

The (Failed) Creation of a Language Change Simulator

The goal of this project is to create a software that simulates language change. Below is a list of steps I have to take towards completing the project. Of course, I was wayyyy too overly optimistic when I first envisioned the project...

Describing the language

In order to simulate language change, we must first describe the current state of the language. Therefore, our software needs to be able to read in and store the grammar of the language. The user should be able to input descriptions of an existing language, or they could use the software to create their own!

1 Phonemic inventory

The most basic component of a (spoken) language is a phoneme – the smallest unit of sound. Every language has a phonemic inventory, which is a collection of sounds that a speaker of that language perceives as distinctive. Each phoneme can be described using a feature matrix – a set of minimally necessary binary features that distinguishes it from all other phonemes in the language.

Thus, our software must be able to (1) store a phonemic inventory, (2) associate each phoneme to its set of features, and (3) allow the user to access phonemes via feature matrixes, and vice versa. As a bonus, we could also incorporate an algorithm that outputs a feature matrix given a list of feature values, and a system that allows the user to describe or create a phonetic-based writing system and associate each phoneme to an orthographical element.

2 Phonology

Every language has phonological rules that govern how phonemes are realized in speech. These rules are encoded using the same feature matrixes described above.

Thus, our software must be able to (1) allow the user to input and store phonological rules, and (2) correctly apply these rules to phonemic representations and yield the appropriate phonetic forms.

Time out...

So by then I realized that even getting that far is a bit of a stretch... And I was right...

An account of my unfortunate struggles

1 Creating a user interface

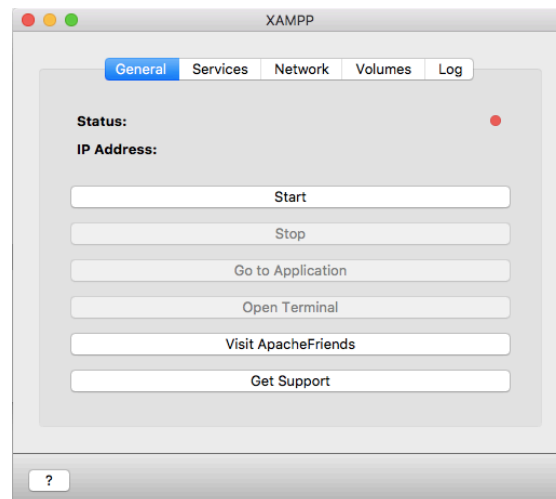
When I first started the project, I decided that the first thing I needed to do was to create a user interface. I found the *Qt5 C++ GUI Programming Cookbook* (URL:

https://culturalengineerassociation.weebly.com/uploads/8/6/7/7/86776910/qt5_c__gui_programming_cookbook.pdf) and followed instructions. I learned to use stylesheets, layout options, spacers, and various clicking reactions.

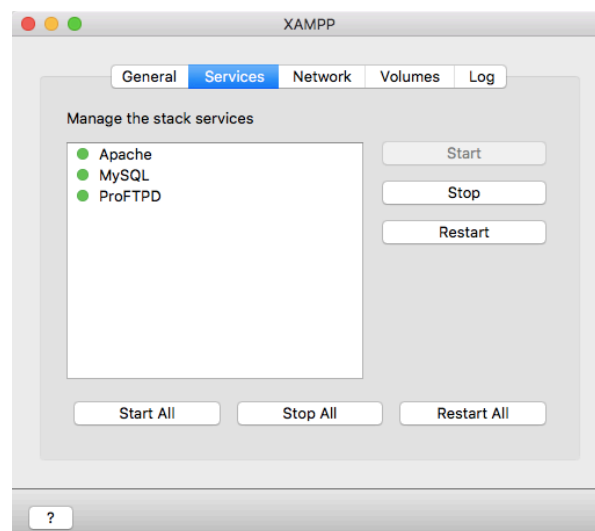
2 Setting up a database

Following the Qt Cookbook, I attempted to create and connect to a database. I downloaded XAMPP, which is a package that includes Apache and MySQL, which I need in order to create a database. Since the version of XAMPP that I downloaded has a very different user interface than the one described in the cookbook, I had to figure out how to make it work. Below is a list of steps (with screenshots!) that I took to set up a simple database for any unfortunate soul to follow:

- (1) Launch XAMPP and click “Start”



- (2) Click on “Services”. If the circles next to Apache and MySQL are red, click on each and click on start. If they are both green, you’re good to go!



- (3) Go back to “General” and click on “Go to Application”. It should take a page that looks like this:



Welcome to XAMPP for 7.2.5-0

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the [FAQs](#) section or check the [HOW-TO Guides](#) for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the [FAQs](#) to learn how to protect your site. Alternatively you can use [WAMP](#), [MAMP](#) or [LAMP](#) which are similar packages which are more suitable for production.

Start the XAMPP Control Panel to check the server status.

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our [Forums](#), adding yourself to the [Mailing List](#), and liking us on [Facebook](#), following our exploits on [Twitter](#), or adding us to your [Google+](#) circles.

Contribute to XAMPP translation at translate.apachefriends.org.

Can you help translate XAMPP for other community members? We need your help to translate XAMPP into different languages. We have set up a site, translate.apachefriends.org, where users can contribute translations.

- (4) Click “phpMyAdmin” at the top right corner of the page. You will most likely see this error message:

Access forbidden!

New XAMPP security concept:

Access to the requested directory is only available from the local network.

This setting can be configured in the file "httpd-xampp.conf".

If you think this is a server error, please contact the [webmaster](#).

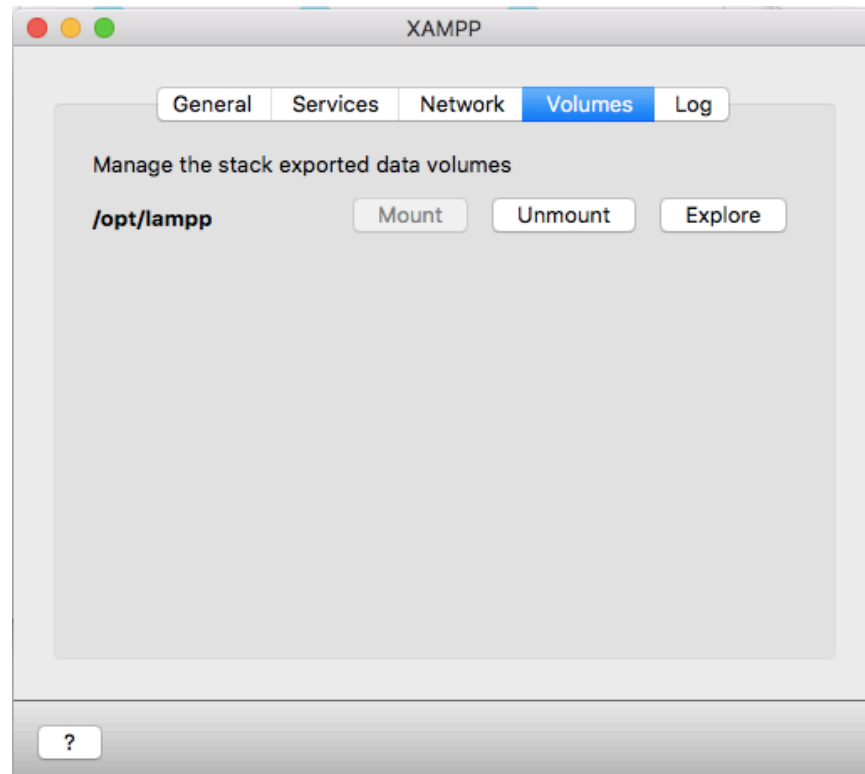
Error 403

[192.168.64.3](#)

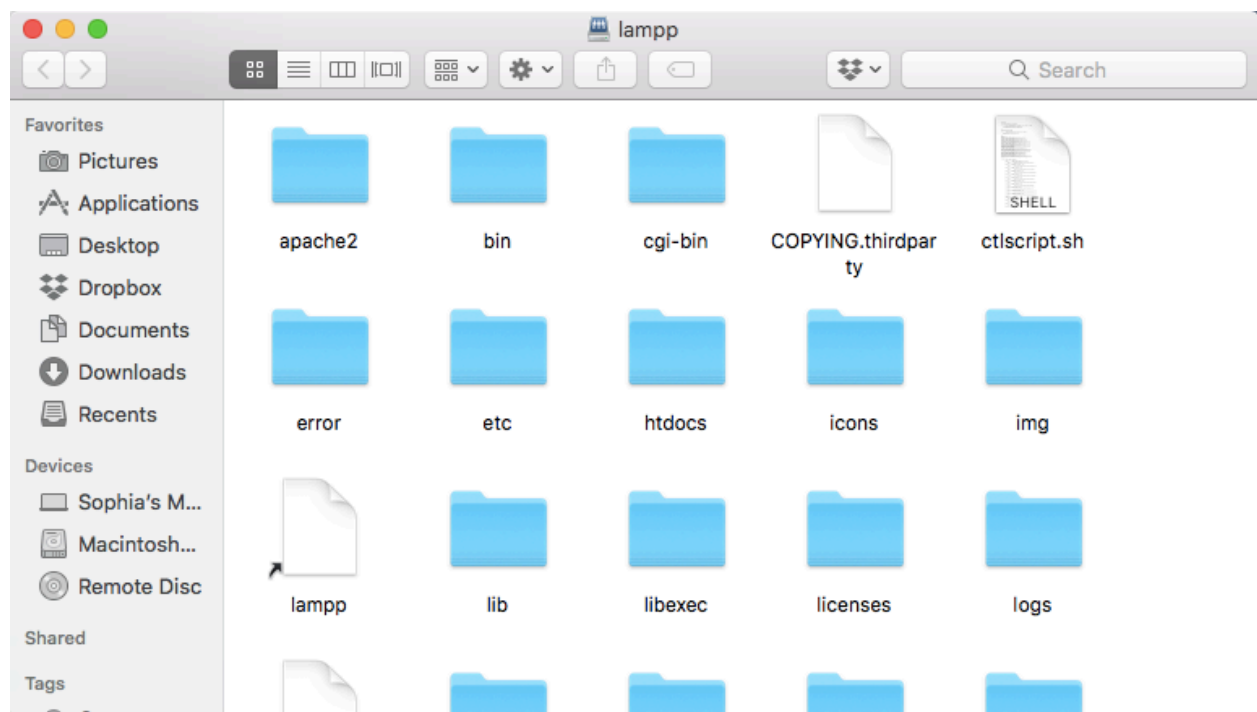
Apache/2.4.33 (Unix) OpenSSL/1.0.2o PHP/7.2.6 mod_perl/2.0.8-dev Perl/v5.16.3

Note: It took me foreeeever to figure out how to fix this. You’re welcome!

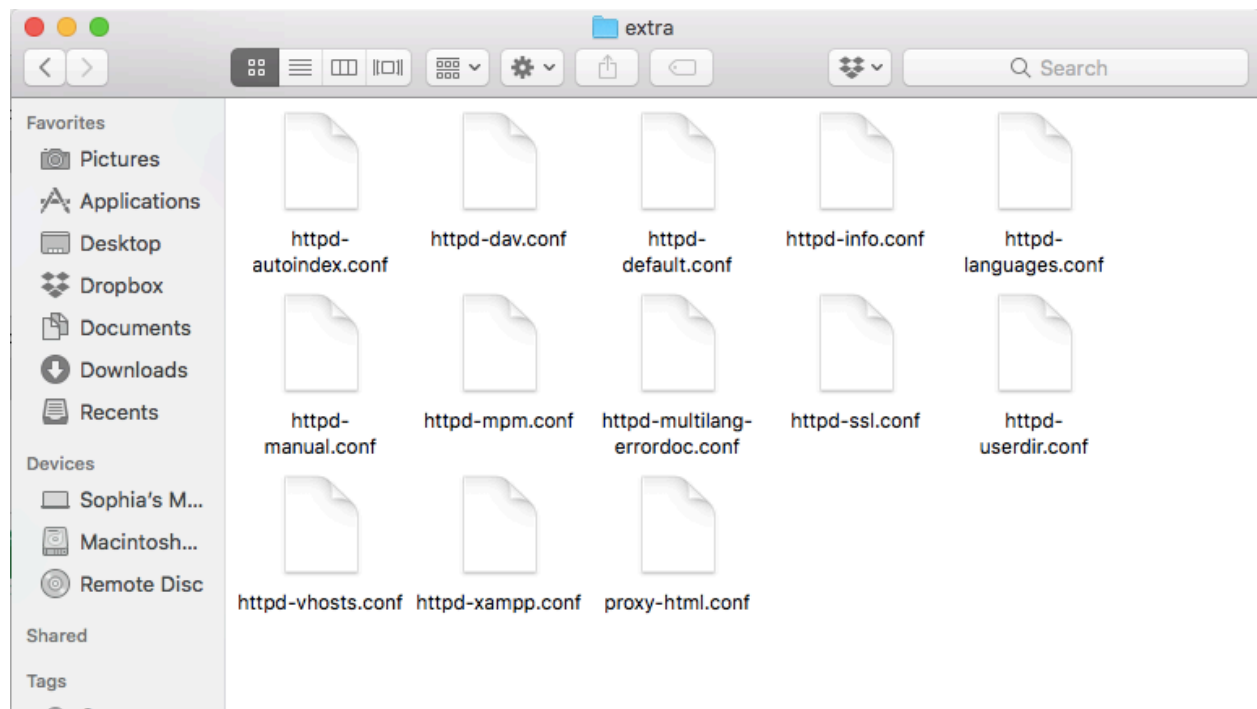
- (5) Go back to the XAMPP interface and click on “Volumes”. Then, click on “Mount”.



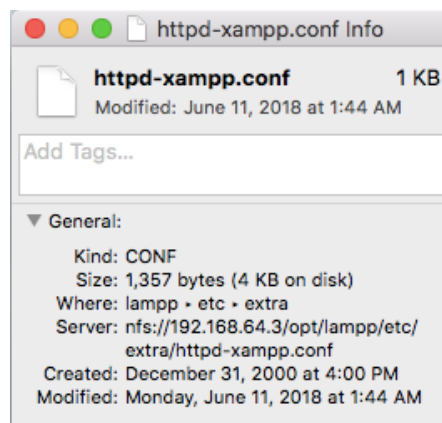
- (6) Click on “Explore”, and you’ll be taken to a Finder window that looks like this:



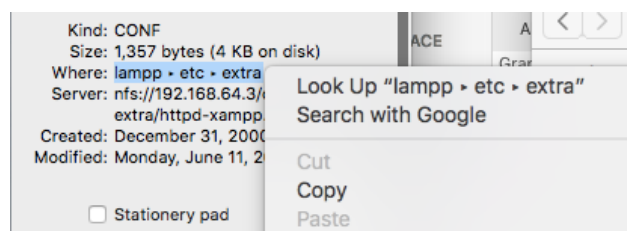
- (7) Click into the folder “etc”, then into another folder “extra”, where you’ll see a file named “httpd-xampp.conf” (second one in last row). We need to modify this file in order to fix the Access Forbidden error.



- (8) Right click on the file and select “Get Info”. A window like this will pop up:



- (9) Select the information after “Where” and copy it.



- (10) Open Terminal, type in **cd**, space, and paste the address you just copied. Hit enter. Your Terminal should look something like this:

```
[Sophias-MacBook-Air:~ sophialuo$ cd /Users/sophialuo/.bitnami/stackman/machines/]
xampp/volumes/root/etc/extra
Sophias-MacBook-Air:extra sophialuo$ █
```

- (11) Type **vim httpd-xampp.conf** and hit enter. You will see something like this:

```
##<IfDefine PHP4>
#LoadModule php4_module          modules/libphp4.so
#</IfDefine>
#<IfDefine PHP7>
#LoadModule php7_module          modules/libphp7.so
#</IfDefine>

# We will enable it by default
#<IfDefine PHP>
    LoadModule php7_module          modules/libphp7.so
#</IfDefine>

LoadModule perl_module          modules/mod_perl.so

Alias /phpmyadmin "/opt/lampp/phpmyadmin"

# since XAMPP 1.4.3
<Directory "/opt/lampp/phpmyadmin">
    AllowOverride AuthConfig Limit
    Require local
    ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
</Directory>

"httpd-xampp.conf" 51L, 1357C
```

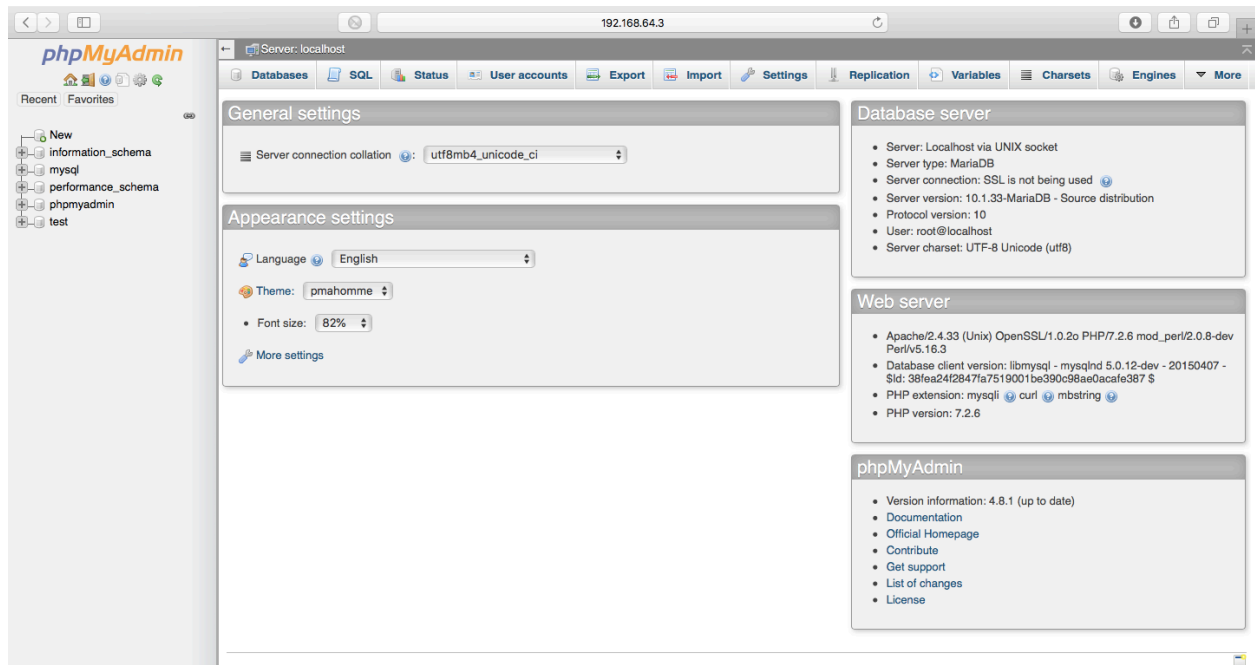
- (12) Hit **i** (stands for “insert”) so we can modify the file. Find the following section:

```
## since XAMPP 1.4.3
<Directory "/opt/lampp/phpmyadmin">
    AllowOverride AuthConfig Limit
    Require local
    ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var
</Directory>
```

- (13) Modify this section as follows:

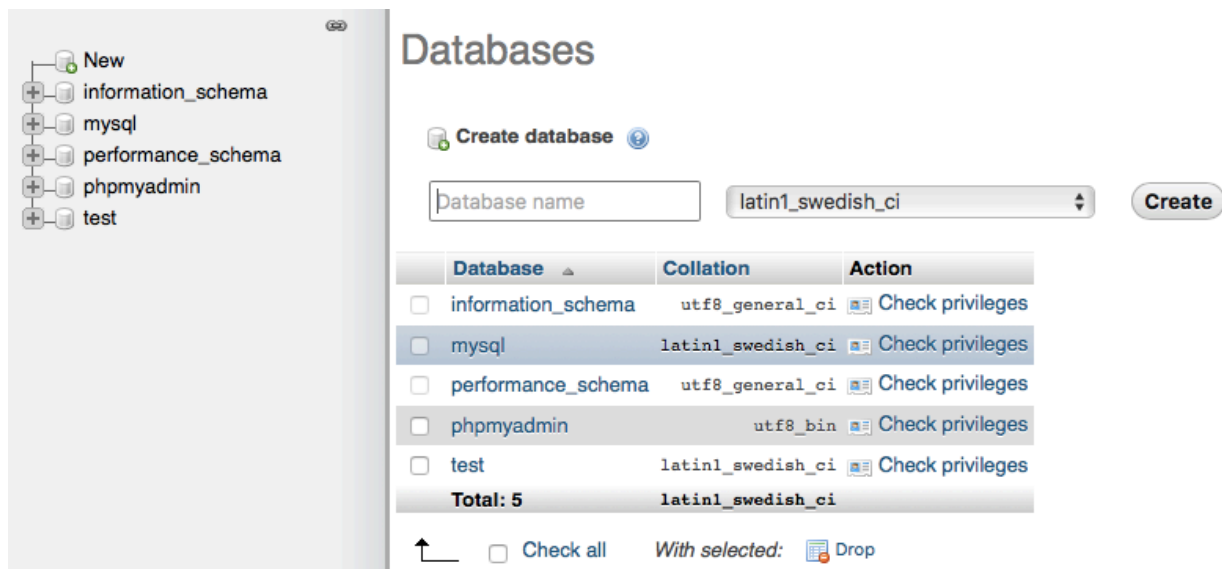
```
# since XAMPP 1.4.3
<Directory "/opt/lampp/phpmyadmin">
    AllowOverride AuthConfig Limit
    Order allow,deny
    Allow from all
    Require all granted█
</Directory>
```

- (14) Hit **esc**, type **wq**, then hit enter to save and quit. Now, restart XAMPP following steps (1) to (4). This time, you should be taken to an interface like this:

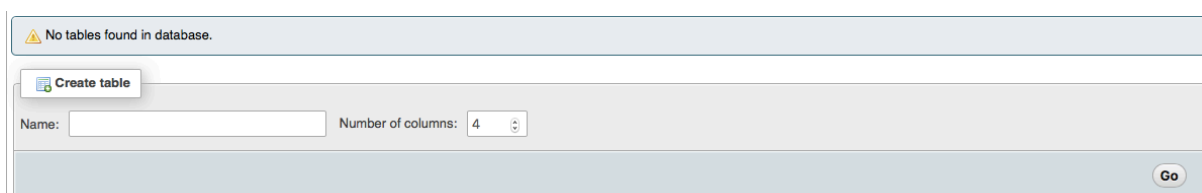


We're finally ready to start creating a database!

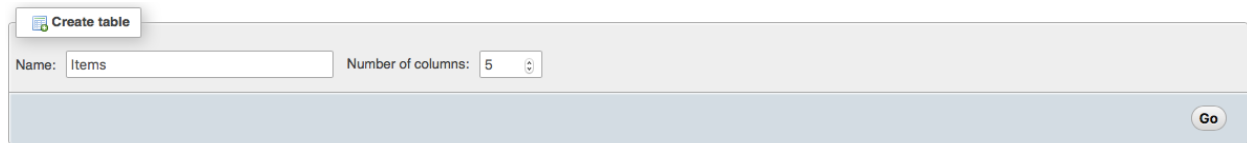
(15) Click on “New” at the top of the left side bar, and you will see something like this:



(16) Type in a database name of your choosing (I will use “Tutorial”) and click on “Create”. You will then see this:



- (17) Now, suppose that, in our database, we want to store some items and their colors – whether they are black, white, red, green, or some of the above, or all of the above. For this purpose, I will create a table named “Items”, and give it five columns. You create any table of your choosing. Once you have entered the name and the number of columns you want, click on “Go”.

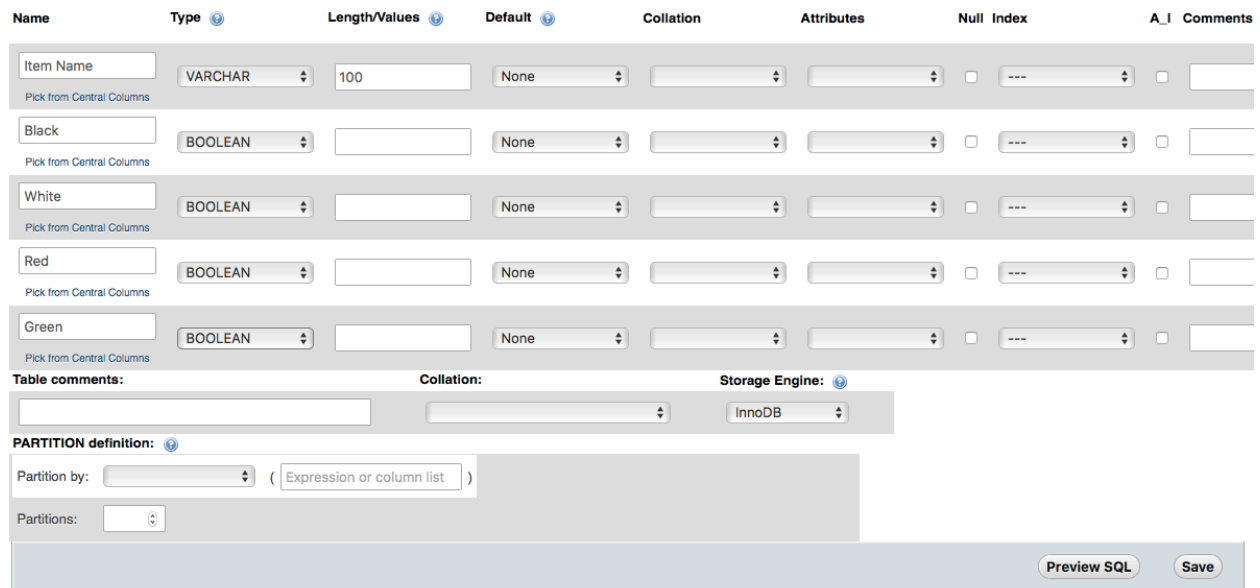


Create table

Name: Items Number of columns: 5

Go

- (18) Now, you will have the chance to set up your table. When you’re ready, click on “Save”. Here is how I set up mine:



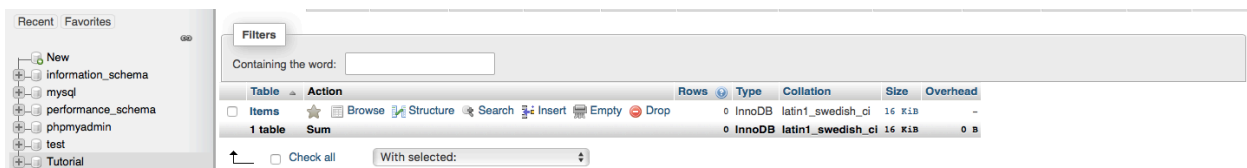
Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
Item Name	VARCHAR	100	None						
Black	BOOLEAN		None						
White	BOOLEAN		None						
Red	BOOLEAN		None						
Green	BOOLEAN		None						

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition: Partition by: (Expression or column list) Partitions:

Preview SQL Save

- (19) In order to add entries into our database, click on the database name in the left sidebar (mine is “Tutorial”), and click on “Insert” under “Action” at the center of the page.



Recent Favorites

- New
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- Tutorial

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
Items	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 K	16
1 table	Sum	0	InnoDB	latin1_swedish_ci	16 K	16

Check all With selected:

- (20) Add each item to the form, and click “Go” at the bottom of the page when you’re done. Here is an example:

Column	Type	Function	Null	Value
Item Name	varchar(100)			Lady bug
Black	tinyint(1)			1
White	tinyint(1)			0
Red	tinyint(1)			1
Green	tinyint(1)			0

Go

☐ Ignore

Column	Type	Function	Null	Value
Item Name	varchar(100)			Polar bear
Black	tinyint(1)			0
White	tinyint(1)			1
Red	tinyint(1)			0
Green	tinyint(1)			0

Go

and then

(21) Now, click on “Items” in the left sidebar, and you should see a table with items we entered:

Tutorial

- New
- Items
- Columns

☐ Show all | Number of rows: 25 | Filter rows:

+ Options

Item Name	Black	White	Red	Green
Polar bear	0	1	0	0
Lady bug	1	0	1	0
Penguin	1	1	0	0

We did it! Hooray!!!

3 Connecting Qt to a database

So God knows how many hours I have spent looking at tutorials and forums trying to figure this out, but it just *would not work*.

The code that various tutorials and forums said to put in **main.cpp** are pretty much all the same, and look something like this:

```
// creating a database connection

QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
db.setHostName("localhost");
db.setUserName("root"); // default mysql username for xampp is 'root'
db.setPassword(""); // xampp mysql has no password in default
db.setDatabaseName("qtdatabase");

// lets test the connection
if(db.open()){
    cout << "Database connected" << endl;
}
else{
    cout << "Database connect failed" << endl;
}
```

It never works.

Following tutorials and forums, I have tried to

- (1) Copy and paste the folder “sqldrivers” in .../Qt/5.11.0/clang_64/plugins into the debug location of my Qt project. It didn’t work.
- (2) Download MySQL Connector/C++ (download link here: <https://dev.mysql.com/downloads/connector/cpp/>) and add the library files in my Qt project. Since all the tutorials I could find were done on Windows OS, the library files are slightly different. Nonetheless, I tried including, one, some combination, or all of the library files in my project, and it DOES NOT WORK.

Moral of the story: Maybe I should try Python instead...

Attempt two: Python

And that is what I shall do!! Below is a step-by-step documentation of my progress:

1 Setting up

- (1) Downloaded and installed Python 3.6.5 for MacOS from <https://www.python.org>
- (2) Found YouTube tutorial for connecting to MySQL database with Python: <https://www.youtube.com/watch?v=JcOzJLO2V7k>
- (3) Created file mysqlcode.py
- (4) Then she started talking about pymysql and I don't know what that is...
- (5) Found a seemingly more straightforward YouTube tutorial: <https://www.youtube.com/watch?v=1ji8lqiBJe0>
- (6) Downloaded Python connector for MySQL – Platform Independent (Architecture Independent), ZIP Archive Python – from <https://dev.mysql.com/downloads/connector/python/>
- (7) Typed the following commands in Terminal and hit enter to install:

```
[Sophias-MacBook-Air:~ sophialuo$ cd Desktop/
Sophias-MacBook-Air:Desktop sophialuo$ ls
mysql-connector-python-8.0.11
Sophias-MacBook-Air:Desktop sophialuo$ cd mysql-connector-python-8.0.11
Sophias-MacBook-Air:mysql-connector-python-8.0.11 sophialuo$ ls
CHANGES.txt  PKG-INFO      examples      setupinfo.py  unittests.py
LICENSE.txt   README.txt    lib           src
MANIFEST.in   docs          setup.py      tests
Sophias-MacBook-Air:mysql-connector-python-8.0.11 sophialuo$ python3 setup.py install
```

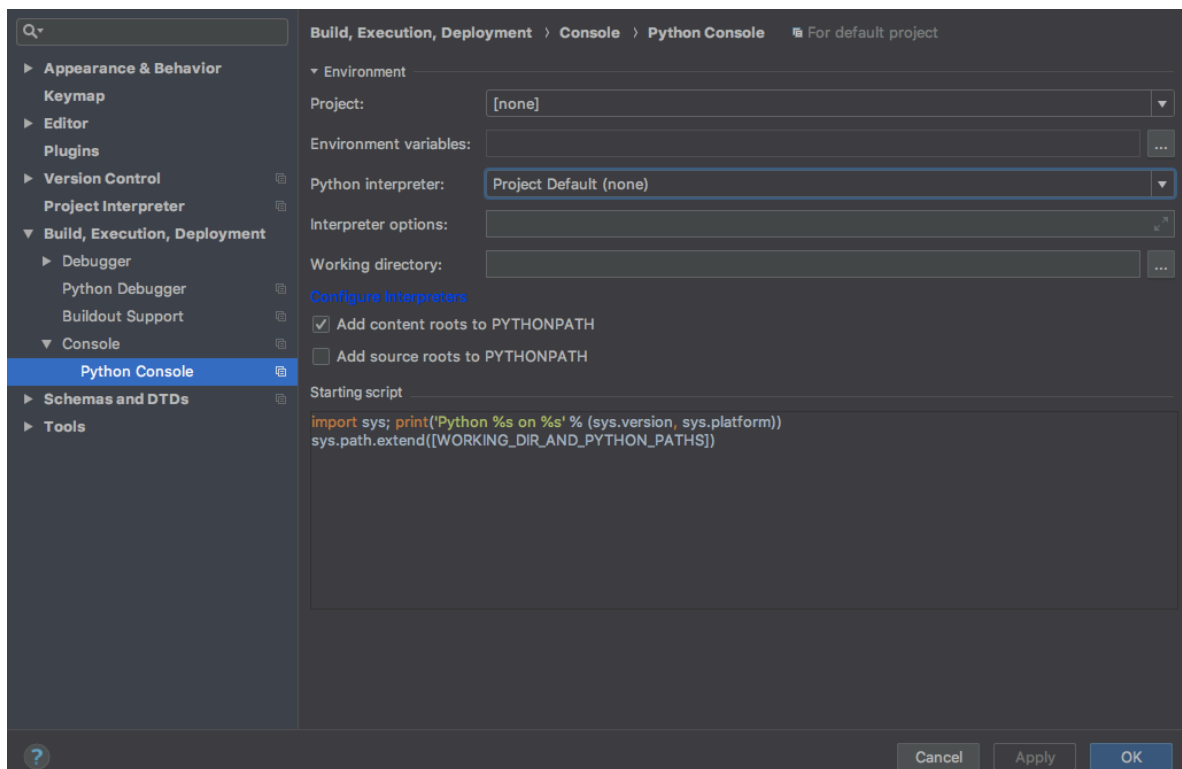
- (8) Downloaded and installed PyCharm CE from here: <https://www.jetbrains.com/pycharm/download/#section=mac>
- (9) Created a new Python project in my repository → OH NO I HAVE RUN OUT OF DISK SPACE T_T X_X
- (10) Cleaned up disk space and kept following YouTube tutorial. Created new Python file **index.py** and started writing code following tutorial, but I don't think it's working!
- (11) Tried to put MySQL Python connector into my Python project. Still doesn't work. I don't even know how this PyCharm IDE works... or even how Python works... Maybe I should start with some basic Python tutorials?

2 Getting acquainted with Python

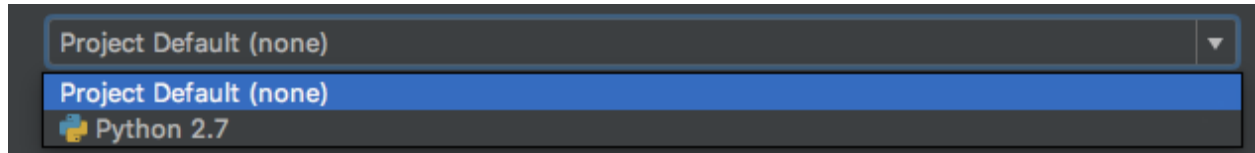
- (1) Found this YouTube crash course on Python:
<https://www.youtube.com/watch?v=N4mEzFDjqtA>
- (2) Set up PyCharm following tutorial:
 - i. Launch PyCharm and go to Preferences:



- ii. Go to “Build, Execution, Deployment” > “Console” > “Python Console”, and make sure “Project interpreter” and “Python interpreter” both match the version of Python that is installed



- iii. OH NOOOOOO... I believe something has gone wrong... Either something went wrong during installation (maybe because my disk space was low), or I will have to find the appropriate interpreter somewhere on my computer. WHY DOES NOTHING EVER WORK OUT SMOOTHLY?!



A note on branches

There are four branches in this repository:

- (1) **master**: This is the branch that I started out with. It documents my Qt-to-database struggles.
- (2) **experimental**: Trial branch for **master**
- (3) **LaTeX**: So I had a spur-of-the-moment thought that I would learn to use LaTeX to make my README look prettier. I didn't go through with that because I ran out of time...
- (4) **python**: I created this branch after abandoning my attempt to connect Qt to a database. It documents my struggles with setting up python (because everything is a struggle...)

So what does all of this have to do with PIC 10C?

1 Git

I feel like I have gone from a total amateur to a Git master. (Emphasis on *feel* like.) This project has given me a lot of good practice on staging, committing, branching, merging, cherry-picking, and all the other Git fun.

2 STL containers

Originally I had thought that maybe I could use STL containers to store my phonemic inventory. For example, since a phoneme, which is usually represented by an International Phonetic Alphabet (IPA) symbol, can be identified by a unique set of phonological features, we can store the phonemes of our language by storing their corresponding IPA symbols in **std::map**, and the set of features that identifies that phoneme would be the unique key value. However, this idea was not useful because I went down the road of databases, but even so maybe it would become applicable when the project is more developed?

3 Function pointers

Although there is very little code at this stage of the project, function pointers could become very useful in the future for two reasons:

- (1) A lot of phonological processes, such as stress and syllable assignment, work through “recursive rule application”, and the idea is basically exactly the same as callback functions in programming. Thus, function pointers would come in handy when we want to implement callback functions to simulate linguistic processes that work through recursive rule application.
- (2) Sound change occurs in sequence. A historic form from Middle English does not suddenly turn into its corresponding form in Modern English. Instead, it undergoes a series of intermediate stages, some of which must occur in a certain order. Function pointers could be useful here because, instead of defining individual functions that represent a sound change from start to end, we can write much simpler function pointers and store them in arrays, and each array of functions would represent a sequence of sound change

4 Inheritance and polymorphism

Although the rough concept of inheritance can (kind of) be observed in natural language – phonology “inherits” from phonetics, morphology from phonology, syntax from morphology, and so on – implementing these relationships as derived classes would be... forcing the issue, especially if we’re using a database to store phonemes, words, and everything else.

However, inheritance did come into play when I was trying to build a UI using Qt, since Qt is (kind of) just a big jumbo of numerous base classes and derived classes. But alas, Qt didn’t quite work out for me, and I haven’t gotten far enough with Python to know if inheritance will come into play. And so the mystery continues... *dramatic music*

5 Templates

Besides using existing STL stuff, I really can’t think of uses for this one... especially since all of my objects/variables are either type bool (for binary phonological features) or type string (for everything else).

6 Introduction to Qt

HA! So yes, of course, this is very relevant. I think Qt is a very effective platform to customize a UI. I did some exploring with layout design, and tried very hard, for many hours, to find a way to connect Qt to a database. But it just would not work!! So I decided to switch to Python.

One of the big things we discussed about Qt is the RAII idiom, which is not a feature in Python. I found this Stack Overflow thread that explains why (<https://stackoverflow.com/questions/5071121/raii-in-python-automatic-destruction-when-leaving-a-scope>), and it lists three reasons:

- (1) Python is a GC language

A garbage collector is a feature that releases memory no longer being used, thereby preventing memory mismanagement. In PIC 10C, we learned that C++ does not use a garbage collector for the following reasons (taken from smart points hand out: <https://bitbucket.org/rikis-salazar/10c-smart-pointers/overview>):

- i. Control over execution time must be ‘handed over’ to the garbage collector, and it could take a while to get control back
- ii. The collector runs independently of the programming logic, kicking in usually when memory is running low

In other words, garbage collectors are incompatible with the C++ philosophy that puts most value in speed and efficiency.

On the other hand, Python does use a garbage collector, which means that it doesn’t see the need to reclaim memory as long as there is enough memory elsewhere for new objects (memory is fungible). Non-fungible resources are specially treated using the **with** statement. (This is in contrast with C++, which treats all resources the same way.)

(2) Python doesn’t have stack variables (WAT?! O_O)

“In C++ terms, *everything* is a **shared_ptr**”

(3) Scoping in Python works differently than in C++

This makes RAII incompatible with Python, because RAII is based on scoping: associated heap-memory is released when an object goes out of scope.

7 Memory Management