Odd Semester (2021)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Group Work)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Naman**  **Nicholas**  **Rino**  **Salya** | | **Student ID Number**  **2201798420**  **2201798446**  **2201798736**  **2201798143** | |
| 1. | | **Vohra**  **Jovanka**  **Santoso**  Wirachman |  | |
|  |  |
| **Course Code** | **: COMP6340** |  |  | | **Course Name** | | **: Introduction to Programming** | |
| **Class** | **: L3AC** |  |  | | **Name of Lecturer(s)** | | : 1. Maria Seraphina Astriani | |
|  |  |  |  | |  | |  | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment**  (if any) | : Voice to Macro | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 29 -10- 2019** |  |  | | **Submission Date** | | **: 29 - 10 - 2019** | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

**CONTENTS**

**Contents………………………………………………………………………………….2**

1. **Introduction……………………………………………………………………………3**
   1. **Background……………………………………………………………………3**
   2. **Problem Description…………………………………………………………3**
2. **Related Work………………………………………………………………………….3**
3. **Implementation ……………………………………………………………………….4**
   1. **Formal Description of the Problem……………………………………….4**
   2. **Design of the Program……………………………………………………….4**
   3. **Libraries Used………………………………………………………………….4**
   4. **Implementation Detail…………………………………………………………5**
4. **Program Manual………………………………………………………………………..8**
5. **Discussion………………………………………………………………………….....10**
6. **Links and References……………………………………………………………….10**

Introduction

**Background**

Computers have become a part of our daily life and to use them efficiently is to master the use of shortcuts in the keyboards. As computer scientists we love to use shortcuts in order to make our computing life much more convenient. There are many kinds of commands that are executed by using multiple key presses on the keyboard, The Foundation or reason for us to choose this problem was 2 things. One of them was purely because a multitasking shortcut tool is a nice thing for a computer scientist and we see the potential of it being helpful in future assignments and projects. We were inspired by a person with motor disabilities such as hand tremors, cramps, and muscular dystrophy. People who suffer from these situations might not be able to use a computer properly. We saw the opportunity to make a program where certain complicated or convoluted processes are made easier just by using voice input. This project could also prevent other hand injuries when using a keyboard for too long such as carpal tunnel syndrome, Posterior cervical dorsal syndrome, and Tennis Elbow. To conclude, we believe that this project fits the requirements and will be a helpful tool for computer users.

**Problem Description**

The main problem that arises when talking about shortcuts is the amount of them to memorize and by its nature not being an inherent design or easy to understand design for the users for example who would think that the key combination for force closing and application is Alt + F4. For some people who have a hard time memorizing they would always have to look up the correct key presses meaning the supposed shortcuts are not shortcuts at all. The next problem is that using these shortcuts sometimes require capable motoric abilities for either pressing multiple keys simultaneously or pressing a series of keys fast after another. For people who have a motoric disability or for children and the elderly this proves to be a problem for them since they would have problems using a keyboard. In the end we decided the voice activation program to solve these issues and let the user have a more personalized shortcut.

Related Work

Our Project will uses these current applications that exist which are Windows software called Voice Macro.  Home assistants such as Siri, Alexa, Cortana, Google Assistant. But our program target is focuses on keyboard controls and shortcuts.

Implementation

**Formal Description of the Problem**

We saw that there are 2 main problems in terms of shortcut usage which are:

* Inconvenient Key Combinations
* Memorization

The first problem stems from key combinations that are uncomfortable or unorthodox to press. such examples are Ctrl + Shift + Esc making your hand shape twist in an unnatural way. Research states that using a keyboard for a long period of time causes injuries to the hand due to its design forcing your hand to over stretch on some keystrokes.

The second problem is the huge amount of shortcuts that exist. Each program could average around 40-50 Shortcuts, that means there are thousands of key shortcuts that are programmed by the creator’s sense of logic. We decided that we want to fix those problems to create a more ease of access to use shortcuts that makes working on a computer more efficient.

**Design of the program**

The Design of the program is using microphone to pick up any sounds and create and audio file, then that audio file is checked and converted to a data type of either a string or a integer. It creates a microphone object to detect when to record the audio input. And with a recognizer object to determine which noises are background noises by setting a threshold.

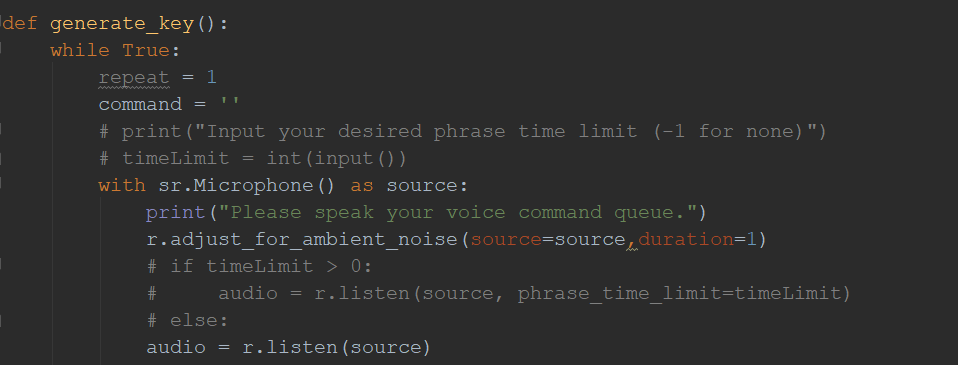
Then the program stores the audio input and let the user choose which keyboard interaction that is bound to the audio file. Afterwards the program awaits until the command is said and stimulates key press, key release, and delay at which it was set by the user. In short terms it goes form Voice to text to action type to macro.

**Libraries Used**

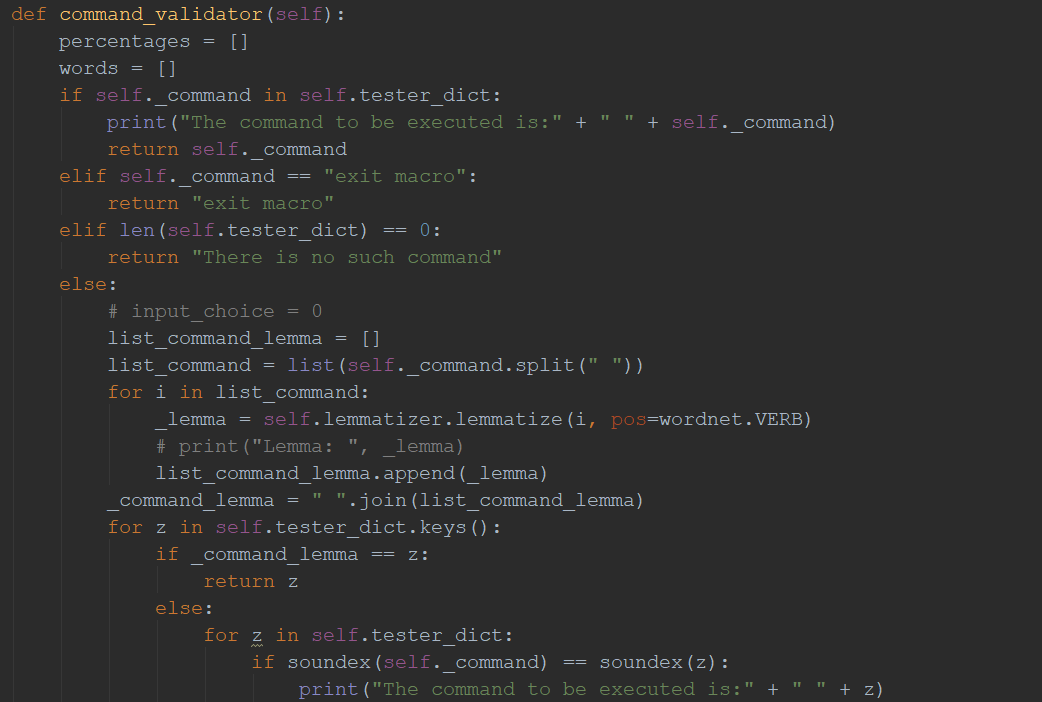
* PyAudio: Used for Audio capture purposes
* Speech Recognition: Detects Audio Inputs and their types
* NLTK: Used for Lemmatizer Function.
* Jellyfish: Used for word and phrase validation
* Pickle: Used for Storing user made commands
* Time: Time related functions
* OS: Dependent Operating System Functionality
* Numpy: Used for number and integer management

**Implementation Details**

The implementation of the program relies on a few key libraries which are SpeechRecognition and Keyboard. These two libraries are used to first detect a microphone for audio input. Creates an object called microphone that records an audio input for a certain amount of time. Then it creates a recognizer object to detect and filter the type of audio input,



Next we used a lemmatizer to turn a word to its base form by removing the prefix and suffix of the said phrase by the user. afterwards it tries to convert the audio and creates a string or integer based on the keyword that is recognized within the audio file.

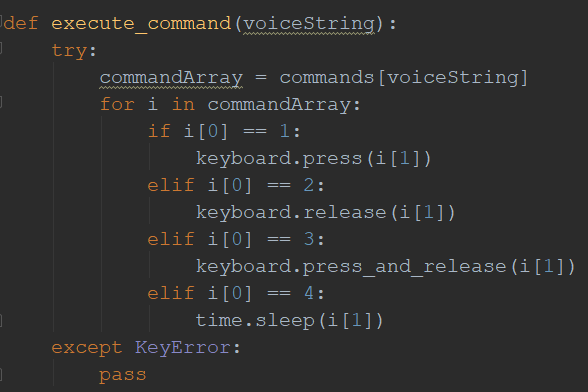


There are four levels of validation in the processing of the audio commands (user input). The first level simply checks whether the user input is in the dictionary (list of commands which the user created earlier) .If the voice command is exactly the same with one of the keys in the dictionary, it will return the key (command). But, if the voice command differs, it will proceed to the next step which implements lemmatisation process. This process transforms the word into its base form. For example, different->differ, drove->drive, testing->test. This can be used in a case (for example) where test is one of the command name (key) in the dictionary. If the user inputs testing, it will be converted into its base form which is test and then would match the key in the dictionary. The key is then returned so that it can be executed.

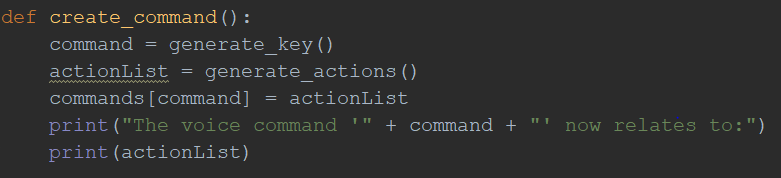


The third level of validation targets words which sounds similar. This is achieved with the help of soundex algorithm which is a part of the jellyfish library. Soundex is a system whereby values are assigned to names in such a manner that similar-sounding names get the same value. This can be used in a case where one of the commands in the dictionary sounds similar to the input voice command. For example, if forth is one of the keys in the dictionary and the user's voice command is fourth (though what he meant to say was Forth), the system would still recognize the input as forth. The reason for this is that both voices will produce the same soundex encoding as both have similar pronunciations. The fourth and the last step is executed when the first three steps fail to return a value. It will pass the original dictionary and user input to a function called dict\_slicer which slices the original dictionary and append its keys into a list. The keys which will be appended are the ones which have the same length with the user input or the keys which form a part in the command and vice versa. For example, if the key is the word 'age' and the command is 'mirage' the key 'age' will be appended to the list as the word age is a part of the word mirage. Then the percentage similarity is calculated. The formula for the case above (where the key is age) is the length of the common letters (3) over the length of the key or the command (takes the one with the more length) \* 100. As mirage (user command) contains more characters than age, it will take the length of user command as the denominator. It will calculate the percentage and validates it with the lexer\_sensitivity (the range of tolerance level for the similarity between key and user input) before appending the word into a list. The key which matches the most with the command Is the one which will be returned. If the sensitivity level is to low, it may be possible that none is returned.

Then a Dictionary is created to store a key and value, that we decide should be the corresponding voice command and action. In that dictionary the key is the converted and recognized voice audio and the value is a 2d array where it consists of 4 action types which are key press, key release, key press and release, and delay.



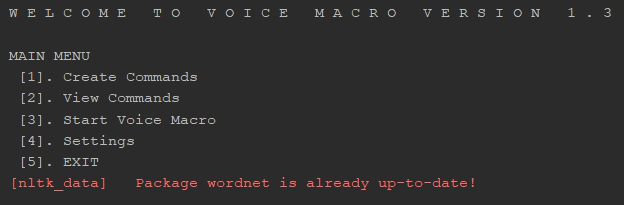
Which then is used with the keyboard module to know what action should the module stimulate in the windows virtual keyboard. The user could set what kind of macro’s that they need with a function. The program then waits for the listen key-bind to be pressed and then executes the action that was set beforehand.



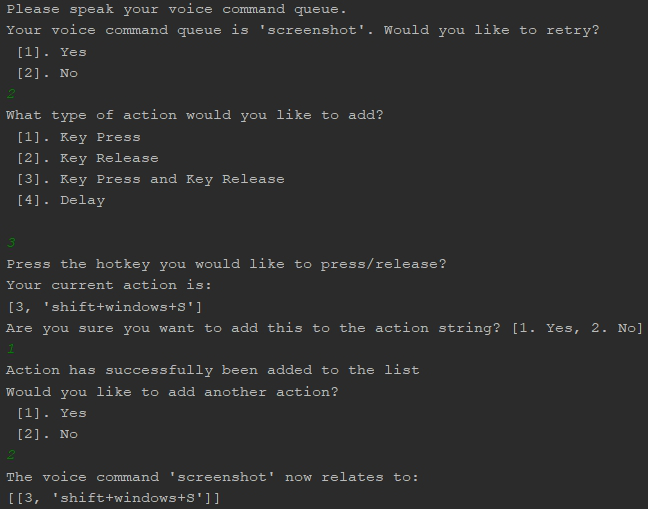
Lastly when the program is exited the user set command are saved outside of the terminal using Python pickle to make future uses easier. In conclusion creates a voice activated shortcut.

Program Manual

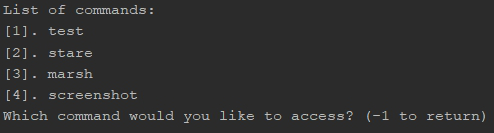
* The Program runs on a terminal and the user is given a main menu. In the main menu the user is given 5 options. but to start the user should first create commands by pressing the 1st option



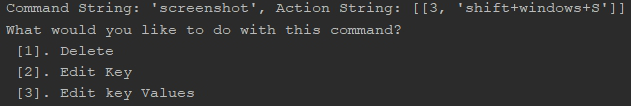
* The program then listens and waits for the audio input. If the input wasn’t heard then the program asks if the user wants to retry. If it is heard and the voice is converted to text, then they are prompted with what kind of action that correlates to the command. The 1st choice is to push down a key, 2nd is to let go of pushing down a key, 3rd on is the push and let go of that key. And lastly the 4th one is to add a delay between the press and release if you want to hold down a key for a certain amount of time.
* Afterwards the user is asked to press the key or keys that they want to have that action. Next the program prints out the inputs for confirmation and asks if you want to chain more commands. If not then it continues on.



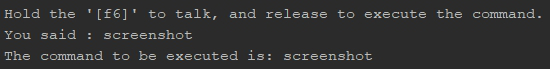
* Next it is saved in the dictionary and the main menu will show up again. Where the user can view the list of commands already created.



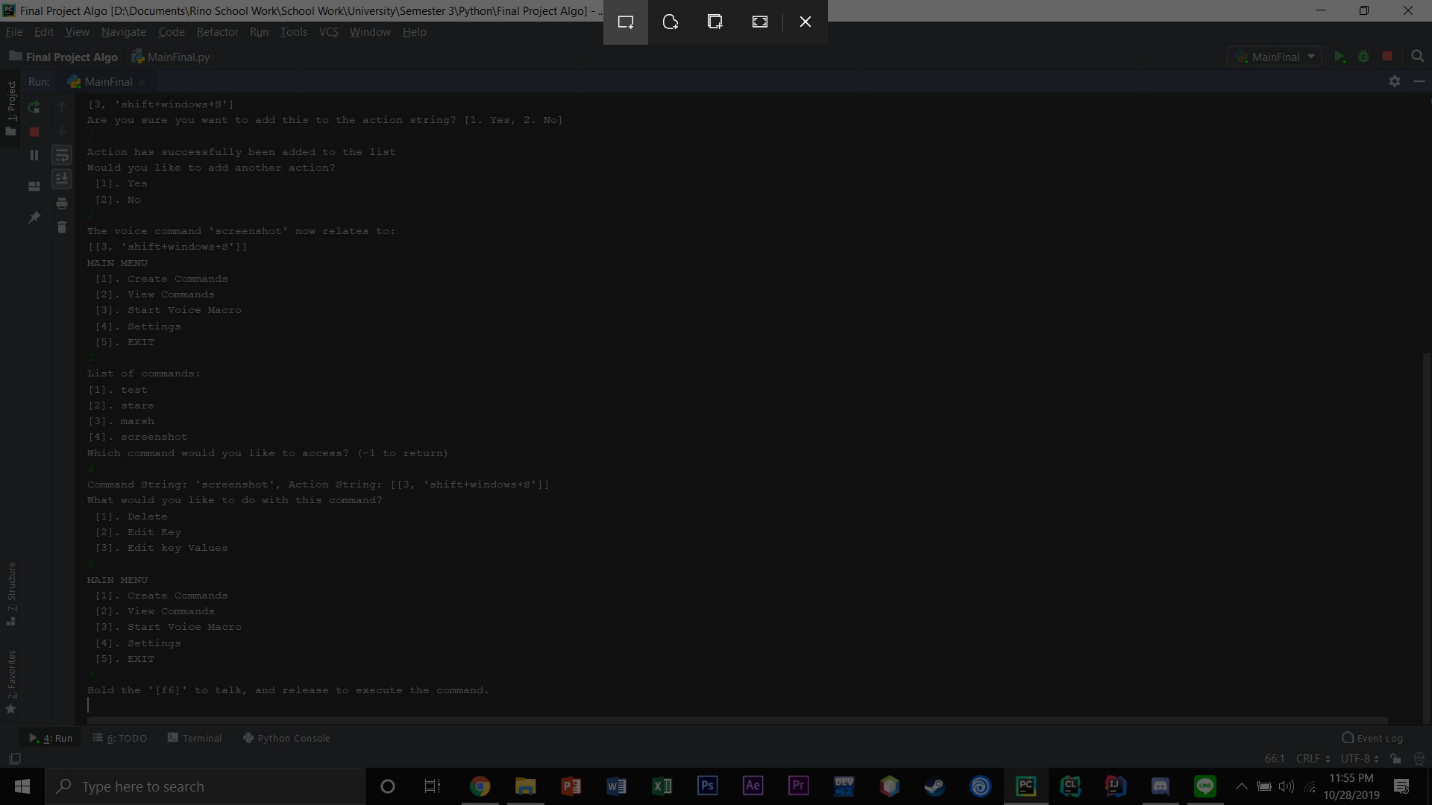
* If the user chooses to access the command, they will have more options to either alter, update, or delete said command.



* Then when the user chooses the 3rd option which is executing the voice macro the program will listen in on the audio file for the predetermined keyword.

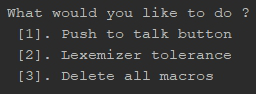


* The result of the program in this example is taking a screenshot.



* Lastly is the 4th option which are settings that consist of rebinding the listen (push to talk button), adjusting the lexemizer, and to delete all saved macros.

The lexemizer tolerance is the setting that controls how much percentage of closely sounding words that you want to tolerate, this helps



Discussion

**Conclusion**

In Conclusion our program of aiming to create a voice to macro program to solve the problem of memorization, inconvenient key combinations, and motoric requirement to activate shortcuts were achieved by the program that we created. Although we realize that this project still has some flaws and in the future we might improve and reutilize this concept our aims of creating an inclusive voice activation shortcut for all audiences was fulfilled.

Links and References

Github Links: [nicholasjovanka/analysisofalgorithmfinalproject](steam://openurl/https:/github.com/nicholasjovanka/analysisofalgorithmfinalproject)

References:

**Libraries**

PyAudio:

<http://people.csail.mit.edu/hubert/pyaudio/#downloads>

SpeechRecognition:

<https://pypi.org/project/SpeechRecognition/>

<https://github.com/Uberi/speech_recognition/blob/master/README.rst>

Keyboard Module:

<https://pypi.org/project/keyboard/>

NLTK:

<https://www.nltk.org/>

Pickle:

<https://docs.python.org/3/library/pickle.html>

Numpy:

<https://numpy.org/>

**Concept References:**

<https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63>

<https://stackoverflow.com/questions/2474015/getting-the-index-of-the-returned-max-or-min-item-using-max-min-on-a-list>

<https://stackoverflow.com/questions/55331723/how-to-get-the-similar-sounding-words-together>