Assignment 1

Ting Hu

S330823

# Program

Hangman version 1:

Play stage:

A picture containing drawing

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated



A screenshot of a cell phone

Description automatically generated

A picture containing clock

Description automatically generated

A picture containing clock

Description automatically generated

A screenshot of a cell phone

Description automatically generated



A close up of a logo

Description automatically generated

A close up of a logo

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing drawing

Description automatically generated

In this first version of HANGMAN game, it is well-organized with prompt. The game starts with 6 lives, which allows user to guess 6 wrong times, with each wrong guess user’s lives will be taken away once and the diagram will indicate one piece of the hanging man’s body. Eventually, when the user use up all the lives, a whole body of hangman will show up and the game ends.

If the user guessed the word, the user will receive the win message and also with a prompt to choose play another game or no.

If the user input with invalid letter, they will not get live deduction but still receive a prompt to guess a letter, as the definition of HANGMAN game is guess the English word, so the invalid input include symbols or numbers will not be counted.

In this version, the program can be run functionally, but with the last stage of losing game, there will show up 2 hangman’s diagram.

**Fixed of version 1-1:**

A picture containing clock

Description automatically generated

In this fixed version, the last stage will only show one hangman diagram, and problem is fixed.

# Refactoring

In the version 1, when user guessed the wrong letter, there will always be deducted one life and show the hangman diagram. In the code, it can functionally show the diagram but each wrong attempt has its own diagram showing statemen, which means we can refactor this part of code.

**Before**A screenshot of a cell phone

Description automatically generated

**After refactoring (version 1-2)**

A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

A close up of a logo

Description automatically generated

As playing the HANGMAN game in the real life, the hangman diagram is always showing , so here we recode it and reduce the code duplication and the hangman can be shown logically.

# Codeversion 1-2

import random

from words import words

import string

#randomly get a word from the words list

def get\_word(words):

word = random.choice(words)

while '-' in word or ' ' in word:

word = random.choice(words)

return word.upper()

#hangman game function

def hangman():

word = get\_word(words)

word\_letters = set(word) # letters in the word

alphabet = set(string.ascii\_uppercase)

used\_letters = set() # what the user has guessed

#test lives

lives = 6

print("Let's play Hangman!")

print(display\_hangman(lives))

print("\n")

# getting user input

while len(word\_letters) > 0 and lives > 0:

#test used letters

print('You have', lives, 'lives left.')

# what current word is

word\_list = [letter if letter in used\_letters else '-' for letter in word]

print('Current word: ', ' '.join(word\_list))

#test remove function

user\_letter = input('Guess a letter: ').upper()

if user\_letter in alphabet - used\_letters:

used\_letters.add(user\_letter)

if user\_letter in word\_letters:

word\_letters.remove(user\_letter)

print('')

#test remove live function

else:

lives = lives - 1 # takes away a life if wrong

print('\nYour letter,', user\_letter, 'is not in the word. \nAnd you have used these letters: ', ' '.join(used\_letters))

elif user\_letter in used\_letters:

print('\nYou have already used that letter. Guess another letter.')

else:

print('\nThat is not a valid letter.')

print(display\_hangman(lives))

print("\n")

# gets here when len(word\_letters) == 0 OR when lives == 0

if lives == 0:

print('You died, sorry. The word was', word)

else:

print('YAY! You guessed the word', word, 'You win!!')

def display\_hangman(lives):

stages = [ # last state: head, body, both arms, and both legs

"""

--------

| |

| O

| \\|/

| |

| / \\

-

""",

# head, body, both arms, and one leg

"""

--------

| |

| O

| \\|/

| |

| /

-

""",

# head, body, and both arms

"""

--------

| |

| O

| \\|/

| |

|

-

""",

# head, body, and one arm

"""

--------

| |

| O

| \\|

| |

|

-

""",

# head and body

"""

--------

| |

| O

| |

| |

|

-

""",

# head

"""

--------

| |

| O

|

|

|

-

""",

# initial empty state

"""

--------

| |

|

|

|

|

-

"""

]

return stages[lives]

def main():

hangman()

while input("Play Again? (Y/N) ").upper() == "Y":

hangman()

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Github

<https://github.com/DylanCDU/Hangman>