

**Assignment Week 07**

Sadia Afrin Jarir (n0033816619)

International American University

CSE550: Software Quality Assurance And Test Automation (Java)

Professor Zakir Hossain

December 09th, 2024

**Title: Reading an external file (csv/excel/txt) and printing every line in the VS Code Console.**

**Introduction**

Integrating file handling and web automation provides useful chances to expedite data processing operations in the current era of software testing and automation. Since CSV, Excel, and TXT formats are frequently used for storing and exchanging structured data, reading external files in these formats is an essential competency for quality assurance and automation engineers. In addition to increasing productivity, automating the real-time display of this data reduces the possibility of human error. The two main responsibilities of this assignment are to read data from external files and report the results in the Visual Studio Code (VSCode) console, and then utilize Selenium WebDriver to show the same data in a Chrome browser window.

By utilizing Python's robust libraries for file handling and the powerful automation capabilities of Selenium, we can simulate the dynamic nature of real-world data processing and display tasks. The output is shown on a random webpage, such as [RapidTables Notepad](https://www.rapidtables.com/tools/notepad.html), demonstrating how automation tools can bridge the gap between backend operations and web-based user interfaces. This approach reflects how software automation processes can be extended beyond traditional console outputs to web environments, making the results more accessible and interactive. Such integration is crucial for modern software testers and developers, who often need to validate data and system responses across multiple platforms. This assignment exemplifies the potential of web automation to increase productivity, improve accuracy, and enhance the flexibility of software testing workflows.

**Step-by-step process**:

1. First, we need to ensure that we have a desired file (csv, xlsx, txt) at our disposal. Here, I’ve made an excel sheet (.xlsx) with pretty simple data.

A table with numbers and text

Description automatically generated

1. Next, in a new file in Visual Studio, we need to set up our necessary databases (pandas, selenium) using pip function in the terminal (pip install pandas).
2. Once we have that set up, we can start writing our main code.
3. We start by defining the different databases we are going to use.

A screen shot of a computer

Description automatically generated

1. Then, we use the data function (df) to use our database pd to read our preexisting xlsx file. We use the following code to do so:



Here, **na\_values** removes all cells with no data in the output while **usecols** determines which columns the code is going to interpret the data from. I have chosen columns **A** through **E**.

1. Next, we want to establish a Selenium Webdriver in which we can display the output of our code. I have decided to use Chrome for this instance by using the following code:

A black background with white text

Description automatically generated

1. Now that our webdriver has been set up, we need to use a website we can use to represent our desired code in. Since, I have created an information based excel sheet, my goal is to display the information as I see fit on a notepad. For this I have chosen [RapidTables Notepad](https://www.rapidtables.com/tools/notepad.html).



1. We then create an iteration using for loop. What this iteration does is that it reads every data in each cell of a row before passing onto the next column. Here I have designated variables to each data for each iteration as follows:

A screen shot of a computer program

Description automatically generated

The **time.sleep(1)** delays each iteration by 1 second.

1. After this, we need to allow the code to find the editable area in the website and insert the data we have read in each iteration.



Here, the code is designed to find the element in the webpage denoted by “area” as its ID, which is the body of the notepad we are trying to display our result in. We then use **send.keys** function to enter each data we have read in the iteration into the “area” segment of the webpage. The sequence shown in the code is how the output is displayed. The ‘\n’ at the end of the line indicates a line break/new paragraph.

1. Then we end the code by adding a time delay (here I’ve used 5s) and a code to exit our webdriver.



1. Now our code is complete and if we run it, our webdriver will show the following:

A screenshot of a computer

Description automatically generated

*Pop-up of our designated Selenium Webdriver*

A screenshot of a computer

Description automatically generated

*Our destination webpage ([RapidTables Notepad](https://www.rapidtables.com/tools/notepad.html" \t "_new))*

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

*Our desired output from the data received from an excel file visualized in the webdriver*

**Conclusion**

As you can see, we have just created a python code that allowed us to first read specific data from an external file, which was an excel sheet we created with randomized data, and source that data onto a webpage displayed on our Selenium Webdriver. The application of each of these steps and the overall significance of learning how to source data from external files is extremely essential in python coding. I am extremely delighted to have had a good understanding of how this works and implement my own version of this code in this paper.

