Written Responses - Section 2

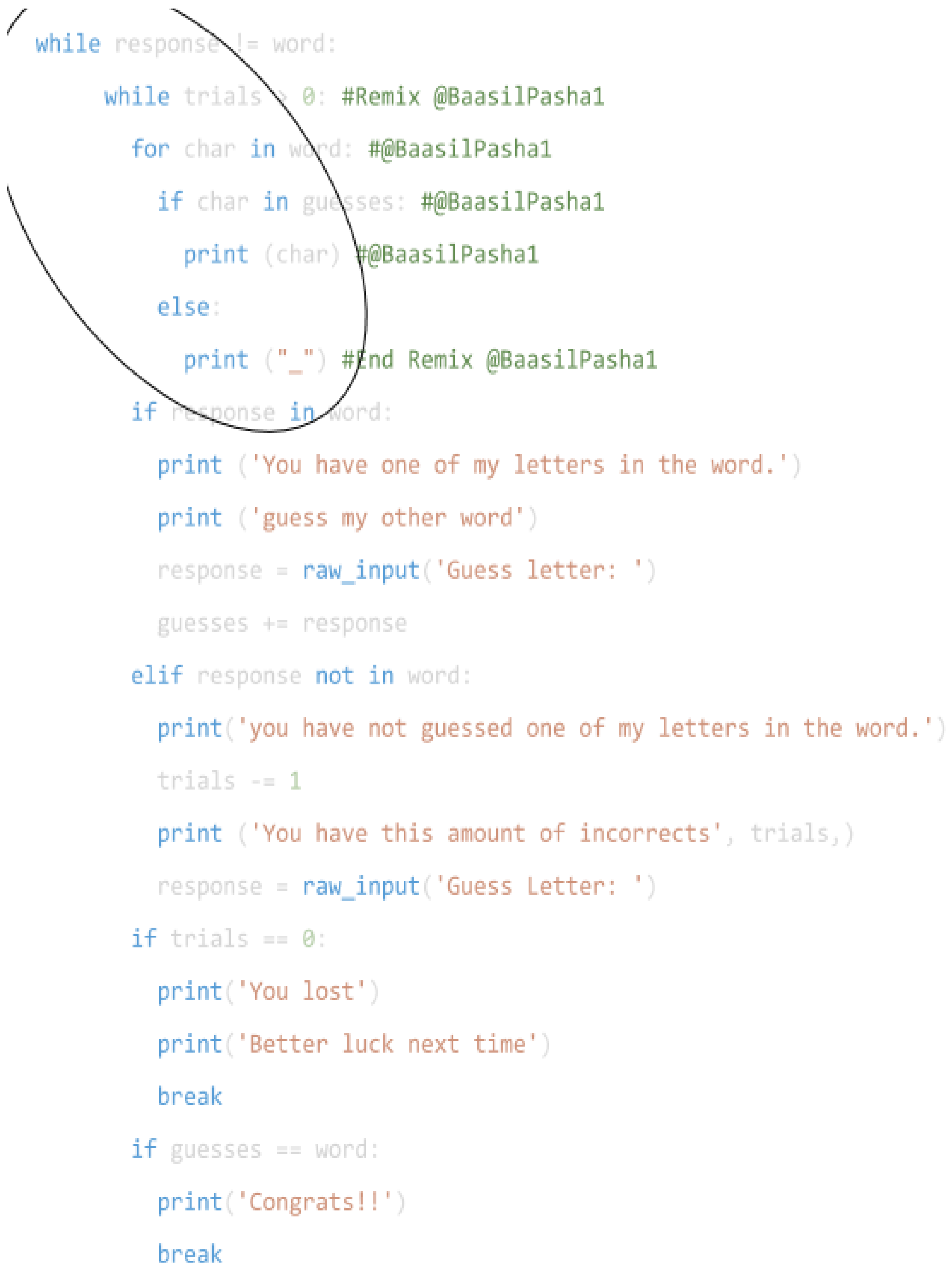
2a. Provide a written response or audio narration in your video that:

I use the program called REPL that uses the python program. I created a game using REPL and the game I made is hangman. The goal of this game is to correctly guess the secret word with little few incorrects guesses before losing the game if you used up all the guesses. The video shows several algorithms working and also shows how my hangman game work. The video also show what happens if you enter an incorrect guess or a correct guess and also shows the end result when you win the hangman game.

2b.

I start off typing the code for my program using the original plan I have of what my program will look like. First, I created an introduction in my game for when the user starts the hangman game. As I was typing the lines of code for the introduction, I was thinking of using a while loop for the core of my hangman game, the most important part of the game. Once I reached to this point I was able to make the while loop work but not in the way I wanted. Major challenges I faced were figuring out how to keep the while loop true and when would the while loop stop once the secret word is guessed. I used someone's lines of code, @BaasilPasha1, since I figure out how to use it. Also I ask Mr. Orkney to help me find where my code does not work and to give me suggestions in making it work and also I came up with some ideas to make it work like a word bank. I put my ideas down in codes by doing a nested while loop and for loop I finally made my hangman game to work properly.

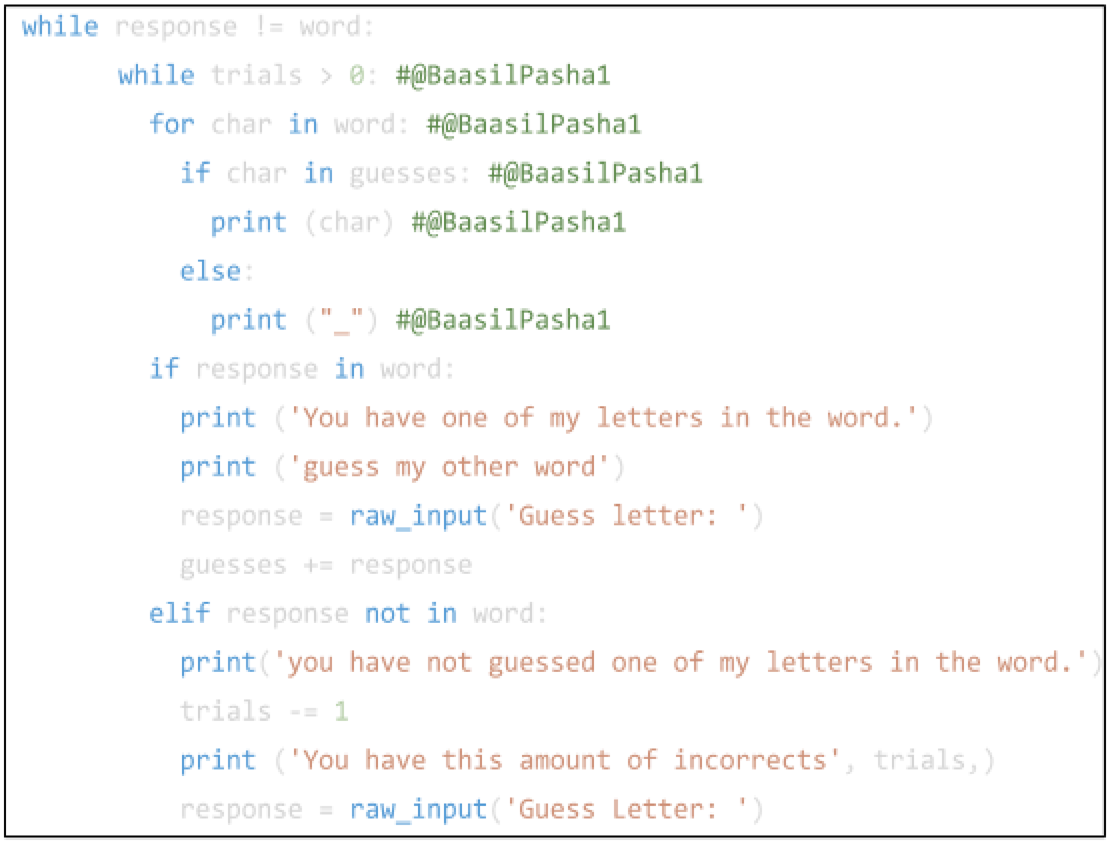
2c.



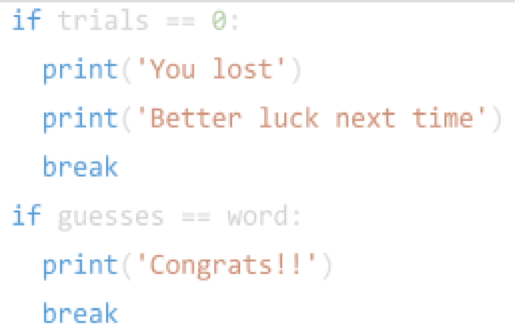
```
while response != word:
    while trials > 0: #Remix @BaasilPasha1
      for char in word: #@BaasilPasha1
        if char in guesses: #@BaasilPasha1
          print (char) #@BaasilPasha1
        else:
          print ("_") #End Remix @BaasilPasha1
      if response in word:
        print ('You have one of my letters in the word.')
        print ('guess my other word')
        response = raw_input('Guess letter: ')
        guesses += response
      elif response not in word:
        print('you have not guessed one of my letters in the word.')
        trials -= 1
        print ('You have this amount of incorrects', trials,)
        response = raw_input('Guess Letter: ')
      if trials == 0:
        print('You lost')
        print('Better luck next time')
        break
      if guesses == word:
        print('Congrats!!')
        break
```

This section of program code from my game implements several algorithms that are very essential in my program. The first while loop starts the entire block of codes underneath it. The main while loop is use when the response value does not occur in the variable word, which contain the secret word game racecar. The main while loop would also start the second while loop making it a nested while loop. This while loop is also very special because the while loop makes my codes beneath the while loop to keep running while it is true. The while loop is from a user name @BaasilPasha1 and I use his code as an essential part in my game. The next few lines of codes are also from the user. The for loop creates a temporary inside the variable word. After the temporary variable, char, is created the if statement then is executed. The temporary variable char is scan in the variable guesses and the if statement says that if char is true in the variable guesses then it prints the specific value of char or if false then it would print an underline to represent the missing value of char.

2d.



```
while response != word:
      while trials > 0: #@BaasilPasha1
        for char in word: #@BaasilPasha1
          if char in guesses: #@BaasilPasha1
            print (char) #@BaasilPasha1
          else:
            print ("_") #@BaasilPasha1
        if response in word:
          print ('You have one of my letters in the word.')
          print ('guess my other word')
          response = raw_input('Guess letter: ')
          guesses += response
        elif response not in word:
          print('you have not guessed one of my letters in the word.')
          trials -= 1
          print ('You have this amount of incorrects', trials,)
          response = raw_input('Guess Letter: ')
```



```
        if trials == 0:
          print('You lost')
          print('Better luck next time')
          break
        if guesses == word:
          print('Congrats!!')
          break
```

This section of program code from my game shows the complexity of my behind the scenes program codes of the game. I use a logical concept while loops that makes the codes beneath it to run without stopping when the statement is true. Doing nested while loops helps build up the complexity of my program because it will repeat until the user correctly guesses the secret word. A mathematical concept that I apply is the while loop that keeps the program running when the trials is greater than 0. This helps because it prevents stopping the program and I also have other mathematical concepts in the if/elif statements. For example when the trials is equal to 0 the program will then stop. The way I use logical and mathematical concepts helps build up my complexity of my program because if one line of code like a for loop does not work the whole entire program would not work and also there are different logical statements that keeps the program running until the secret word is guessed.