TASTEVENS Elvis, DI SALVO Salvatore, BIALASIK Eryk

Abstract

A step-by-step guide on

Raspberry Pi 400 project

Manual

Step by step Manual

[Introduction: 2](#_Toc400933955)

[Step-by-step manual: 2](#_Toc418718864)

[Presentation of code: 2](#_Toc770752563)

[Importing libraries: 2](#_Toc41500025)

[Creating a 2D Array: 3](#_Toc1575705827)

[Adding Functions: 3](#_Toc858999167)

[Translation to morse code: 4](#_Toc479494312)

[Creating audio play function: 5](#_Toc907466822)

[Main function 6](#_Toc1242901210)

[Our findings: 8](#_Toc726965687)

[Conclusion: 8](#_Toc1250008177)

[Meeting minutes: 8](#_Toc7706185)

[References: 9](#_Toc450777317)

# Introduction:

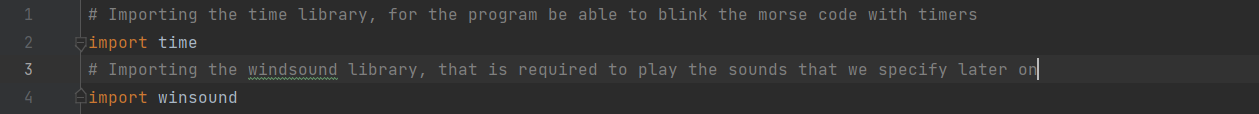
Our goal in this project was to create a program that would be able to translate the user input from Roman letters to Morse code that would be displayed on the monitor. Our secondary goal was to install a loudspeaker that would play the sound of previously translated user input into morse code. We’ve used built-In libraries, loop, lists, many functions and arrays to complete the project

# Step-by-step manual:

1. First the program prompts with a message that asks you to enter text or type “quit” to exit. Please write your input under the message.
2. You will be able to see “Filtered message:” output, this message lets you know that all special characters got removed and shows you filtered version of your input.
3. Under the filtered message you will notice “morse code” message output that follows with written morse code that is your translated message.
4. Now the prompt asks if you want to play the morse code in audio. You need to input either “Y” for yes or “N” for no.
5. After you have selected “Y” a sound message will play audio in morse code with your message.
6. When the audio finishes playing the prompt will start over.

# Presentation of code:

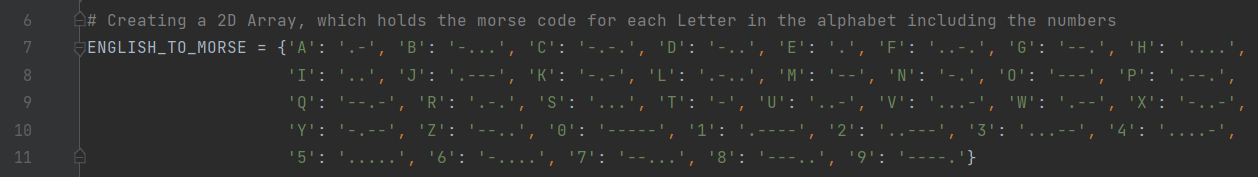
## Importing libraries:

Two libraries we require for this project are ’time’ & ’winsound’

Time – allows us to wait during code execution. This is used once we need to provide an audio representation of Morse code.

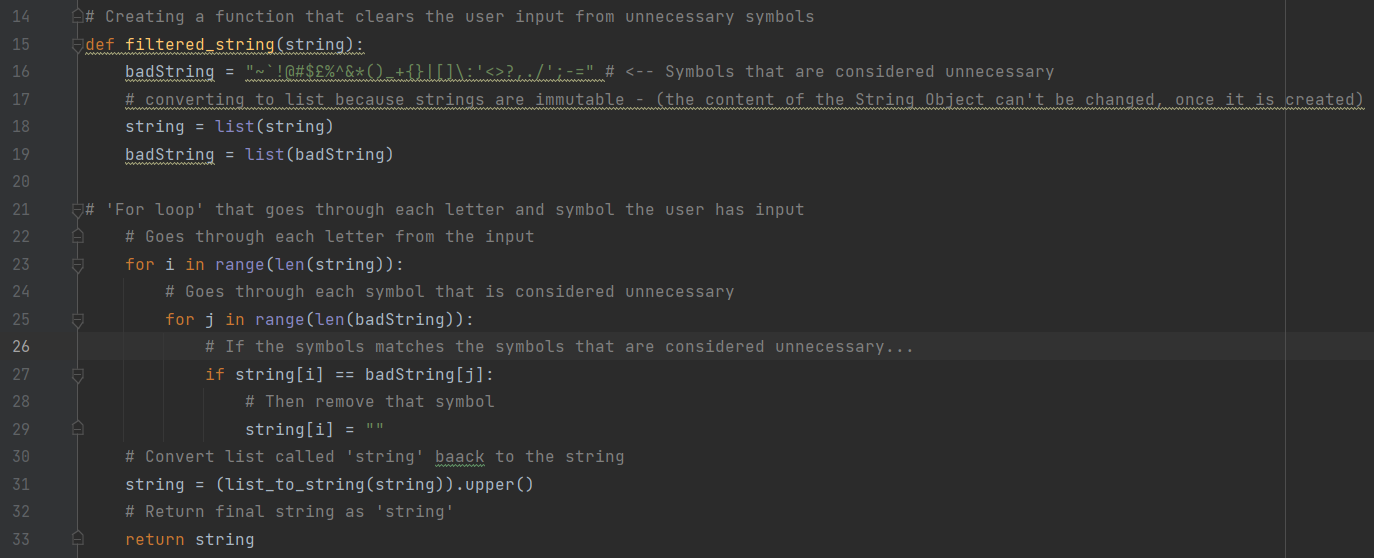
Winsound – is a library used to create custom frequency sounds. There we can change the custom frequencies to be played as well as duration of each sound.

## Creating a 2D Array:

2D array will serve as a translator between English characters and Morse Code.

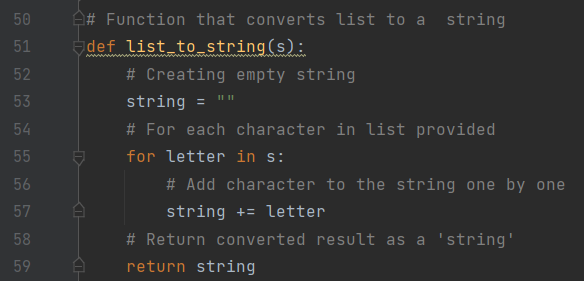
For example, the Morse code for the letter ‘A’ is ‘. -’, and the 2D array we have created holds that information for each letter in the alphabet including the numbers 0 to 9.

## Adding Functions:

The first function that needs to be created is a filter function. The filter function allows the program to filter out any unnecessary symbols that the user might input, which could cause errors in the future. It also provides conversion between strings and lists, since strings are immutable meaning the content of the String object can't be changed, once it is created.

The FOR loop that was added inside the function makes the program go through each character the user inputs, and if it finds the unnecessary symbols it will erase them from the list.

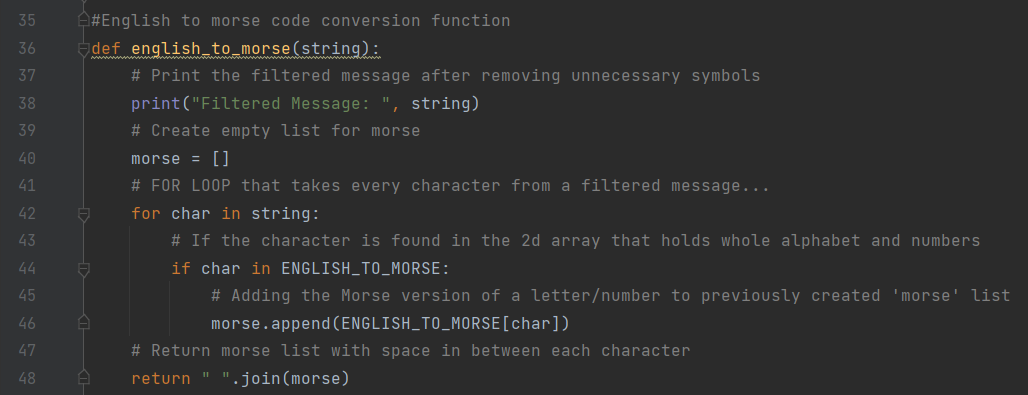
The list that was fully cleared from the unnecessary symbols then needs to be converted into back into the string. For that reason, we need to create another function that would do just that.



This function provides the conversion between lists and strings, then returns the final result back to the program.

## Translation to morse code:

Next part of the program needs to be actual translation between the English letters and the Morse code.

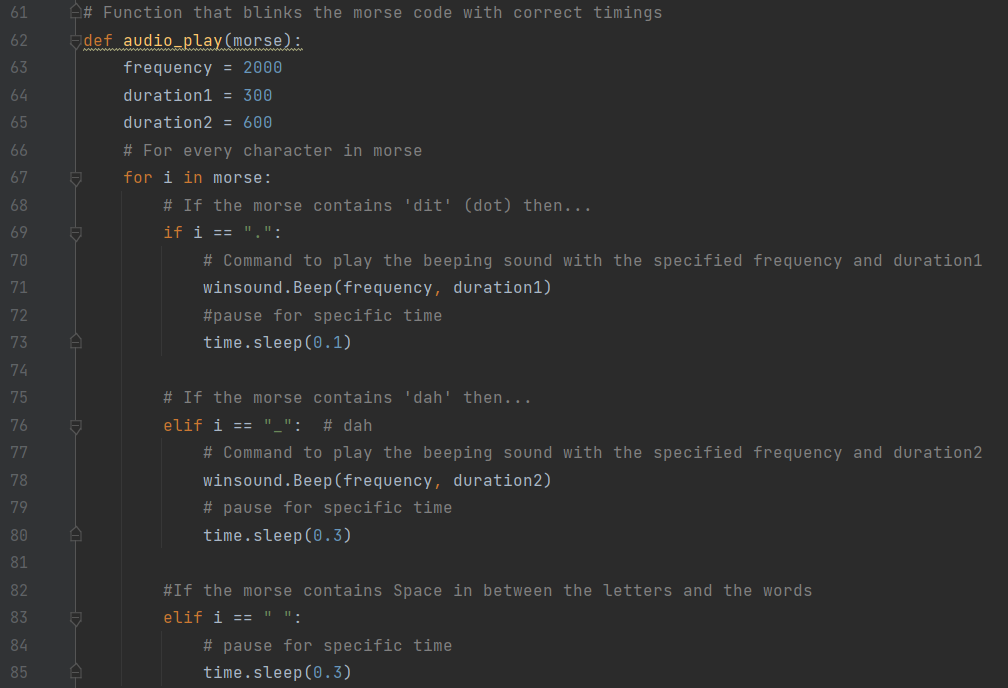
Function is called ‘english\_to\_morse’. This function translates between English and Morse.

The program first prints out the filtered version of the user input. Then the empty morse code list is created which is the place where the morse code will be written to.

This function includes a FOR loop, which goes through each character that the filtered input has. It then compares them with the previously created 2D Array and adds corresponding Morse code to the Morse list. Each new character adds more Morse code to the list. And in the end, returning the completed Morse code message.

## Creating audio play function:

The next function to be created is the Audio playing function.



The first step is to specify the frequency and the duration of the beeps. There are 2 types of beeps, one is for the ‘dit’ which are represented as a dot in the Morse code, and another one is ‘dah’ usually represented as a line in the Morse code. The ‘dah’ sounds are longer than the ‘di’ sounds, therefore need to be different duration, that is why we have duration1 and duration2.

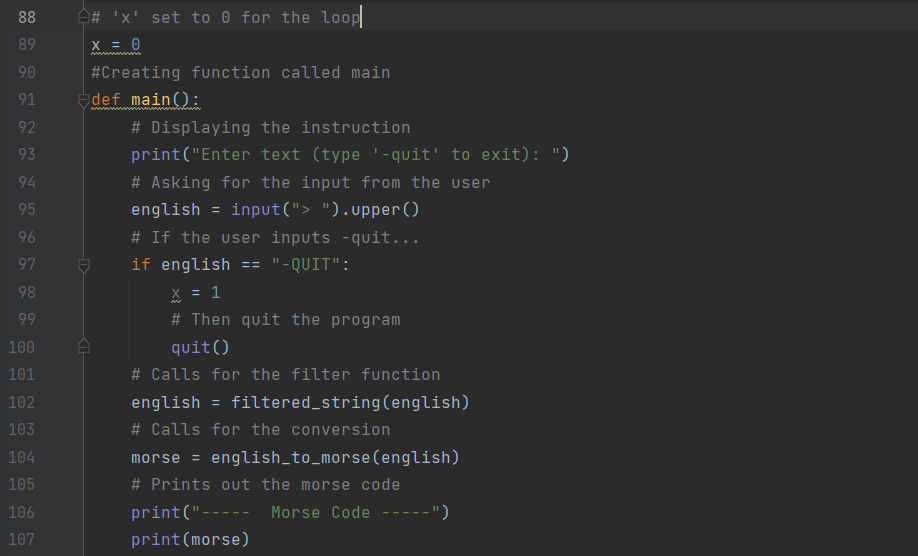
The loop that is inside the function is going to scan every character in the morse code and if it sees ‘dit’ sound then the sound is going to start playing with the frequency that was specified previously, as well as the duration1 that is only 300ms long. The program then is put to sleep by the command time.sleep(0.1) which a very short duration.

The loop then repeats the steps and once the ‘dah’ character was found, everything else is the same as the ‘dit’, but this time a different duration of the sound is provided which is marked as a duration2. As well as longer sleep time in between the beeps.

If the space is found by the loop, then there is no sound played back but only sleep command is used, this marks a space in between the Morse code characters.

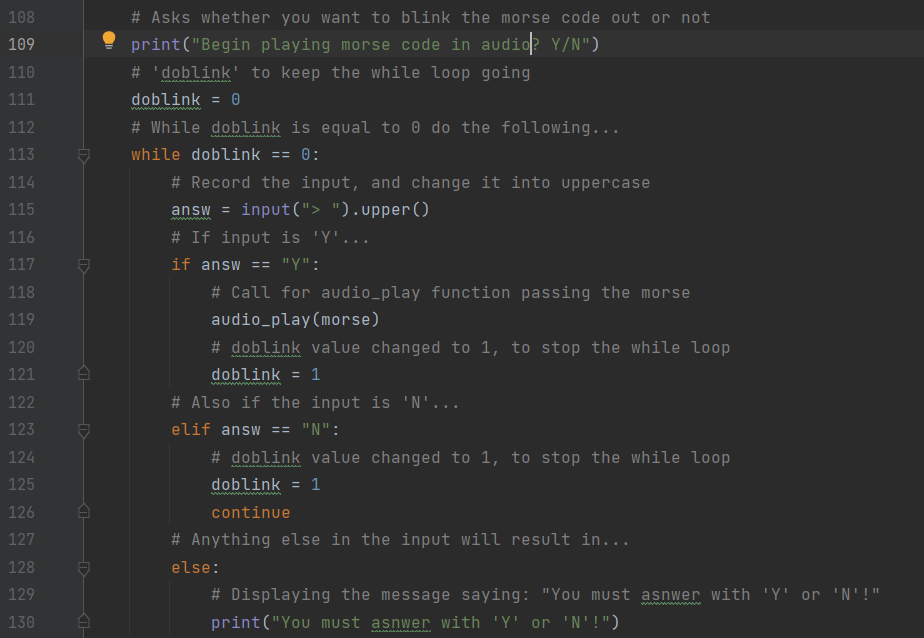
## Main function

The last function that we require is going to be the main() function, this is where all the other functions are going to be called and will be returning the final outputs.

Once the main function is called, it will start off by displaying the instruction on the screen. In this instance the instruction is to ”enter the text” user wants to translate into a morse code, additionally the user can input”-quit” that will result in program being terminated completely and will require to be run again.

The input is then recorder and changed into an uppercase. The program then checks the input for the “-quit” and will terminate the program if the input contains it.

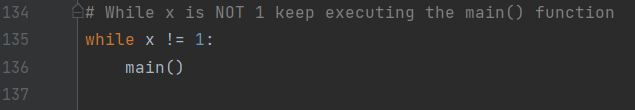
Program proceeds with the filtering stage, where any unnecessary symbols are removed. Right after that the text is translated into Morse code and printed out fully.

The program then proceeds to ask the user whether they want it to start playing audio representation of Morse code. The input form the user reads and saved in the uppercase. If the input is ’Y’ then call for the function that plays the audio, that would play the audio that was mentioned earlier. And ends the loop.

If the input is ‘N’ then the program is going to continue to the next step without executing the audio.

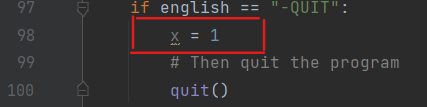
If the input is anything else, then the program is going to prompt the user by saying that they should input ‘y’ or ‘n’ until the user inputs one of the letters.

And finally, the loop that starts and keeps the program running:



While variable x IS NOT 1 it will keep repeating the main() function that we have mentioned earlier.

The variable x is being changed inside the main() function:



That concludes the whole code of the program.

# Our findings:

Our biggest find during this project was a necessity to think outside the box if you desire to reach the final goal. We have also seen the importance of making sure that all the components are compatible with each other to avoid problems in the future. Another discovery we made during development was that the operating system commands are different for each operating system, which made simple programming more challenging.

# Conclusion:

During our project we have encountered many obstacles to reaching the final goal. Ultimately, we have managed to create a working code that is able to translate from letters to morse code by using display and sound thanks to built-in libraries as well as arrays created by us.

In the future we would approach the topic differently with more preparation and research beforehand. Making sure that all the equipment is compatible with each other as well as having access to all the libraries that are going to be required turned out to be a vital part of the whole project.

# Meeting minutes:

|  |  |
| --- | --- |
| Meeting | 1 |
| Meeting Minutes | 30 |
| Group Number |  |
| Attendees | Eryk, Elvis, Salvatore |
| Actions from Previous Meeting | Research projects |
| Current Meeting | 11/02/2022 |
| Absentees | **Sam Lewis** |

|  |  |
| --- | --- |
| Meeting | 2 |
| Meeting Minutes | 30 |
| Group Number |  |
| Attendees | Eryk, Elvis, Salvatore |
| Actions from Previous Meeting | Research alternate project since project proposal got rejected |
| Current Meeting | 25/02/2022 |
| Absentees | **Sam Lewis** |

|  |  |
| --- | --- |
| Meeting | 3 |
| Meeting Minutes | 20 |
| Group Number |  |
| Attendees | Eryk, Elvis, Salvatore |
| Actions from Previous Meeting | Assign roles to members |
| Current Meeting | 07/03/2022 |
| Absentees | **Sam Lewis** |

|  |  |
| --- | --- |
| Meeting | 4 |
| Meeting Minutes | 60 |
| Group Number |  |
| Attendees | Eryk, Elvis, Salvatore |
| Actions from Previous Meeting | Find alternate way of expressing morse code since displaying it through light is not working |
| Current Meeting | 06/05/2022 |
| Absentees | **Sam Lewis** |

# References:

Morsecodeclassnet. (2020). Morse Code Timing. [online] Available at: https://www.morsecodeclassnet.com/ch3-timing/ [Accessed 8 May 2022].

B. Gold, "Machine recognition of hand-sent Morse code," in IRE Transactions on Information Theory, vol. 5, no. 1, pp. 17-24, March 1959, doi: 10.1109/TIT.1959.1057478.