dinner  1708
          14      No   Sun  Dinner  1708
          15      No   Sun  Dinner  1708
          16      No   Sun  Dinner  1708
          17      No   Sun  Dinner  1708
          18      No   Sun  Dinner  1708
          19      No   Sat  Dinner  1708
          20      No   Sat  Dinner  1708
          21      No   Sat  Dinner  1708
          22      No   Sat  Dinner  1708
          23      No   Sat  Dinner  1708
          24      No   Sat  Dinner  1708
          25      No   Sat  Dinner  1708
          26      No   Sat  Dinner  1708
          27      No   Sat  Dinner  1708
          28      No   Sat  Dinner  1708
          29      No   Sat  Dinner  1708
          ..     ...   ...     ...   ...
          214    Yes   Sat  Dinner  1708
          215    Yes   Sat  Dinner  1708
          216    Yes   Sat  Dinner  1708
          217    Yes   Sat  Dinner  1708
          218    Yes   Sat  Dinner  1708
          219    Yes   Sat  Dinner  1708
          220    Yes   Fri   Lunch  1708
          221    Yes   Fri   Lunch  1708
          222    Yes   Fri   Lunch  1708
          223     No   Fri   Lunch  1708
```

```
224    Yes   Fri   Lunch  1708
225    Yes   Fri   Lunch  1708
226    Yes   Fri   Lunch  1708
227     No   Sat  Dinner  1708
228     No   Sat  Dinner  1708
229    Yes   Sat  Dinner  1708
230    Yes   Sat  Dinner  1708
231    Yes   Sat  Dinner  1708
232     No   Sat  Dinner  1708
233     No   Sat  Dinner  1708
234    Yes   Sat  Dinner  1708
235     No   Sat  Dinner  1708
236    Yes   Sat  Dinner  1708
237    Yes   Sat  Dinner  1708
238     No   Sat  Dinner  1708
239     No   Sat  Dinner  1708
240    Yes   Sat  Dinner  1708
241    Yes   Sat  Dinner  1708
242     No   Sat  Dinner  1708
243     No  Thur  Dinner  1708

[244 rows x 4 columns]
```

In [97]:
```
#SUPPOSE WE WANT ALL COLUMNS EXCEPT ONE
#EXAMPLE
tips.drop("smoker", axis = 1)
#axis = 1 means drop "smoker" from the columns
```

Out[97]:

	total_bill	tip	sex	day	time	size
0	16.99	1.01	Female	Sun	Dinner	1708
1	10.34	1.66	Male	Sun	Dinner	1708
2	21.01	3.50	Male	Sun	Dinner	1708
3	23.68	3.31	Male	Sun	Dinner	1708
4	24.59	3.61	Female	Sun	Dinner	1708
5	25.29	4.71	Male	Sun	Dinner	1708
6	8.77	2.00	Male	Sun	Dinner	1708
7	26.88	3.12	Male	Sun	Dinner	1708
8	15.04	1.96	Male	Sun	Dinner	1708
9	14.78	3.23	Male	Sun	Dinner	1708
10	10.27	1.71	Male	Sun	Dinner	1708
11	35.26	5.00	Female	Sun	Dinner	1708
12	15.42	1.57	Male	Sun	Dinner	1708
13	18.43	3.00	Male	Sun	Dinner	1708
14	14.83	3.02	Female	Sun	Dinner	1708
15	21.58	3.92	Male	Sun	Dinner	1708
16	10.33	1.67	Female	Sun	Dinner	1708
17	16.29	3.71	Male	Sun	Dinner	1708
18	16.97	3.50	Female	Sun	Dinner	1708

```
19        20.65  3.35    Male   Sat  Dinner  1708
20        17.92  4.08    Male   Sat  Dinner  1708
21        20.29  2.75  Female   Sat  Dinner  1708
22        15.77  2.23  Female   Sat  Dinner  1708
23        39.42  7.58    Male   Sat  Dinner  1708
24        19.82  3.18    Male   Sat  Dinner  1708
25        17.81  2.34    Male   Sat  Dinner  1708
26        13.37  2.00    Male   Sat  Dinner  1708
27        12.69  2.00    Male   Sat  Dinner  1708
28        21.70  4.30    Male   Sat  Dinner  1708
29        19.65  3.00  Female   Sat  Dinner  1708
..          ...   ...     ...   ...     ...   ...
214       28.17  6.50  Female   Sat  Dinner  1708
215       12.90  1.10  Female   Sat  Dinner  1708
216       28.15  3.00    Male   Sat  Dinner  1708
217       11.59  1.50    Male   Sat  Dinner  1708
218        7.74  1.44    Male   Sat  Dinner  1708
219       30.14  3.09  Female   Sat  Dinner  1708
220       12.16  2.20    Male   Fri   Lunch  1708
221       13.42  3.48  Female   Fri   Lunch  1708
222        8.58  1.92    Male   Fri   Lunch  1708
223       15.98  3.00  Female   Fri   Lunch  1708
224       13.42  1.58    Male   Fri   Lunch  1708
225       16.27  2.50  Female   Fri   Lunch  1708
226       10.09  2.00  Female   Fri   Lunch  1708
227       20.45  3.00    Male   Sat  Dinner  1708
228       13.28  2.72    Male   Sat  Dinner  1708
229       22.12  2.88  Female   Sat  Dinner  1708
230       24.01  2.00    Male   Sat  Dinner  1708
231       15.69  3.00    Male   Sat  Dinner  1708
232       11.61  3.39    Male   Sat  Dinner  1708
233       10.77  1.47    Male   Sat  Dinner  1708
234       15.53  3.00    Male   Sat  Dinner  1708
235       10.07  1.25    Male   Sat  Dinner  1708
236       12.60  1.00    Male   Sat  Dinner  1708
237       32.83  1.17    Male   Sat  Dinner  1708
238       35.83  4.67  Female   Sat  Dinner  1708
239       29.03  5.92    Male   Sat  Dinner  1708
240       27.18  2.00  Female   Sat  Dinner  1708
241       22.67  2.00    Male   Sat  Dinner  1708
242       17.82  1.75    Male   Sat  Dinner  1708
243       18.78  3.00  Female  Thur  Dinner  1708

[244 rows x 6 columns]
```