Name: Anh Phan

Assignment: hard hangman

ID: ajp6959

Text

Description automatically generated

def read\_file(directory):

    """

    Open file and read its content

    Param:

    - directory: directory of a file

    Return:

    - a list that each value is one line of the txt

    """

    try:

        with open(directory) as file:

            text = file.readlines()

            for index in range(0, len(text)):

                word = text[index]

                if "\n" in word:

                    text[index] = word[:-1]

            return text

    except FileNotFoundError:

        print("File is not found")

def word\_len(word\_list):

    """

    Count how many letter in the word to determine the number of guess

    Param:

    - word\_list: the given world list to be played

    Return:

    - word\_len: an int represent number of guesses

    """

    word\_len = 0

    for word in word\_list:

        word\_len = len(word)

    return word\_len

def match\_no\_match(char, word\_list):

    """

    Count the option of words that will match and will not match with the letter that is passed in

    Param:

    - char: a character that is chosen

    - word\_list: a list that was imported from outside containing words that could be used

    Return:

    - match\_list: a list of words that would match with the selected letter

    - no\_match\_list: a list of words that would not match with the selected letter

    """

    match\_list = []

    no\_match\_list = []

    for word in word\_list:

        if char in word:

            match\_list.append(word)

        else:

            no\_match\_list.append(word)

    return match\_list, no\_match\_list

def remove\_unsatisfied\_index\_match(word\_list, guess\_index, char):

    """

    Removing the word that at a given index does not contain the required letter

    Param:

    - word\_list: the list of word that can be used

    - guess\_index: the index that is required to have a given letter

    - char: the letter that is needed at the given index

    Return:

    - word\_list: a word list that contains all the valid words that satisfied the condition that there is a given letter that at certain index

    """

    for word in word\_list:

        if word[guess\_index] != char:

            word\_list.remove(word)

    return word\_list

def list\_init(value, length):

    """

    Initializing a new list

    Param:

    - value: what will be the default value of the list

    - length: the length of the list

    Return:

    - list: a list contains all given value with a given length

    """

    return [value] \* length

def choosing\_word(guess\_list, correct\_match, no\_match):

    """

    Selecting the word that will continue be used for the game.

    - If there is more match than no match with a certain letter, the match words will be chosen to continue the game

    - If there is more no match or equal match and no match, the no match words will be chosen to continue the game

    Param:

    - guess\_list: the list of letter that was correctly guessed

    - correct\_match: the list of words that is determined to be matching with the chosen letters

    - no\_match: the list of words that is determined to no tbe matching with the chosen letters

    Return:

    - The word list that is chosen to continue the game

    - A boolean: True if the match word list was used/ False if the no match list was used

    """

    if len(correct\_match) < len(no\_match) or len(correct\_match) == len(no\_match):

        return no\_match, False

    else:

        return correct\_match, True

def choosing\_index(char, word\_len, word\_list):

    """

    Choosing the index that most word in the list contain the given letter

    Param:

    - char: the letter that will be used to determine match

    - word\_len: how many letters are therein the word

    - word\_list: the list of word that will be processed for the game

    Return:

    - index in the word that match with the given letter for majority of the words in word list

    """

    index\_count\_list = list\_init(0, word\_len)

    for word in word\_list:

        for index in range(len(word)):

            if char == word[index]:

                index\_count\_list[index] += 1

    return index\_count\_list.index(max(index\_count\_list))

def guess\_word\_format(guess\_list):

    """

    Creating a string version of the correctly guessed letter list for display

    Param:

    - guess\_list: a list of correctly guessed letters

    Return:

    - guess\_word: a string version of the guess list

    """

    guess\_word = ""

    for char in guess\_list:

        guess\_word += char

    return guess\_word

def guess\_complete(guess\_word):

    """

    Checking if the word is completely guessed.

    If so, the game ends and player wins

    Param:

    - guess\_word: the word that is being chosen for guessing (string)

    Return:

    - boolean: True if completed/ False if not completed

    """

    if "\_" not in guess\_word:

        return True

    else:

        return False

def result(is\_complete, word):

    """

    Printing the result of the game

    Param:

    - is\_complete: a boolean True for finish, False for not finish

    - word: the word that is being guessed

    """

    if is\_complete:

        print(f"You have guessed {word} correctly!")

    else:

        print("0 guesses left you lost!")

def main():

    words = read\_file("PSU\CMPSC 131\HW\Hangman\wordlist.txt")

    guess\_chances = int(input("Number of guess: "))

    word\_length = word\_len(words)

    guess\_times = guess\_chances

    guess\_list = list\_init("\_", word\_length)

    guess\_word = guess\_word\_format(guess\_list)

    is\_complete = guess\_complete(guess\_word)

    while ((not is\_complete) and (guess\_times > 0)):

        letter = input(f"Please guess a letter for the word ({guess\_chances} guesses left): {guess\_word}\nGuess: ")

        possible\_match, no\_match = match\_no\_match(letter, words)

        words, is\_match = choosing\_word(guess\_list, possible\_match, no\_match)

        if is\_match:

            chosen\_index = choosing\_index(letter, word\_length, words)

            guess\_list[chosen\_index] = letter

            words = remove\_unsatisfied\_index\_match(words, chosen\_index, letter)

        guess\_word = guess\_word\_format(guess\_list)

        is\_complete = guess\_complete(guess\_word)

        guess\_times -= 1

    result(is\_complete, guess\_word)

main()