# FPP Report- <Chloe Yeo>,<2458600Y>

## What? < Mini games and a dive into history >

This program provides the user 3 mini activities to choose from, which includes hangman, guess the turtle game, and history of women activity.

The "Hangman" game is where the user has to guess the letters for the random word the program holds.

"Guess the turtle game" is where the user has to click on each square on a grid of squares to make it disappear until they find a turtle hidden underneath within the time limit for each round. Once the turtle is found, the user wins the round.

Finally, the "History of Women" activity shows user a set of buttons to click on – each button corresponding to a particular century. If the user clicks on a button, the information about women's historical achievements in that particular century appears on the screen.

## Brief user instructions

Unzip the folder then open the main file called "Mini games and a dive into history.py" and run it. The 3 mini programs are already imported in this file. So, run this file to play any of the 3 mini games or activities. However, if you want to take a look at the code for each mini program, click on these 3 files: "Hangman.py", "Guess the turtle game.py", and "History of women.py" – which are in the same zip folder.

While playing the mini activity called "History of women", when the screen is full of the historical events displayed, the user can use the horizontal and vertical scrollbars to read the full line of each historical event or move down to read the newly added lines below on the screen respectively.

## How it works

[Mini games and a dive into history]

In "Mini games and a dive into history", the user is asked to input a number corresponding to one of the 2 games or a history learning activity from the menu. Then, the user can play the game or activity she chooses. Now I will explain how each mini game or activity works.

1 – Hangman:

For "Hangman", first the window screen is set up to change size, background colour and add a descriptive text. Random library is used for the program to pick a random word in the list called words for user to guess. The words in the list called words are read in from each line of the file called "nouns.txt".

Then, onkeypress is used so that when user presses a letter in her keyboard, which is one of the alphabet letters in the string assigned to the variable name 'letters', either the Hangman

is or is not drawn, and the letters that the user pressed so far are printed out. Each time the user presses a wrong letter, the integer value associated with the variable named count goes up by 1. Using if statements and Boolean expressions, depending on the number of count, different lines of the Hangman are drawn using the turtle drawing.

If the user fails to guess all the correct letters in word to guess before the whole hangman is drawn, the user is informed that she has failed and that she has 0 lives left. Each time the user fails to guess the correct letter, the integer value associated with the variable name lives goes down by 1 and the user is informed how many lives are left. Otherwise if the user guesses all letters in the word before the whole Hangman is drawn, then it is drawn on the screen "You guessed it right!".

2 – Guess the turtle game:

For "Guess the turtle game", it also begins with the window screen being set up to change size, background colour and add a descriptive text on it. The display also includes "3-2-1-Go!" which is shown on the screen, before game starts, in a countdown.

Then first, the user is asked how many rounds she wishes to play, then a function called setup_round is called. Afterwards, brief instructions of how to play the game along with the round number is shown on screen, then the user is asked to input the number of rows and columns for drawing out the grid of squares each round.

While the integer value associated with the variable called rounds_passed is less than the total number of rounds the user wants to play, and while the number of rows and columns are small enough to fit into the screen, the index for the correct square is randomly chosen and the function called beginRound is called.

Every time the beginRound function is called, the timer is newly set depending on how many rows and columns there are. Also within this function, the function called drawSquare is called, in which a for loop is used to draw out each square and the values for shape and x,y coordinates for each square is returned, which is assigned to each square in the dictionary called squares by a nested for loop. When the timer runs out, the function called stopRound is called.

After the stopRound function is called, it is displayed on screen that time is up and user loses this round. As the round has finished, the user can no longer click any squares to make it disappear before the next round starts. At last, the setup_round function is called again to start a new round.

As the user clicks on each square on screen, the function called changeBox is called, and a black square is drawn on top of the red square to erase it. Using an if statement and a Boolean expression, if the index of the current square, which is returned from the function called findbox, is equal to the index of the correct square – i.e. if the user clicks on the correct square - a white turtle is shown on top of the black square and the user wins the round. The screen displays "You are the champion!" and after 2 seconds, the function setup_round is called to start a new round.

Each time the function setup_round is called, the turtle is reset and the integer value associated with the variable rounds_passed goes up by 1. Using another if statement with a Boolean expression, if the value of rounds_passed is larger than the number of rounds that the user inputted, the game scores are shown on the screen with a confetti effect made from many different shapes moving in random directions on screen, and then the screen closes.

3 – History of women:

For "History of Women", a screen is shown with a frame with listbox, vertical scrollbar, horizontal scrollbar, and buttons for each century that the user can click inside. At first the window size and title is set up. Each line in the file called historyOfWomen.txt is read in and each part of the line is split by ":::" and stored in a list. Lists with the same first two number of the year are stored in the same list which is associated to the key named via first two number of the years in the list of lists in the dictionary called history.

As the user clicks each button for a particular century, the function called click_century, with the century number shown on the button clicked used inserted as an actual parameter, is called. Afterwards, the relevant information for each century is displayed in the listbox in the screen in the format below -
(name) from (country) in (year):
(historical event).

## Data structures

Hangman:

I have used a list called words to store all the words in each line of the nouns.txt file that has been read. The list called letter_list stored all the letters in the word selected by the program randomly from the words list. The list called letters_pressed stores all the letters that user pressed in her keyboard so far that is in the string called letters which is a string of all alphabets.

Guess the turtle game:

Dictionaries are used in the form of dictionaries inside a dictionary called squares which has the index of each square as the key and where the value associated with each key is another dictionary with the shape, x and y coordinates, and width of square as the keys. Nested for loops are used in the function beginRound to assign a value for the x,y coordinates and width for each square index. The number of keys(the key is the index of square) in the squares dictionary is the number of squares in the rows*columns grid where rows and columns for each round(and thus for every time the beginRound function is called) are set by user input.

Women in history:

At first an empty dictionary called history is created. Then, the year, country, name and description of history read in from the historyOfWomen.txt file are stored in a list in order. The lists with the same first two numbers for the year are stored inside the same list. Then this list of lists becomes the value associated to the key in the history dictionary, where the key is the first two numbers of the years in that list of lists.

### Interesting / tricky / subtle aspects of the code meriting further explanation

In “History of women”, for each button for each century, I encapsulated the function call click_century(num_th_century) within a lambda so that it is not automatically called when assigned to command but still has an association with the argument.

In “Guess the turtle game”, the turtle is actually not ‘underneath’ the square and the square is not actually ‘erased’ – instead, after the red squares are drawn, as the background is a black colour, black square is drawn on top of the red squares for each square in the dictionary called squares to make it look like it’s been erased. Then, the white turtle is shown on top of that black square.

In “Hangman” – in the beginning, an empty list called correctLetter is created. Then for n times where n is the length of the word the user has to guess, an underscore is appended into the correctLetter list. Then, if the letter that user pressed is one of the correct letters and also the one that has not been pressed yet, then the underscore(s) in the correctLetter list is replaced with that letter the user just pressed. This is done by using a for loop for the length of the word to guess – for each letter in the word to guess, if it is the same letter as the one that user pressed, then the index of that letter in word to guess is used so that the underscore in the correctLetter list with the same index is replaced with the letter.

## Particular challenges I overcame

In the guess the turtle game, it took a long time until I figured out how to make each round finish so the user can no longer click any other squares in that round after having selected the correct square with the white turtle underneath. I overcame this challenge by using turtle.onscreenclick(None) so that nothing happens when I click the screen.

Another error that took me a long time to solve was the white turtle not showing up on the screen. By using turtle.clearscreen() before the code for showing the white turtle was executed, I could see that the turtle was successfully being drawn in the background, however I still could not see it on screen when I removed the turtle.clearscreen(). For this and similar errors where for example the black square did not appear to draw out(to remove the red square) when it was actually being drawn in the background, I used turtle.update() in the following line.