**Data Cleansing Tools**

Camilo Hoyos

DAT-375 Data Analysis Techniques

Jacqueline U. Clauss, M.S.

February 7, 2021

**Introduction**

I will be comparing the data cleansing capabilities of Excel and Python. For simplicity there will be two data sets to work off of. A mixed list of string and integer and a list of just integers. Each section of actions will show the initial data set, the method for Excel, the results from Excel, the method for Python, and the results from Python. It should also be mentioned that the examples provided are simple. There are more elaborate methods that work on more elaborate data sets.

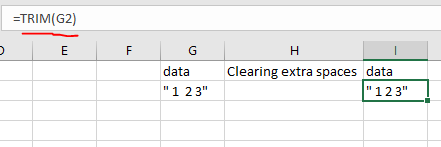
**Datasets**

data = [" 1 2 3", 456, "", " ", "123", 456, 456, 605]

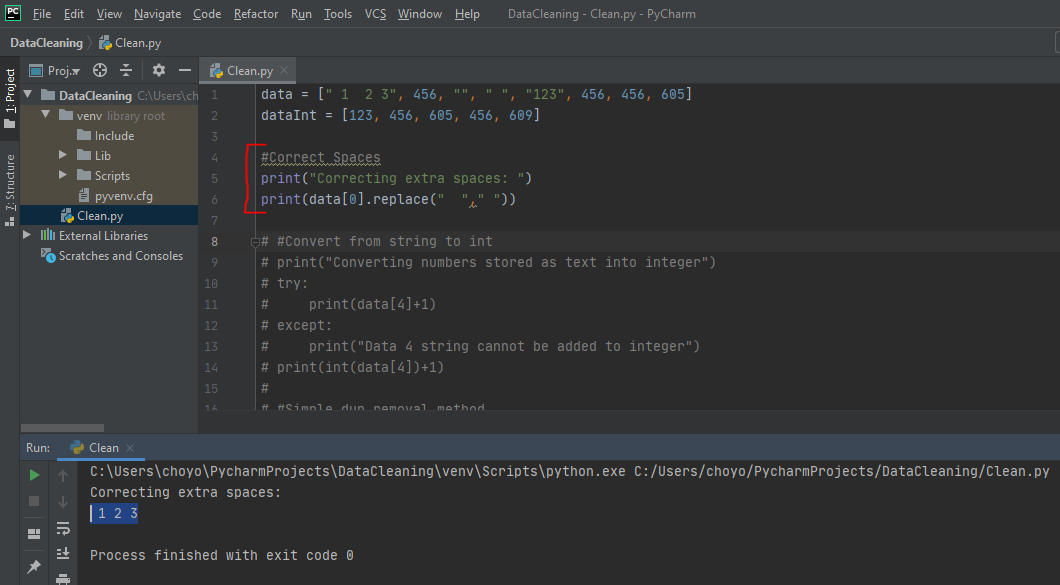
dataInt = [123, 456, 605, 456, 609]

**Clearing extra spaces**

In Excel, there is a function to remove multiple spaces that are sequential called ‘TRIM()’. There are two spaces in between 1 and 2 which it converts to one space.

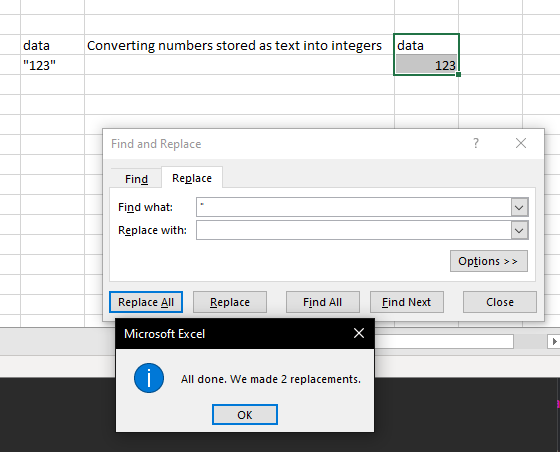


In Python, it can be done through a .replace() function. There are more advanced options such as .strip() which can trim excess whitespace in the front and back of a string.



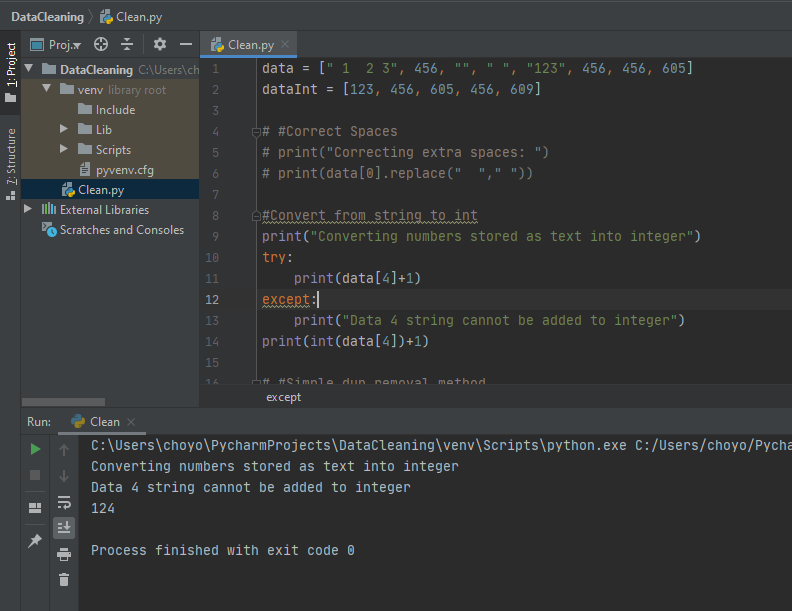
**Converting numbers stored as text into numerals**

In Excel there are a handful of ways to convert text to numbers. It depends on how the cell is considering the value as text or integer. In the scenario that has been created, it is simply because double quotation marks surround the number which Excel will forcefully interpret as a string.



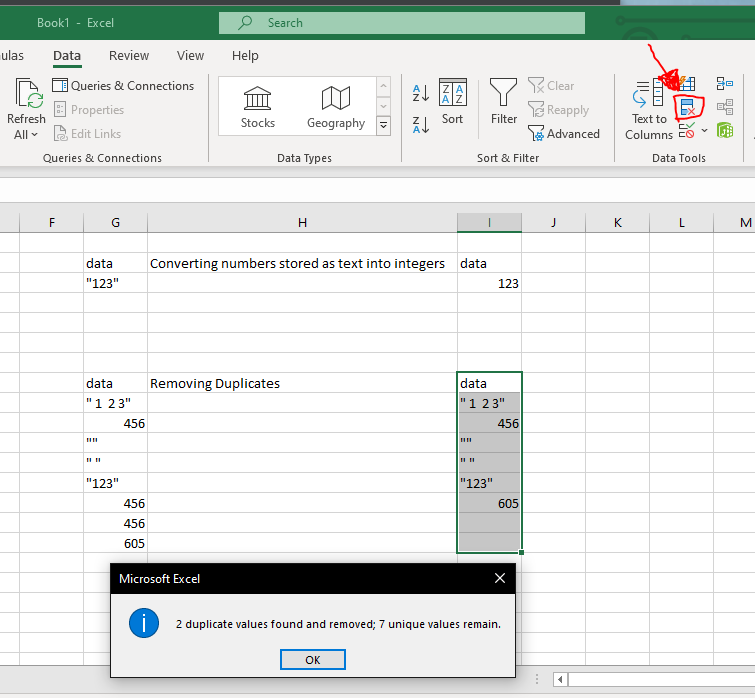
By removing the quotation marks, Excel will automatically interpret the cell as an integer. You can identify this since strings in Excel are left-aligned and integers are right aligned.

In Python there is a slightly more systematic approach as you can cast a value using int(), str() and many more. In the example below you can see the exception triggers a message to print instead of the added value as it is not casted to int. However, in line 14, the value is casted to int and then is capable of applying arithmetic.

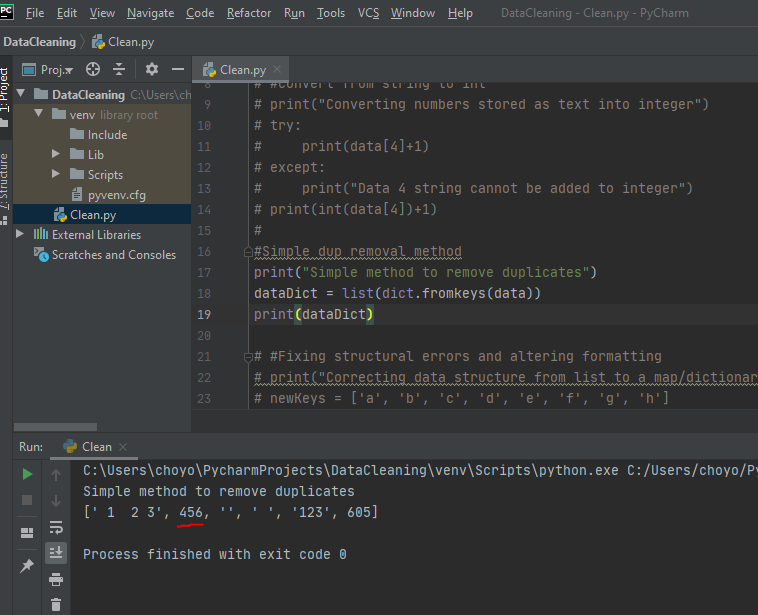


**Removing irrelevant or duplicate data**

In Excel there is a built in command for removing duplicates. In the data tab there is an icon (circled below) that will remove duplicates. Simply select the cells you want evaluated and navigate through the wizard.



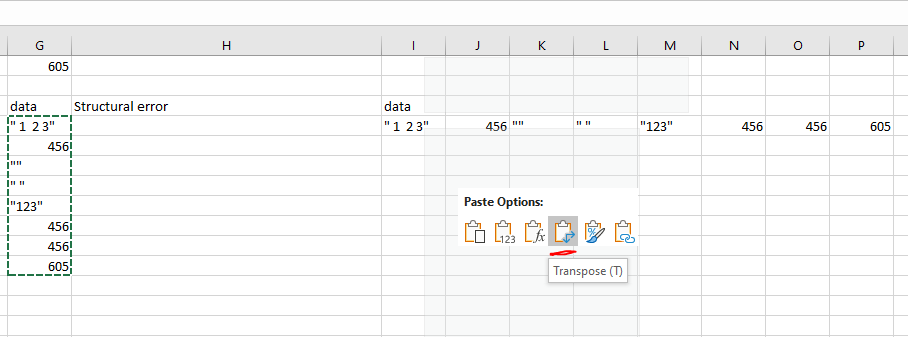
In Python, a simple solution is to assign an entire dataset to a dictionary as this will assign the values to keys. Dictionary key rules do not allow for duplicate entries so they will simply be removed and you will end up with a distinct list of values as seen below.



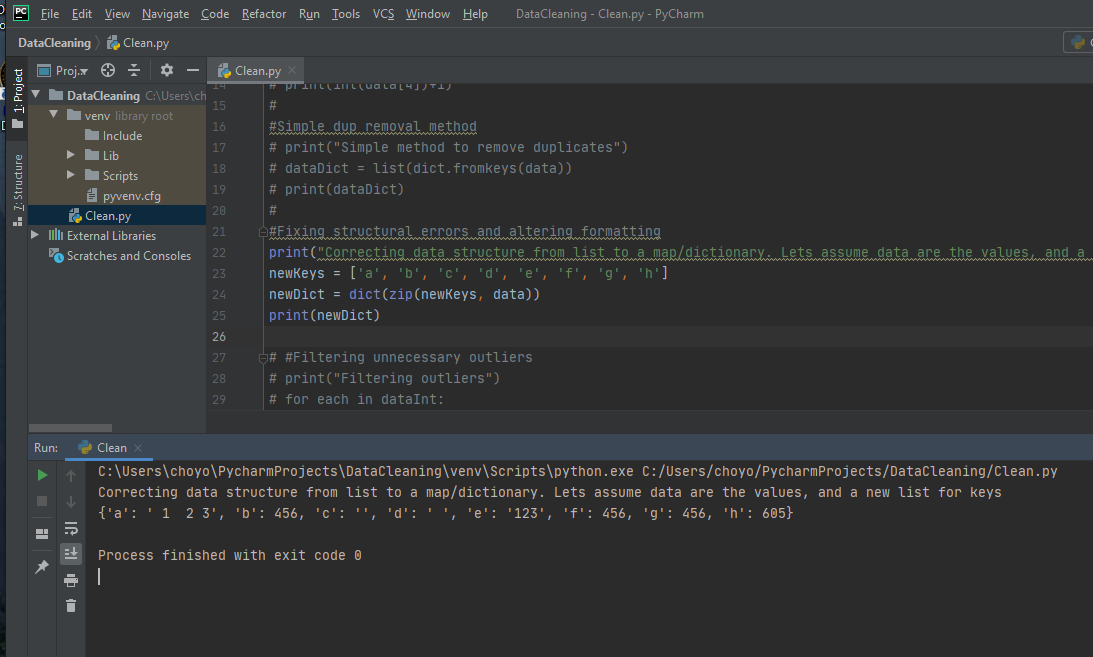
**Fixing structural errors and altering formatting as needed**

This scenario is vague so I will be using two different methods just to show flexibility in each tool.

In Excel let's say you receive a vertical list that needs to be reconfigured to a horizontal list. You can use the ‘transpose’ function. This can be valuable whenever you have to convert columnar data into row data. There have been many times I have needed to create a horizontal list for consumption or vice versa.

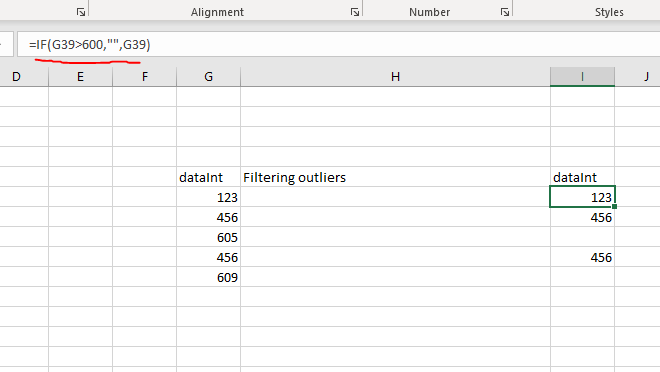


In Python, I will show how to combine two lists to create a dictionary. If you have a list of values in which you want to assign a set of keys you can use the zip() function. This is particularly useful if you want to key value pair a set of values for further data manipulation. Note that if you have duplicated key values it will add any paired values to the same key.

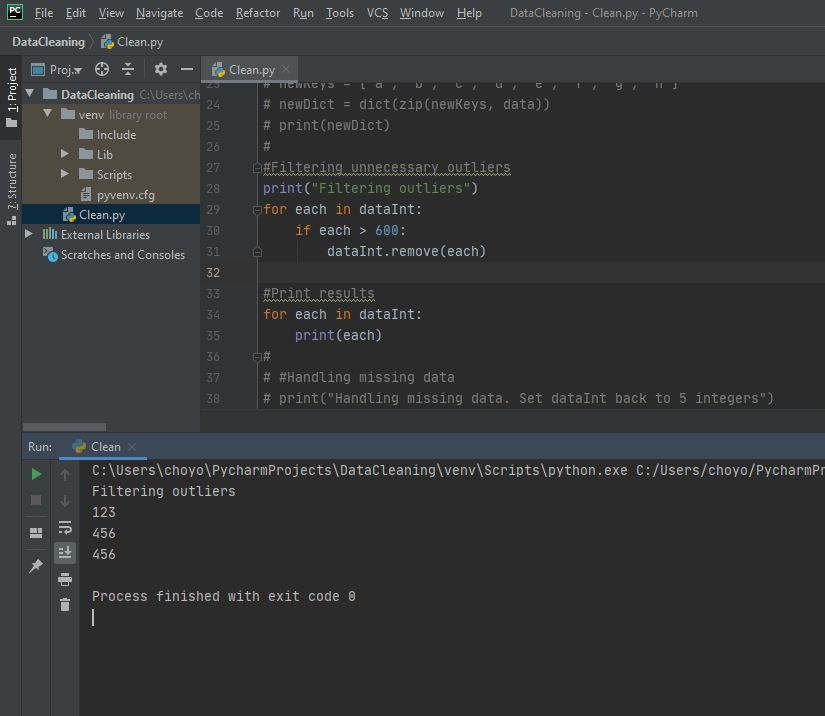


**Filtering unnecessary outliers**

Using the new dataset ‘dataInt’, let's assume outliers are greater than 600. In Excel you can create a conditional statement to clean out outliers. Using IF() to evaluate the value and determine if to assign as empty (or “”) or leave the value alone.



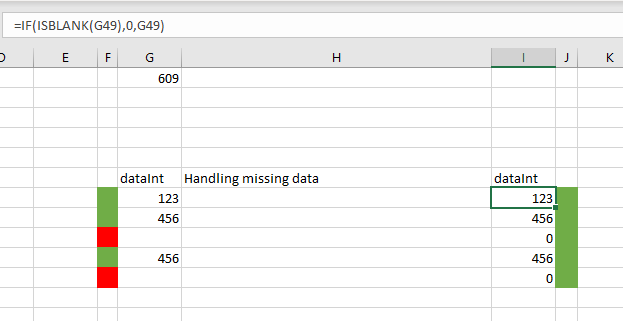
In Python, you would execute an equivalent procedure. Evaluate each value in the list and remove those who pass the outlier criteria.



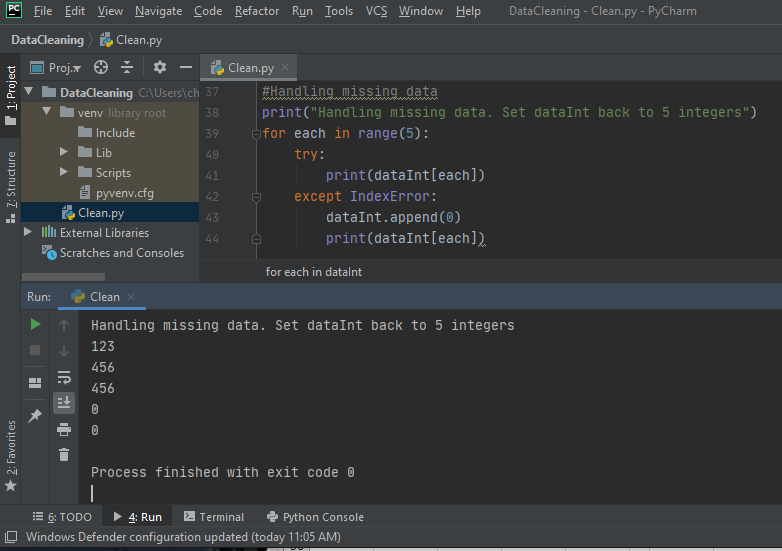
**Handling missing data**

In this scenario we’re going to insert a placeholder value of 0 into a dataset with missing values. Since we removed values greater than 600 in the last scenario, we will insert 0s back into the dataset.

In Excel you can do this as a conditional using IF() and ISBLANK(). ISBLANK() will return true if the field does not contain any values. Our IF() statement will insert a 0 where it cannot find a specific value.



In Python, we have to take a similar approach. First we have to define the range of expected data and evaluate if we find an error. By using try with an exception for IndexError we can find empty values where something was expected. You’ll note this provides different answers compared to Excel. This can all be combined in either method to give the same results. Excel saves row positions which is why the inserted 0s are in the old positions of the values that were greater than 600. Whereas Python cleaned old positions and condensed the list.



**Summary**

In summary, both tools have some similarities in how to execute data cleansing. Where Excel is better is that some of the common cleansing needs are available through built in features. However, Python has pure flexibility in that it is a general purpose language that can handle nearly any scenario given to it. (Devlin, 2019) Python also has a multitude of libraries that can assist with data cleansing, whereas Excel is limited to built-in features. In the end, for larger unstructured data sets, Python will be better set to cleanse data. Excel works for one-off, structured, small data set cleaning. (Chou, 2019)

References

Chou, L. (2019, May 28). Comparison of data analysis tools: Excel, R, Python and bi tools. Retrieved February 06, 2021, from https://towardsdatascience.com/comparison-of-data-analysis-tools-excel-r-python-and-bi-tools-6c4685a8ea6f

Devlin, J. (2019, December 13). Excel vs python: How to do common data analysis tasks. Retrieved February 06, 2021, from https://www.dataquest.io/blog/excel-vs-python/