```
@@ -0,0 +1,15 @@
+sentence = input("Enter sentence: \n>")
+sentence = sentence.lower()
+sentencelist = sentence.split()
+search = input("Word to find:\n>")
+search = search.lower()
+position = [i for i, found in enumerate(sentencelist) if found == search]
+if len(position) >1: plural = "s"
+else:
+    plural = ""
+if any(n in position for n in range(0, 9999999)):
+    print("The word,", search, ", was found at position"+ plural, position, ".")
+else:
+    print("The word", search, "wasn't found.")
+input()
+
```

Developing task 1 was simply following the criteria given by the task sheet. I had to make sure I did specific things like case sensitivity etc. I started with the user inputting their sentence, shortly followed by turning the sentence into lowercase to prevent case sensitivity. After that, I turned the sentence into a list using the .split() function. The next thing to do was to tell the user to input the word they wanted to find. Again, once they input that word I made sure it turns to lowercase to match the sentence. Finding the position of the word was a tricky task. Well, actually it was quite simple for one word. It was when it came to finding the position of the same word twice. I started with the simple index function. “position = sentencelist.index(search)” This worked if the word only appeared once. But if it appeared twice, it only showed the position of the first time it appears. After doing some research, it seemed there was a way around this, the enumerate function. Basically, I managed to make a massive for loop which used the enumerate function so that it followed the for loop with each and every item in the list, getting all the indexes for the one word and storing it in it’s own list. After that you can see I added a plural thing which is actually pretty neat. All it does is add an s to the end of the word “position” if the word that the user searched for appeared more than once. It did this by seeing if there was more than 1 item stored in the position list where the indexes were stored. Ok so this next bit was poorly done by me, don’t worry I correct it later, I basically wanted to say if there was anything in the positions list, output that the word was found at the position. At the time I wasn’t aware you could just say “if not position”. Finally the last bit was just outputting the results using simple strings and adding bits to the strings.

```
@@ -1,4 +1,10 @@
-sentence = input("Enter sentence: \n>")  #user enters sentence which is stored in the variable as sentence.
+empty = True
+while empty is True: #just a while loop to make sure the user actually enters somthing as entering nothing could cause errors..
+    sentence = input("Enter sentence: \n>")#user enters sentence which is stored in the variable as sentence.
+    if sentence == "":
+        print("You didn't enter anything...")
+    if sentence !="":
+        empty = False #ends while loop
 sentence = sentence.lower() #turns the sentence into all lower case to prevent case senstivity.
 sentencelist = sentence.split() #splits the sentence into individual words and stores it in this variable as a list.
 search = input("Word to find:\n>") #user enters word they wish to find which is also stored as a variable.
```

So next I thought about what would happen if the user entered nothing. And after going through a test plan, I believe it would cause a few logical errors. This is why I added a while loop with a condition. The condition was if the sentence that the user inputted literally equalled nothing, it would output “you didn’t enter anything…” and loop back to the input. Then I added another condition where if the sentence didn’t equal nothing, the while loop would end and the program

would proceed. However, although this was a good idea, I didn't do it very efficiently.

```
...  ...  @@ -1,12 +1,15 @@
     1    +import string #acts like a dictionary so later I can use short phrases which can be lookedup in this
1    2     empty = True
2    3     while empty is True: #just a while loop to make sure the user actually enters somthing as entering nothing could cause errors..
3         -    sentence = input("Enter sentence: \n>")#user enters sentence which is stored in the variable as sentence.
4         -    if sentence == "":
5         -        print("You didn't enter anything...")
6         -    if sentence !="":
7         -        empty = False #ends while loop
     4    +    sentence = input("Enter sentence (without punctuation): \n>")#user enters sentence which is stored in the variable as sentence.
     5    +    if not sentence: #if the sentence is left blank (nothing was entered)
     6    +        print("You didn't enter anything...") #output that nothing was entered
     7    +    elif any(punctuation in sentence for punctuation in string.punctuation): #or if any item in the string.punctuation list is in t
     8    +        print("Punctuation found!")
     9    +    else: #if the sentence passes all of these conditions then
     10   +        empty = False #end the while loop and procede
8    11    sentence = sentence.lower() #turns the sentence into all lower case to prevent case senstivity.
9         -sentencelist = sentence.split()#splits the sentence into individual words and stores it in this variable as a list.
     12   +sentencelist = sentence.split()#splits the sentence into individual words and stores it in this variable as a list.
10   13    search = input("Word to find:\n>") #user enters word they wish to find which is also stored as a variable.
11   14    search = search.lower() #again, the word the user wishes to search is turned into lower case to match the sentence, avoiding case s
12   15    position = [i for i, found in enumerate(sentencelist) if found == search] #this loop goes through each induvidual word in the sente
           @@ -18,4 +21,3 @@
18   21    else: #self explanitory, if the word was found, output the success, and it's positions.
19   22        print("The word,", search, ", was found at position"+ plural, position, ".")
20   23    input() #creates a break so the user can press enter when they wish to close the program.
21        -
```

This is what led me to make these changes. Here, I changed the way I set the conditions. Instead of if sentence == "" I used if not sentence. I also added a punctuation check since it was a part of the criteria. To check for punctuation, I used a for loop which goes through each word in the sentence and checks if they match any punctuation from the string.punctuation list which was imported. If any of them do, it outputs "Punctuation found!" and loops back to the input using the while loop. But if both these conditions are met, the while loop ends, allowing the program to proceed.

```
1  import string #acts like a dictionary so later I can use short phrases which can be lookedup in this
2  empty = True
3  while empty is True: #just a while loop to make sure the user actually enters somthing as entering nothing could cause errors..
4      sentence = input("Enter sentence (without punctuation): \n>")#user enters sentence which is stored in the variable as sentence.
5      if not sentence: #if the sentence is left blank (nothing was entered)
6          print("You didn't enter anything...") #output that nothing was entered
7      elif any(punctuation in sentence for punctuation in string.punctuation): #or if any item in the string.punctuation list is in the sent
8          print("Punctuation found!")
9      else: #if the sentence passes all of these conditions then
10         empty = False #end the while loop and procede
11 sentence = sentence.lower() #turns the sentence into all lower case to prevent case senstivity.
12 sentencelist = sentence.split()#splits the sentence into individual words and stores it in this variable as a list.
13 search = input("Word to find:\n>") #user enters word they wish to find which is also stored as a variable.
14 search = search.lower() #again, the word the user wishes to search is turned into lower case to match the sentence, avoiding case senstivi
15 position = [i for i, found in enumerate(sentencelist) if found == search] #this loop goes through each induvidual word in the sentence lis
16 if len(position) >1: plural = "s" #used to help the following output make more sense. basically, if the word has been found more than one
17 else:
18     plural = "" #sets the variable plural as nothing so it knows not to add an s on the end of positions.
19 if not position: #basically, if no positions are found, then output it no word found.
20         print("The word", search, "wasn't found.")
21 else: #self explanitory, if the word was found, output the success, and it's positions.
22     print("The word,", search, ", was found at position"+ plural, position, ".")
23 input() #creates a break so the user can press enter when they wish to close the program.
```

This is the final result for the task 1 python code.