How to import Excel data into Python scripts with Pandas

Instead of manually copying data into a database, here is a quick guide on how to load Excel data into Python using Pandas.

Microsoft Excel is the most widely used spreadsheet software in the world and has good reason for that. Excel has a user-friendly interface and powerful integration tools that make it easy to work with data.

But if you want to process data more advanced, you'll need to use things beyond Excel's capabilities and start using a programming language like Python. Instead of manually copying data into a database, here is a quick guide on how to load Excel data into Python using Pandas.

Note : If you've never used Python before, this tutorial can be tricky. You should start with websites to learn Python and the basic Python examples that **TipsMake.com** has suggested.

How to load Excel data into Python with Pandas

1. What is pandas?
2. Install Pandas
3. Prepare Excel data
4. Write Python scripts
    1. Enter the Python library
    2. Working with file paths
    3. Extract Excel data with Pandas.Read_Excel ()
    4. Run the Python Script
    5. Take a closer look at the DataFrame object!

What is pandas?

Python Data Analysis Library (Pandas) is an open source library for the Python programming language used to analyze and manipulate data.

Pandas loads data into Python objects called Dataframes, which store data in rows and columns like a traditional database. Once a Dataframe is created, it can be manipulated in Python, opening up a multitude of possibilities.

Install Pandas

Note : You must have Python 2.7 or higher to install Pandas.

To start working with Pandas on your computer, you will need to enter the Pandas library. If you are looking for an advanced solution, you can download **Anaconda Python Distribution** , which has Pandas integrated. If you are not using Anaconda, Pandas is very simple to install in the terminal.

Pandas is a PyPI package, which means you can install it using PIP for Python via the command line. Modern Mac systems come with PIP. For Windows, Linux and many other older systems, it's easy to learn how to install PIP for Python.

Once you have opened the terminal, the latest version of Pandas can be installed using the command:

```
>> pip install pandas
```

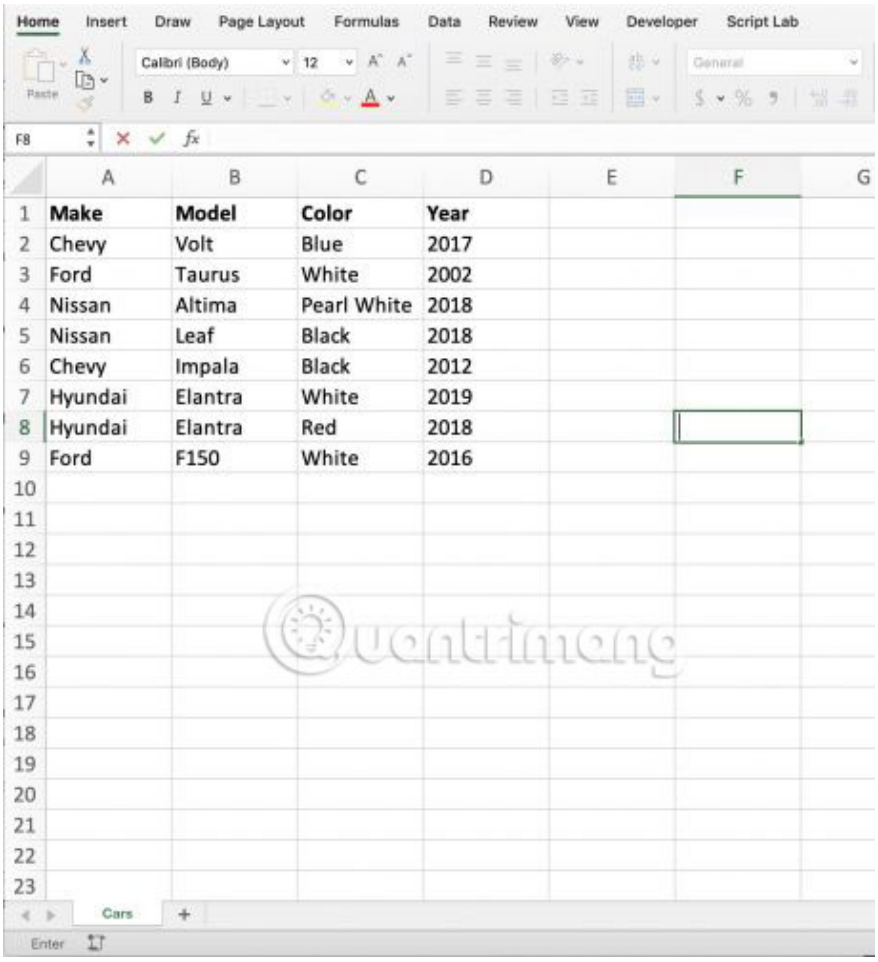
Pandas also requires the NumPy library, please install this library on the command line:

```
>> pip install numpy
```

Now you have Pandas installed and ready to create your first DataFrame!

Prepare Excel data

In this example, use a sample data set: An Excel workbook named **Cars.xlsx.**



	Make	Model	Color	Year
2	Chevy	Volt	Blue	2017
3	Ford	Taurus	White	2002
4	Nissan	Altima	Pearl White	2018
5	Nissan	Leaf	Black	2018
6	Chevy	Impala	Black	2012
7	Hyundai	Elantra	White	2019
8	Hyundai	Elantra	Red	2018
9	Ford	F150	White	2016

This data set displays the brand, style, color and year of manufacture of the cars entered into the table. The table is displayed as an Excel range. Pandas is smart enough to read the data appropriately.

This workbook is saved to the **Desktop** folder **,** this is the file path to be used:

```
/Users/grant/Desktop/Cars.xlsx
```

You will need to know the file path of the workbook to use Pandas. Start by opening Visual Studio Code to write the script. If you do not have a text editor, the article recommends using Visual Studio Code or Atom Editor.

Write Python scripts

You now have the text editor you need. Next, the article will combine Python and workbook Cars to create a DataFrame Pandas.

Enter the Python library

Open a text editor and create a new Python file. Call it **Script.py.**

To work with Pandas in the script, you will need to enter it into your code. This is done with one line of code:

```
import pandas as pd
```

Here, the example is loading the Pandas library and attaching it to a variable **'pd'.** You can use any name you want, and the example currently uses **'pd', which** stands for **Pandas.**

To work with Excel with Pandas, you need an additional object named **ExcelFile.** ExcelFile is integrated into the Pandas ecosystem, so you can import directly from Pandas:

```
from pandas import ExcelFile
```

Working with file paths

To give Pandas access to the workbook, you need to direct your script to the file location. The easiest way to do this is to give the script the full path to the workbook.

Recall the path in this example: **/Users/grant/Desktop/Cars.xlsx**

You will need this file path referenced in the script to extract the data. Instead of referencing the path inside the **Read_Excel** function **,** keep the code 'clean' by storing the path in a variable:

```
Cars_Path = '/Users/grant/Desktop/Cars.xlsx'
```

You are now ready to extract data using the Pandas function!

Extract Excel data with Pandas.Read_Excel ()

With Pandas entered and the path variable set, you can now use functions in the Pandas object to complete the task.

The function you will need to use is called **Read_Excel.** The Read_Excel function takes the file path of the Excel workbook and returns a DataFrame object with the contents of the workbook. Pandas sets this function to:

```
pandas.read_excel(path)
```

The **'path'** argument is the path to the Cars.xlsx workbook and the article has set the path string to the **Cars_Path** variable **.**

You are now ready to create DataFrame objects! Let's put it all together and put the DataFrame object into a variable named **DF:**

```
DF = pd.read_excel(Cars_Path)
```

Finally, you will want to see the DataFrame, so print the results. Add a print statement to the end of the script, using the DataFrame variable as an argument:

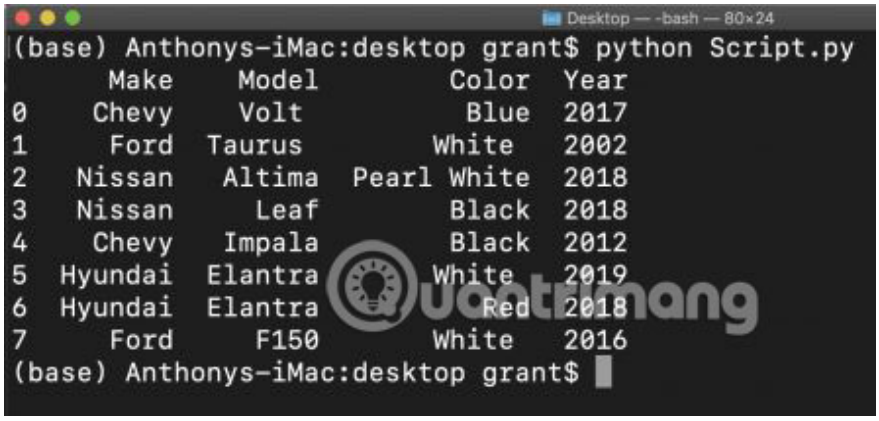
```
print(DF)
```

Time to run scripts in the terminal!

Run the Python script

Open a terminal or command line and navigate to the directory where your script is located. In this case, we have '**Script.py**' immediately on the desktop. To execute the script, use the python command followed by the script file:

Python will pull data from 'Cars.xlsx' into the new DataFrame and print the DataFrame to the terminal!



Take a closer look at the DataFrame object!

At first glance, DataFrame looks very similar to regular Excel tables. This makes Pandas DataFrame very easy to understand.

Headers are labeled at the top of the dataset and Python has filled the rows with all the information read from the Cars.xlsx workbook.

Note the leftmost column, an index starting at 0 and numbering the columns. By default, Pandas will apply this index to DataFrame, which may be useful in some cases. If you do not want this index to be created, you can add an additional argument to the code:

```
DF = pd.read_excel(Cars_Path, index=False)
```

Setting the '**index**' argument to **False** removes the index column, leaving only Excel data.

You now have the ability to read data from an Excel spreadsheet. You can apply Python programming any way you choose. Working with Pandas is a simple way for experienced Python programmers to work with data stored in Excel workbooks.

The ease of using Python to analyze and manipulate data is one of the many reasons why Python is the programming language of the future.

Hope you are succesful.