# GUI FLASH CARD: A FLASH HAT

Trương Lê Quỳnh Hoa: Coder

Nguyễn Ngọc Thái An: Coder

Nguyễn Hà Mỹ Tâm: Designer & Reporter

Nguyễn Thị Ngọc Hà: Designer & Reporter

Fulbright University Việt Nam

CS101: Computer Science I

Dr. Phan Thanh Trung

December 29, 2022

## Literature Review

Of all essential skills for the globalization age of the 21st century, being fluent in foreign languages is increasingly important and popular for most generations, from children to the old. In a highly tech-savvy world of population like these days, learning foreign languages always goes along with the application of up-to-date technology so that learners can always enhance their motivation, efficiency and productivity. When it comes to learning English at most public schools in Vietnam, vocabulary memorization has always been an inevitable part for any student to reinforce their knowledge of every textbook chapter, as well as pass periodic exams with flying colors. Understanding this concern of most Vietnamese students, our group are inspired to apply our course content related to Python programming language into launching a Flashcard app that can alleviate users' pressure from memorizing a specific list of new words, provide them with a scientifically-proved effective method of consolidating memory named spaced repetition through games, and incentivize their intrinsic self-study motivation thanks to trendy interface user design. Our Flashcard app is expected to be a supportive supplement to millions of Vietnamese students' personalized toolbox besides their compulsory English courses at school or any English extra-classes. Specifically, our main goal was to create a flashcard which runs on the terminal (will be interpreted more on the **BASIC FLASHCARD ON THE TERMINAL** part). However, our group wants users to understand how our flashcard program will look as an application (will be interpreted more on the **FLASHCARD ON PYGAME** part). Therefore, the group has applied the interface on the Review part of the program.

The idea of a Flashcard game is said to be no longer unfamiliar for most of us, especially with the rapid development of the Tech-Edu industry for the past few years. That being said, no single available language learning app perfectly fits with our target group of users. To clarify, we

take some language learning apps available on the market into consideration to gain some deeper insights about several limitations waiting to be resolved. The first and foremost place calls for Duolingo, the most widely used app for language lovers allowed to select within a variety of 39 languages, and over 61,000 exercises following carefully designed programs by levels, such as beginning, intermediate, and advanced (Brown, 2022). However, it is this huge freedom of Duolingo that unfortunately prevents users from creating their personal set of vocabulary required by their pre-existing learning programs. For the second place, it will be a shortcoming of us if Quizlet is not mentioned. This flashcard-based application is undeniably attractive for many students thanks to its numerous sources of learning materials contributed by users from worldwide not only in language learning but also core theory of textbooks in different fields at high schools, or colleges (*Quizlet*, n.d.). Nevertheless, having been created a long time ago, Quizlet's interface is quite academic, conservative, and boring for young students at lower school levels. Last but not least, most available apps served for learning and memorizing foreign languages are charging users to some extent, which is another burden on those having difficulty accessing technology in general and smartphones, or computers in particular. As a result, our Flashcard card is made to fill in the gaps related to users' personalized review content, and eye-catching interface.

## Result

### 1. Overall Progress

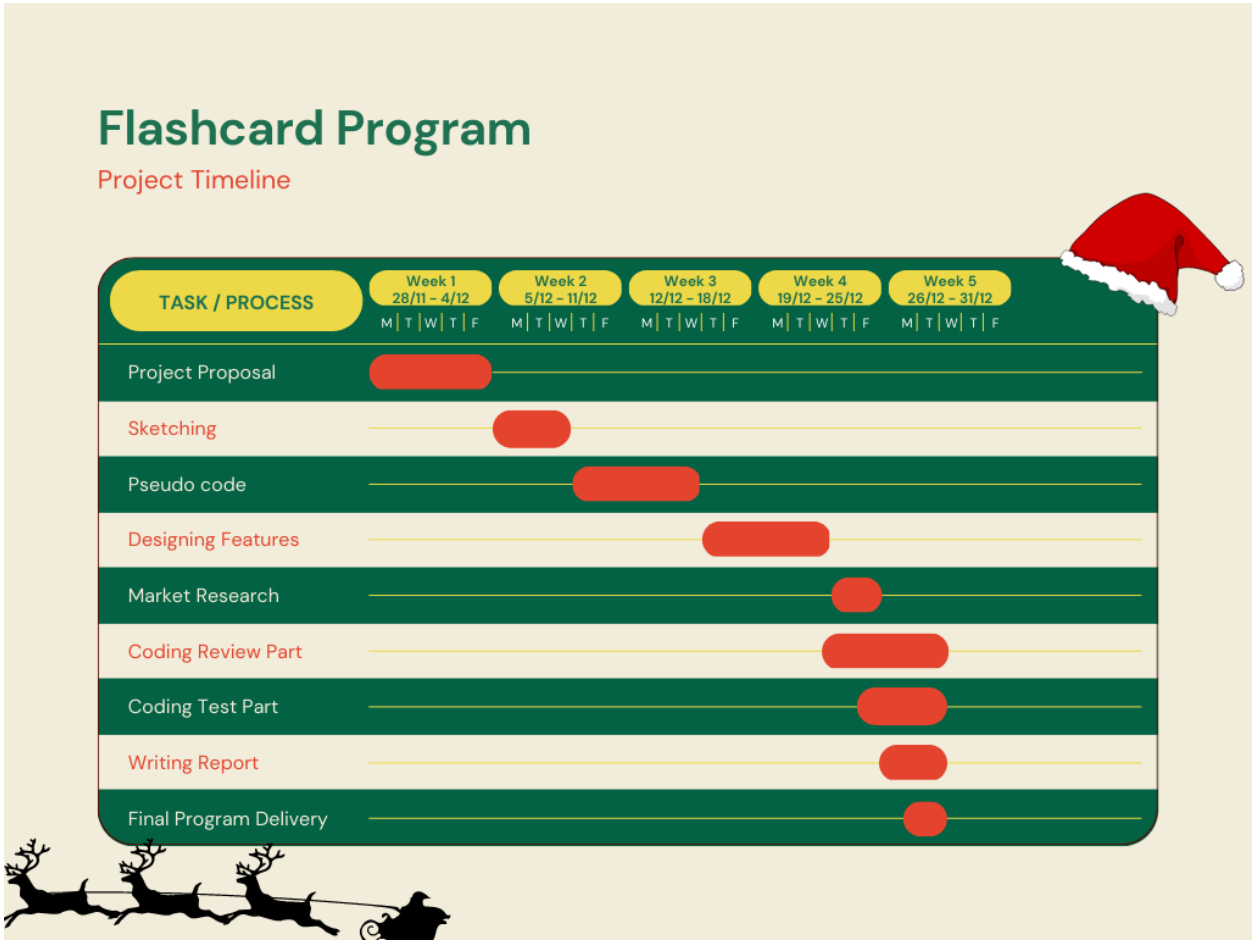


*Figure 1: Our Project Timeline using Gantt Chart*

There are some outcomes that we have achieved so far based on our Project Proposal as follows:

- Completed “Must have” features and nearly half “Should have” functions.
- Acquired the application of Object Oriented Programming

## 2. Program Explained

First, before starting coding the Flashcard application, our group had to mock up the program using hand sketches. In the first step, we sketched everything in our mind about the program's interface on papers:

1) Which function do we want the Flashcard to have?

2) What will each button look like in each feature? (for example: “next” button, “return” button, “start" button, etc.)

3) What will each page look like? (for example: Home page, Review page, Test page, etc.).

Then, we decided on the final interface for our Flashcard and started to convert it from paper to Canva design. We then used Powerpoint for easier downloading of each features' images in the future.

**BASIC FLASHCARD ON THE TERMINAL**

Our group needed to learn how to create a simple flashcard program using Python and Pygame. We began the program in the Main Module in Pycharm which contains many functions and classes so that we can easily follow the basic overview of the program structure.

We began with the def *readVocabCSV* for Python to read the filenames which we had already created. We used a tuple whose features have the same function as a list. Tuple is used in dictionaries and sets in which the element is immutable (Lemonaki, 2021). The *clrscr* def is used to clean the screen whenever the users switch to another function. We created a function to ask users which set of cards or rounds they want to choose (def *chooseVocabList*). We have 4 default sets of flashcards:

[1] B2 Vocabulary

[2] Environment

[3] Climate change

[4] Most common english verbs in spanish

Users are required to input a number (1/2/3/4) that is compatible with the topic. For example, if they want to learn vocabulary about Environment, input “2”. Or if they input anything that is not a number, the program will automatically exit (return -1).

The *game_Review* def function is to show all the words in the flashcard set that the user has chosen. The words will be printed line by line, each line spaced 0.1 second. In each line, it contains the number of a word, the word and its definition. After sets of words have been shown, the user has 3 options to choose: enter 0 if users want to go back to the **Menu**, enter 1 if users want to **Review** another set of words, and enter 2 if users want to take the **Test**.

Our group created 2 types of Test: **Multiple Choice** and **Short Answer**. For each test's form, we choose 15 words in the set to random (list *current_words*). The list *word_counter* is used to count the number of repetitions of the correct answer. Each correct answer will be counted as 5 points in the *current_points* variable. If the user answers correctly 4 times for a word that adds up to 20 points (5 points x 4 times), this word will not appear again on the Test.

With the def *game_MultipleChoice*, we decide to set the question to be a word (j = 0) and the answer is the definition of that word (j = 1). We checked the user's answer with the correct answer by checking the definition that the user chose for the word (in the question) to see whether they are matched with each other or not. We input the correct answer first, then later input the 3 wrong answers to the *lst_user_answer* and then use *random.shuffle* function to make sure the game will have different correct answers that change for every new question.

Unlike the def *game_MultipleChoice*, in the *def game_ShortAnswer*, we decide to make the question the definition of words (j = 1), and the answer is the word (j = 0). When the user inputs their word (the answer), we check it with the word to see whether the users' answer is correct or not. Everything else works exactly the same with the def *game_MultipleChoice*.

After finishing the Test, the def *game_Test* function is to prompt the user to select the test or go back to the menu. Users have 3 choices: input 1 if users want to choose Short Answer, 2 for Multiple Choice, and 0 to Back to Menu.

The *game_Introduction* def is where the user gets the instruction of the program, about how to learn the vocabulary through tests and make use of its functions. The *chooseMode* def is to prompt the user to select an option from the menu and the *main* def is to control the flow of the program. We also emulate the *main* function to make sure when the code is running, the function works correctly.

Note that our group faced some difficulty while applying the features of each function in the Test part. However, we could apply the features on the Review part in which each word in a vocabulary list would appear in the Christmas theme that we have designed before. We used *screen.blit* function to apply the features by inputting the pictures' name which we want them to be displayed on the screen, then the x and y position of the pictures (*screen.blit(image name, (x position, y position)*).

**FLASHCARD ON PYGAME**

After creating a basic flashcard on the terminal, our group also wrote another code file to create the general interface of A Flash Hat. First, in the **classes_of_main.py** file, there are a total

of 3 classes. With the ***class Button***, it will be used to create the button which has available images. The images' positions will be shown on the *def __init__*:

+ self.x_pos and self.y_pos (position that the group wants the button to be appeared on)
+ self.image (selected image for each button)
+ self.rect (create a boundary whose the left corner's position is the same with the self.x_pos and self.y_pos)

This class also contains 3 more functions: *def draw* to show the button's image on the screen, *def check_click_only* to check whether users have clicked the button or not (this def does not show the image) and *def update_n_check_click* which is the combination of the 2 functions above. Next, in the ***class Card***, it defaults the text fonts of our flashcard to the Marykate-Regular.ttf and the width of one card is 325 px. The main purpose of this class is to show the word and its definition on the screen. The last class in this file is the ***class Page***, it defaults there will be 3 words per page and its main purpose is to create many pages based on the sum of words on 1 dictionary file. Beside the 3 classes, there are 2 more functions: ***def wrap_text*** is for putting the text in line based on the cards' length. The other def is ***render_text_list*** which draws multiline text to a single surface with a transparent background.

Second, our group coded the **main.py** file for the program's interface which was divided into 4 parts: Menu, Info, Test, and Review. A quick note is that our Christmas-theme interface is just only applied on the Review part of the Flash Card program. Regarding the Test part, it has not been completed yet. There is a demo of the interface of multiple choice and short answer, not an official model so we are working and updating on it with the hope that we will have the interface on both the Review and Test part (*Figure 2*). To read the csv file, our group used *with open (filename, mode='r')* as file and *csv.reader()*. We have 4 files defaults for the program: B2_vocab.csv, Global-Climate-Change_vocabulary.csv, environment_vocab.csv and

spanish_vocab.csv. In addition, this folder will be updated in the future for diversity in topics. In the ***def menu***, the main purpose is to create an interface containing 3 buttons: Test, Review, and Info. These buttons were created by the **class Button** that we have mentioned before.

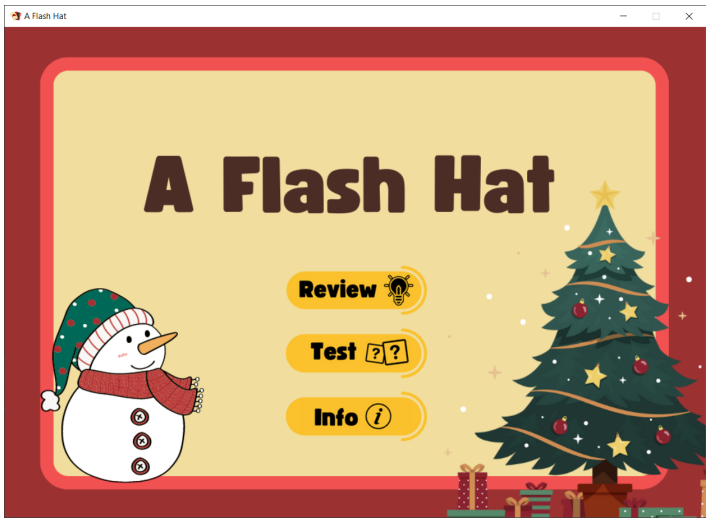


*Figure 2: Menu screen with applied Christmas-theme interface*

In the ***select_deck_function***, it will show 4 decks based on 4 files csv we have already read. When clicking on a deck, words and definitions (using the ***class Card***) will be appeared on the screen divided by ***class Page***.


*Figure 3: Deck selection screen with the applied Christmas-theme interface*

In ***def info_function***, it creates an introduction about A Flash Hat and test rules.


*Figure 4: Info function screen(s) with applied Christmas-theme interface*

The ***def review_function*** creates an interface for users to view the words with their definitions.

There are 3 cards per page.

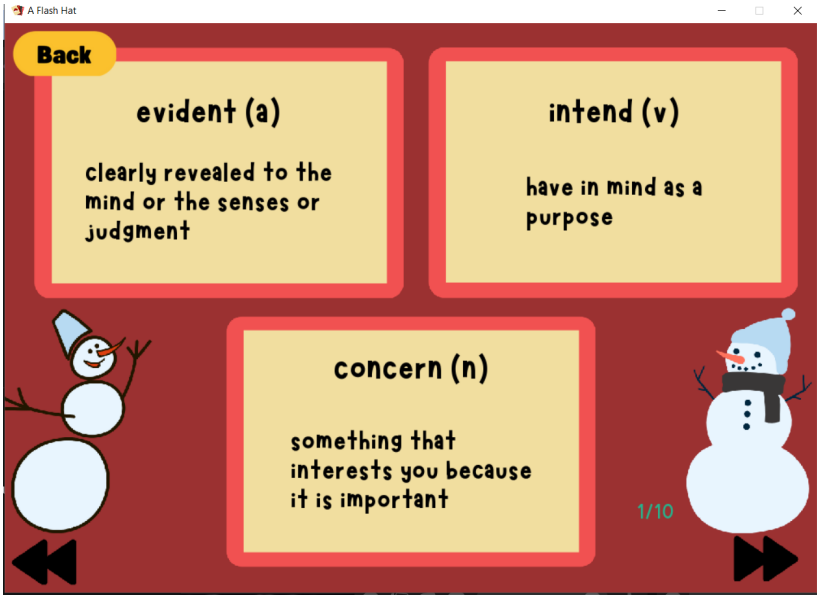


*Figure 5: Review function screen with the applied Christmas-theme interface*

In ***def test_function***, users can choose either one of the 2 buttons (Writing and Multiple Choice) which represent 2 types of tests: Multiple Choice (*def multiple_choice_function*) and Short Answer (*def writing_function*).

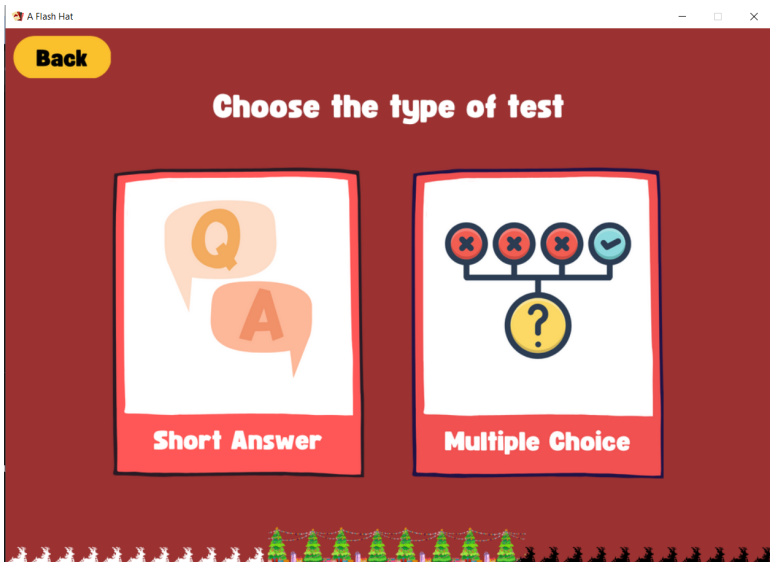


*Figure 6: Selection screen with applied Christmas-theme interface*

After users click on the option that they chose, the screens will appear respectively first the Ready screen, next the Question and 4 answers (for Multiple Choice test) or a box to input answer (for Short Answer test), and lastly the Congrats screen.

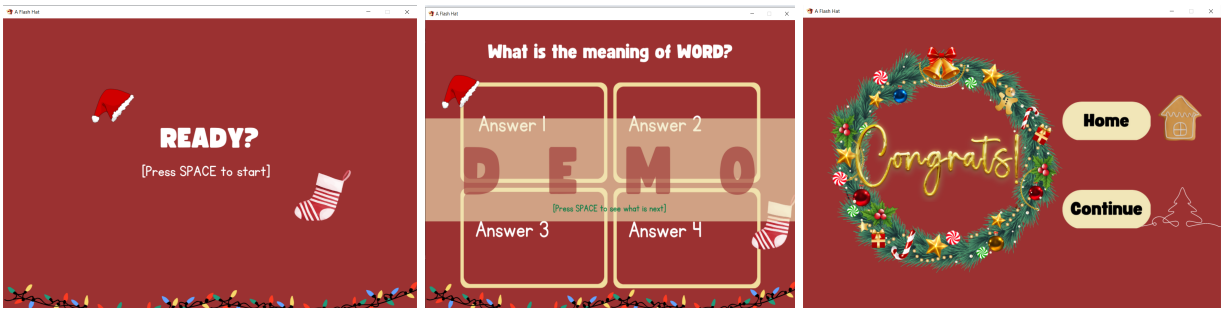


*Figure 7: Demo interface of Multiple Choice test screen*

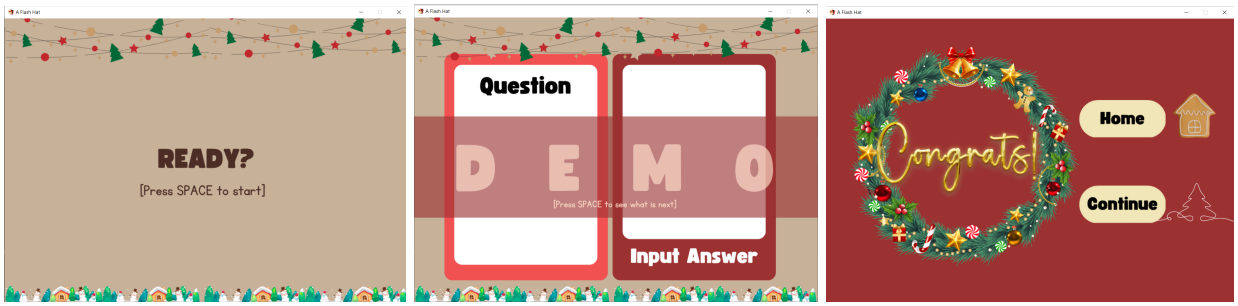


*Figure 8: Demo interface of Short Answer test screen*

Our group also coded one “could have" thing in the Project Proposal which was to add the background music for A Flash Hat. Our group used the mixer module (*mixer.music.load()*) in pygame to add background music while the program runs. Moreover, our group also created an app icon for our program (Figure 9).

*Figure 9: App icon for A Flash Hat*

**APPENDIX**

These are sources that we explore and apply while writing our program:

- Create button, and check if the user's mouse click at button is released or not : https://www.youtube.com/watch?v=G8MYGDf_9ho
- How we create out **Menu**:
    - Create transitions for multiple stages mentioned in the **Menu**: https://www.youtube.com/watch?v=GMBqjxcKogA&t=61s
    - Use images to make eye-catching button: https://www.youtube.com/watch?v=G8MYGDf_9ho&t=1s
- Basic knowledge about Pygame (how to change the app's logo, display images, add text, add background): https://www.youtube.com/watch?v=FfWpgLFMI7w&t=7354s
- Basic knowledge about how *class* operates: https://www.youtube.com/watch?v=rJzjDszODTI&list=PLFAUPcKFC97rygIbSOpl2oAFPMK1d338p&index=8&t=37s
- Understand the necessity of __name__ = "__main__" in a program: https://realpython.com/if-name-main-python/ https://www.youtube.com/watch?v=JQAeQUZxiz4&list=PLFAUPcKFC97rygIbSOpl2oAFPMK1d338p&index=7&t=141s
- Add background music to the game: https://www.youtube.com/watch?v=pcdB2s2y4Qc

**Command Prompt (CMD)**

- F-strings in Python:

  This string is used for more flexibility in changing between different formats (string and integer). We did not need to notice the space and we could adjust the formats that the

group wanted to be shown on the screen:

https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/

- Tuples

  Tuples are used to store multiple items in a single variable and it works exactly the same with the nested list but in a simpler way. People often use tuples in coding the question part in the flashcard especially when users need to choose 1 option in many options. (consult from https://stackoverflow.com/questions/65682576/how-to-make-multiple-choice-games-on-python).

  https://www.w3schools.com/python/python_tuples.asp

- Making a menu in Python:

  https://www.youtube.com/watch?v=63nw00JqHo0&t=195s

- How to clear the screen while the program is still running on CMD:

  https://docs.python.org/3/library/sys.html#sys.platform

**References**

Andy Dolinski (Director). (2020, May 11). *Making a menu in Python*. https://www.youtube.com/watch?v=63nw00JqHo0

BaralTech. (n.d.). *HOW TO MAKE A MENU SCREEN IN PYGAME! - YouTube*. Retrieved 26 December 2022, from https://www.youtube.com/watch?v=GMBqjxcKogA&t=61s

buildwithpython (Director). (2019, September 15). *Pygame Tutorial—16—Adding Sounds and background music*. https://www.youtube.com/watch?v=pcdB2s2y4Qc

Breuss, M. (2022, September 21). *What Does if __name__ == '__main__' Do in Python? – Real Python*. https://realpython.com/if-name-main-python/

Brown, S. (2022, December 14). *Duolingo Review: A Fun, Functional, Fresh Approach to Learning a Language*. CNET. https://www.cnet.com/tech/services-and-software/duolingo-review-a-fun-functional-approach-to-learning-a-language/

Chris Strange. (n.d.). *Python and Pygame: Creating a Simple Flashcard Program—YouTube*. Retrieved 26 December 2022, from https://www.youtube.com/watch?v=JQAeQUZxiz4&list=PLFAUPcKFC97rygIbSOpl2oAFPMK1d338p&index=7&t=141s

Coding With Russ. (n.d.). *PyGame Beginner Tutorial in Python—Adding Buttons—YouTube*. Retrieved 26 December 2022, from https://www.youtube.com/watch?v=G8MYGDf_9ho

freeCodeCamp.org. (n.d.). *Pygame Tutorial for Beginners—Python Game Development Course—YouTube*. Retrieved 26 December 2022, from https://www.youtube.com/watch?v=FfWpgLFMI7w&t=7354s

*f-strings in Python—GeeksforGeeks*. (n.d.). Retrieved 26 December 2022, from

https://www.geeksforgeeks.org/formatted-string-literals-f-strings-python/

Hallden. (n.d.). *Learn Classes in Python in 4 Minutes—YouTube*. Retrieved 26 December 2022, from https://www.youtube.com/watch?v=rJzjDszODTI&list=PLFAUPcKFC97rygIbSOpl2oAFPMK1d338p&index=9&t=37s

Lemonaki, D. (2021, September 20). *Python Tuple VS List – What is the Difference?* FreeCodeCamp.Org. https://www.freecodecamp.org/news/python-tuple-vs-list-what-is-the-difference/

*Pygame—How to make multiple choice games on python—Stack Overflow*. (n.d.). Retrieved 26 December 2022, from https://stackoverflow.com/questions/65682576/how-to-make-multiple-choice-games-on-python

*Python Tuples*. (n.d.). Retrieved 26 December 2022, from https://www.w3schools.com/python/python_tuples.asp

*Quizlet*. (n.d.). Quizlet. Retrieved 26 December 2022, from https://quizlet.com/vi

*sys—System-specific parameters and functions*. (n.d.). Python Documentation. Retrieved 26 December 2022, from https://docs.python.org/3/library/sys.html