Application on building an Interactive English Dictionary using Python programming language

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

What is Interactive Dictionary

It is a console application, made by using python programming language. It takes the word from the user and returns its definition from the data.json file. It also gives best-matched word when the user enters the word with the wrong spelling.

JSON file

A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format. It is primarily used for transmitting data between a web application and a server. ...

Steps :

- Load JSON file into python.
- Ask the user for a Word.
- Pass that word into a word_meaning() function where the code will be looking for the meaning of input word into the words.json file.
- word_meaning() function will not just look for the meaning into words.json file but also analyze the word to check if a user somehow mistypes the word and meant something else, which will be making our dictionary interactive.
    - *In the below code I have used **get_close_matches**() function of **difflib** module inside word_meaning() function to analyse the word and making our application interactive, **difflib** module provides classes and functions for comparing sequences.*
    - ***get_close_matches**: Return a list of the best 'good enough' matches. Word is a sequence for which close matches are desired (typically a string), and possibilities is a list of sequences against which to match word (typically a list of strings).*
-

```
import json

#load JSON data
data = json.load(open("data.json"))

#take word from user
word = input('Enter word: ')

#function to return meaning of the word from data
def getMeaning(w):
    return data[w]

#function call to get meaning of the word entered by user
meaning = getMeaning(word)

#printing meaning of the word in console
print("Meaning :", meaning)
```

Case:

By implementing this, we implemented logic to give an appropriate message if the word doesn't exist in data. But now, the issue is if we will enter "rain", it will give definition but what if we will write "RAIN", "Rain", "raIN"?

2 Now, what if the user will enter a word with wrong spelling? Here, I am talking about giving a suggestion like if by mistake user enters "rainn" instead of "rain", we have to give an appropriate message for a suggestion.

```
def translate(w):
    w = w.lower()
    if w in data:
        return data[w]
    elif len(get_close_matches(w, data.keys())) > 0:
        yn = input("Did you mean %s instead? Enter Y if yes, or N if no: " % get_close_matches(w, data.keys())[0])
        if yn == "Y":
            return data[get_close_matches(w, data.keys())[0]]
        elif yn == "N":
            return "The word doesn't exist. Please double check it."
        else:
            return "We didn't understand your entry."
    else:
        return "The word doesn't exist. Please double check it."
```

When we display definition or meaning as an output, it shows square bracket and commas. We don't want to show that as an output **to make it more user friendly**. So, just use for loop and print each element of the list like shown below:

```
output = translate(word)
if type(output) == list:
    for item in output:
        print(item)
else:
    print(output)
```

Snapshots of output

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: praise
the act of expressing approval or admiration; commendation; laudation.
To extol in speech or writing.
```

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: PRAISE
the act of expressing approval or admiration; commendation; laudation.
To extol in speech or writing.
```

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: PraISe
the act of expressing approval or admiration; commendation; laudation.
To extol in speech or writing.
```

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: praisse
Did you mean praise instead? Enter Y if yes, or N if no: Y
the act of expressing approval or admiration; commendation; laudation.
To extol in speech or writing.
```

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: PRAISSE
Did you mean praise instead? Enter Y if yes, or N if no: N
The word doesn't exist. Please double check it.
```

```
(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>python app1.py
Enter word: PRAISSE
Did you mean praise instead? Enter Y if yes, or N if no: SDHJF
We didn't understand your entry.

(venv) C:\Users\Blessy_joy\PycharmProjects\interactive dictionry.py>
```