Hanged Man Documentation

By: Nishaal Naseer

Student ID: 22023475

Contents

1. Introduction
2. Imports
3. Global Variables
4. Supporting Functions
5. Game logic

**Introduction**

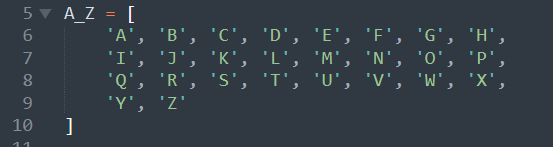
My hanged man game has a list of 20 words, and it selects one word initially at random. Player proceeds to enter a letter to guess the word by going through the alphabet through the B button of the microbit controller. Player selects the desired character with the B button. When a player guesses the correct word the game moves on to the next word, if a player fails to guess a word game ends, else when all 20 words are guessed player wins.

**Imports**

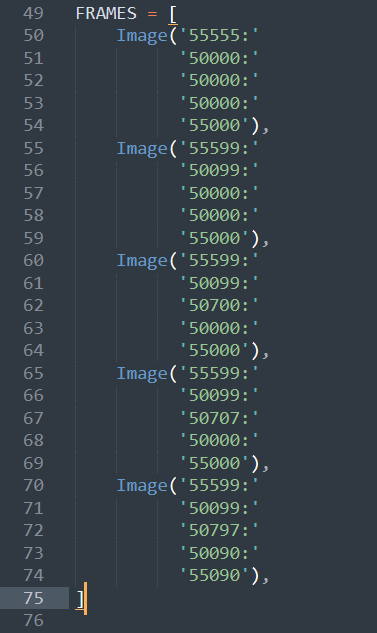


Line 1 imports the microbit modules that is necessary for, in this case display the state of the game. Line 2 imports randint() from the random module that will help select a random number within a given range, this is needed to shuffle the lists of words and to select a random letter from a word to give a hint to the player.

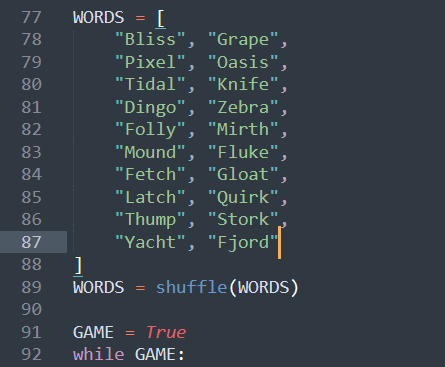
**Global Variables**



‘A\_Z’ is a list that is meant to hold the alphabet so the player can iterate through it to select the desired letter



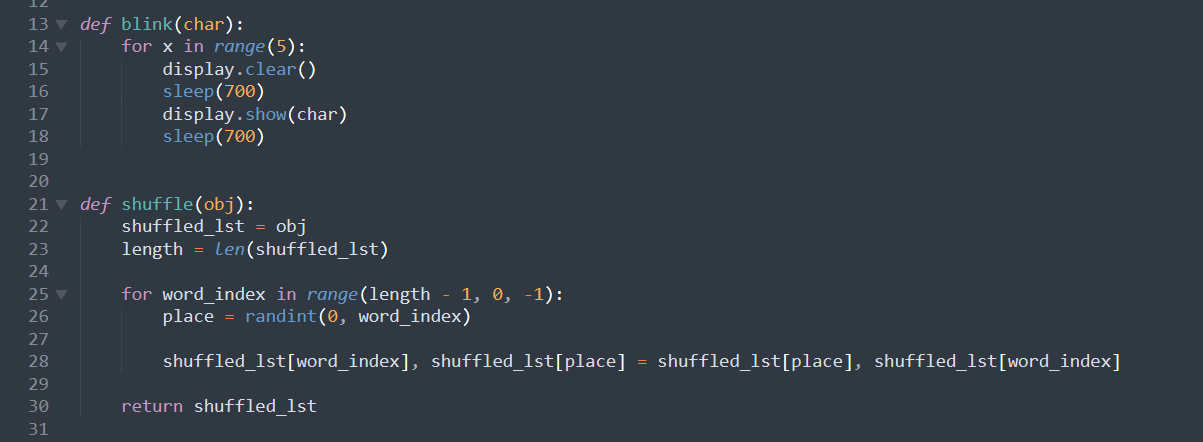
‘FAMES’ is a list of images that are shown to the player to show the progress of the game.



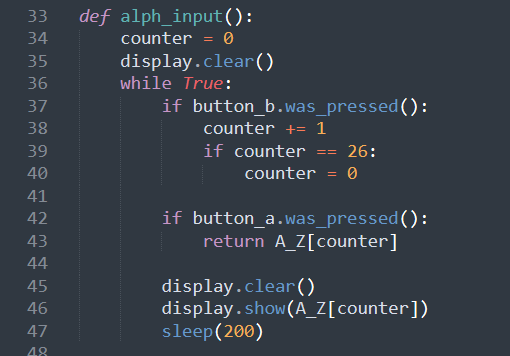
‘WORDS’ is the list of words that are remaining for the player, it is shuffled at the beginning, to give randomness to the game.

‘GAME’ for the main control flow of the game, as long as its ‘True’ game continues

**Functions**

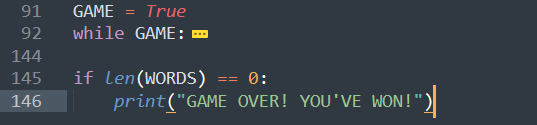


‘blink()’ function blinks a given char on and off screen. This is later used to give a clear indication the player has lost the game, ‘shuffle()’ function shuffles a given iterable.

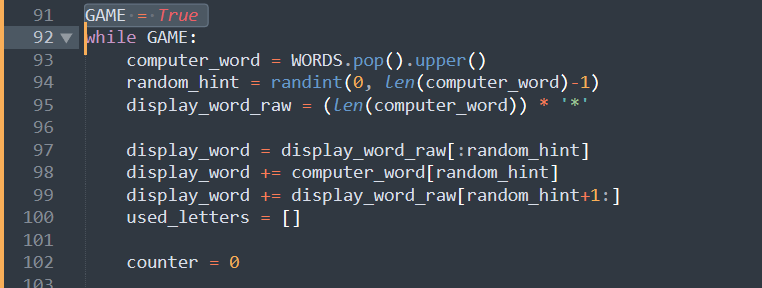


‘alph\_input()’ is the input function of the game, counter is initialized as 0, and the player gets to iterate through the alphabet by pressing button B, when player presses button A, the letter is returned

Game Logic



The whole game is in a while loop with ‘GAME’ variable as the conditional, if it turns False the length of ‘WORDS’ is checked and if it is 0 microbit displays "GAME OVER! YOU'VE WON!".



The above code concerns with the initialization of a new word.

Line 93 removes a word from the list and turns it into upper case, the removal from the list makes the game finite in length, and the upper case nature of the words combined with the values in ‘WORDS’ being upper case ignores case when equating the letter selected by player and the letter in a word.

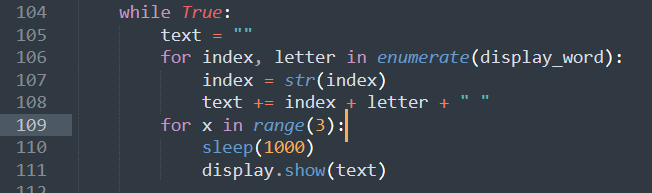
Line 94 selects a random number between 0 and the length of the word.

Line 95 makes a string consisting of only asterisks with the length of the selected word

Line 97 to 99 creates a string, consisting of a asterisks and a single letter, as a hint.

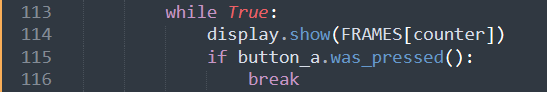
Line 100 creates a list to keep track of used letters, if not a player can keep hitting the same character over and over again and the game would not proceed.

Line 102 creates a variable to show the progress of a hanged man to the player

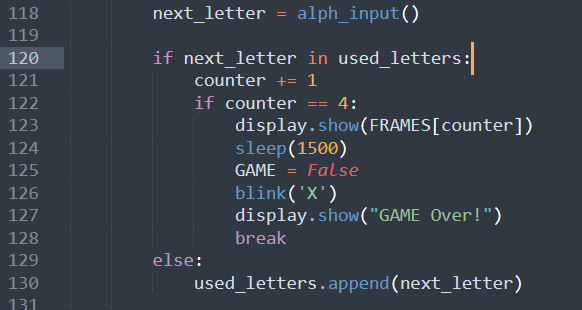


This second loop iterates until a word is completed or the player runs out of tries, and in the latter case the whole game ends

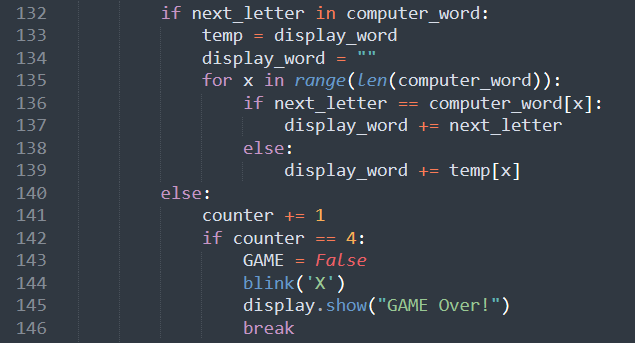
From line 105 to line 111 the current progress of the word is turned into a new string that has the index before the letter and is displayed to the player, the displaying is done three times



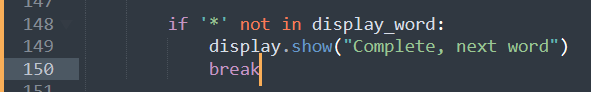
The current progress of hanged man is shown until the player hits button A, this is done so the player can look at the progress until satisfied.



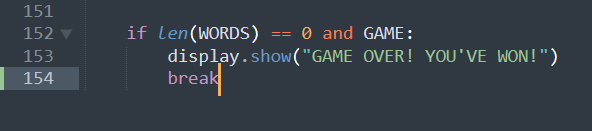
A player input is obtained from the ‘alph\_input()’. It is checked if it has been used before, if so the counter is incremented to keep the game finite else the letter is added to the list of used letters.



It is checked if the letter selected by the player is in the random word selected by the computer, if so the letter is properly added to the progress string, else the counter is incremented and if counter value is equals to 4 the games word loop is broken and main flag is set to false to avoid further iterations. Blinking is done to indicate the player has lost



We can check if a word has completed by checking if that word contains any asterisks and if it doesn’t we break the word loop and this will continue to the next word



This is to check if the player has won the game or not. The first part checks if there any remaining words, and the second part checks if the word loop was broken due to the player reaching the limit of his tries. This breaking is done in line 146 and ‘GAME’ was set to false at line 143