M2 - W4 Assignment: Advanced OOP strategies

In this assignment, you are requested to complete a series of tasks using the best OOP strategies as we defined them in the last workshop. *While there is no universally correct solution,*your solution should be *maintenanable*and *easy-to-read-and-document.*

You are requested to create a class named *UserData.*This class will contain some sentences imported by the user.

* It  must be created in two possible ways:
  + Via a list containing all the sentences
  + Via a txt file that contains one sentence per line
* It should return the number of sentences as a property *(Bonus point if you do that outside the \_\_init\_\_ using the property decorator).*
* Create a helper function that returns a dictionary with the frequency of each unique word in just one sentence. (Tip: Make this function invisible outside the class).
* Use the above helper function to create another function that returns the frequency of each unique word in the whole corpus (Tip: Corpus is the collection of all the sentences).

Please upload **a python file (.py)**for this assignment.

“Write Python codes for the following problem:

Create a class named *UserData.*This class will contain some sentences imported by the user.

* It  must be created in two possible ways:
  + Via a list containing all the sentences
  + Via a txt file that contains one sentence per line
* It should return the number of sentences as a property *(Bonus point if you do that outside the \_\_init\_\_ using the property decorator).*
* Create a helper function that returns a dictionary with the frequency of each unique word in just one sentence. (Tip: Make this function invisible outside the class).
* Use the above helper function to create another function that returns the frequency of each unique word in the whole corpus (Tip: Corpus is the collection of all the sentences).”

class UserData:

def \_\_init\_\_(self, data):

if isinstance(data, list):

self.\_sentences = data

elif isinstance(data, str) and data.endswith(".txt"):

with open(data, "r") as f:

self.\_sentences = f.readlines()

else:

raise TypeError("Data must be a list of sentences or a path to a txt file.")

@property

def num\_sentences(self):

return len(self.\_sentences)

def \_word\_frequency(self, sentence):

"""

Helper function to return the frequency of each unique word in a sentence.

"""

words = sentence.lower().split()

return {word: words.count(word) for word in words}

def corpus\_word\_frequency(self):

"""

Returns the frequency of each unique word in the whole corpus.

"""

word\_counts = {}

for sentence in self.\_sentences:

for word, count in self.\_word\_frequency(sentence).items():

word\_counts[word] = word\_counts.get(word, 0) + count

return word\_counts

# Example usage

data = ["This is a sentence.", "This is another sentence."]

user\_data = UserData(data)

print(f"Number of sentences: {user\_data.num\_sentences}")

print(f"Corpus word frequency: {user\_data.corpus\_word\_frequency()}")

Use code with caution. [Learn more](https://bard.google.com/faq#coding)

content\_copy

This code defines a class named UserData that can take a list of sentences or a path to a text file as input. The class provides a property called num\_sentences that returns the number of sentences in the corpus. It also defines two helper functions:

* \_word\_frequency(sentence): This function takes a sentence as input and returns a dictionary that maps each unique word in the sentence to its frequency.
* corpus\_word\_frequency(): This function iterates over all the sentences in the corpus and uses the \_word\_frequency function to build a dictionary that maps each unique word in the corpus to its frequency.

The example usage shows how to create a UserData object from a list of sentences and then access the num\_sentences and corpus\_word\_frequency properties.

I hope this helps! Let me know if you have any other questions.