Managing Data With Python

There are countless reasons that one might find themselves regularly interacting with data in the form of spreadsheets. Be they massive or many, spreadsheets present ample opportunities to waste precious time and effort searching for and amending nitpicky human errors. Automating such tasks through code is a powerful way to increase your efficiency, saving your energy for more productive endeavors than making copy pasta. Don't worry if this already sounds overwhelming. Much like you don't need to be fluent in Spanish to ask where the baños are, knowing a little bit of coding can provide tremendous utility.

This guide includes instructions for:

- setting up your system in the Terminal
- installing and importing Python libraries
- acquiring and setting up a code editor
- writing a Python script that find specific data in in a larger collection and write a new, more manageable spreadsheet file containing only the desired data.

You will need:

- a computer running Mac OS 10.15.4+
- Python 2.7
- Microsoft Excel, Numbers, or similar
- a web browser
- an internet connection

Before You Begin

If you've never coded anything before, you'll need to know a few key concepts to understand the processes this guide explains.

Python

Coding, simply put, is the act of writing instructions for computers. Python coding language is generally regarded as one of the easier, more intuitive coding languages to learn and, thusly, is very broadly used.

Code Editor

Code editors are programs used to compose and test code which is then saved as, in this case, a .py file. Opening (by double-clicking) the .py file will run your program which will write and save the new .xlsx file to your hard drive.

Terminal

The terminal where you give your computer instructions about itself. For our purposes, the terminal is the tool we'll use to set up Python and install two necessary libraries.

Libraries

Think of libraries as encyclopedias in the language of Python; each library helps Python understand different subjects. This tutorial requires a library called "pandas," used for data analysis, along with a library called "XlsxWriter," used for creating .xlsx (Excel) files.

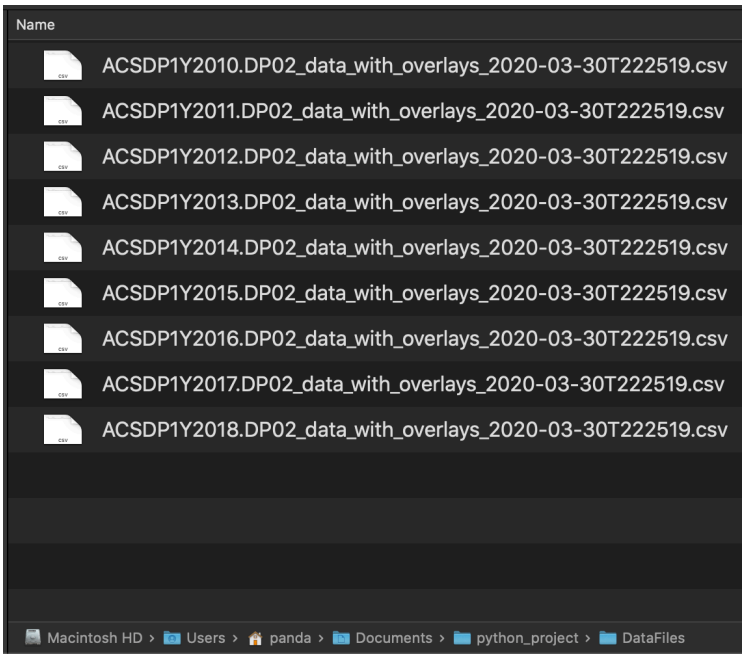
PHASE ZERO

READYING THE DATA

The absolute first thing you should do before getting started with this tutorial is to save some sample .csv files to your hard drive. This tutorial works with several .csv files that contain data from the annual American Community Survey downloaded from the Census Bureau at https://data.census.gov/cedsci/. In this tutorial, the script will index a part of each year's ACS data spanning from 2010 – 2018. The data has been saved into a folder /Users/panda/Documents/python_project/DataFiles.

The file format of the data you're working with dictates which Python libraries you'll need to access. Pandas is a popular Python library that can read .csv, or "comma separated values," files, so that is one of two libraries this tutorial uses. The file names on these are quite long, but don't worry much about that.

NOTE: Each of these files has the year of the survey in its name. If you choose to rename your files, you must include the year number in the new name.



The script will search these reference tables for two specific statistics:

- the percentage of Los Angeles county's population holding bachelor's degrees for each year of data,
- and the percentage of Los Angeles county's population holding graduate or professional degrees for each year of data.

LOCATING DATA AND COPYING COLUMN LABELS

1. Open the first year's file in Excel.
2. Press **control + F**
3. Type "bachelor's degree" into the search box.
4. Press **return**.
5. Scroll through the results until you find the desired data.
   NOTE: read carefully, some columns list data for "bachelor's degree or higher" while some list data for bachelor's degrees alone.
6. Copy exactly the contents of the column label into a blank document for later use.
7. Repeat steps 2-5, but search for the term "graduate."

8. Copy exactly the contents of the column label into a blank document for later use.

9. Repeat both searches in each remaining data table.
   NOTE: Given the time span of these data tables, it is safest not to assume they'll be uniformly labelled. Searching for the same data across 9 tables/years produces different labels used for each of our two variables.

PHASE ONE

Before any code can be written, you'll need to set up a few features in the command line terminal. Mac OS X should include Python and pip installer by default. You should check the version of Python and pip installer on your machine before proceeding.

ASSESSING PYTHON

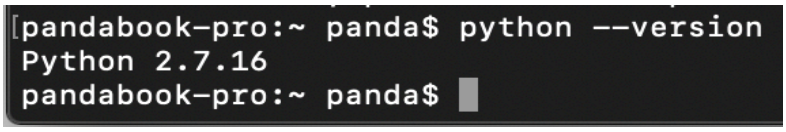
1. In any window, press **command** + **space bar** to summon Spotlight Search.

2. Type "terminal" into the search bar that appears.

3. In the list of search results, select **Terminal,** which will either appear as the top hit or in the Applications section.
   The Terminal window will appear, ready for input.

4. Type "python --version."

5. Press **return**.
   The line below will display the version of Python currently running on your computer. This image shows that the computer, pandabook-pro, is running Python version 2.7.16.

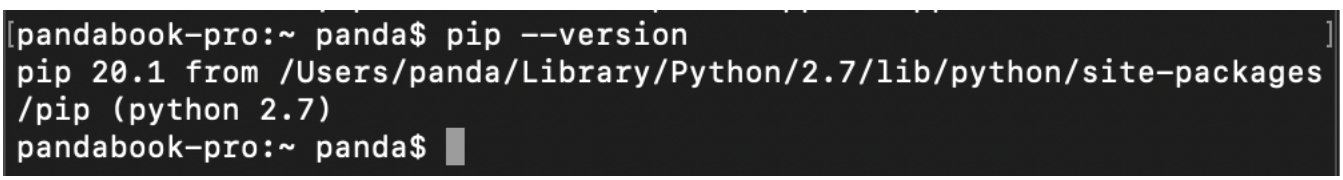


ASSESSING PACKAGE INSTALLER (PIP)

Package installer for Python, or pip, is the utility with which Python libraries are installed. Like Python, your system should already have pip installer on it, but you should check the version on your computer and update to the most recent version if necessary.

1. In the terminal, type "pip --version."

2. Press **return**.



The next line will display the version of pip installer on your system. pandabook-pro is running pip 20.1, the most recent version as of the writing of this guide.

NOTE: to update older versions of the pip installer, type "pip install -U pip" and press **return**.

INSTALLING PYTHON LIBRARIES

Once pip installer is ready, it can be used to install the libraries necessary for Python to interpret the script and perform its functions. The two libraries needed for this guide are pandas and XlsxWriter. Pandas is a very popular library used for data analysis and XlsxWriter is used to create spreadsheet files that can be read in Excel, Numbers, and Google Sheets, among others.

1. In the terminal window, type "pip install pandas." ress **return**.
   The installer will perform its function and produce several lines of technical jargon that you don't need to worry about. As long as the last line reads "successfully installed pandas," you can continue on to step 2.
   NOTE: if pandas does not successfully install, visit packaging.python.org for troubleshooting FAQs and forums.

2. Press **return**.

3. In the same window, type "pip install xlsxwriter."
   Like the pandas library, you will know the XlsxWriter library installation is complete when the last line reads "successfully installed xlsxwriter."

4. Press **return**.

5. Quit the Terminal application.

PHASE TWO

The next phase of the process is to download and set up a code editor, Visual Studio Code. VSC is a free application that can be used to efficiently code in many programming languages and will automatically format Python coding which relies on accurate indentation to function.

ACQUIRING A CODE EDITOR

1. Launch your web browser.

2. Go to code.visualstudio.com

3. Click the **link** on the launch page to download the application in a zip file.

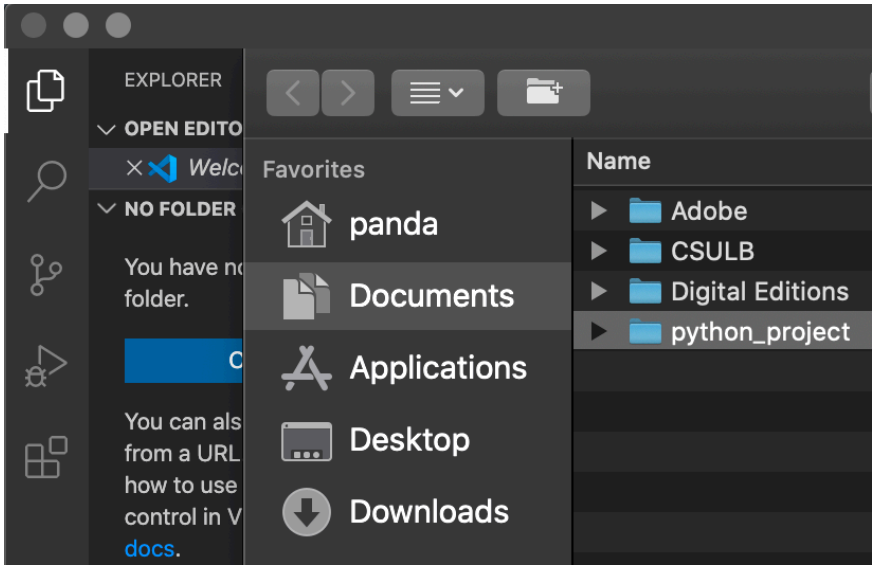
4. Once the file is downloaded, open the folder in which you saved it.

5. Right-click the **file**.
6. Click **Open With**.
7. Click **Archive Utility**.
8. The file will unzip into the current folder.
9. Drag the unzipped application file into the **Applications** folder.

SETTING UP VISUAL STUDIO CODE

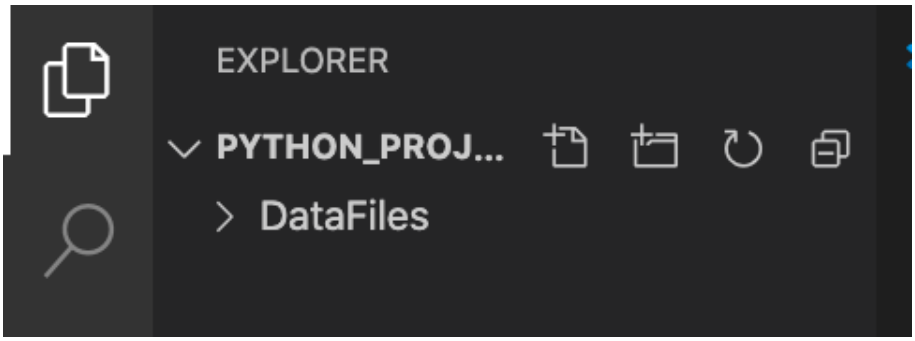
Visual Studio Code's default settings need to be adjusted so that it knows to format your code according to the rules of Python.

1. Double-click **Visual Studio Code** in the Applications folder to launch the program.
2. The launch screen will display several options for customizations. Find the **Tools and languages** section in the **Customize** area.
3. Click **Python** to install necessary components for coding in Python.
4. Once the installation has finished, type **Shift** + **Command** + **E** to call the explorer sidebar into view.



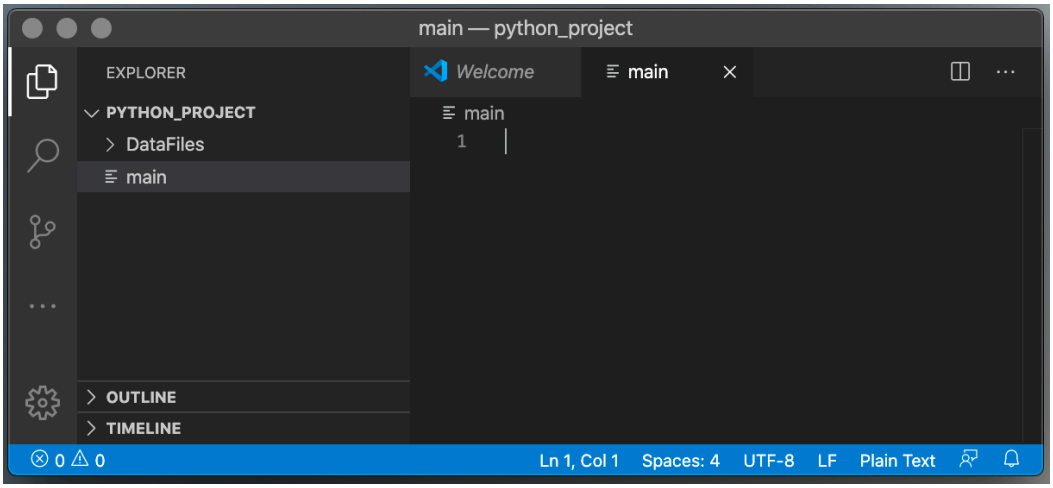
5. Click **Open Folder**.
   NOTE: Opening a folder from the Explorer sidebar tells VSC what folder to look in to find the documents it will reference while running the code you write. Choose the folder in which you saved your reference data. In this example, the data files that will be referenced by the program are saved in /panda/Documents/python_project/DataFiles.
6. Select **python_project**.
7. Click **Open**.
8. The explorer sidebar will show the selected folder and its contents, the data files.

9. Open a new file in this folder by clicking the first icon to the right of the folder name.
NOTE: the new file is the workspace in which you will compose your code. The file will be saved as a .py program.



10. Name the new file. The example file for this tutorial is named **main**.

NOTE: Once the new file has been created and named, the editing view will appear in place of the welcome screen.



11. From the options in the blue bar that runs along the bottom of the application window, click **Plain Text** to reveal a search bar.

12. In the search bar that appears, type "Python" and press **return**.

PHASE THREE

Now that your system and code editor are prepared for the task, you can begin building your script, which is composed of two main portions. The first portion is where you import the libraries, so that the program knows where to look to interpret some of the instructions it will contain; it is also where you tell it where to reference the original data files and what to name the new spreadsheet it will create and save to your hard drive. The second portion is the main body of the program which will contain the bulk of the instructions for the functions you want it to perform.

IMPORTING LIBRARIES

The libraries you installed earlier are available on your machine, but your program still has to be told to utilize them. This is how the script will know what libraries to reference.

1. On the first line of your file, type "import pandas as pd"
   NOTE: importing the pandas library "as pd" is essentially assigning it a nickname. From here on out for the specific script you're writing, refer to pandas as "pd."

2. Press **return**.
   NOTE: Pressing return in the code editor only moves the cursor to the next line.

3. On the next line, type "import XlsxWriter"
   NOTE: Like all aspects of Python, XlsxWriter is case-sensitive and must be input as shown to be correctly interpreted.

4. Press **return** twice.

READYING REFERENCES AND OUTPUT

You'll also need to name the file or files that the script should reference to gather the data you want to copy. Because we opened the folder in which they are saved, that doesn't need to be specified, but the files within that folder do.

1. Type "DATAFILENAME = ""

2. Into the closed quotes, **copy and paste** the name of the first .csv file.

3. Delete the year number from the file name.

4. In place of the year number, type "%d."
   NOTE: This step sets up the code to be able to reference multiple files; % is a variable like X in algebra, it can stand for a number. Adding the "d" next to the variable dictates that the value will be expressed as an integer, or whole number.

5. Press **return** twice.

6. Type "OUTPUT_XLSX_NAME = 'output.xlsx'" to set the name of the file.

7. Press **return** twice.

```
1   import pandas as pd
2   import xlsxwriter
3
4   DATAFILENAME = "ACSDP1Y%d.DP02_data_with_overlays_2020-03-30T222519.csv"
5
6   OUTPUT_XLSX_NAME = 'output.xlsx'
```

The code should appear identical to this with color coding automatically being applied to VSC-recognized elements, assuming they've been correctly input. If any of the above text in pink, orange, or blue is not pink, orange, or blue in the editor by the end of step six of this section, check for spelling, spacing, and punctuation errors.

Main Script Functions

The script will now be able to interpret the specific instructions to follow. None of the following actions will occur until the entirety of the script's main functions have been processed. See section "Finalizing the Program" on p. 11.

Establishing Workbook and Worksheet

Steps 3 through 6 of this section establish a workbook, or .xlsx spreadsheet file and then situate a worksheet, sometimes called a tab or page, within the file.

1. Type "def main():" to signal the end of the script set up and the actions it will take.
2. Press **return**.
3. Type "workbook = xlsxwriter.Workbook(OUTPUT_XLSX_NAME)."
4. Press **return**.
5. Type "worksheet = workbook.add_worksheet()."
6. Press **return** twice.

Defining the Header

This section establishes the column titles for the two statistics the script will snip from the original data. Columns and rows on an Excel sheet start at 0 and are read (row, column). So the header row will run horizontally across row 0, making row 0, column 0 the top-left cell on the spreadsheet, and the first of the labels to be applied. Adding 1 to the column number moves one cell to the right, to row 0, column 1, or (0, 1) .

1. Type "worksheet.write(0, 0, "Year")"
2. Press **return**.
3. Type "worksheet.write(0, 1, "Bachelor's Degree")"
4. Press **return**.
5. Type "worksheet.write(0, 2, "Graduate or Professional Degree")"

Indexing Rows

The following section takes up much of the bulk of the code because it contains a fair portion of the action the script will execute, which is to find and copy the data listed under each of the four possible column labels in each of the original .csv files.

Essentially, the "if" and "break" combination instructs the program to search the .csv files for the first of two possible labels for a given value by saying "if the first label is found, your job is done." However, the next few lines of code provide instructions for scenarios in which the second label by using the "elif," or "else if," which is how Python understands "if the first label is not found, then search for the second label, then stop when it's located."

1. Type "row_num = 1."
2. Press **return**.
3. Type "for i in range(2010 ,2018):"
4. Press **return.**
5. Type "filename = DATAFILENAME % i."
6. Press **return** twice.
7. Type "df = pd.read_csv("DataFiles/" + filename, delimiter=',')."
8. Press **return.**
9. Type "list_of_rows = [list(row) for row in df.values]."
10. Press **return** twice.
11. Type "bach_idx = 0."
12. Press **return**.
13. Type "for column_name in list_of_rows[0]:."
14. Press **return**.
15. Type " if column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Bachelor's degree":."
16. Press **return**.
17. Type "break."
18. Press **return**.

19. Type "elif column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Population 25 years and over!!Bachelor's degree":."
20. Press **return**.
21. Type "break."
22. Press **return**.
23. Type "bach_idx +=1."
24. Press **return**.
25. Type "grad_idx = 0."
26. Press **return**.
27. Type "for column_name in list_of_rows[0]:."
28. Press **return**.
29. Type " if column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Graduate or professional degree":."
30. Press **return**.
31. Type "break."
32. Press **return**.
33. Type "elif column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Population 25 years and over!!Graduate or professional degree":."
34. Press **return**.
35. Type "break."
36. Press **return**.
37. Type "grad_idx +=1."
38. Press **return** twice.

Copying Values From the Index

Once the values have been located, they can be copied.

1. Type "bach_value = list_of_rows[1][bach_idx]."
2. Press **return**.

3. Type "grad_value = list_of_rows[1][grad_idx] ."
4. Press **return** twice.

Pasting Values From the Index

With the row values copied to the computer's clipboard, they can be pasted into place on the worksheet within the workbook.

1. Type "worksheet.write(row_num, 0, i)."
2. Press **return**.
3. Type "worksheet.write(row_num, 1, bach_value + "%")."
4. Press **return**.
5. Type "worksheet.write(row_num, 2 ,grad_value + "%"."
6. Press **return** twice.

Cell Management

The following line of code is small, but crucial—it instructs the program to move from one cell to the next after pasting a value, so that each value gets its own cell on the spreadsheet.

1. Type "row_num +=1."
2. Press **return** twice.

Closing the Workbook

The following lines of code close and save the workbook.

1. Type "workbook.close()."
2. Press **return** twice.

FINALIZING THE PROGRAM

As explained in section "Main Script Functions," none of the preceding code has actually been executed yet. Like a very conscientious test-taker, the script reads all of its instructions before beginning. The following portion of code signals that there are no more instructions to read before beginning to work.

1. Type "if __name__ == "__main__":."
2. Press **return**.
3. Type "main()."

The entirety of the code will look like this:

```
import pandas as pd
import xlsxwriter

DATAFILENAME = "ACSDP1Y%d.DP02_data_with_overlays_2020-03-30T222519.csv"

OUTPUT_XLSX_NAME = 'output.xlsx'

def main():
    workbook = xlsxwriter.Workbook(OUTPUT_XLSX_NAME)
    worksheet = workbook.add_worksheet()

    worksheet.write(0, 0, "Year")
    worksheet.write(0, 1, "Bachelor's Degree")
    worksheet.write(0, 2, "Graduate or Professional Degree")

    row_num = 1
    for i in range(2010, 2018):
        filename = DATAFILENAME % i

        df = pd.read_csv("DataFiles/" + filename, delimiter=',')
        list_of_rows = [list(row) for row in df.values]

        bach_idx = 0
        for column_name in list_of_rows[0]:
            if column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Bachelor's degree":
                break
            elif column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Population 25 years and over!!Bachelor's degree":
                break
            bach_idx += 1

        grad_idx = 0
        for column_name in list_of_rows[0]:
            if column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Graduate or professional degree":
                break
            elif column_name == "Percent!!EDUCATIONAL ATTAINMENT!!Population 25 years and over!!Graduate or professional degree":
                break
            grad_idx += 1

        bach_value = list_of_rows[1][bach_idx]
        grad_value = list_of_rows[1][grad_idx]

        worksheet.write(row_num, 0, i)
        worksheet.write(row_num, 1, bach_value + "%")
        worksheet.write(row_num, 2, grad_value + "%")

        row_num += 1

    workbook.close()

if __name__ == "__main__":
    main()
```

NOTE: Again, the indentation levels and color coding are all automatically formatted by Visual Studio Code. If your code doesn't look like the above image, check for errors in spacing, spelling, and punctuation.

SAVING THE PROGRAM

Until the code in the editor is saved, there is nothing to run.

1. Press **command + S** to save your code as a .py file.
   NOTE: The new .py file will be saved in the same folder that was opened in the explorer side bar of VSC. If the sidebar view is still enabled, the file will appear there.

RUNNING THE PROGRAM

Now that the file has been saved, the .py is ready to execute its functions.

1. Press the green triangle **run button** at the upper right-hand corner of the screen. The editor view will split to show the Terminal view below the editing window.

2. Several lines of text will appear; this may take a few moments.

3. You'll know the program has run completely when a file named "output.xlsx" appears in the folder shown in the explorer sidebar.

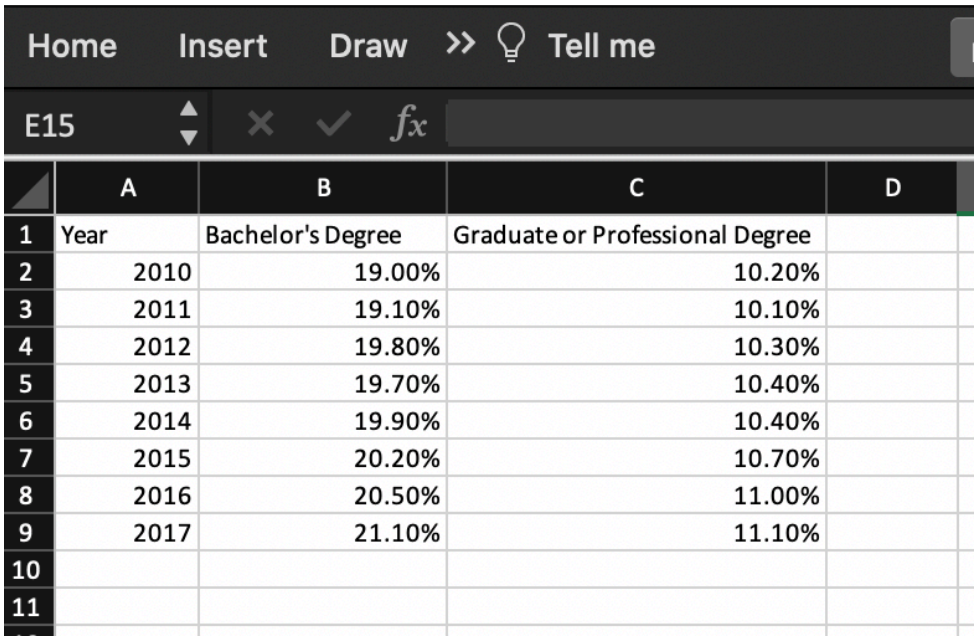
CHECKING YOUR WORK

To make sure everything went well, open your new .xlsx file.

1. Right-click the **file name** in the explorer sidebar.

2. Click **Reveal in Finder**.

3. Double-click **main.xlsx** It should display the header, rows, and data like this:



	A	B	C	D
1	Year	Bachelor's Degree	Graduate or Professional Degree	
2	2010	19.00%	10.20%	
3	2011	19.10%	10.10%	
4	2012	19.80%	10.30%	
5	2013	19.70%	10.40%	
6	2014	19.90%	10.40%	
7	2015	20.20%	10.70%	
8	2016	20.50%	11.00%	
9	2017	21.10%	11.10%	
10				
11				

NAILED IT!

Now that your computer is set up for Python and you've managed to compose your first functional bit of coding, you can ride your own coat tails on this accomplishment for as long as you find it useful. Because the basic functions of Python are standard, this script can be used as a template for future data-snipping endeavors. Simply copy and paste new reference files and rename your variables to save yourself as much time and effort possible.

When issues arise—and they will—visit github.com for a wealth of community-authored coding assistance. Search the existing resources before submitting your question—especially for relatively simple tasks like these, any related questions you might come across in this process have probably already been asked and answered.