RE: PDF & Web Scraping in Python

FROM: A. Zhu

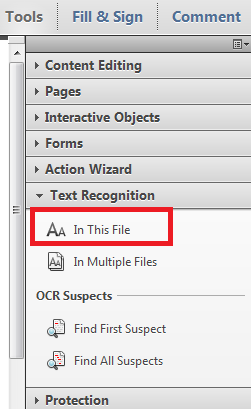
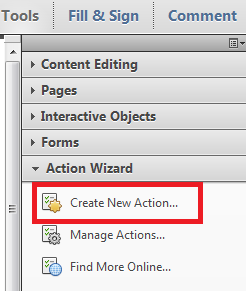
DATE: January 22, 2020

# Introduction

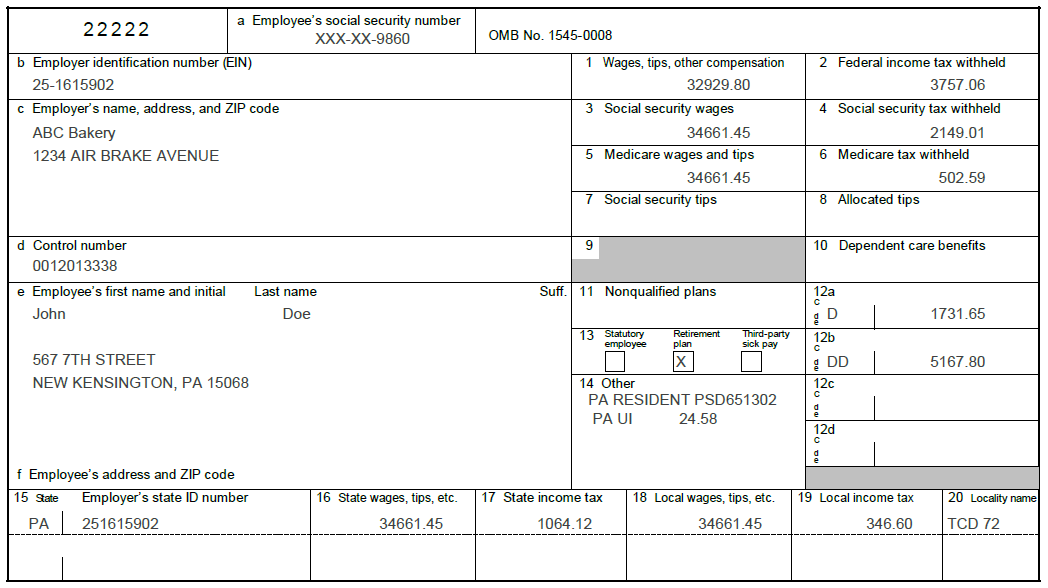
1. In most cases, we receive data from clients and third-parties in structured text formats (e.g., txt, csv, xls), which we can conveniently use Stata and SAS to import and analyze data. However, sometimes we might receive data in PDF format or need to get data from webpages. In many cases, these tasks will be difficult, if not impossible, with Stata and SAS. Python offer many utilization libraries that are well suited to develop automated scraping tools for PDF and web pages.

# PDF Scraping

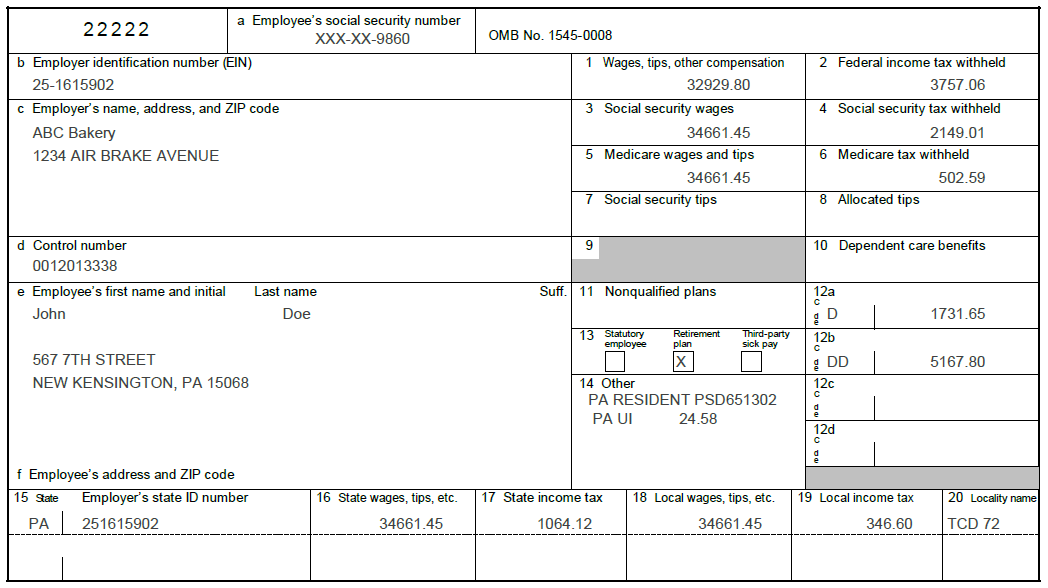
1. Python has several libraries (e.g. PyPDF2, tabula, etc.) that can deal with PDF files. In this memo, Python library, “pdfquery” will be used to demonstrate how to import PDF files in Python.
2. Before we import PDF data in Python, we need to check and see whether the texts are OCRed. OCR (optical character recognition) converts images of text into a machine-encoded text which allows a computer to interpret the text of the file (so it can be searched/copied/scraped). If you’ve ever opened a PDF and not been able to select the text, that is an example of a PDF that has not been OCRed. If a PDF is not OCRed, we will need to do some extra steps in Adobe Acrobat to get it ready for scraping.
3. In Adobe Acrobat, go to Tools -> Text Recognition -> In This File. Adobe Acrobat should start to OCR the PDF file. If you have multiple PDF files, we can set up an “Action Wizard” to automate the process and OCR all the PDF pages.

1. Once the PDF files are OCRed, we can start using Python to import the data. In following example, this PDF file includes unstructured W2 data, in which we don’t have row-column structure. Relevant information (e.g. employee’s SSN, name, address, employer, wage, etc.) are scattered all over in this W2 form.

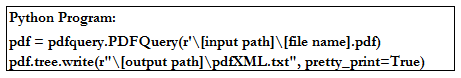


1. In the first step, we need to convert PDF into Extensible Markup Language (XML), which includes data and metadata of a given PDF page.



XML FILE

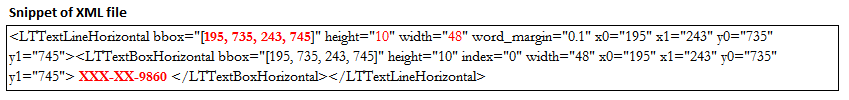
(Data and Metadata)



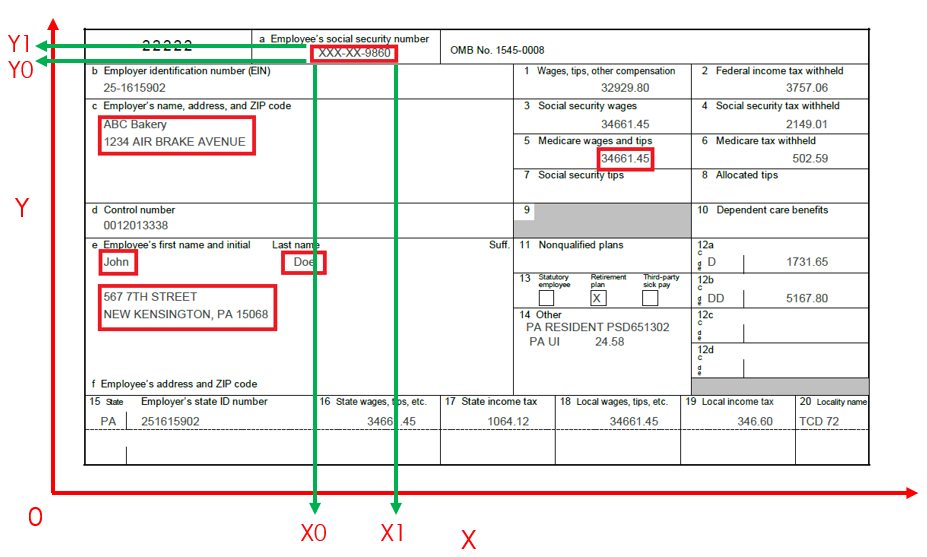
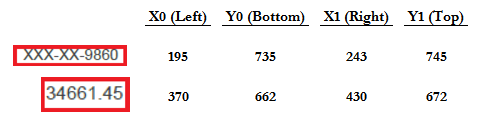
1. XML defines a set of rules for encoding PDF in a format that is both human-readable and machine-readable. Following is a snippet of XML for employee’s SSN. It includes both data (XXX-XX-9860) and metadata (e.g., text box coordination, height, width, etc.)

A screenshot of a cell phone

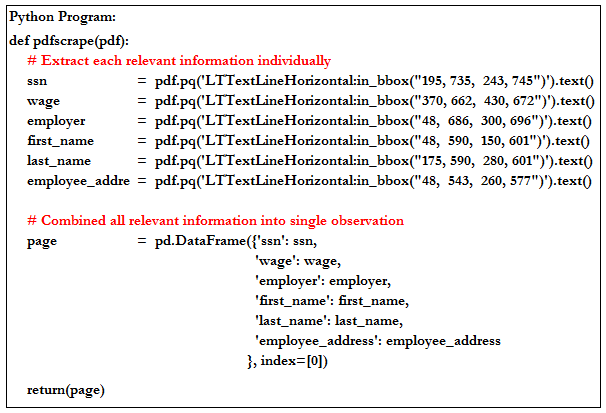
Description automatically generated



1. When you want to extract an element in a PDF, you should think about the page and the element location in terms of X-Y coordinates. The X-axis spans the width of the PDF page and the Y-axis spans the height of the page. Every element has its bounds defined by a bounding box which consists of 4 coordinates. These coordinates (X0, Y0, X1, Y1) represent left, bottom, right and top of the text box, which would give us the location of information we are interested in the PDF page. In following example, coordination [195, 735, 243, 745] indicates employee’s SSN (XXX-XX-9860) and [370, 662, 430, 672] indicates wage (34661.45).

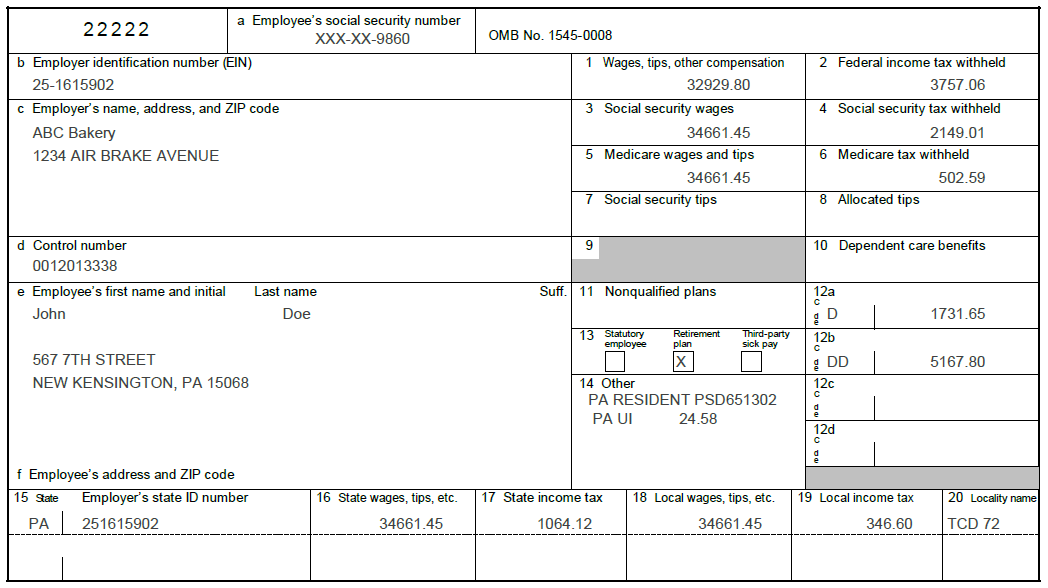
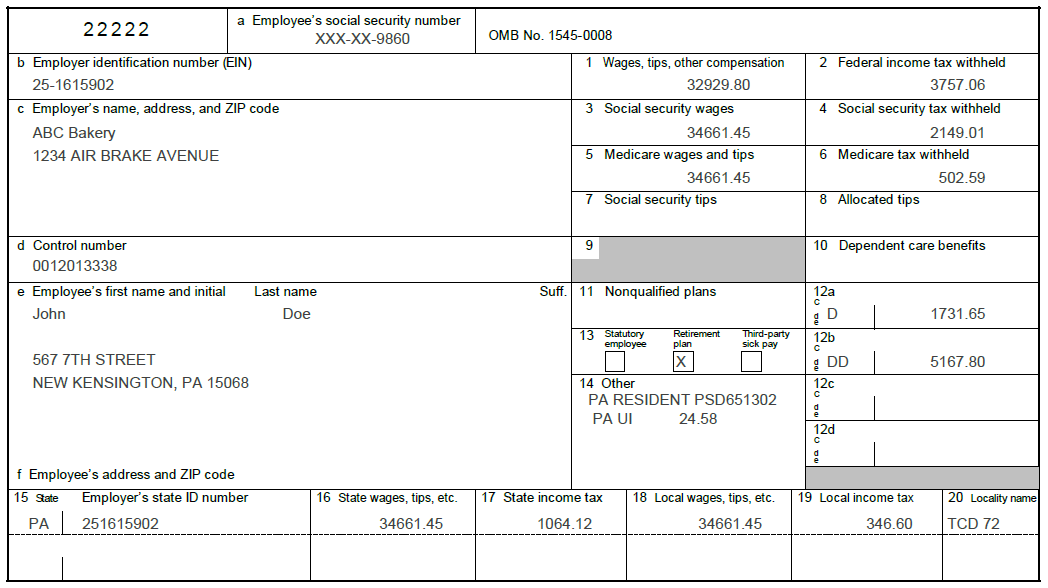
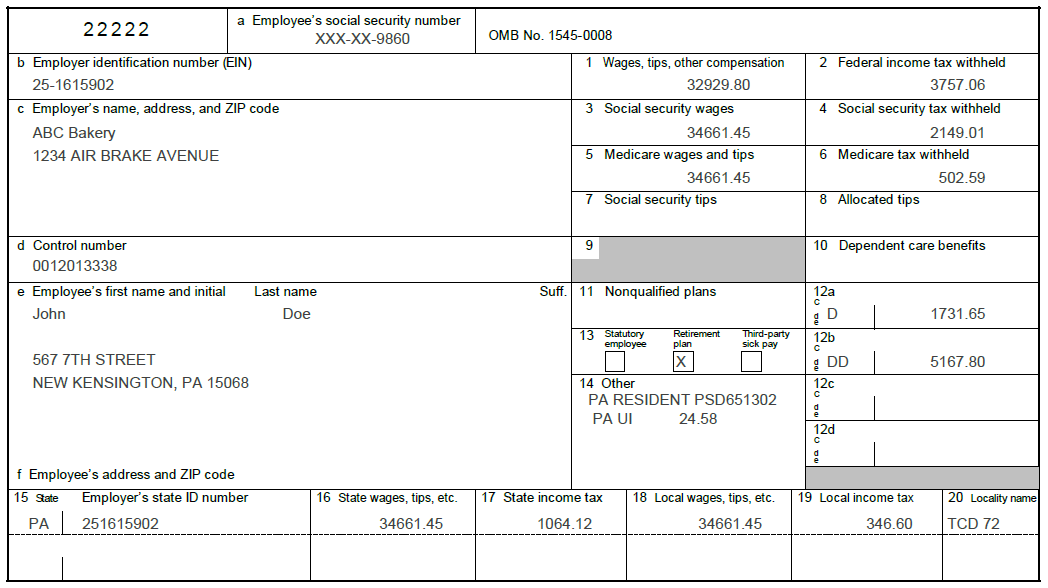
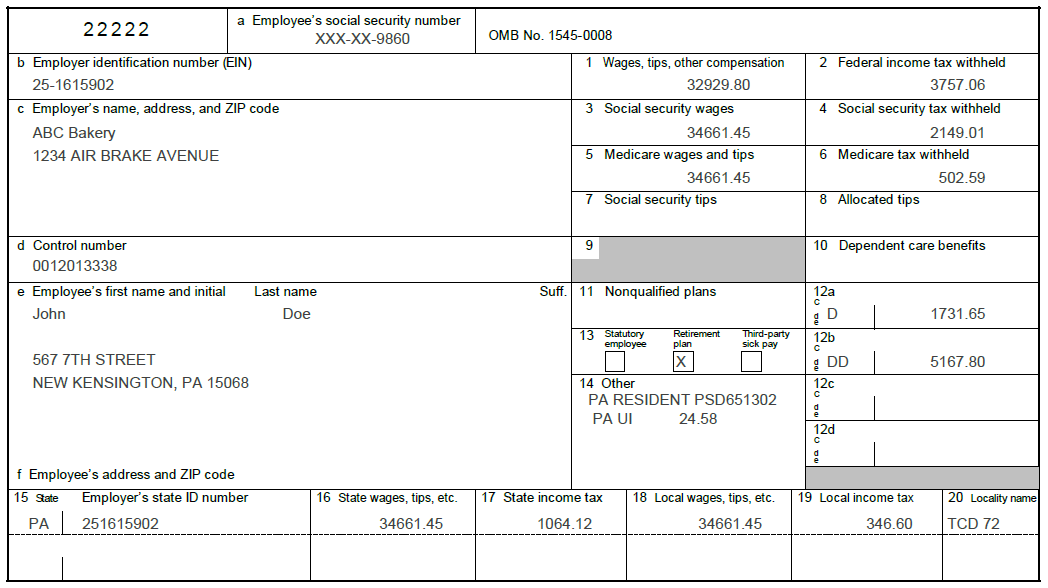
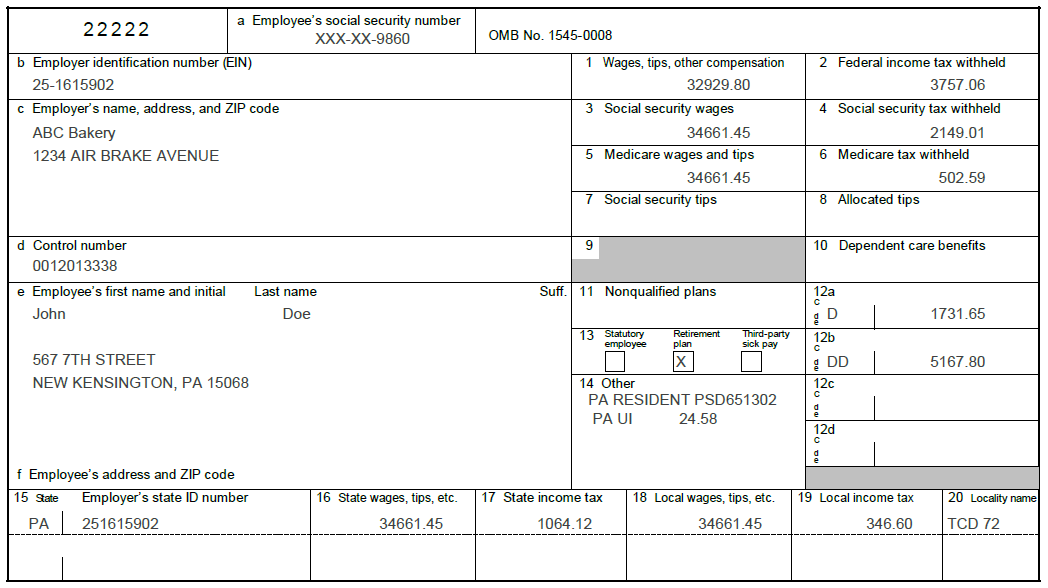
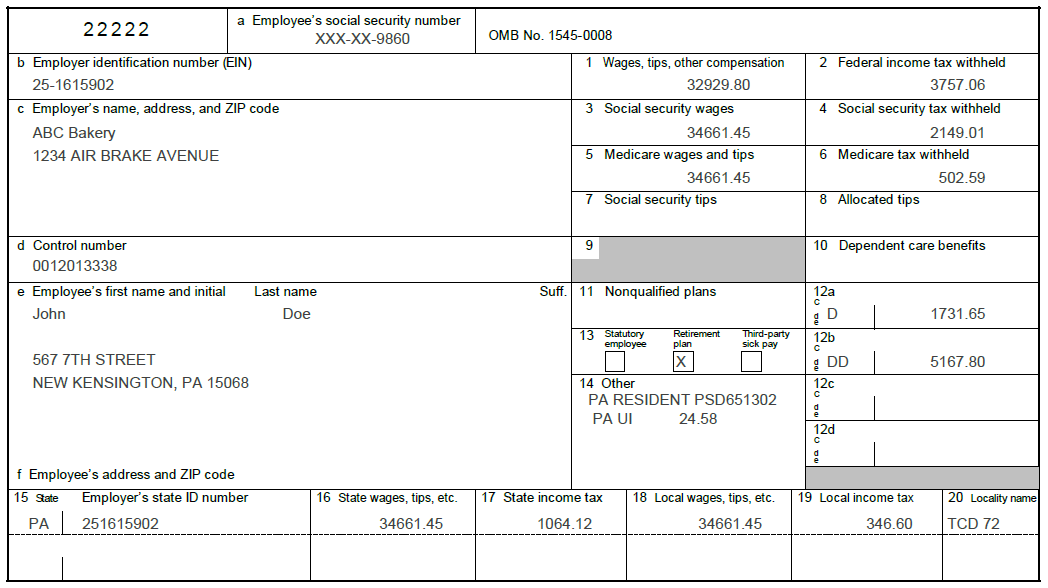
1. We can extract each piece of relevant information individually using their corresponding text box coordination, and then combined all scraped information into single observation.

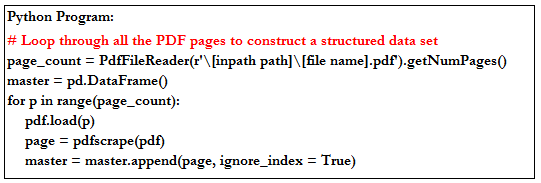
A screenshot of a social media post

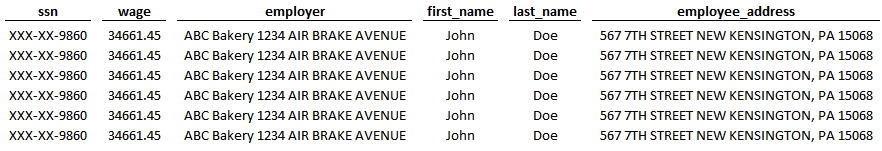
Description automatically generated



1. Information in subsequent pages/documents should not always be bounded in text boxes with the same coordinates as the first page/document. For example, length of home address varies from employee to employee so that, the coordinates of home address should be different from page to page. To overcome this issue, “pdfquery” allows us to use option, “LTTextLineHorizontal:overlaps\_bbox(X0, Y0, X1, Y1)”. As soon as the text box (X0, Y0, X1, Y1) we specify initially overlaps the text boxes in subsequent page, we should extract the correct information in all pages. For example, the text box [370, 662, 430, 672] overlaps all the text boxes for SSN in a PDF.
2. Once we correctly define the all text box coordinates, we should automate this process with the use of for loop and combine rows of observations into a data table. Now we’ve constructed a structured data table from an unstructured PDF file.







# Web Scraping

1. Many companies and government agencies keep records online. Price listings, employee directories, historic data (with the use of the internet archive), charts and graphics, file repositories, data in nearly any format can be scraped from the web and turned into a useful dataset.
2. Python has some useful libraries (Selenium and BeautifulSoup) that we can use for navigating and scraping websites.
3. Selenium is a framework that controls the browser interactions programmatically such as clicks, form submissions and mouse movement. In addition, selenium comes in handy when scraping data from JavaScript generated contents in a webpage.
4. BeautifulSoup is a Python library for parsing HTML, XML and other markup languages. If a webpage display data relevant to your project, such as, table, price, date or address, but that don’t provide a directly way of downloading the data. BeautifulSoup can help you parse specific content from a webpage, remove the HTML markup and save the information.
5. Following is what a typical website looks like. Within this given webpage, we might only be interested in some components. In this example, we are most interested in the bar chart for Apple worldwide shipments.



1. In the first step, we need to extract HTML (HyperText Markup Language) from a webpage. We can download pages using the Python requests library. The requests library will make a GET request to a web server, which downloads the HTML contents.



**HTML**

**(Source Code of Web Content)**

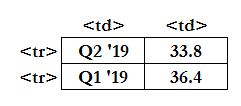


1. HTML uses “makeup” to annotate text, images and other content for display in a Web browser. HTML markup includes special “elements”, such as, <head>, <title>, <div>, <tr>, <td> and many others. Following is a snippet of HTML script for bar chart. In the following example, it includes both data (quarter and iPhone shipments) and HTML markup elements (<tr>, <td>), which we can consider them as row and column in a structured data.

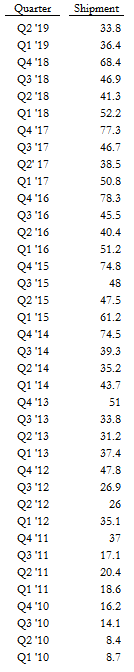
Source Code of Bar Chart

A screenshot of a cell phone

Description automatically generated

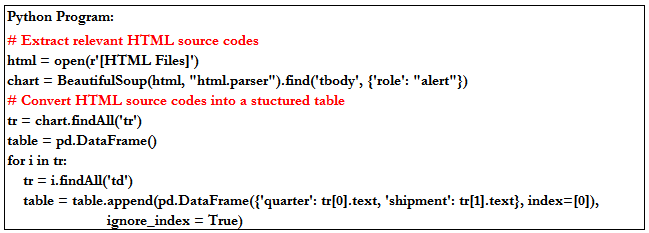


1. In the next step, we can use BeautifulSoup functions, such as, “find” and “findAll” to parse and clean the relevant content in the HTML by specifying the attribute of these contents. In this example, our data is included within <tboday> and has attribute of “role” equal to “alert”.
2. Another Python library, Pandas can also come in handy for data manipulation and creating a analysis-ready structure data.



**HTML Source Code**

**Structured Data**



# Relevant Tutorials

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PDF Scraping |  |  |  |  |  |
| <https://github.com/jcushman/pdfquery> | | | |  |  |
|  |  |  |  |  |  |
| Web Scraping |  |  |  |  |  |
| <https://www.dataquest.io/blog/web-scraping-tutorial-python/> | | | | | |
| <https://selenium-python.readthedocs.io/> | | | |  |  |