Data Processing and Database Project

By

Juan Moctezuma

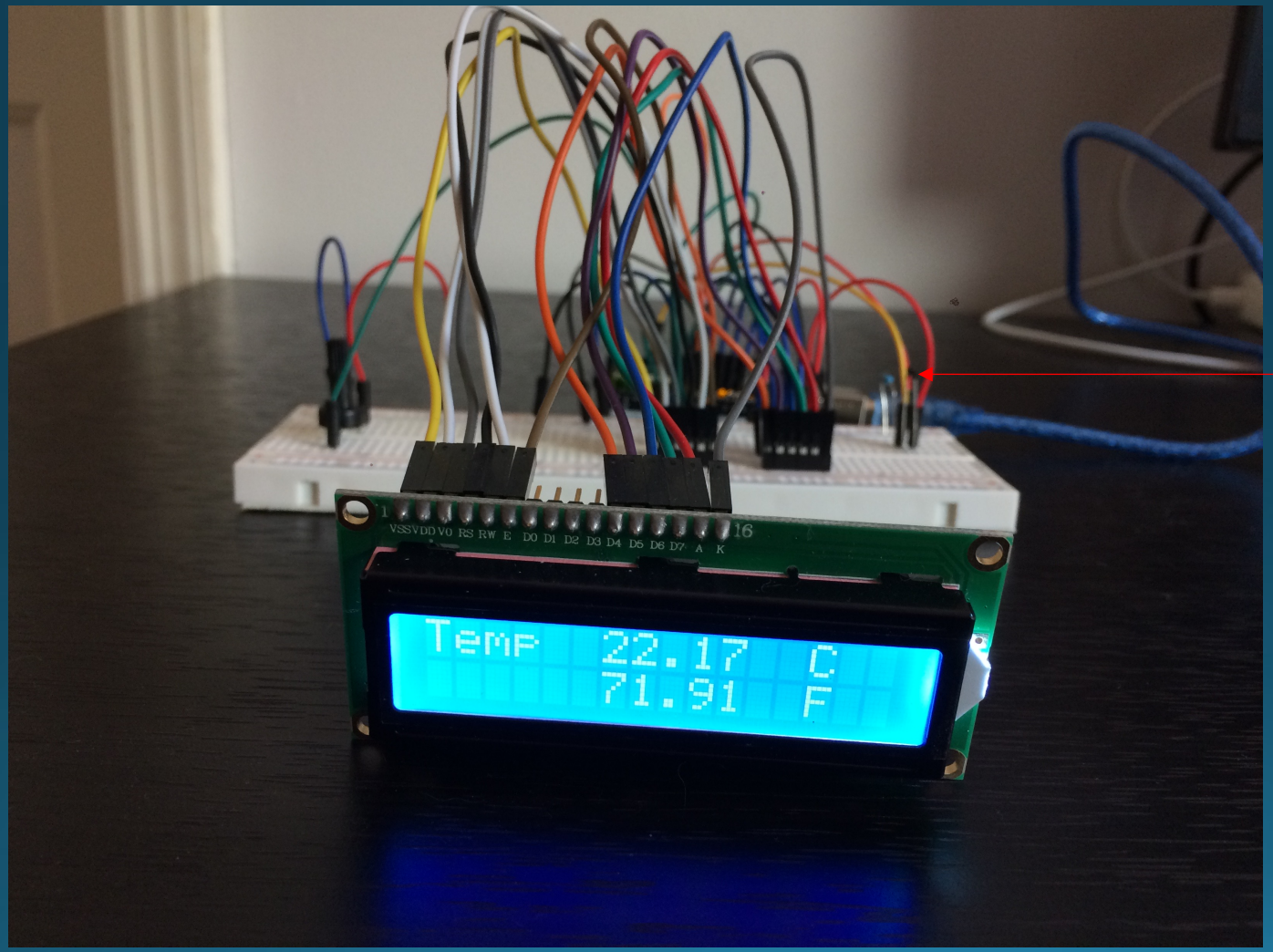
What is this project about?

- Demonstrating how raw data (room temperature) can be obtained from a source; in this case a device.
- Preparing and modifying the information obtained directly from the source throughout a series of steps.
- Making use of the available technological tools required to develop a final product (an Excel spreadsheet with results nicely organized and displayed) while maintaining high quality data.
- Creating a database (hence a SQL table) for storage or record keeping purposes.

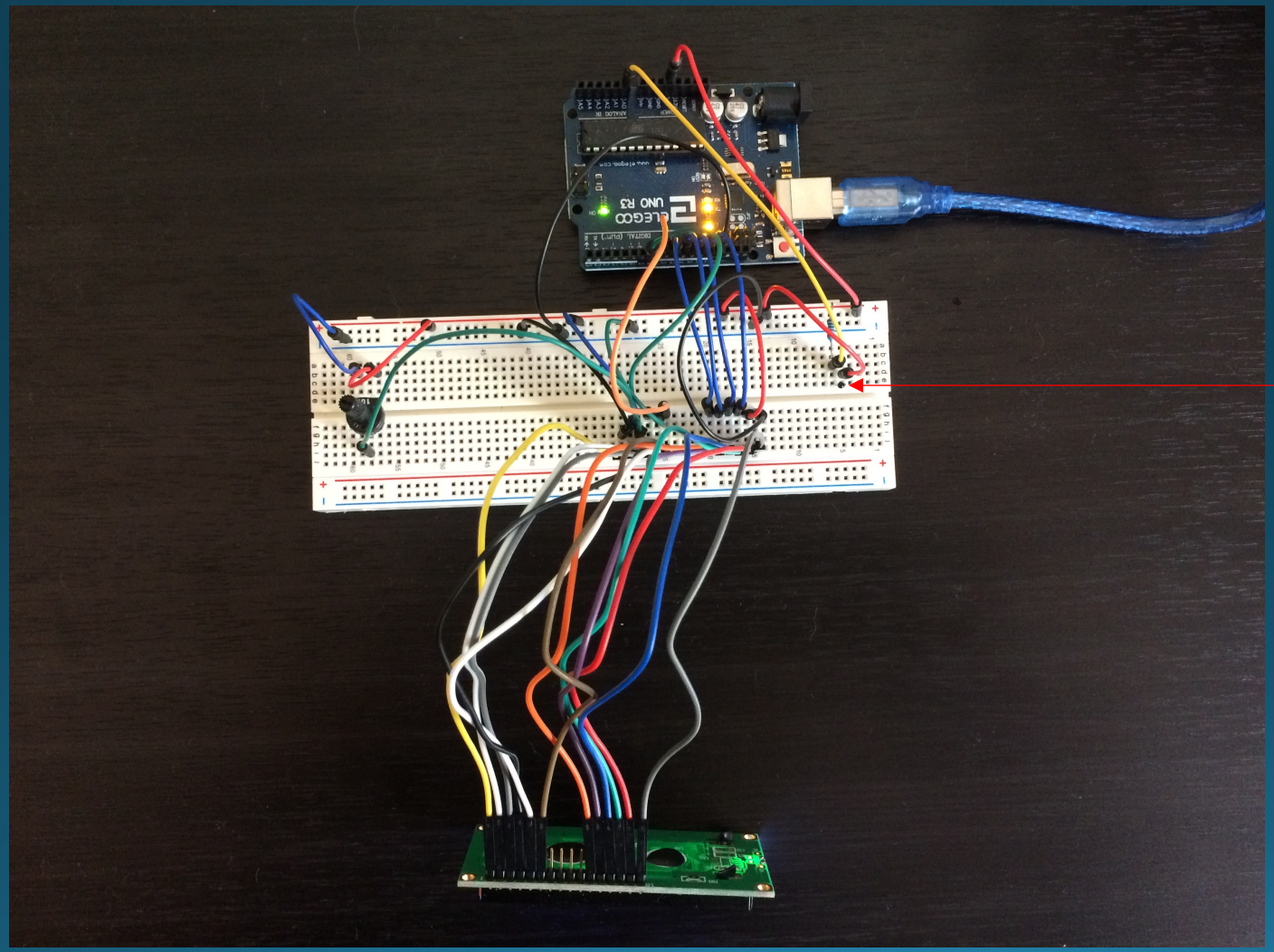
Where is the raw data coming from?

- The source producing data or information is a temperature sensor attached to an *Arduino UNO R3* circuit board and other essential electrical components. Basically a digital thermometer. Please note that this project will not go into details related with Electrical Engineering or the principles of electricity.
- The 3 following slides will show images of the actual thermometer, temperature sensor and the connection schematic (instructions of how the device needs to be physically connected).

Digital Thermometer – Front View

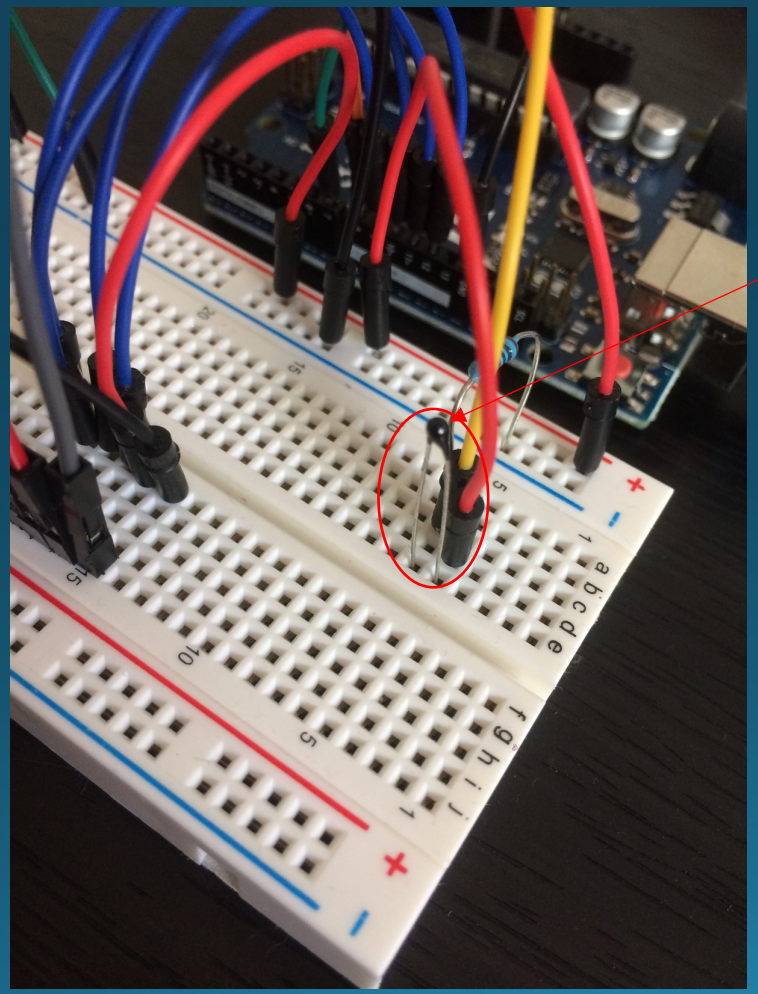


Digital Thermometer – Top View



Temperature Sensor

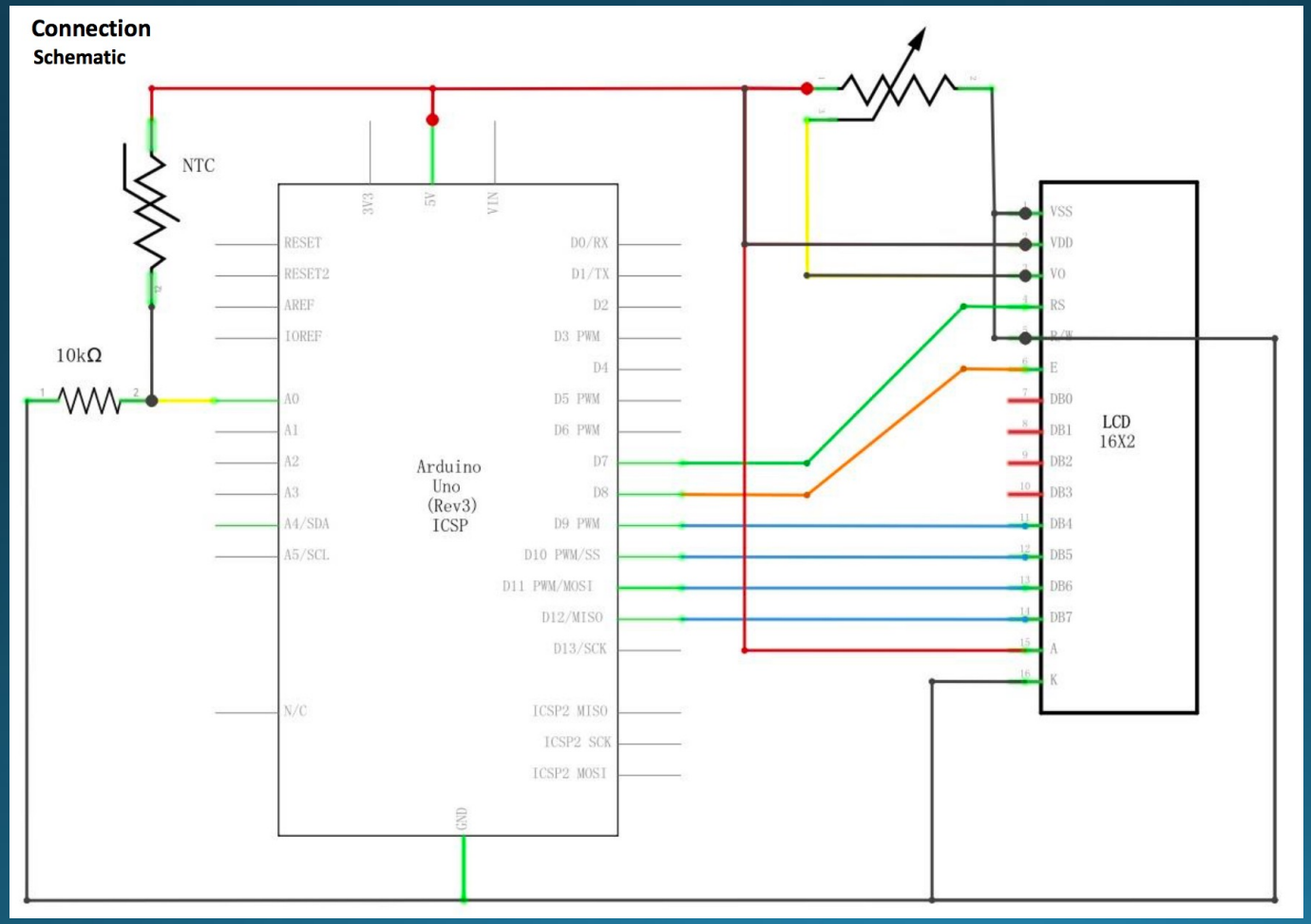
- Readings are measured by the small black structure within the red oval.



Note: If this black component is touched directly, readings will reflect body temperature instead of room temperature.

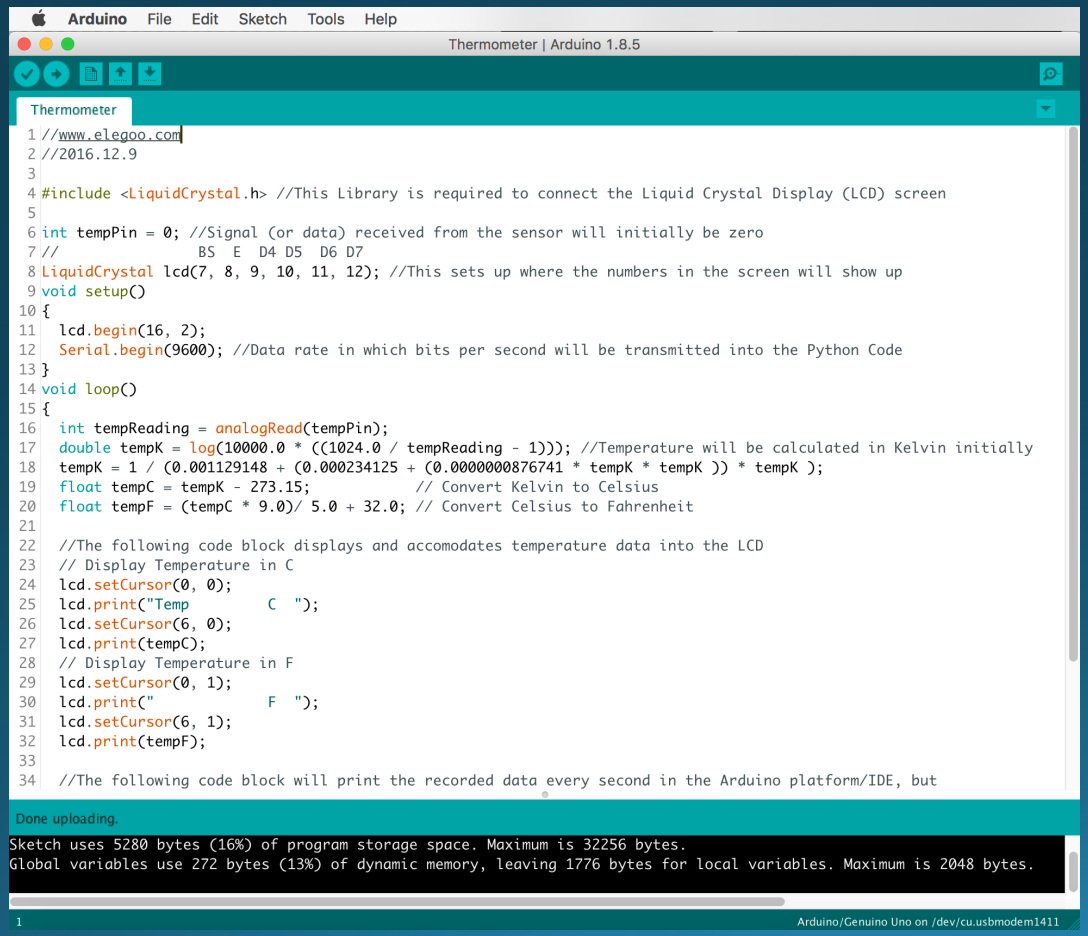
Electric Diagram or Schematic

- Note: The following image was retrieved from *Elegoo Super Starter Kit for UNO*. I did not create or modified the picture provided in this slide.



How do we make our thermometer work?

- After the device is wired up and connected to a computer via USB, we need to open Arduino's open-source Integrated Development Environment (IDE) and run the code written in C/C++.



What role does Python play?

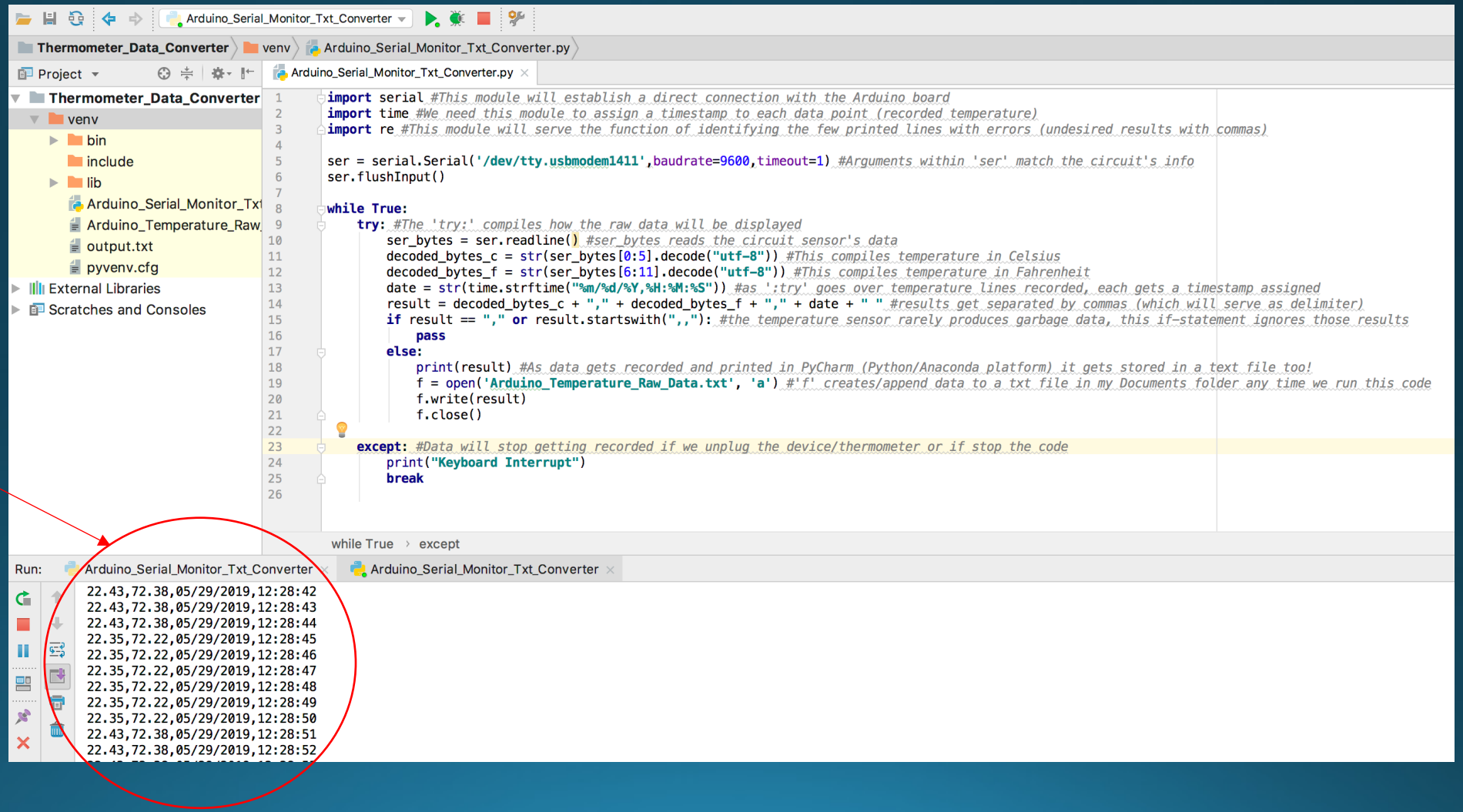
- We need Python 3.6 in order to establish a direct connection with the digital thermometer and obtain its data or measurements. This needs to occur as the code from the other device is running at the same time as well.
- Once these are connected and PyCharm (Python's platform or IDE) starts receiving live/real time data, each record (temperature in Celsius and Fahrenheit) will get joined along with a timestamp and then separated by a comma; timestamps also get split into date and time, and separated by a comma.
  - For instance, if room temperature was 70° at some point in time, then the record gets compiled as "21.47,70.64,05/28/2019,17:59:05".

What will happen to the compiled room temperature records?

- The Python script is also programmed to gather each record into a text file conveniently named *Arduino_Temperature_Raw_Data.txt*, and then get that file exported into my personal computer's Documents folder.
- This text file containing the raw data is the Python code's 'output', which will continue to modified (with Microsoft Excel – Macro/VBA) after recording the desired amount of data.
- For the sake of simplicity only 2-day worth of data was obtained by leaving the python script along with the thermometer running for several hours during two consecutive days.

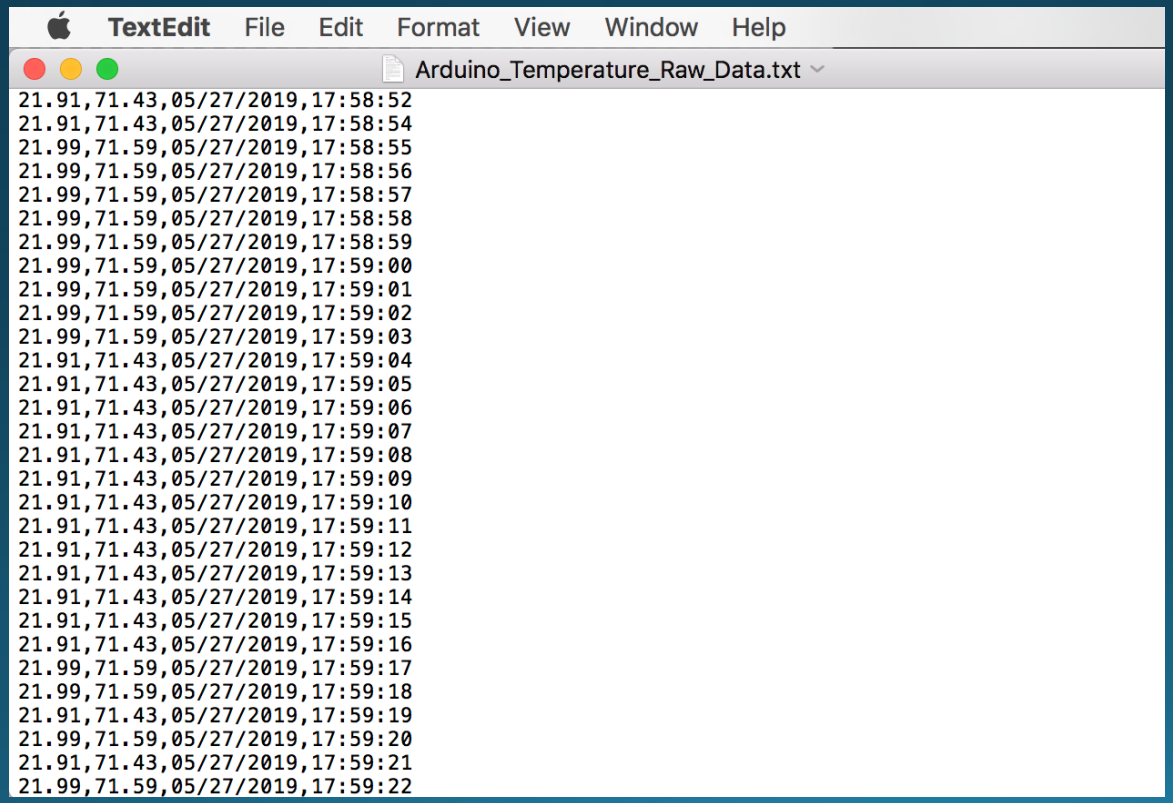
How does the Python script and IDE look like?

Each record gets displayed every second in the same format as in the example provided in slide 9. As long as the thermometer is running, data won't stop compiling.



How does the output look?

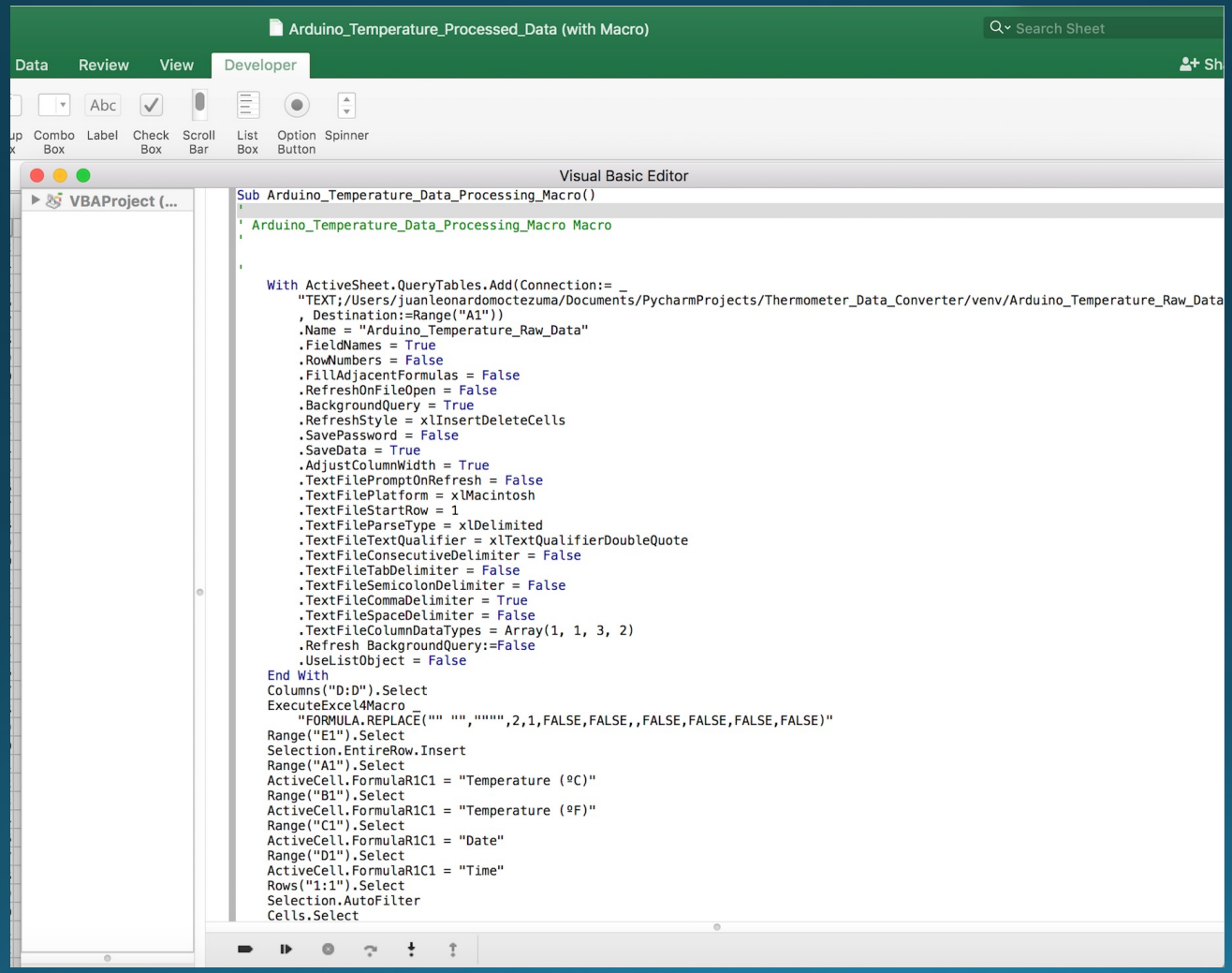
- As mentioned previously, the output containing the raw data is simply a list within *Arduino_temperature_Raw_Data.txt* containing every record.



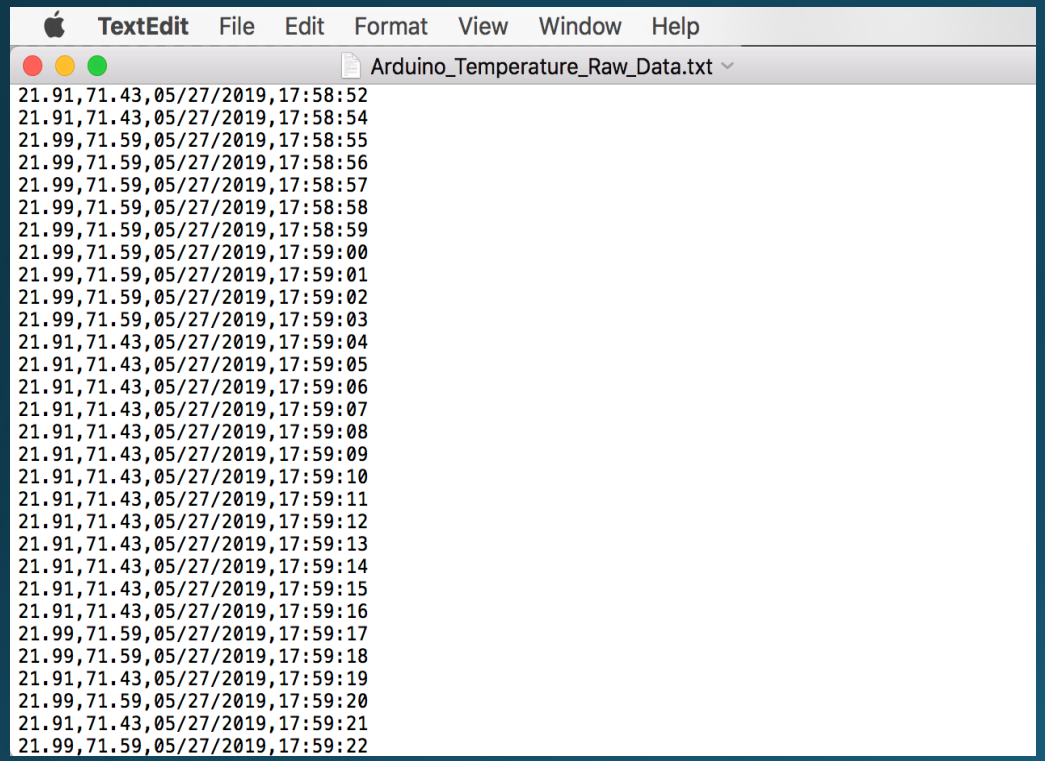
What will the Excel Macro do to the output text file?

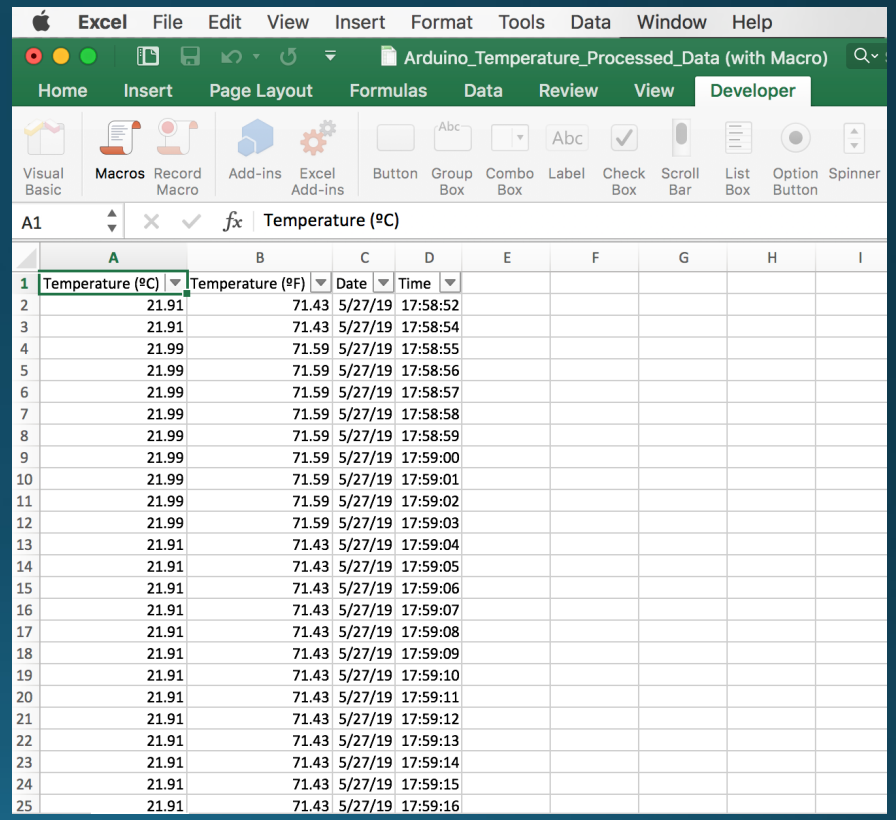
- Once the raw info was imported in Microsoft Excel as a text file, a Macro recorded every step that was performed in order to obtain the processed (completed) data.
- The steps were as followed:
  - Created columns based on delimiters (commas).
  - Replaced every blank space created after Time column (Column D) was imported. Note: I wasn't able to figure out why my Python script was creating an unwanted space in every record, therefore this step was required to be done in Excel in order to ensure quality and avoid formatting-related errors.
    - For instance: "17:58:52 " has an extra place to the right of number two, which was replaced with no blank spaces at all, such as "17:58:52".
  - Inserted four headers in the first row and assigned them filters. Headers consist of Temperature (ºC), Temperature (ºF), Date and Time.

How does the Macro look like?



Temperature Records: Before and After





Raw data: Python script's output after compiling Measurements. Please note that each data point is separated by commas and each row represents an element of a list.

Processed data: Resulting spreadsheet after running macro. This would be the FINAL output.

What is the purpose of the Macro?

- To automate some of the steps regarding data manipulation.
- Simulate and redo every step performed manually to a dataset. Please note that data must have the same structure otherwise the macro will produce erroneous results.
- Save time by avoiding repetitive manual work in Microsoft Excel which most of the time is unnecessary.
- Re-use the same recently created Macro or VBA code in case the data needed an update and/or data points were to be appended to the raw data text file.

What happens to the final output?

- The Excel file now needs to get saved as a CSV file in order to get loaded into a table (a subset of a database).
- Data gets stored for record keeping purposes. Although this step might not affect the quality of our data directly, we still need to have the appropriate formatting for every column. Not doing might cause pgAdmin3 (PostgreSQL) to reject the CSV file depending on the conditions in which that table was created.

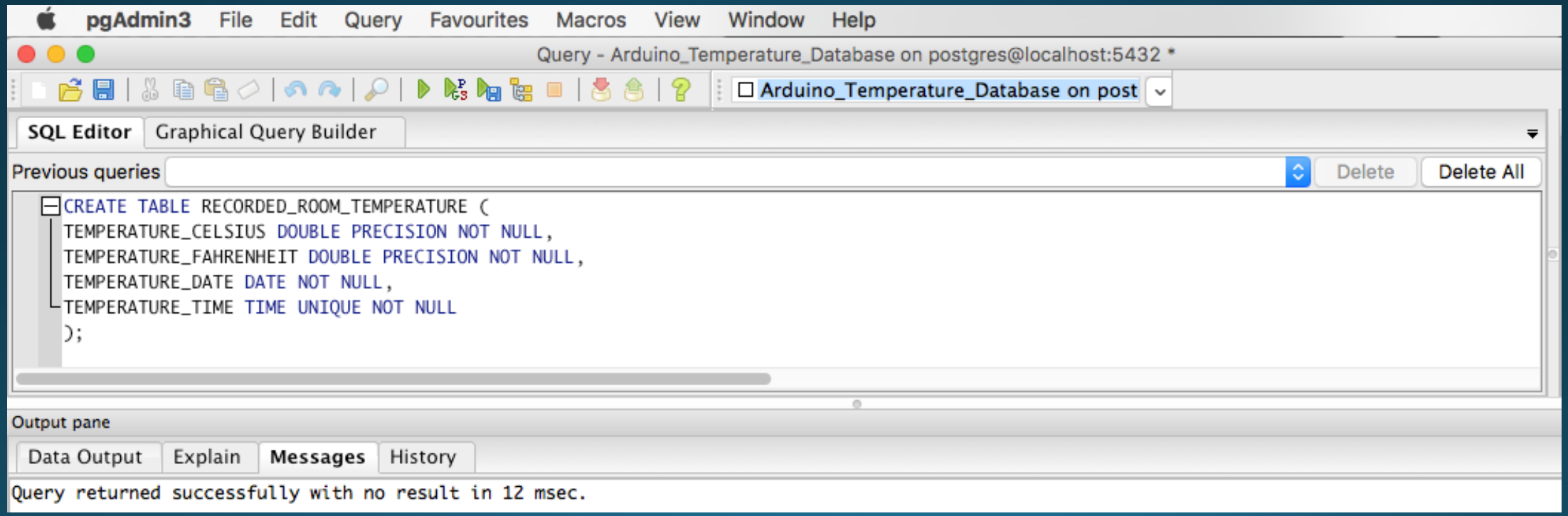
PostgreSQL

Loading Data into a Table

- As mentioned on previous slide (17), in most cases tables require each data set to meet certain conditions.
- In this project, the table was intentionally set to only accept CVS containing the following:
  - Temperature data (decimal numbers with two places).
  - Date with a valid format (there are several acceptable formats).
  - Time with a valid format (there are several acceptable formats).
  - None of the records can have missing information (blank cells within the spreadsheet).

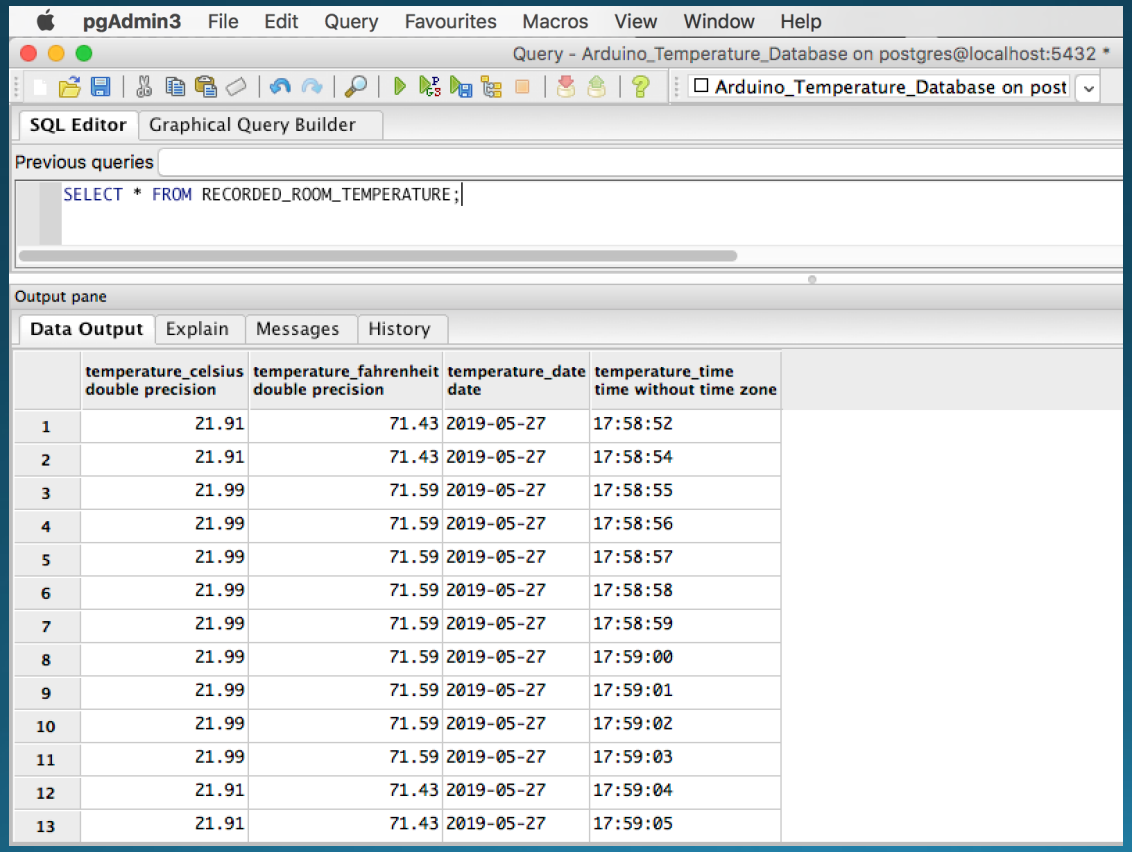
SQL Table

- The image below demonstrates how the RECORDED_ROOM_TEMPERATURE table was created with SQL syntax and through pgAdmin3:



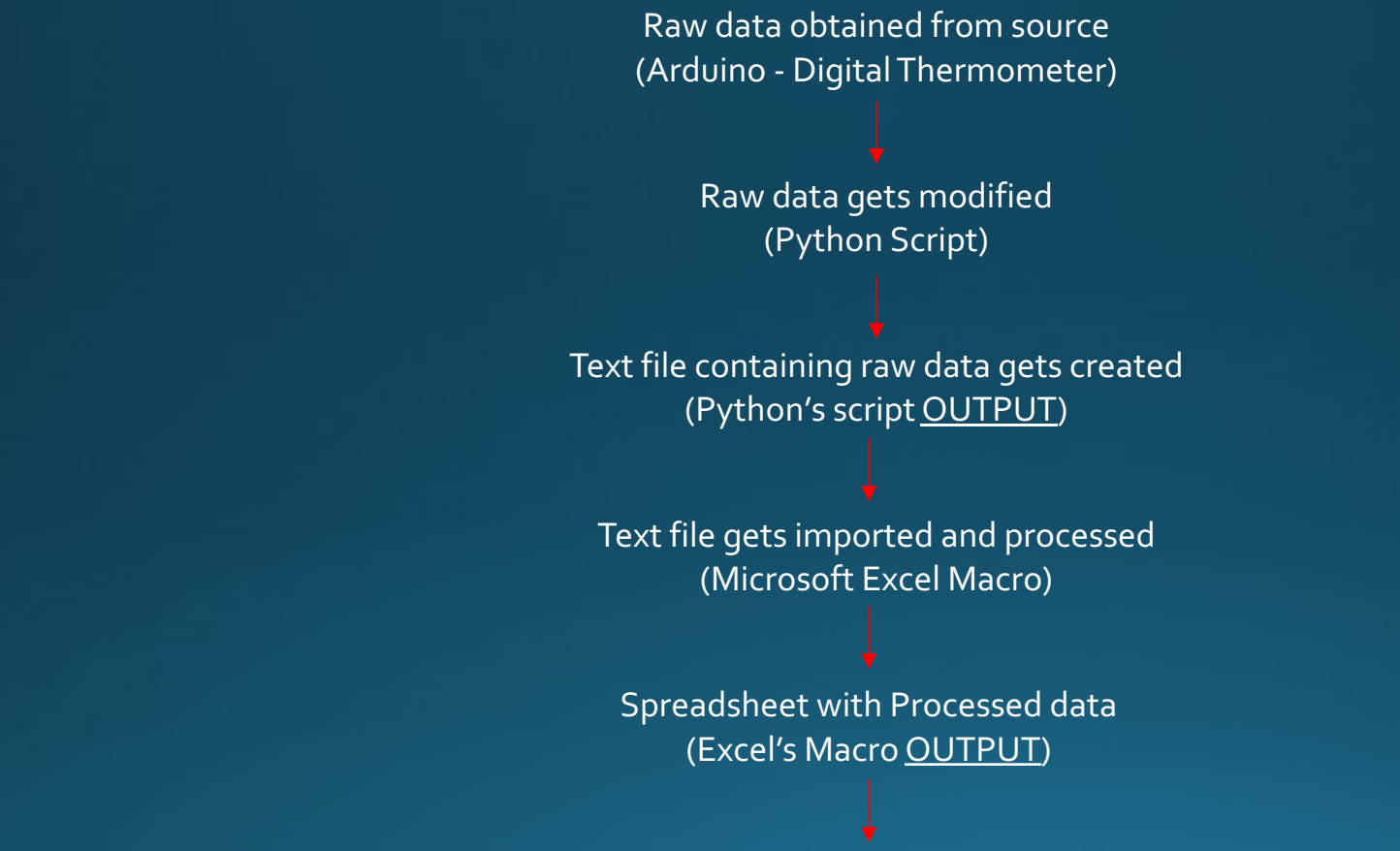
SQL Table (cont.)

- The following picture displays how our data looks like after writing a simple query selecting all data, after being stored into the previously mentioned table:



	temperature_celsius double precision	temperature_fahrenheit double precision	temperature_date date	temperature_time time without time zone
1	21.91	71.43	2019-05-27	17:58:52
2	21.91	71.43	2019-05-27	17:58:54
3	21.99	71.59	2019-05-27	17:58:55
4	21.99	71.59	2019-05-27	17:58:56
5	21.99	71.59	2019-05-27	17:58:57
6	21.99	71.59	2019-05-27	17:58:58
7	21.99	71.59	2019-05-27	17:58:59
8	21.99	71.59	2019-05-27	17:59:00
9	21.99	71.59	2019-05-27	17:59:01
10	21.99	71.59	2019-05-27	17:59:02
11	21.99	71.59	2019-05-27	17:59:03
12	21.91	71.43	2019-05-27	17:59:04
13	21.91	71.43	2019-05-27	17:59:05

Summary Diagram



Thank You for Watching!