Python program for word suggestion

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route("/", methods=["POST", "GET"])

def home():

    if request.method == "GET":

        languages = ["C++", "Python", "PHP", "Java", "C", "Ruby",

                    "R", "C#", "Dart", "Fortran", "Pascal", "Javascript"]

        return render_template("index.html", languages=languages)

if __name__ == '__main__':

    app.run(debug=True)
```

html file

```
<!DOCTYPE html>

<html>

<head>

        <title>AutoComplete</title>

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.js">

        </script>

        <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.16/jquery-ui.js">

        </script>

        <link href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.16/themes/ui-lightness/jquery-ui.css"

        rel="stylesheet" type="text/css" />

</head>

<body>

        <h1>Welcome to  SJCET</h1>

        <input type="text" id="tags">
```

```
        <script>
$( function() {
        var availableTags = [
                {% for language in languages %}
                        "{{language}}",
                {% endfor %}
        ];
        $( "#tags" ).autocomplete({
        source: availableTags
        });
} );
</script>
</body>
</html>
```

Output

To run this app open cmd or terminal and run the below command.

python app.py

AUTOCOMPLETE PROGRAM

```
# Python3 program to demonstrate auto-complete
# feature using Trie data structure.
# Note: This is a basic implementation of Trie
# and not the most optimized one.


class TrieNode():
  def __init__(self):
    # Initialising one node for trie
    self.children = {}
    self.last = False
```

```
class Trie():
  def __init__(self):

    # Initialising the trie structure.
    self.root = TrieNode()

  def formTrie(self, keys):

    # Forms a trie structure with the given set of strings
    # if it does not exists already else it merges the key
    # into it by extending the structure as required
    for key in keys:
      self.insert(key) # inserting one key to the trie.

  def insert(self, key):

    # Inserts a key into trie if it does not exist already.
    # And if the key is a prefix of the trie node, just
    # marks it as leaf node.
    node = self.root

    for a in key:
      if not node.children.get(a):
        node.children[a] = TrieNode()

      node = node.children[a]

    node.last = True

  def suggestionsRec(self, node, word):

    # Method to recursively traverse the trie
    # and return a whole word.
    if node.last:
      print(word)

    for a, n in node.children.items():
      self.suggestionsRec(n, word + a)

  def printAutoSuggestions(self, key):

    # Returns all the words in the trie whose common
    # prefix is the given key thus listing out all
    # the suggestions for autocomplete.
    node = self.root
```

```
        for a in key:
            # no string in the Trie has this prefix
            if not node.children.get(a):
                return 0
            node = node.children[a]

        # If prefix is present as a word, but
        # there is no subtree below the last
        # matching node.
        if not node.children:
            return -1

        self.suggestionsRec(node, key)
        return 1


# Driver Code
keys = ["hello", "dog", "hell", "cat", "a",
        "hel", "help", "helps", "helping"] # keys to form the trie structure.
key = "h" # key for autocomplete suggestions.

# creating trie object
t = Trie()

# creating the trie structure with the
# given set of strings.
t.formTrie(keys)

# autocompleting the given key using
# our trie structure.
comp = t.printAutoSuggestions(key)

if comp == -1:
    print("No other strings found with this prefix\n")
elif comp == 0:
    print("No string found with this prefix\n")
```

OUTPUT

```
hel
hell
hello
help
helps
helping
```

Get similar words suggestion using Enchant in Python

For the given user input, get similar words through Enchant module.

Enchant is a module in python which is used to check the spelling of a word, gives suggestions to correct words. Also, gives antonym and synonym of words. It checks whether a word exists in dictionary or not. Other dictionaries can also be added, as, (“en_UK”), (“en_CA”), (“en_GB”) etc.

To install enchant :

pip install pyenchant

EXAMPLES

Input : Helo

Output : Hello, Help, Hero, Helot, Hole

Input : Trth

Output : Truth, Trash, Troth, Trench

CODE

```
# Python program to print the similar
# words using Enchant module
# Importing the Enchant module
import enchant
# Using 'en_US' dictionary
d = enchant.Dict("en_US")
# Taking input from user
word = input("Enter word: ")
d.check(word)
# Will suggest similar words
# form given dictionary
print(d.suggest(word))
```

OUTPUT

Enter word: **aple**

['pale', 'ale', 'ape', 'maple', 'ample', 'apple', 'plea', 'able', 'apse']

Python – Spelling checker using Enchant

Enchant is a module in Python, which is used to check the spelling of a word, gives suggestions to correct words. Also, gives antonym and synonym of words. It checks whether a word exists in the dictionary or not.

Enchant can also be used to check the spelling of words. The check() method returns True if the passed word is present in the language dictionary, else it returns False. This functionality of the check() method can be used to spell check words.

The suggest() method is used to suggest the correct spelling of the incorrectly spelled word.

```
# import the enchant module

import enchant

# create dictionary for the language

# in use(en_US here)

dict = enchant.Dict("en_US")

# list of words

words = ["cmputr", "watr", "study", "wrte"]

# find those words that may be misspelled

misspelled =[]

for word in words:

        if dict.check(word) == False:

                misspelled.append(word)

print("The misspelled words are : " + str(misspelled))
```

```
# suggest the correct spelling of

# the misspelled words

for word in misspelled:

        print("Suggestion for " + word + " : " + str(dict.suggest(word)))
```

OUTPUT

```
The misspelled words are : ['cmputr', 'watr', 'wrte']

Suggestion for cmputr : ['computer']

Suggestion for watr : ['wart', 'watt', 'wat', 'war', 'water', 'watr']

Suggestion for wrte : ['rte', 'write', 'wrote', 'wert', 'wite', 'w rte']
```