# Using the UMSL Home Repair Cost Estimate (CE) Tool

The UMSL Home Repair Cost Estimate (CE) tool facilitates the process of estimating home repair costs from datasets of any size that meet its input criteria.

## How does the UMSL Home Repair Cost Estimate (CE) Tool Work?

At a high-level, the tool is a program that reads in a dataset that includes survey and respondent data. It also reads in a standard 'csv' file that ties a specific numeric cost to a home repair parameter (i.e. window replacement, 400). Finally, it checks whether the participants individually need a given repair, and if so, it indicates which and calculates that individual's total estimated cost.

## What do you need to do this?

1. From a personnel standpoint, it would be helpful to have an individual with at least some experience with computer programming with any popular coding language. This is not necessary, however, as these instructions are intended to be able to guide anyone generally comfortable with computers toward a successful outcome through following the steps carefully.
2. You will also need a few software tools, to be detailed below, and familiarity with Microsoft Excel.

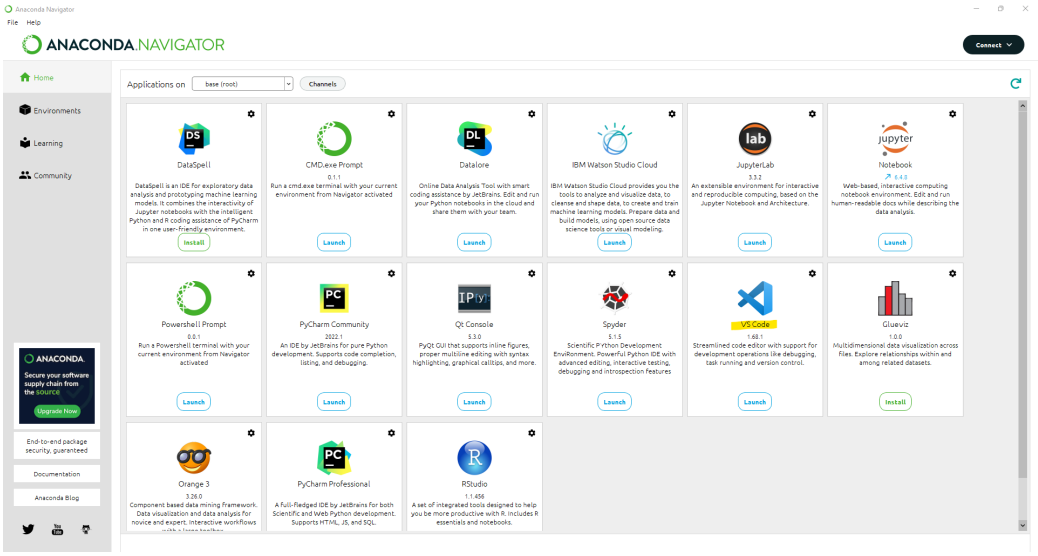
## How can I prepare my dataset to be used with the CE Tool?

1. Most datasets can be easily utilized and adapted to fit the tool as it is created with flexibility and efficiency in mind. However, the dataset must include the survey responses for each participant.
    a. In our dataset, we have a column that has a survey ID. We then have over 50 columns with identifiers like "Q1", "Q2", "Q3", and so forth. The "Q#" columns hold a numerical value generally between 1 and 4 indicating a different response to a survey question.
2. Create a csv or similar file that associates costs with given repairs. Ours is very basic. One column name is "Name", the other is "Cost", and under each we tie a given home repair to a cost. You can have as many or as few as you wish and you can change the costs easily at any time.
    a. This csv file is included in this guide as a separate file and the formatting is very simple.
    b. Example:
    name,cost
    btu,1.249
    serviceheatingequipment,284
    replaceheatingequipment,4655
    serviceacequipment,267
    …

## Getting Started with the CE Tool

1. The CE Tool was developed using the **Python** programming language within **Microsoft's Visual Studio Code (VSCode) program.** VSCode, from Microsoft, is free for educators and students. You

can download VSCode from Microsoft directly, although many Integrated Development Environments (IDE), through which you enter and run code, exist that work with Python.

2. The CE Tool also utilizes a popular data science extension for the Python language called PANDAS. You can quickly download and install PANDAS through a company and website 'Anaconda', via the link below:
    a. https://anaconda.org/anaconda/pandas
3. Once PANDAS and Anaconda are installed, open VSCode or your preferred tool through the Anaconda interface on your computer (see below).
    a. 

4. Once VSCode is open through the Anaconda interface (just above), you can proceed by importing the attached .py (Python) file. Additionally, there is a csv file (previously mentioned) for costs that you can utilize and update to your liking.

## Understanding and Adapting the Code

1. For your costs, simply change the numerical value for an associated cost, delete whatever costs you don't wish to utilize, and add whatever costs that you feel might be missing.
2. In VSCode, Create a "Workspace" -> File -> Select a Folder on your hard drive. This is going to be the folder that VSCode reads the data from and where you should store the files relevant to this tool (the costs.csv file as well as the home_repair.py file).
3. Within VSCode, open the costs.csv and home_repair.py files.
4. Now direct your attention to the home_repair.py file within VSCode. The 'import' statements at the top of the code are necessary for it to perform its functions, so you must keep them. These are three lines with technical and necessary meanings that won't be necessary to cover, but they must be included for the code to run.
5. Reading in your dataset and sheet name:
    a. Take the "file_name" variable and set it equal to file and directory path of your Excel worksheet containing your survey or similar data. This is on LINE 5 of the home_repair.py file.
    b. Take the "sheet" and set it equal to the sheet you are looking to read in from the Excel worksheet. That may simply be "sheet1" or something more specific. LINE 6.

c. Next is the “answerkey”, a line that allows the tool to read the information processed in step a on line 5 as text information and index the information around each individual’s survey ID. LINE 7.
d. The line for the costs variable reads in the costs.csv sheet that has costs and variables together. You can, of course, change this to whatever csv file you like. LINE 8.

6. You may, of course, have certain multipliers or numerical variables that you’ll need to perform math with. The program otherwise reads in the dataset as a String type of variable, which you can consider essentially as text. We want to perform calculations that involve building square footage and the number of floors, however, so you can convert whatever columns to numeric form as seen in lines 11-12.
7. Below is an example of the code logic that checks for given repairs and then returns a value for that participant later in the spreadsheet that the program outputs:

```
def check_weatherization1(x):
  if x["Q3"] == "1": return costs.weatherization1
  if x["Q3"] == "2": return costs.weatherization1
  return 0
```

a. This function is utilized to check whether a participant needs weatherization work completed on their home.
b. If a participant responds with a 1 or a 2 to question 3, then the program returns the weatherization cost determined in the costs csv file.
c. You can do this logic for any variable that you are exploring.

5. We found it valuable to have a column for each variable in the final output that showcases whether a given participant had any type of repair. For example, we wanted to see whether a participant had their heating equipment serviced, so we would set a column in the final output file as such:

```
df["serviceheatingequipment"] = df.apply(check_serviceheatingequipment, axis=1)
```

6. To sum up an individual’s costs, we have a line near the bottom of the code titled df[“TotalCost”] which creates a Total Cost column in the final output file. It simply adds the costs for each participant for whatever repairs they needed.

## See the End Result

1. Decide what you want your output file to be named. For us, we used “CEOutput” for Cost Estimate Output.
   xlsxfile = 'CEOutput.xls'
2. Click Run and select Start Debugging or Run Without Debugging. More likely than not, you’ll have a few typos and syntax errors before it works. However, the terminal in VSCode should help you identify where the issues lie.
3. Once the program runs successfully, you should have an output file in your directory that you can view.