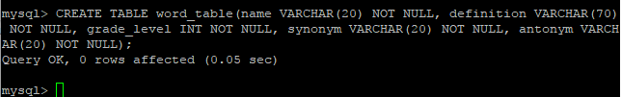
Vocabulary Table

James O'Donnell

**About this Project:**

For this project, I decided to go with a vocabulary table. The table stores words, the definition of the word, the word’s grade level, and a synonym and antonym of the word. The words are added to the database by users through the use of HTML forms. The HTML forms are processed by Python scripts and added to the database. There is also an option for the user to display the current list of words in the database for a particular grade level. The practical use for this could be for a teacher that is preparing a vocabulary quiz. They would be able to select a group of words from a certain grade level with ease.

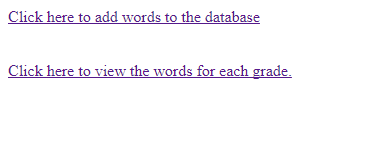
This project runs on a CentOS 6 machine and uses the MySQL database. The server is run using Apache 2.2.15 and the HTML forms are handled by Python scripts. The MySQL-Python module for Python allows the Python scripts to easily transfer the information from the HTML forms into the MySQL database.

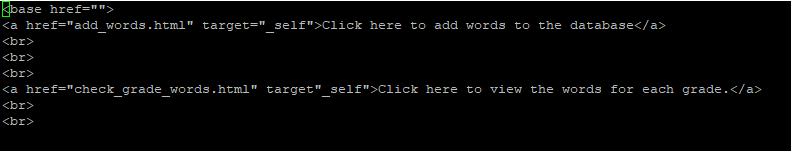
**Creating the Table**

This table was created in MySQL, and is the basis of the project. It is where all of the information that is inputted into the HTML forms is stored. The table, known as word\_table, contains five elements; a name (of the word), a definition for the word, a grade level, and a synonym and antonym for a word. One table is sufficient to hold all of the information in this project, so no other tables were made. This table also does not have a primary key, as words can have multiple definitions and multiple definitions could be assigned to different words.

**Starting up the project**

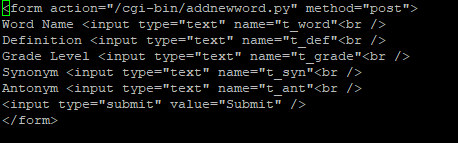
https://i.gyazo.com/14d7bd97c0e8cfd1f01d324b20f3468a.pngBefore anything can be done with the project, Apache must be started up.

When Apache is started and the site is launched (in this case, it’s a local host), the user is brought to a default index screen. The index screen looks like this:



This menu is a very basic start-up for the user, and prompts them to click on either option to either add a word to the database, or to view the words that are currently in the database.

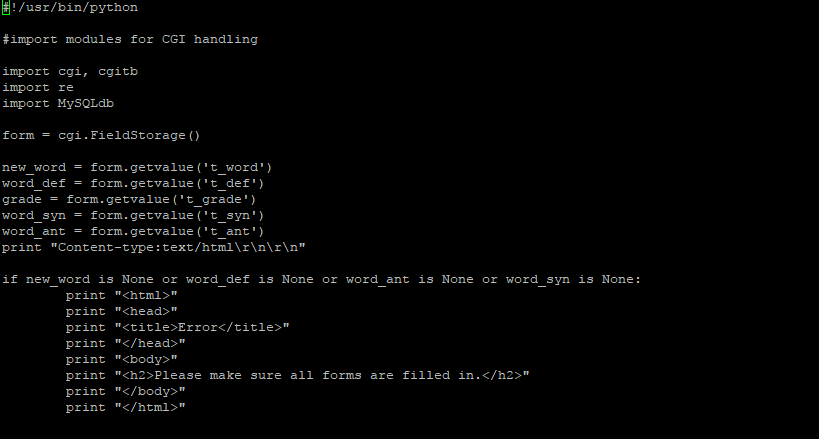
**Adding to the Database: The HTML Forms**



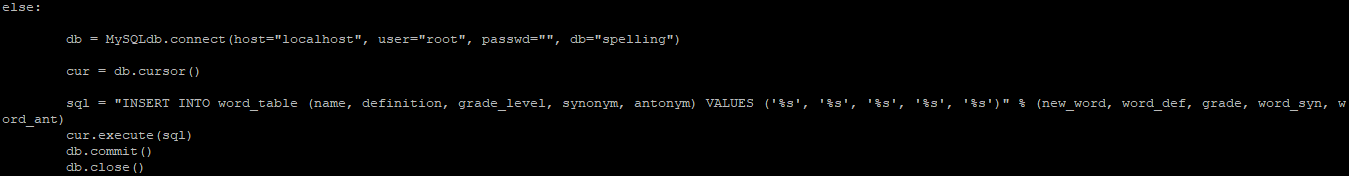
After creating a table, the next step was to figure out a way to add data to the database. I decided to use HTML forms to allow users to add data to the database. The inputted data is then processed by the Python script, which is seen in the “form action” at the top of the HTML file.

This is what the HTML file looks like when it is run. When the five forms are filled in and the user hits “submit”, the Python script is run and the word is added to the database.

**Adding to the Database: The Python Script**



This part of the Python script does a few things. First, it creates variables using the data from the HTML forms. Then, it does a quick check to make sure that there was data inputted into all of the forms; if any of the forms were left blank, it will not add anything to the database and tell the user to make sure that they filled in all of the forms.

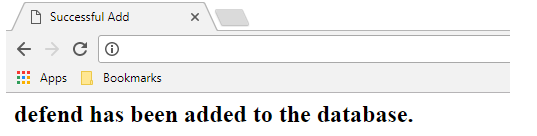


If the data inputted into the forms is valid, the python script then connects to the database. It runs an SQL statement that inserts the values of new\_word, word\_def, grade, word\_syn and word\_ant into word\_table. It then commits the change to the database, and closes the database. When the word is added to the database, the user is directed to a screen acknowledging that their word has been added to the database.

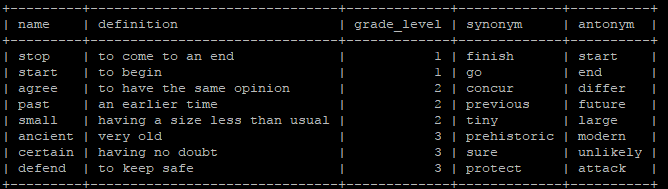
An example of a word being added to the database is on the next page.

**Example of adding a word to the database**

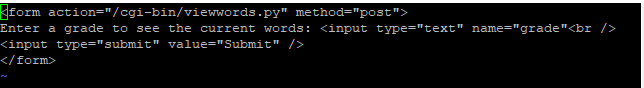
In this example, “defend”, a 3rd grade word, is being added to the database. After all of the information is filled in and the “Submit” button is clicked, a page is loaded that notifies the user that the word they inputted was successfully added to the database.

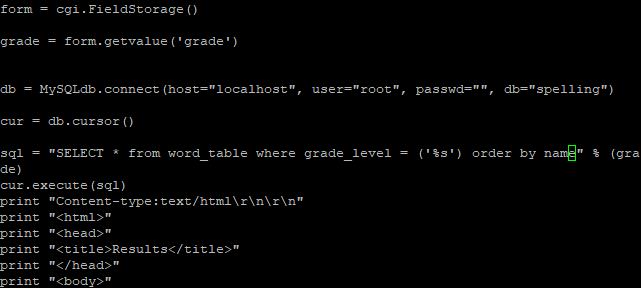


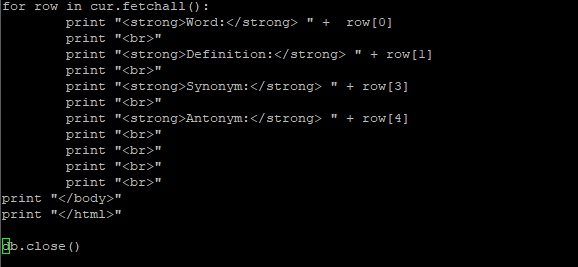
When we look in the database, we can see that the word “defend” has been added to it, along with the rest of the values that were inputted.



**Viewing each grade’s words: HTML Form and Python Script**

Similarly to the above example, this HTML form prompts the user to input the grade number they want to see the words for.

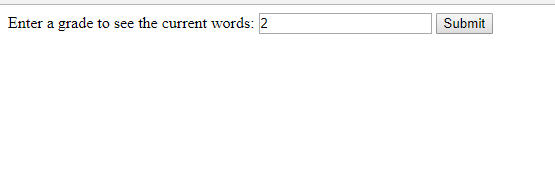
The input is then processed by this Python script, which runs SQL code to select the elements of the word table where the grade level matches the input in the form. At the end of the SQL statement, there is an “order by name” command that sorts the outputted data by the “name” element. The “name” refers to the main vocabulary word.



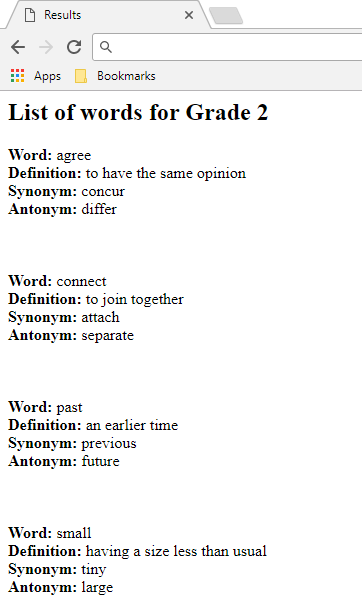
This part of the Python script shows how the data will be printed out. It prints it out in rows and has all of the elements of the table in the database.

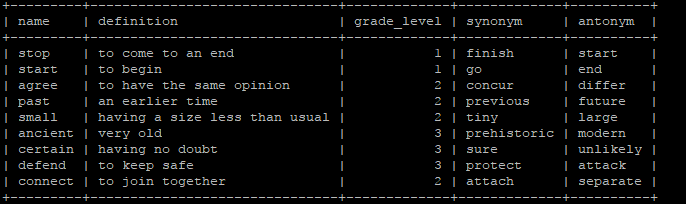
An example of the input and output of the program can be found on the next page.

**Example of checking the words in a grade level:**



In this example, I want to see which words are currently in the database for grade 2. I enter 2 into the HTML form and click submit.



When I submit the form, it outputs the data that currently has the grade level “2”. It gives me all of the details of the word. The SQL code sorts the output so the list of words that are outputted appear in alphabetical order. For example, if “order by word” was not in the SQL statement, then “connect” would be the last word to be outputted for grade 2.

**Conclusion and Final Thoughts**

This project uses MySQL, Python, HTML and Apache to create a small web server that can function in two ways. It can take in user input to populate a database with vocabulary words, with a word, its definition, the grade level of the word, a synonym and an antonym. It can also output words of a certain grade level, depending on what grade level the user inputs to the form. It sorts these alphabetically and output them, along with all of their attributes. I considered creating two more tables, a synonym and antonym table, and adding the values of just the word and the synonym/antonym into those tables when a user submitted data. With how the word table was set up, however, it felt redundant.