CECS 174 - Project 4
"CECS 174 - style new and improved Wordle"
Due date: 05/13/22

Team:

Student Name: Joshua Correa
Student ID: 029196984

I certify that this submission is my original work

Joshua Correa

Student Name: Alfredo Regla
Student ID: 027962816

I certify that this submission is my original work

Alfredo Regla

Project Report: Programming Project 4 - "CECS 174-style new and improved Wordle

1. **Goal:**
   Our goal was to improve upon what we created in Project 3 (a Wordle-like game). It was important that we keep similar code from Project 3 in order to ensure that this is an actual improvement and not a completely new game. In order to improve the game it was also important that we use various methods, functions, and techniques that would make the game easier to understand (from a code perspective) but more efficient from a user perspective.

2. **Problem Description:**
   We were required to implement various functions and methods that would replace the relatively crude code in the previous project. This ment making functions for pieces of code that we would frequently use rather than copy and pasting constantly. We also implemented methods that solved some problems that seemed impossible to fix in Project 3. Besides the functions, methods, and other new techniques the code is relatively the same as Project 3.

3. **Program Description:**

a.

Before we begin asking the users to input the **secret** and **player** word, we import a string module and create a list of all 26 letters in the alphabet and store them in the variable **remaining_alphabet**, while we're creating this list we will the lists **in_secret_word_correct_spot,in_secret_word_somewhere,**and **not_in_secret_word** we will use these lists later for storing the letters of player attempts. We will then create the functions **read_dictionary**(open the provided file to make a list of valid words), **enter_a_word**(asks the user to enter a {word_length}-lettered {wordtype}), **is_it_a_word** (checks if the player/secret word is valid according to the wordlist), **enter_and_check** (combines the previous functions in order to validate/enter the player or secret word), and **compare_words** (the function with the most code that compares player and secret word and puts the respectively letters the lists previously stated; and removes letters that have been used on that attempt from the **remaining_letters** list). Once we've done all of that we will now ask the user to input the **secret_word** (which goes through the process above to validate). Once validated, we then ask the user to input a number of attempts for the other user which is stored in variable **N**. After a number of attempts is set the next user enters their guess for the secret word, which is also validated through the same process. Once both words are validated and a number of attempts is set, the **compare_words** function is used to find out which letters in the player word correspond to their respective list (stated above) and then outputs the list as well as a **final_word** which shows the letters in the player words that are correct in correct spot, correct in wrong spot, or just an underscore for not in secret word. Once the words are compared the process repeats until the player either guesses right or runs out of attempts.

b.

Test Case #1:

Welcome to new and improved Wordle - CECS 174 edition!
Enter the 5-letter secret word: cycle
Input allowed number of attempts: 2
Enter your attempt #1
Enter the 5-letter player word: s
You entered a 1-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word:

You entered a 0-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: salt
You entered a 4-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: apple
letter in the right spot: 2
You guessed letters of the secret_word: ___le
Previously attempted letters that are in the correct spot of secret_word:
['l', 'e']
Previously attempted letters that are in some spot of secret_word:
[]
Previously attempted letters that are not in the secret_word:
['a', 'p']
Remaining letters of the alphabet that have not been tried:
['b', 'c', 'd', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
Enter your attempt #2
Enter the 5-letter player word: cycle
letter in the right spot: 5
You guessed letters of the secret_word: cycle
Previously attempted letters that are in the correct spot of secret_word:
['l', 'e', 'c', 'y']
Previously attempted letters that are in some spot of secret_word:
[]
Previously attempted letters that are not in the secret_word:
['a', 'p']
Remaining letters of the alphabet that have not been tried:
['b', 'd', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'z']
Congrats you won using 2 attempt(s)

Explanation: We chose this test case to check if letters in the lists would repeat and different attempts.

Test Case #2:
Welcome to new and improved Wordle - CECS 174 edition!
Enter the 5-letter secret word: 2345678
You entered a 7-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: skill
Input allowed number of attempts: 0

Explanation: We chose this test case to see how our code could handle numbers instead of letters. (the output for 0 in attempts allowed is nothing)

Test Case #3:

Welcome to new and improved Wordle - CECS 174 edition!
Enter the 5-letter secret word: sushi
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: mario
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: asdfg
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: banana
You entered a 6-letter word that is in the dictionary. Please try again!
Enter the 5-letter secret word: 123243546576879
You entered a 15-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: 0987654123456789876543234567876543456765434567654345676543
You entered a 58-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: skill
Input allowed number of attempts: 1234567890
Enter your attempt #1
Enter the 5-letter player word: banana
You entered a 6-letter word that is in the dictionary. Please try again!
Enter the 5-letter player word: sushi
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: mario
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: skill
letter in the right spot: 5
You guessed letters of the secret_word: skill
Previously attempted letters that are in the correct spot of secret_word:
['s', 'k', 'i', 'l']
Previously attempted letters that are in some spot of secret_word:
[]
Previously attempted letters that are not in the secret_word:
[]
Remaining letters of the alphabet that have not been tried:
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'j', 'm', 'n', 'o', 'p', 'q', 'r', 't', 'u', 'v', 'w', 'x', 'y', 'z']
Congrats you won using 1 attempt(s)

Explanation: This test case was chosen to see how the program would handle words that would be valid, but aren't because they aren’t on the list (although they could be if we wanted to).

Test Case #4

Welcome to new and improved Wordle - CECS 174 edition!
Enter the 5-letter secret word: '.[].']/-=[
You entered a 11-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: apple\n
You entered a 7-letter word that is not in the dictionary. Please try again!
Enter the 5-letter secret word: APPLE
Input allowed number of attempts: 3
Enter your attempt #1
Enter the 5-letter player word: lIlIpuTIaN
You entered a 10-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: truck
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: \\\\\\
You entered a 6-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: apple
letter in the right spot: 5
You guessed letters of the secret_word: apple
Previously attempted letters that are in the correct spot of secret_word:
['a', 'p', 'l', 'e']
Previously attempted letters that are in some spot of secret_word:
[]
Previously attempted letters that are not in the secret_word:
[]
Remaining letters of the alphabet that have not been tried:
['b', 'c', 'd', 'f', 'g', 'h', 'i', 'j', 'k', 'm', 'n', 'o', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
Congrats you won using 1 attempt(s)

Explanation: We used this test case to test our programs capability for accepting all uppercase letters as well as rejecting entries that had non-alphabet characters in them.

Test Case #5:

Welcome to new and improved Wordle - CECS 174 edition!
Enter the 5-letter secret word: mario
Input allowed number of attempts: 4
Enter your attempt #1
Enter the 5-letter player word: luigi
You entered a 5-letter word that is not in the dictionary. Please try again!

Enter the 5-letter player word: peach
You entered a 5-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: bowser
You entered a 6-letter word that is not in the dictionary. Please try again!
Enter the 5-letter player word: mario
letter in the right spot: 5
You guessed letters of the secret_word: mario
Previously attempted letters that are in the correct spot of secret_word:
['m', 'a', 'r', 'i', 'o']
Previously attempted letters that are in some spot of secret_word:
[]
Previously attempted letters that are not in the secret_word:
[]
Remaining letters of the alphabet that have not been tried:
['b', 'c', 'd', 'e', 'f', 'g', 'h', 'j', 'k', 'l', 'n', 'p', 'q', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
Congrats you won using 1 attempt(s)

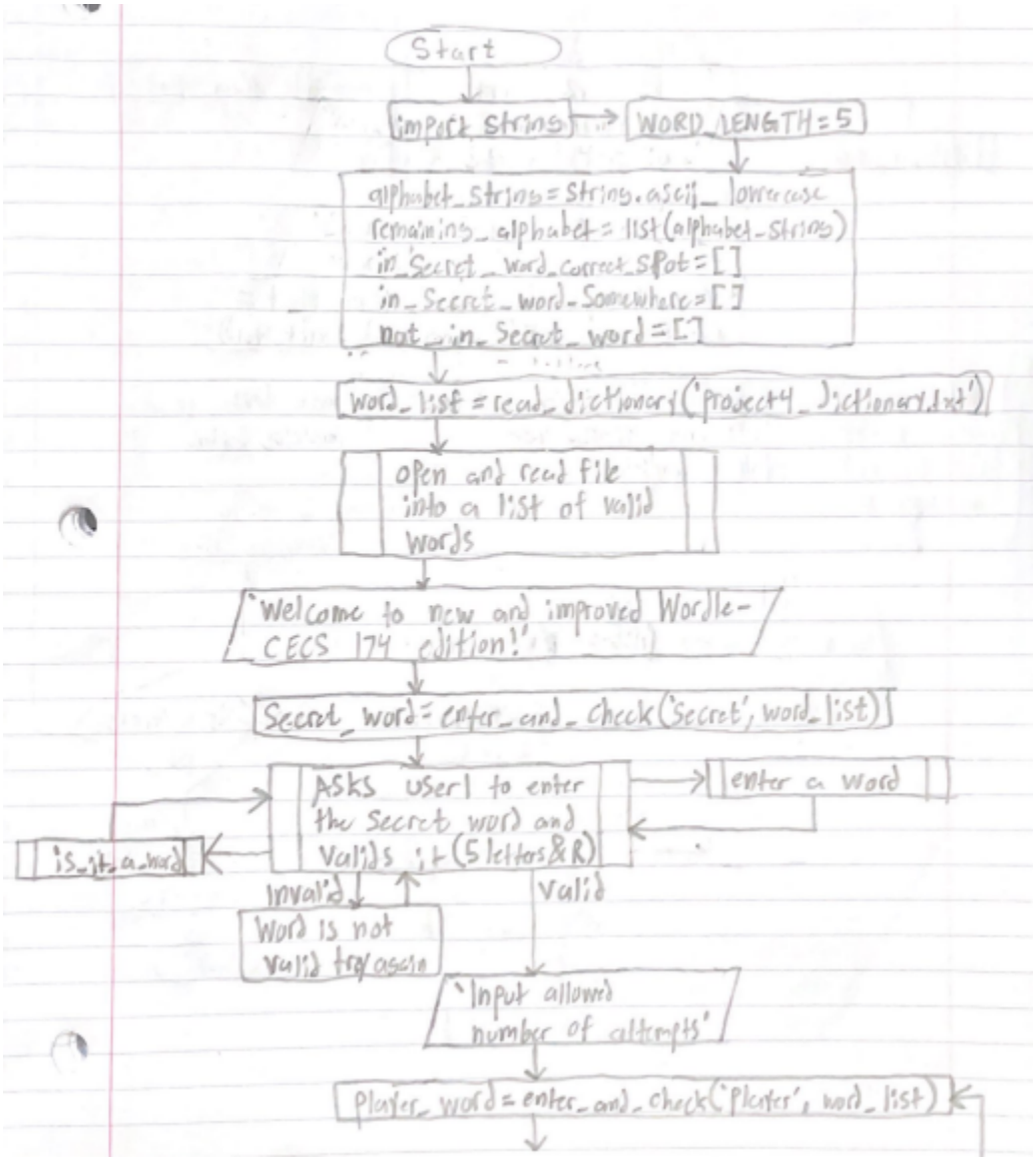
Explanation: We added this piece of code to see if we could add new words to the existing dictionary and have the code still work.

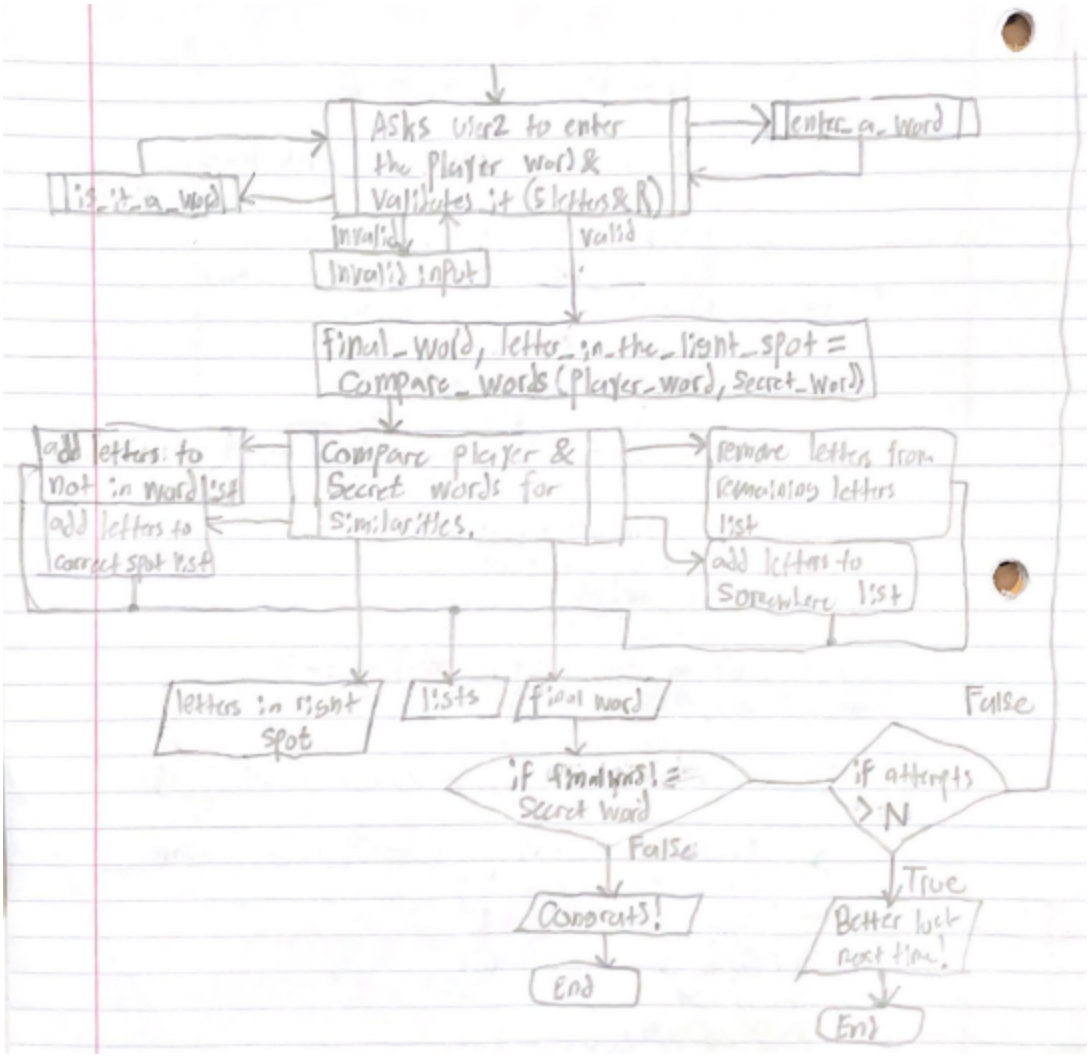
c.

First we ask user1 to input the secret_word, their input is sent through a series of functions to validate if it is 5-letters and an existing word in the list provided to us. We can make a simple if statement to check if the word is 5-letters, and we can make a simple if statement that returns if the letter is in the list. If the word does not meet both of the requirements user1 is asked to input again. After user1 enters a valid word they are then asked the total number of attempts user2 is allowed to guess the secret word. If user1 inputs 0 the program ends, and anything above 0 is valid. Once a total number of attempts is set, user2 steps in to guess what the secret word is by entering a word. That word goes through the same process as the secret word to see if it is valid. Once it is checked user1 and user2 words are compared. If a letter in user2's word is the same as user1's word and is in the same position then the letter is added to a list, is removed from remaining letters, and is added to the final word. If a letter in user2's word is the same as user1's word but is in the wrong position then the letter is added to a different list, is removed from remaining letters, and is added to the final word in parentheses. If a letter in user2's word is not in user1's it is added to a list of letters not in secret word, is removed from remaining letters, and is replaced in the final word with an underscore. If user2 didn't solve the secret word, then

the the program asks user2 to enter a new word and the process repeats. Once user2 either enters the correct guess or runs out of attempts the program ends.

d. .


Start
import strings
WORD_LENGTH=5
alphabet_string=string.ascii_lowercase
remaining_alphabet=list(alphabet_string)
in_secret_word_correct_spot=[]
in_secret_word_somewhere=[]
not_in_secret_word=[]
word_list=read_dictionary('project4_dictionary.txt')
open and read file into a list of valid words
'Welcome to new and improved Wordle- CECS 174 edition!'
Secret_word=enter_and_check('secret', word_list)
Asks user1 to enter the secret word and valids it (5 letters&R)
enter a word
is_it_a_word
invalid
Word is not valid try again
valid
'Input allowed number of attempts'
player_word=enter_and_check('player', word_list)



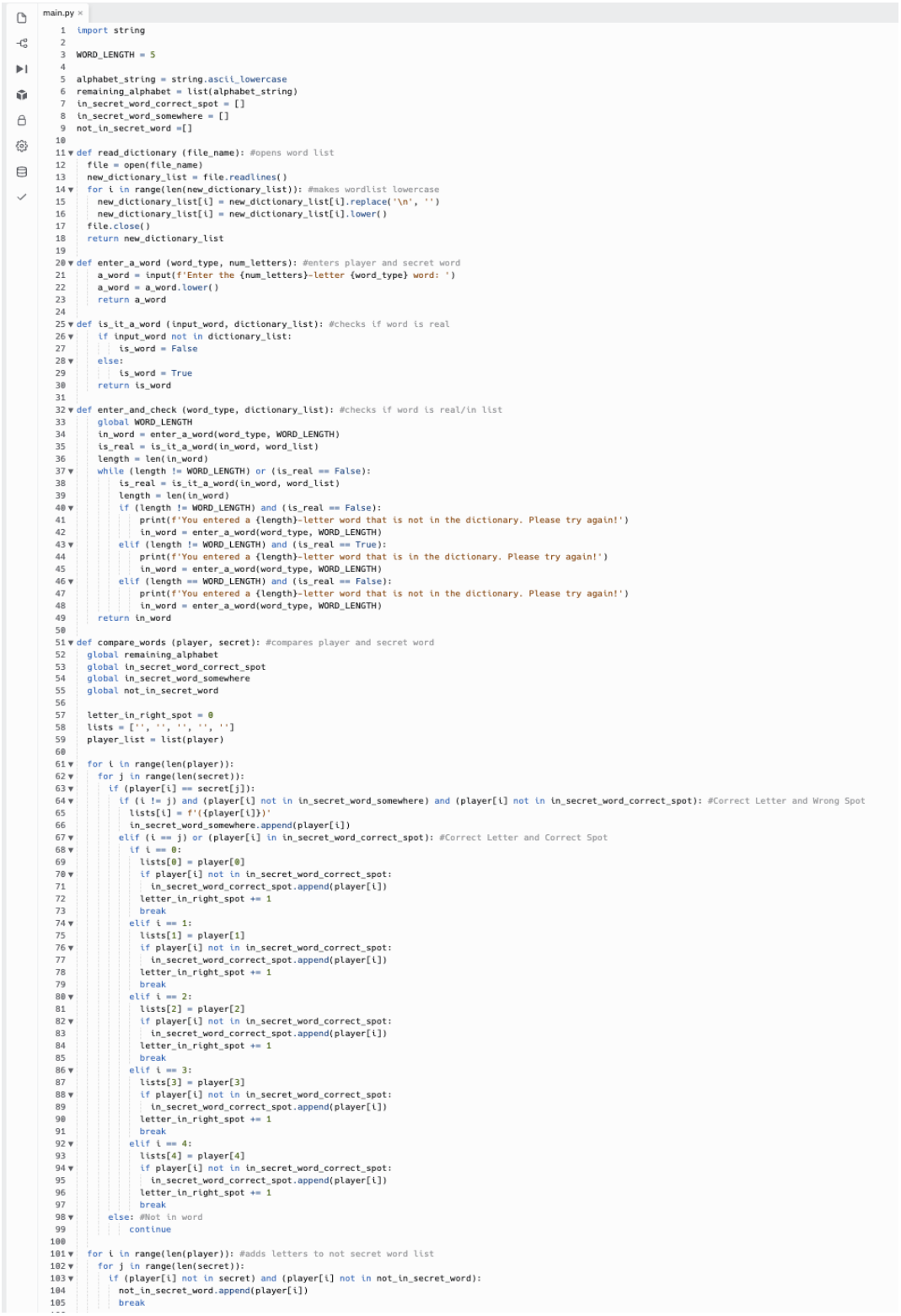
4. **Program Implementation:**
    a. For this code we only used int and str data types. Int was used to grab the number of allowed attempts and str was used for the input when the user either guesses a word or tells you the secret word. The operations that we used were things like !=, ==, <. Does not equal was used for checking if the guessed word was not the same word as the secret word and other things as well and the same goes for Equal Equal as it was used to check if the secret word equaled the guessed word and other things. Finally, in order to keep the loop going with had to use < to keep the number of attempts under the number of the attempts the user wanted. We used for loops and while loops throughout our code to keep attempts under user number and things like using a for loop to check if there are duplicates. in a list

b. The most challenging part for us was when we tried to print, 'Previously attempted letters that are in the correct spot of secret_word:', it would print two of the same letter instead of one. So we were confused on how to fix it until we figured out that we can use a for loop to get rid of the duplicate letter.
c. The most fun part was implementing the print at the end of the code because after doing all that code, and testing if it worked was reliving.
d. Yes we added more test cases after because we didn't know how our program would handle 5 letter words with 2 of the same letters. We knew it worked with the letters like skull and skill but we weren't sure with other letters so we tested many other words that had double letters.
e. Our program handles bad input pretty well as in every single input except one it doesn't allow you to enter anything except a 5 letter word or it just tells the length of the input and tells you that it isn't a word. The only case where our code brick is in the beginning where you are asking for the number of attempts and if you enter anything other than a number the code bricks.
f.
    i. The only bug we had is that when you enter a string into the number of attempts, the code breaks.
    ii. Another error we had is that the code would print double letters when showing the letters that are in the word. We fixed this by adding a for loop in order to get rid of the duplicates.

5. **Conclusion:**
    a. What went well is that we brainstormed ideas in order to allow our code to work. We thought of different ways to use functions to make a specific part of our code work.
    b. I think we could have managed time a little better by figuring out basic functions in the beginning so that in the end we just have to write everything out.
    c. I think giving us the idea of functions we could have used, could've maybe helped us have an easier time coding.
    d. I think coding wordle again was a bit tedious because people had done it already so maybe they didn't want to code it again, so maybe something else instead? although I just enjoyed improving it since we learned new functions.

Appendix:

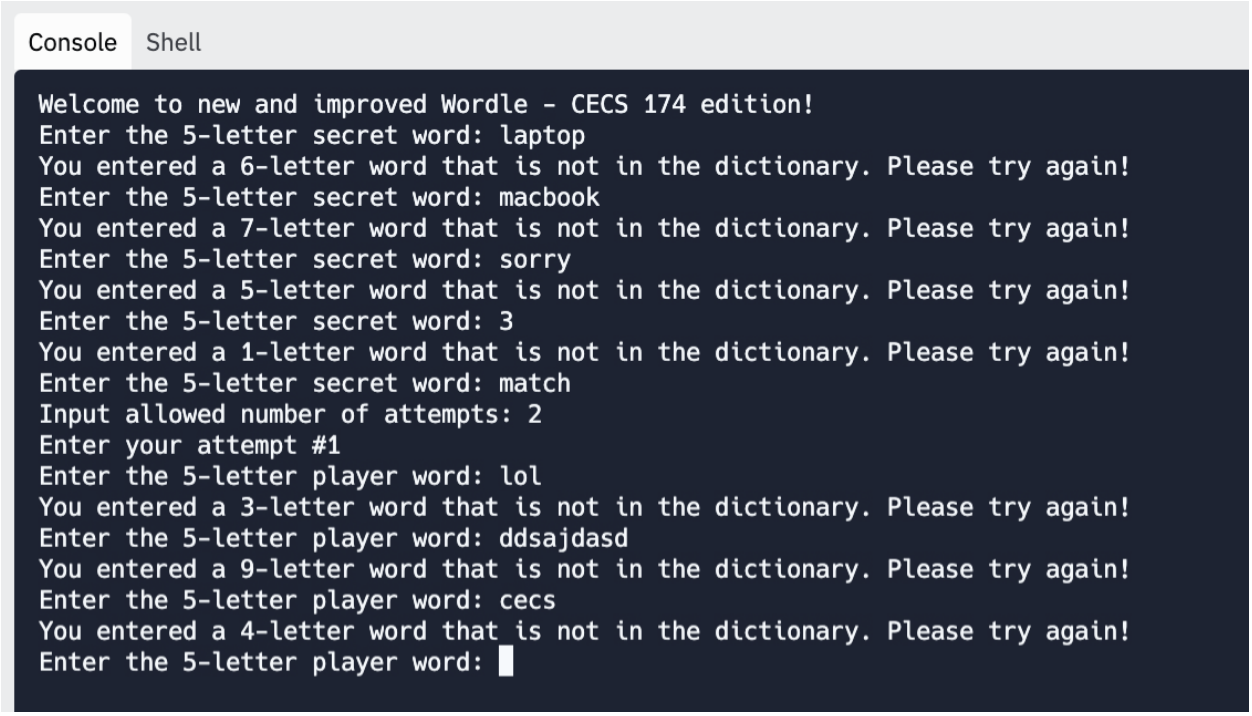
Project Wordle Improved Source code:



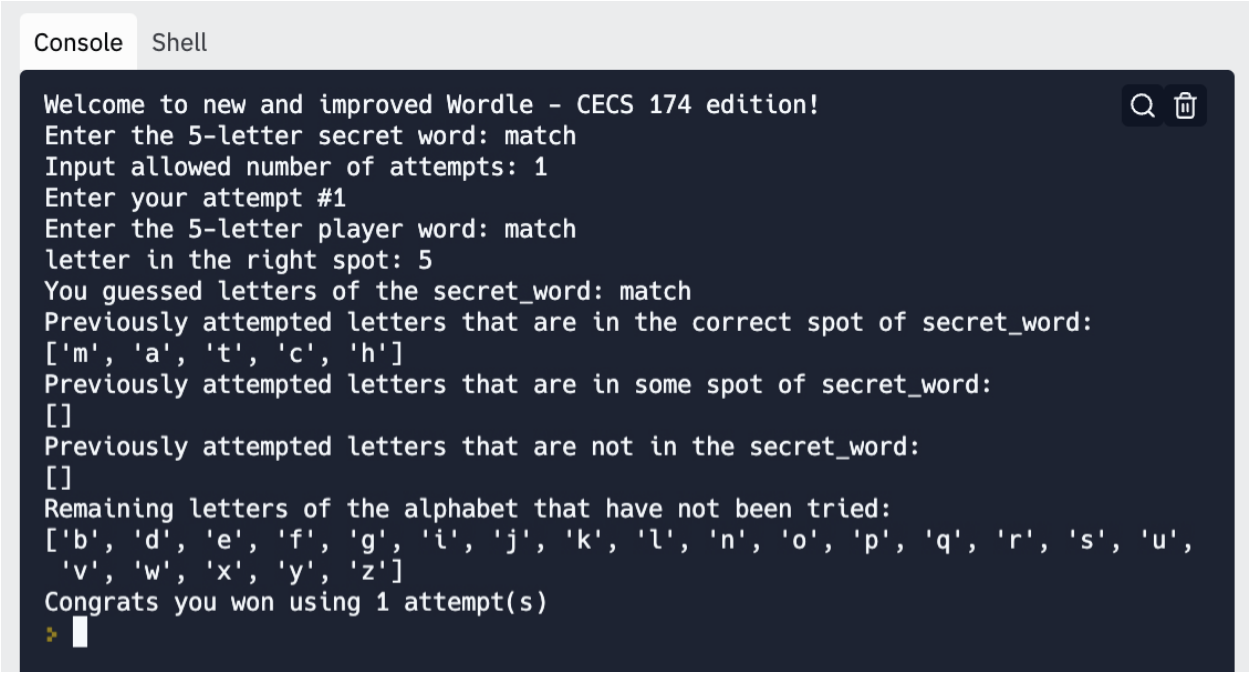
```
import string

WORD_LENGTH = 5

alphabet_string = string.ascii_lowercase
remaining_alphabet = list(alphabet_string)
in_secret_word_correct_spot = []
in_secret_word_somewhere = []
not_in_secret_word =[]

def read_dictionary (file_name): #opens word list
  file = open(file_name)
  new_dictionary_list = file.readlines()
  for i in range(len(new_dictionary_list)): #makes wordlist lowercase
    new_dictionary_list[i] = new_dictionary_list[i].replace('\n', '')
    new_dictionary_list[i] = new_dictionary_list[i].lower()
  file.close()
  return new_dictionary_list

def enter_a_word (word_type, num_letters): #enters player and secret word
    a_word = input(f'Enter the {num_letters}-letter {word_type} word: ')
    a_word = a_word.lower()
    return a_word

def is_it_a_word (input_word, dictionary_list): #checks if word is real
  if input_word not in dictionary_list:
      is_word = False
  else:
      is_word = True
  return is_word

def enter_and_check (word_type, dictionary_list): #checks if word is real/in list
  global WORD_LENGTH
  in_word = enter_a_word(word_type, WORD_LENGTH)
  is_real = is_it_a_word(in_word, word_list)
  length = len(in_word)
  while (length != WORD_LENGTH) or (is_real == False):
      is_real = is_it_a_word(in_word, word_list)
      length = len(in_word)
      if (length != WORD_LENGTH) and (is_real == False):
          print(f'You entered a {length}-letter word that is not in the dictionary. Please try again!')
          in_word = enter_a_word(word_type, WORD_LENGTH)
      elif (length != WORD_LENGTH) and (is_real == True):
          print(f'You entered a {length}-letter word that is in the dictionary. Please try again!')
          in_word = enter_a_word(word_type, WORD_LENGTH)
      elif (length == WORD_LENGTH) and (is_real == False):
          print(f'You entered a {length}-letter word that is not in the dictionary. Please try again!')
          in_word = enter_a_word(word_type, WORD_LENGTH)
  return in_word

def compare_words (player, secret): #compares player and secret word
  global remaining_alphabet
  global in_secret_word_correct_spot
  global in_secret_word_somewhere
  global not_in_secret_word

  letter_in_right_spot = 0
  lists = ['', '', '', '', '']
  player_list = list(player)

  for i in range(len(player)):
    for j in range(len(secret)):
      if (player[i] == secret[j]):
        if (i != j) and (player[i] not in in_secret_word_somewhere) and (player[i] not in in_secret_word_correct_spot): #Correct Letter and Wrong Spot
          lists[i] = f'({player[i]})'
          in_secret_word_somewhere.append(player[i])
        elif (i == j) or (player[i] in in_secret_word_correct_spot): #Correct Letter and Correct Spot
          if i == 0:
            lists[0] = player[0]
            if player[i] not in in_secret_word_correct_spot:
              in_secret_word_correct_spot.append(player[i])
            letter_in_right_spot += 1
            break
          elif i == 1:
            lists[1] = player[1]
            if player[i] not in in_secret_word_correct_spot:
              in_secret_word_correct_spot.append(player[i])
            letter_in_right_spot += 1
            break
          elif i == 2:
            lists[2] = player[2]
            if player[i] not in in_secret_word_correct_spot:
              in_secret_word_correct_spot.append(player[i])
            letter_in_right_spot += 1
            break
          elif i == 3:
            lists[3] = player[3]
            if player[i] not in in_secret_word_correct_spot:
              in_secret_word_correct_spot.append(player[i])
            letter_in_right_spot += 1
            break
          elif i == 4:
            lists[4] = player[4]
            if player[i] not in in_secret_word_correct_spot:
              in_secret_word_correct_spot.append(player[i])
            letter_in_right_spot += 1
            break
        else: #Not in word
            continue

  for i in range(len(player)): #adds letters to not secret word list
    for j in range(len(secret)):
      if (player[i] not in secret) and (player[i] not in not_in_secret_word):
        not_in_secret_word.append(player[i])
        break
```

```
  for i in range(len(remaining_alphabet)): #removes already used letters
    for j in range(len(player_list)):
      if (player[j] in remaining_alphabet):
        remaining_alphabet.remove(player[j])
        break

  for i in range(len(in_secret_word_correct_spot)): #removes l in double l words
    for j in range(len(in_secret_word_somewhere)):
      if (in_secret_word_somewhere[j] == 'l'):
        if in_secret_word_somewhere[j] in in_secret_word_correct_spot:
          in_secret_word_somewhere.remove(in_secret_word_somewhere[j])
          break
      else:
        break

  dup = [x for i, x in enumerate(in_secret_word_correct_spot) if i != in_secret_word_correct_spot.index(x)]
  dup2 = [x for i, x in enumerate(in_secret_word_somewhere) if i != in_secret_word_somewhere.index(x)]
  dup3 = [x for i, x in enumerate(not_in_secret_word) if i != not_in_secret_word.index(x)]

  for i in range(len(in_secret_word_correct_spot)): #removes dups in correct spot
      if in_secret_word_correct_spot[i] in dup:
        in_secret_word_correct_spot.remove(in_secret_word_correct_spot[i])
        break
  for i in range(len(in_secret_word_somewhere)): #removes dups in somewhere
      if in_secret_word_somewhere[i] in dup2:
        in_secret_word_somewhere.remove(in_secret_word_somewhere[i])
        break
  for i in range(len(not_in_secret_word)): #removes dups in not_in
      if not_in_secret_word[i] in dup3:
        not_in_secret_word.remove(not_in_secret_word[i])
        break

  for i in range(len(lists)): #places underscores in final
    if lists[i] == '':
      lists[i] = '_'


  final  = ''.join(lists) #makes final a string
  return final, letter_in_right_spot


word_list = read_dictionary('project4_dictionary.txt')

print('Welcome to new and improved Wordle - CECS 174 edition!')
secret_word = enter_and_check('secret', word_list) #enters secret word

N = int(input("Input allowed number of attempts: ")) #enters number of attempts
attempts = 1

while (attempts <= N): #outputs everything above
  print(f'Enter your attempt #{attempts}')
  player_word = enter_and_check('player', word_list) #enter secret_word
  final_word, letter_in_the_right_spot = compare_words(player_word, secret_word)
  print(f'letter in the right spot: {letter_in_the_right_spot}')
  print(f'You guessed letters of the secret_word: {final_word}')
  print('Previously attempted letters that are in the correct spot of secret_word:')
  print(in_secret_word_correct_spot)
  print('Previously attempted letters that are in some spot of secret_word:')
  print(in_secret_word_somewhere)
  print('Previously attempted letters that are not in the secret_word:')
  print(not_in_secret_word)
  print('Remaining letters of the alphabet that have not been tried:')
  print(remaining_alphabet)
  if secret_word == player_word:
    print(f'Congrats you won using {attempts} attempt(s)')
    break
  attempts += 1

if (N <= 0): #used all attempts
  player_word = 'No'
if secret_word != player_word:
  if (N <= 0):
    pass
  elif (N <= attempts):
    print(f'You already used #{attempts - 1} attempts. Better luck tomorrow!')

```

Project Wordle Improved Output for Bad Input:



Project Wordle Improved Output for Winning:



Project Wordle Improved Output for Losing: