4.3.5 Overview of the Pandas Library

As you are learning this new software, Maria throws a new tool into the mix: the Pandas library. She would like you to use this library to assist you in your analysis. With the Pandas library, you can read raw Excel files, which will help you perform the analysis. Pandas has so much more to offer! So Maria wants you to learn about the Pandas library and what it can do.

Pandas is an open-source Berkeley Software Distribution (BSD)-licensed library that provides high-performance data analysis tools for the Python programming language. The Pandas library is one of the most widely preferred tools for data analysis and carries tremendous power in handling large datasets, which can slow down Excel.

It takes a lot of coding to modify and display datasets using only Python. In Jupyter Notebook, you can use the Pandas library, where raw data can be extracted from a variety of sources, cleaned, transformed, manipulated, analyzed, and visualized, all the while leaving the original dataset intact.

Additionally, with Python, you're limited when it comes to using lists, tuples, and dictionaries to manipulate the data. However, Pandas allows Python programmers the ability to work with two data types, Series and

DataFrames, which are structured lists with many built-in convenience methods that allow for quick and easy manipulation of data.

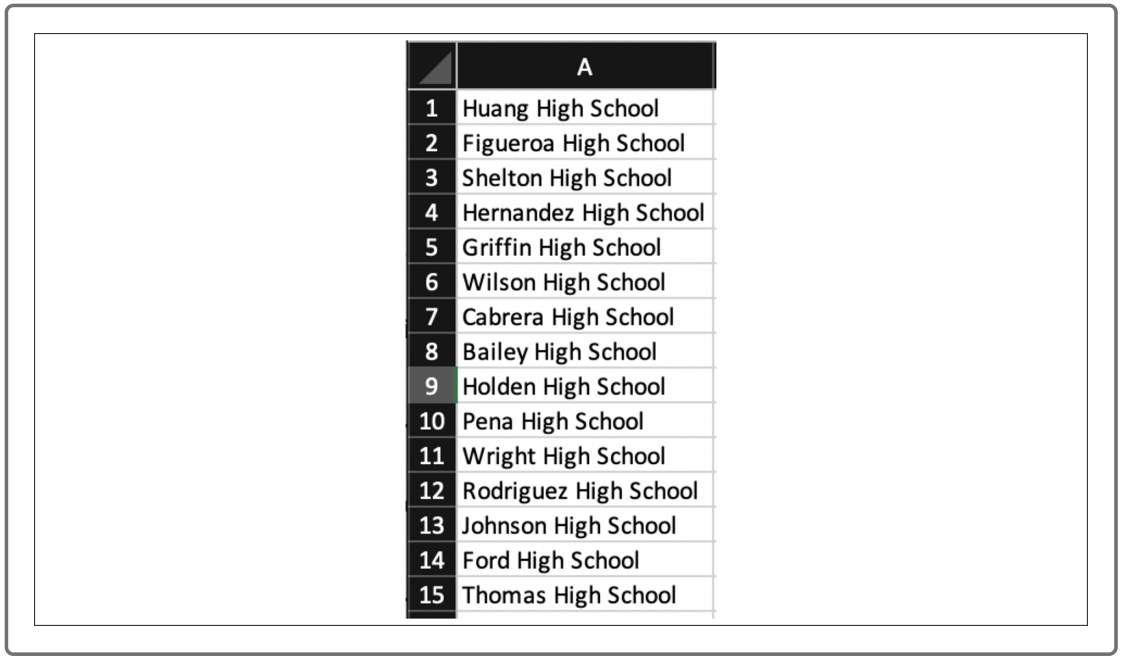
Pandas Series

A Pandas **Series** is a one-dimensional, labeled array capable of holding any data type. This means the data is linear and has an index that acts as a key in a dictionary.

REWIND

A one-dimensional array is a list of objects.

An Excel file containing a list of high schools is an example of a Series. This Series behaves like a Python dictionary in that it has an index and values for each row, which is the high school name.



	A
1	Huang High School
2	Figueroa High School
3	Shelton High School
4	Hernandez High School
5	Griffin High School
6	Wilson High School
7	Cabrera High School
8	Bailey High School
9	Holden High School
10	Pena High School
11	Wright High School
12	Rodriguez High School
13	Johnson High School
14	Ford High School
15	Thomas High School

We can convert the list of high schools in Jupyter Notebook to a Pandas Series, which will allow us to get information from this list.

In the `pandas_practice.ipynb` file, create a new cell, add the list of high schools, and run the cell.

```
# List of high schools
high_schools = ["Huang High School",  "Figueroa High School", "Shelton High
```

To create a Pandas Series, we'll use the `pandas.Series()` method from the Pandas library. But in order to use this method, we first need to import the Pandas library as a dependency.

REWIND

To import a dependency, we use `import`.

In a new cell, type `import pandas as pd`. This code tells the program to import the Pandas library as the alias `pd`.

```
# Add the Pandas dependency.
import pandas as pd
```

IMPORTANT

> It's a best practice to shorten the dependency name, or give it an **alias.** This makes it easier to use the dependency in any preceding code that you write.

In the next cell, type `school_series = pd.Series(high_schools)`. This will create a Pandas Series for the list of high schools.

```
# Create a Pandas Series from a list.
school_series = pd.Series(high_schools)
```

In this code, we assign the variable `school_series` to the conversion of the list of high schools to a Pandas Series by using `pd.Series()` instead of `pandas.Series()`. This is how we use the alias `pd` in our code.

When we run the cell, there is no direct output. The Series we just created is in computer memory. To see the output, we need to retrieve it by calling the variable `school_series`, like this:

```
# Create a Pandas Series from a list.
school_series = pd.Series(high_schools)
school_series
```

When we run the cell, the output looks like it did in the Excel file, but it's formatted differently. There is a default row of index numbers, which is a sequence of incremental numbers starting from 0. In each row, there is a value for each index, which, in this case, is each high school in the list.

```
0          Huang High School
1       Figueroa High School
2        Shelton High School
3     Hernandez High School
4        Griffin High School
5         Wilson High School
6        Cabrera High School
7         Bailey High School
8         Holden High School
9           Pena High School
10        Wright High School
11    Rodriguez High School
12      Johnson High School
13         Ford High School
14       Thomas High School
dtype: object
```

Congratulations, you have created your first Pandas Series!

SKILL DRILL

Like a list in Python, indexing can be used to get specific items from a Pandas Series.

Iterate through the `school_series` and print out each high school.

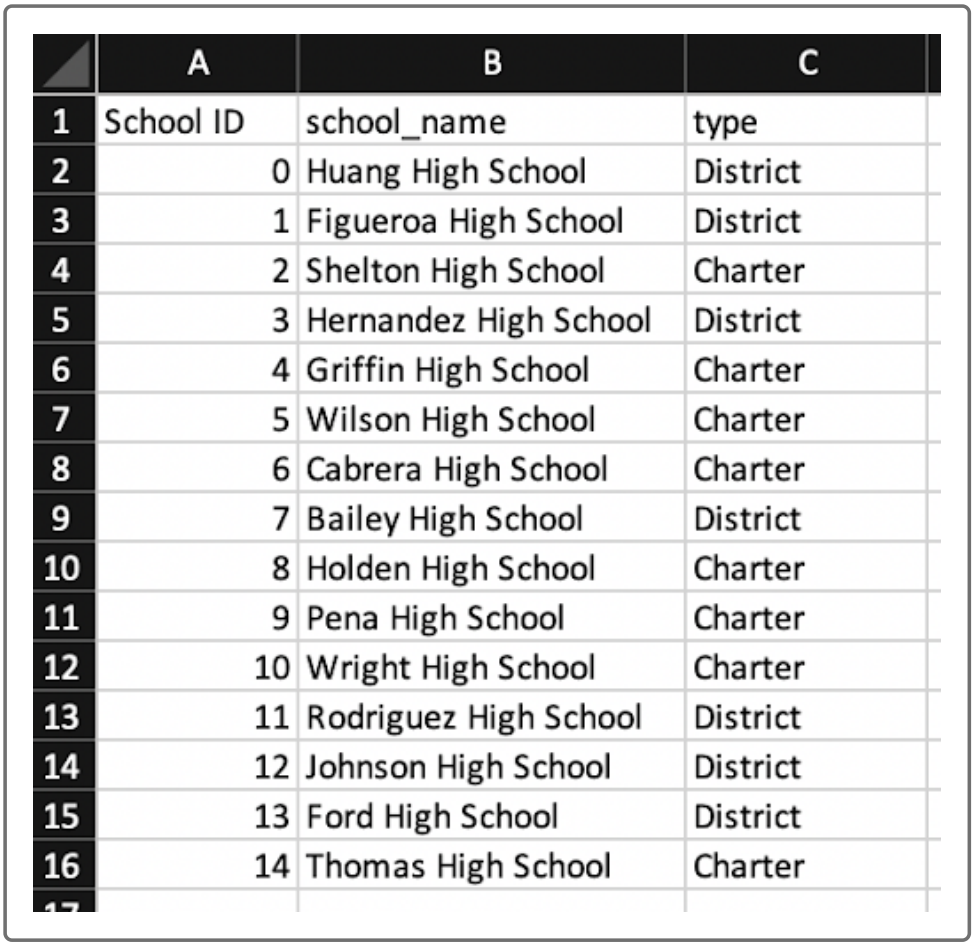
Pandas DataFrames

A Pandas **DataFrame** is a two-dimensional labeled data structure, like a dictionary, with rows and columns of potentially different data types such as strings, integers, and floats (decimal point numbers), where data is aligned in a table.

For example, the following image of the Excel spreadsheet is a two-dimensional labeled data structure where each row and column contains different data types.

In essence, a Pandas DataFrame contains multiple Pandas Series, or lists. Each column in the Excel file is a Series. In the previous section, we worked with a Series, a list of high schools. That Series is like column B in this Excel file.

We can convert the data in this Excel file to a Pandas DataFrame using the `pandas.DataFrame()` method from the Pandas library.



	A	B	C
1	School ID	school_name	type
2	0	Huang High School	District
3	1	Figueroa High School	District
4	2	Shelton High School	Charter
5	3	Hernandez High School	District
6	4	Griffin High School	Charter
7	5	Wilson High School	Charter
8	6	Cabrera High School	Charter
9	7	Bailey High School	District
10	8	Holden High School	Charter
11	9	Pena High School	Charter
12	10	Wright High School	Charter
13	11	Rodriguez High School	District
14	12	Johnson High School	District
15	13	Ford High School	District
16	14	Thomas High School	Charter

Convert a List of Dictionaries to DataFrame

To create the dictionaries, in the `pandas_practice.ipynb` file, copy the code into a new cell.

```
# A dictionary of high schools
high_school_dicts = [{"School ID": 0, "school_name": "Huang High    School",
                      {"School ID": 1, "school_name": "Figueroa High School", "
                       {"School ID": 2, "school_name":"Shelton High School", "t
```

```
    {"School ID": 3, "school_name":"Hernandez High School",
    {"School ID": 4, "school_name":"Griffin High School", "t
```

The column headers in the Excel file are the keys in the dictionary. The value for each of the keys corresponds to a row in the Excel file.

REWIND

A Python dictionary has a **key** and a **value**, or **key-value pairs**.

In the next cell, we will convert the array, or list of dictionaries, to a DataFrame using `school_df = pd.DataFrame(high_school_dicts)`. In a new cell, type and run the following code:

```
school_df = pd.DataFrame(high_school_dicts)
school_df
```

This will create a Pandas DataFrame for the list of dictionaries.

	School ID	school_name	type
0	0	Huang High School	District
1	1	Figueroa High School	District
2	2	Shelton High School	Charter
3	3	Hernandez High School	District
4	4	Griffin High School	Charter

Congratulations on creating your first Pandas DataFrame!

Just like in the image of the Excel file, there are three columns with the corresponding headers "School ID," "school_name," and "type." Also, like in the Excel file, there is an index for each row. However, in Python, indexing starts from 0.

NOTE

> The `df` in `school_df` is short for "DataFrame." It's a best practice to add this `df` to a named DataFrame to help distinguish between DataFrames, Series, and variables.

Convert a List or Series to a DataFrame

We can also create the same DataFrame by adding each column as a list, or Series, to an empty DataFrame.

In a new cell, add the following lists and run the cell.

```
# Three separate lists of information on high schools
school_id = [0, 1, 2, 3, 4]

school_name = ["Huang High School", "Figueroa High School",
"Shelton High School", "Hernandez High School","Griffin High School"]

type_of_school = ["District", "District", "Charter", "District","Charter"]
```

Next, in a new cell, initialize an empty DataFrame, like this:

```
# Initialize a new DataFrame.
schools_df = pd.DataFrame()
```

Now we will add each list to the empty `schools_df` DataFrame by typing the column name in quotes within brackets, like we do when we create a key for a dictionary.

REWIND

The standard format for creating a key in a dictionary is to put the key in single or double quotes and inside brackets, e.g., `counties_dict["Arapahoe"] = 422829`.

Instead of using the dictionary name, we'll use the name of the DataFrame and make it equal to the list, like this:

```
# Add the list to a new DataFrame.
schools_df["School ID"] = school_id
```

```
# Print the DataFrame.
schools_df
```

SKILL DRILL

Add the `school_name` list as "School Name" and the `type_of_school` list as "Type" to the `schools_df` DataFrame. Then print out the `schools_df` DataFrame.

When you run the cells, your output should look like this:

	School ID	school_name	type
0	0	Huang High School	District
1	1	Figueroa High School	District
2	2	Shelton High School	Charter
3	3	Hernandez High School	District
4	4	Griffin High School	Charter

Another way to create this DataFrame is to create a Python dictionary from the three lists, like this:

```
# Create a dictionary of information on high schools.
high_schools_dict = {'School ID': school_id, 'school_name':school_name, 'typ
```

Now we can convert the `high_schools_dict` to a DataFrame.

SKILL DRILL

Convert the dictionary `high_schools_dict` to a DataFrame.

The Anatomy of a DataFrame

Before we move on, let's look at three main parts of a DataFrame, using the `school_df` as an example:

1. **Columns**: The top, or header, rows
2. **Index**: The numbers that run down the left-hand margin
3. **Values**: The values in the columns (the data)

See the following image for a visual representation of a DataFrame's anatomy:

We can access these three main components of a DataFrame with the `columns`, `index`, and `values` attributes.

The Columns Attribute

To get the column names of a DataFrame, use `df.columns`. Here's how you would apply the `columns` attribute to the `school_df` DataFrame:

```
school_df.columns
```

When we run this cell, the output will be a list of the column names:

```
Index(['School ID', 'school_name', 'type'], dtype='object')
```

The Index Attribute

To get the indices of the DataFrame, use `df.index`. Here's how you would apply the `index` attribute to the `school_df` DataFrame:

```
school_df.index
```

When you run this cell, the output is a `RangeIndex` that contains the first and last index value, as well as the "step=1," or how often the index increments. In this case, the index increments a value of 1 from the beginning index to the ending index.

```
RangeIndex(start=0, stop=5, step=1)
```

The Values Attribute

To get the values of the DataFrame, use `df.values`. Here's how you would apply the `values` attribute to the `school_df` DataFrame:

```
school_df.values
```

When you run this cell, the output will be an array of all the values, but without the column names:

SKILL DRILL

Based on the following image, create a list for each column and add each list to a new Pandas DataFrame.

NOTE

> For more information, see the documentation on Series and DataFrames:
>
> - **Pandas Series** (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.sample.html)
> - **Pandas DataFrame** (https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.html)