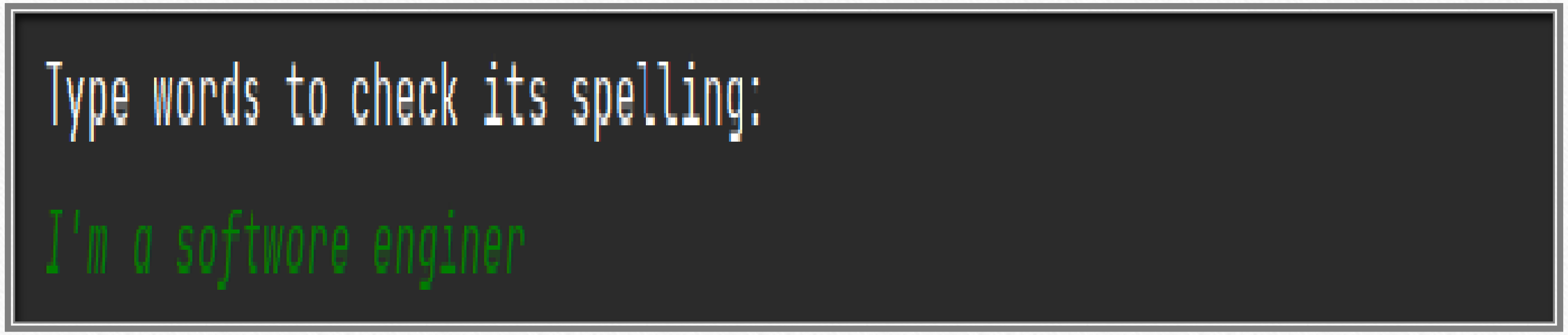
Spell Checker

By: Team 4

EXAMPLE How it work

- *Input*

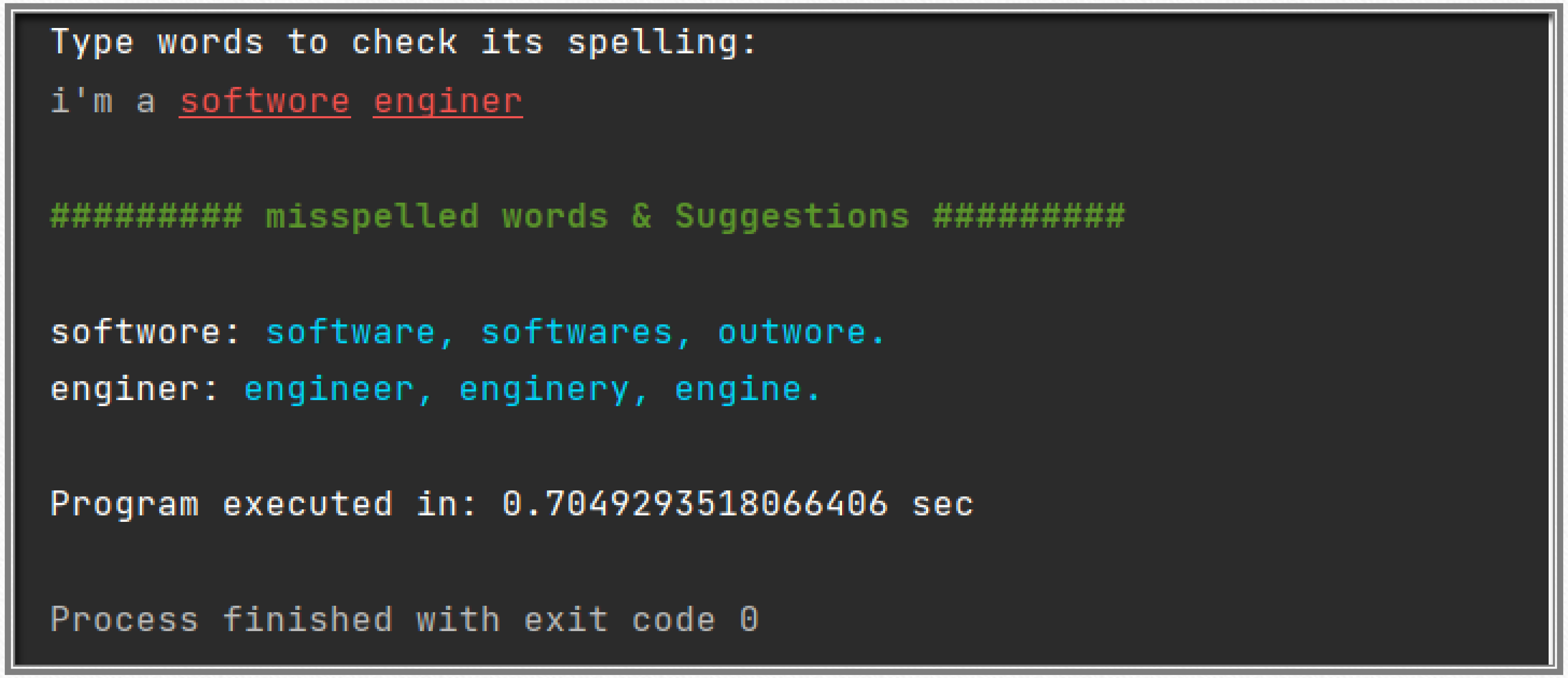


```
Type words to check its spelling:

I'm a softwore enginer
```

EXAMPLE

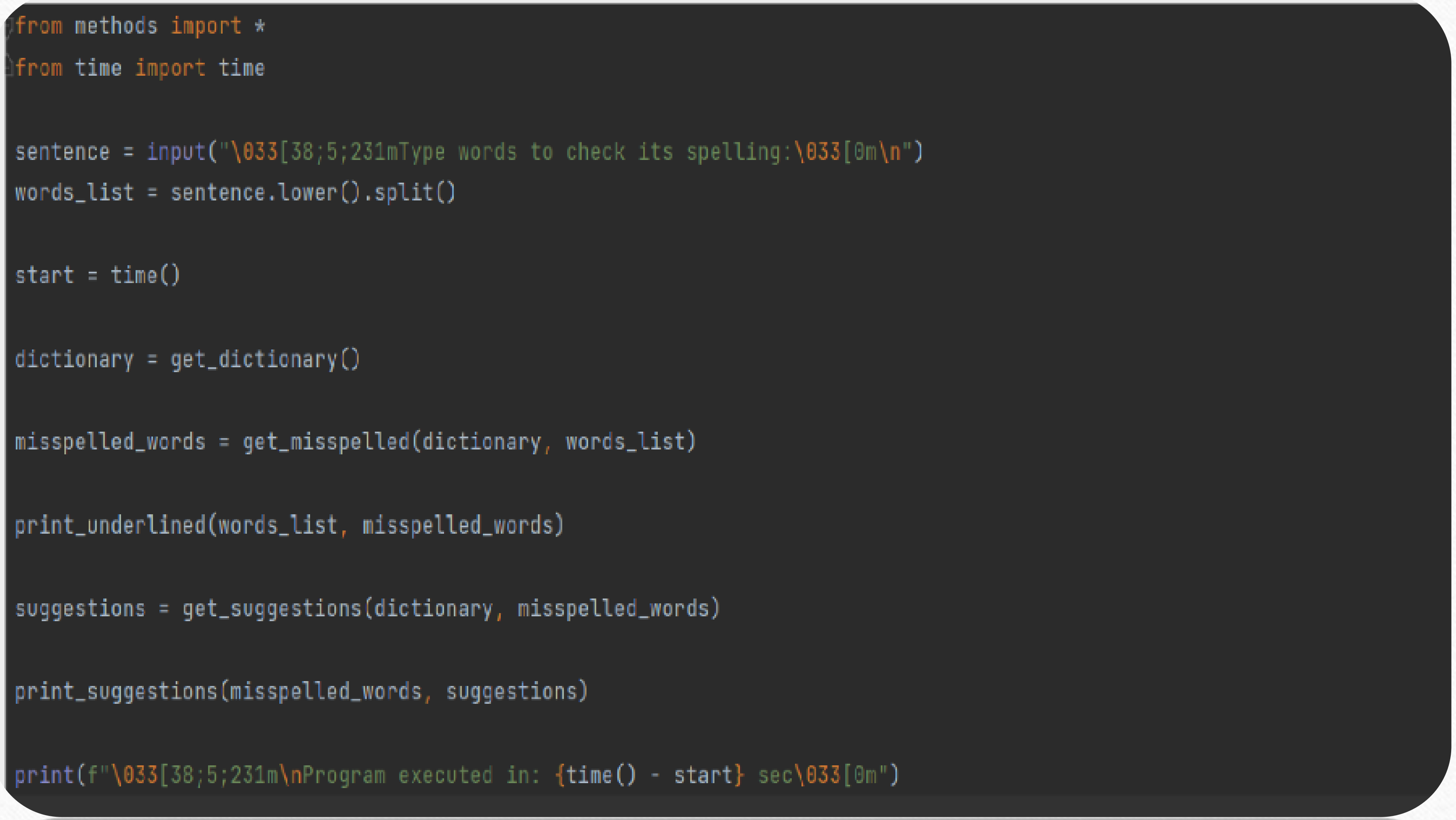
How it work

- *Output*



```
Type words to check its spelling:
i'm a softwore enginer

######### misspelled words & Suggestions #########

softwore: software, softwares, outwore.
enginer: engineer, enginery, engine.

Program executed in: 0.7049293518066406 sec

Process finished with exit code 0
```

Main Code



```
from methods import *
from time import time

sentence = input("\033[38;5;231mType words to check its spelling:\033[0m\n")
words_list = sentence.lower().split()

start = time()

dictionary = get_dictionary()

misspelled_words = get_misspelled(dictionary, words_list)

print_underlined(words_list, misspelled_words)

suggestions = get_suggestions(dictionary, misspelled_words)

print_suggestions(misspelled_words, suggestions)

print(f"\033[38;5;231m\nProgram executed in: {time() - start} sec\033[0m")
```

Import the user words

```
sentence = input("\033[38;5;231mType words to check its spelling:\033[0m\n")

words_list = sentence.lower().split()
```

So

What is the "`sentence.lower().split()`"?

I'm Engineer → i'm engineer → ["i'm" , "engineer"]

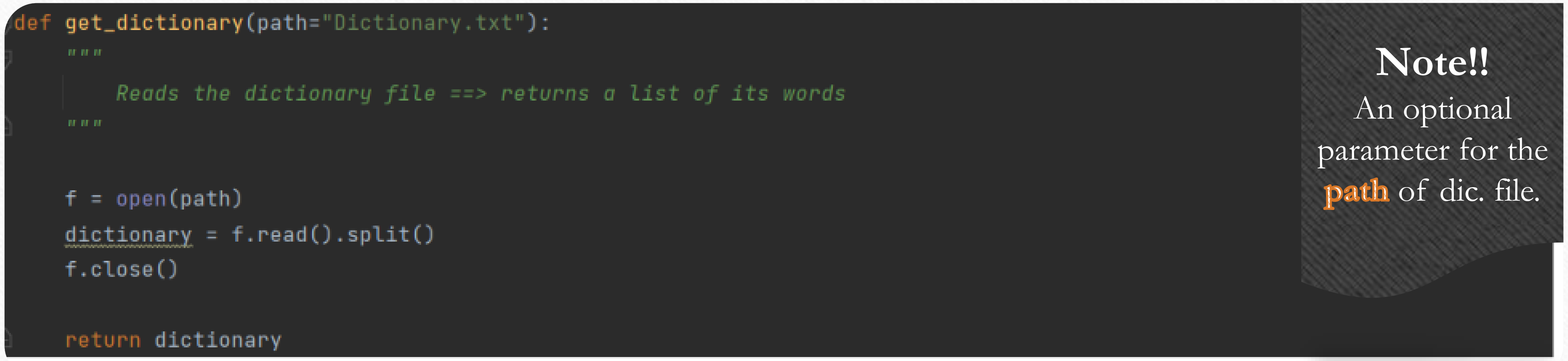
I'm Engineer	i'm engineer	["i'm" , "engineer"]
Hard to control !	Hard to control !	Easy to control !
Wrong word !	Right word!	Right word!

Main Ideas

Main Ideas

1. Import the dictionary.
2. Find the wrong words.
3. Find suggestions.
4. Print results.

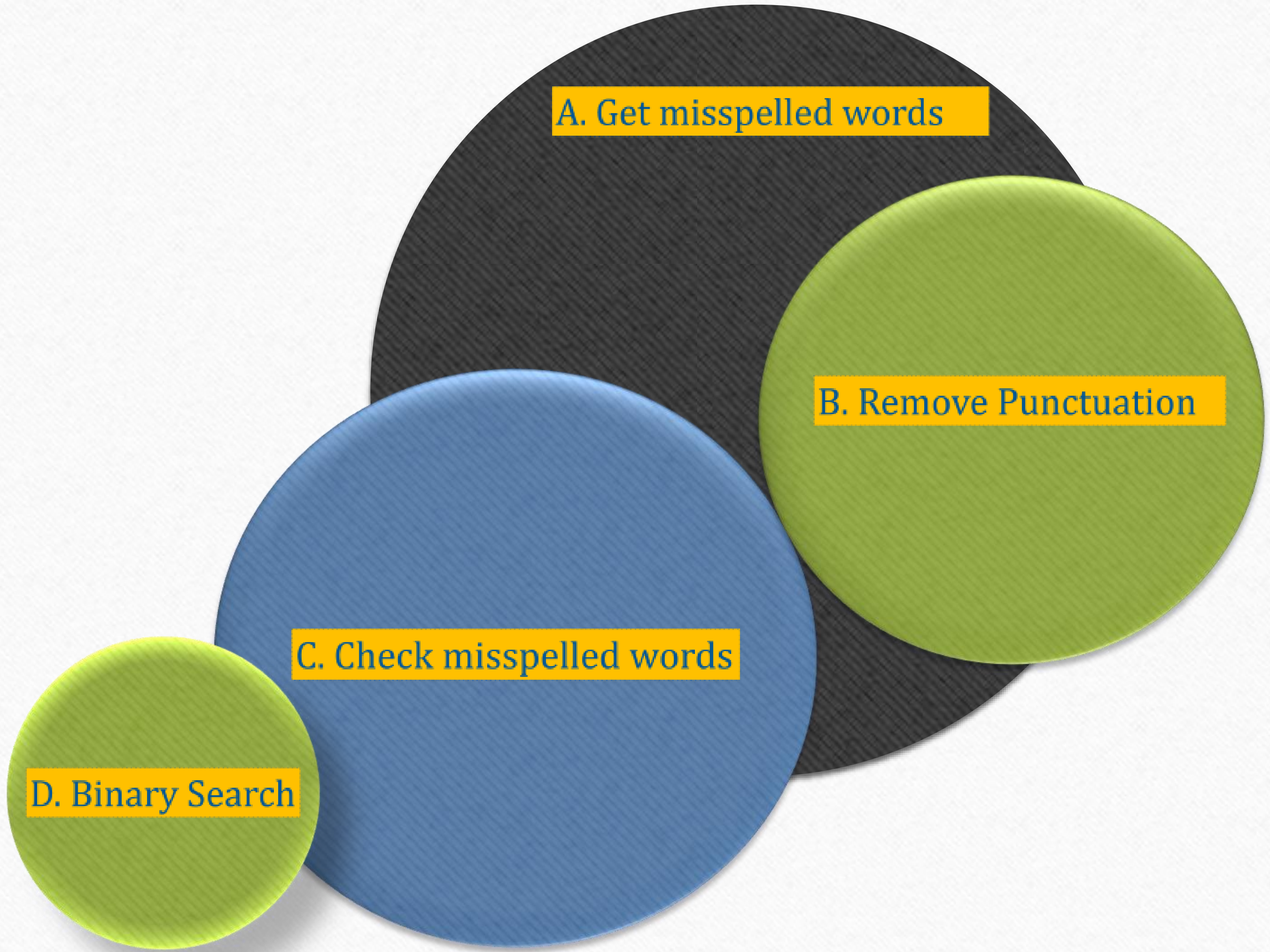
1. Import the dictionary.



```
def get_dictionary(path="Dictionary.txt"):
    """
        Reads the dictionary file ==> returns a list of its words
    """

    f = open(path)
    dictionary = f.read().split()
    f.close()

    return dictionary
```

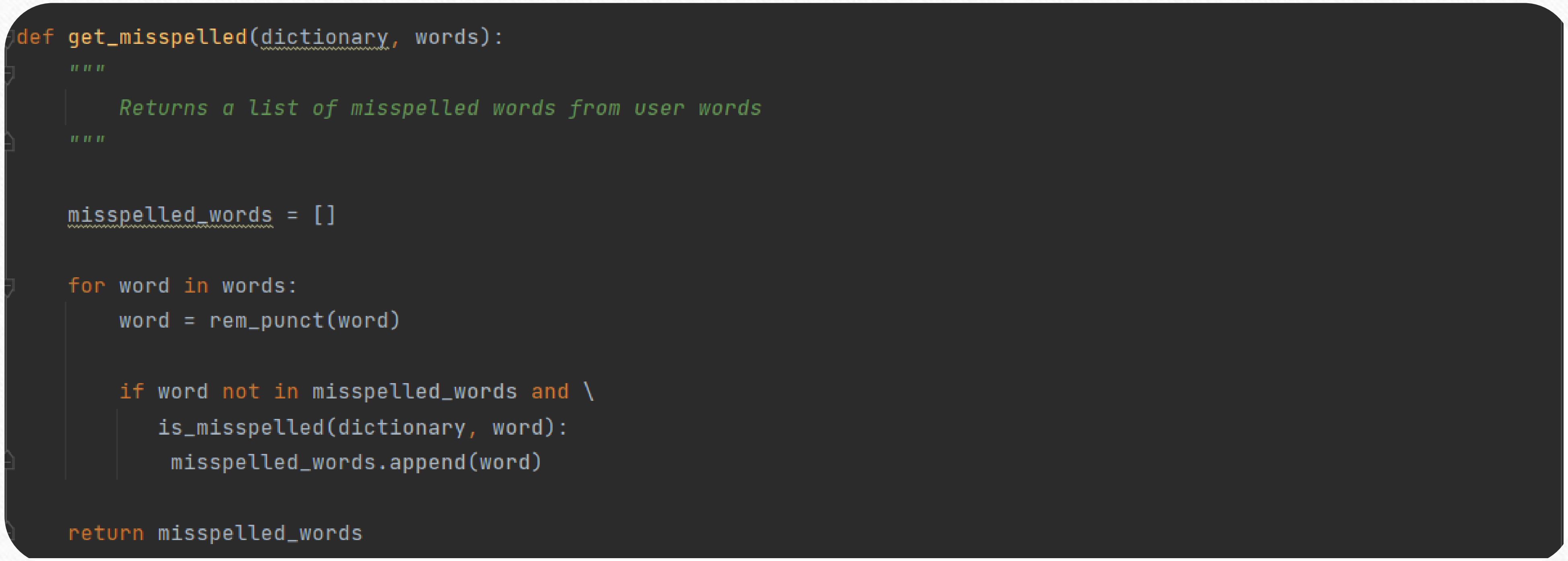
Note!!
An optional parameter for the **path** of dic. file.

❌ 2. Find the wrong words



A. Get misspelled words

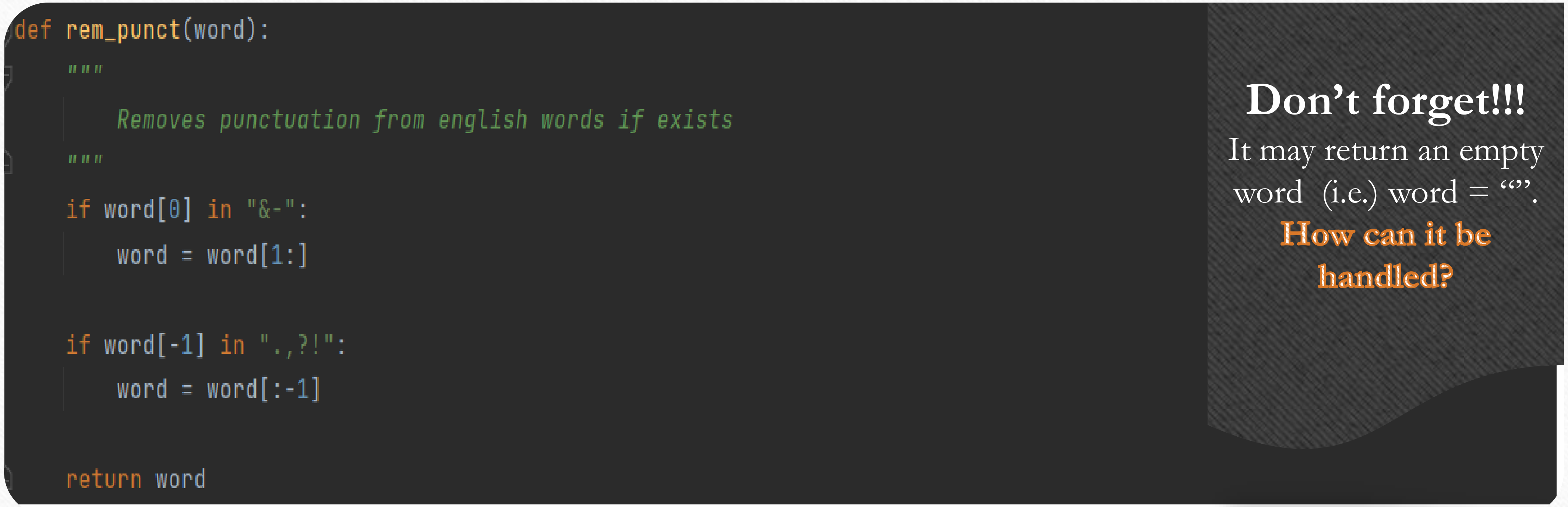
A. Get misspelled words



```python
def get_misspelled(dictionary, words):
    """
        Returns a list of misspelled words from user words
    """

    misspelled_words = []

    for word in words:
        word = rem_punct(word)

        if word not in misspelled_words and \
           is_misspelled(dictionary, word):
             misspelled_words.append(word)

    return misspelled_words
```

B. Remove Punctuation

B. Remove Punctuation



```
def rem_punct(word):
    """
        Removes punctuation from english words if exists
    """
    if word[0] in "&-":
        word = word[1:]

    if word[-1] in ".,?!":
        word = word[:-1]

    return word
```

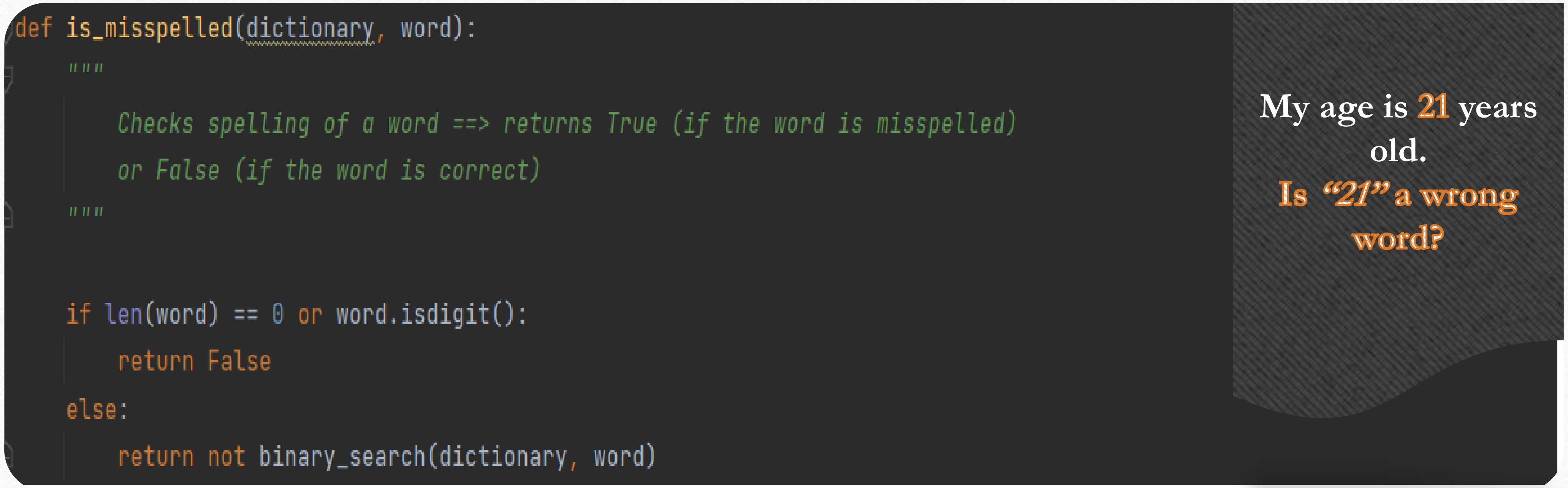
Don't forget!!!

It may return an empty word (i.e.) word = "".

How can it be handled?

C. Check misspelled words

C. Check misspelled words



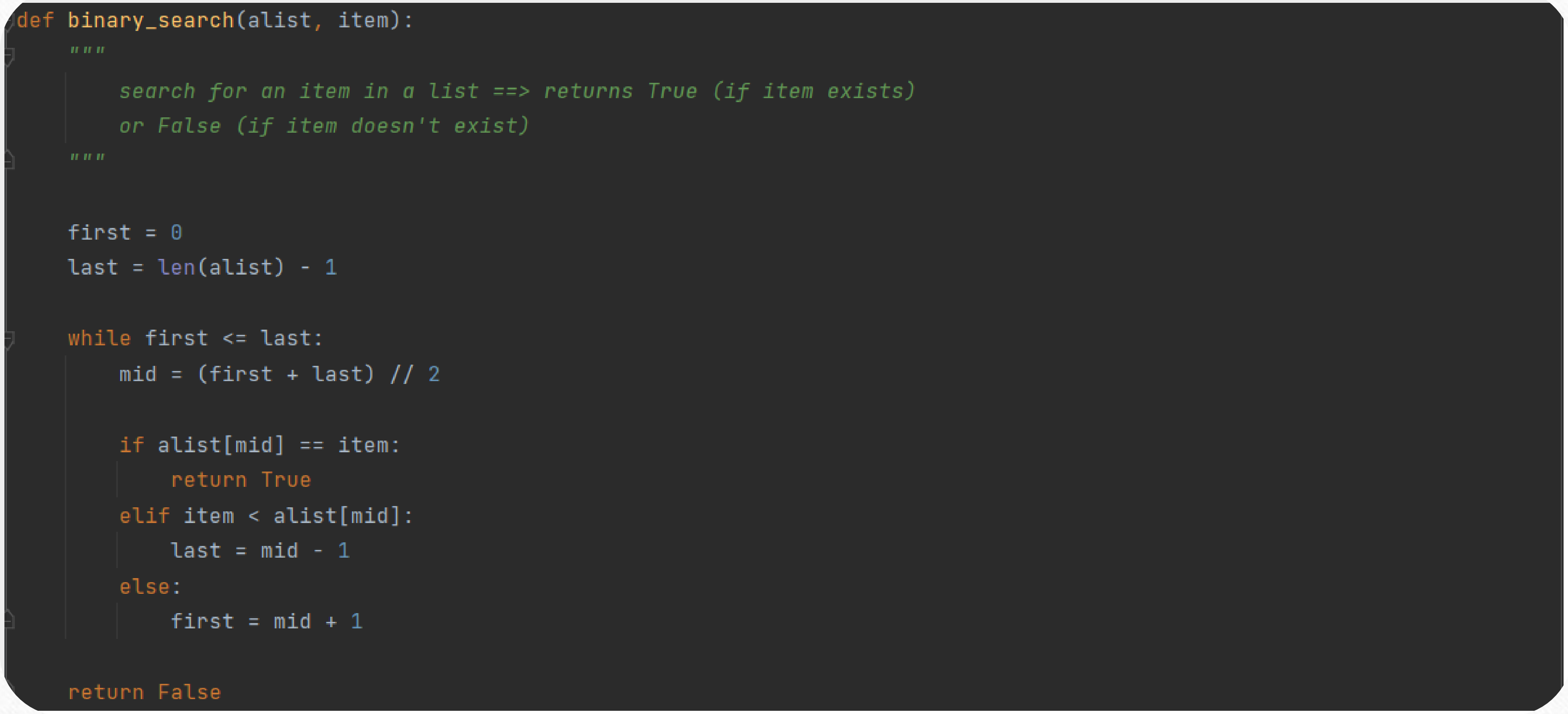
```
def is_misspelled(dictionary, word):
    """
        Checks spelling of a word ==> returns True (if the word is misspelled)
        or False (if the word is correct)
    """

    if len(word) == 0 or word.isdigit():
        return False
    else:
        return not binary_search(dictionary, word)
```

My age is 21 years old.

Is "21" a wrong word?

D. Binary Search

D. Binary Search



```python
def binary_search(alist, item):
    """
        search for an item in a list ==> returns True (if item exists)
        or False (if item doesn't exist)
    """

    first = 0
    last = len(alist) - 1

    while first <= last:
        mid = (first + last) // 2

        if alist[mid] == item:
            return True
        elif item < alist[mid]:
            last = mid - 1
        else:
            first = mid + 1

    return False
```

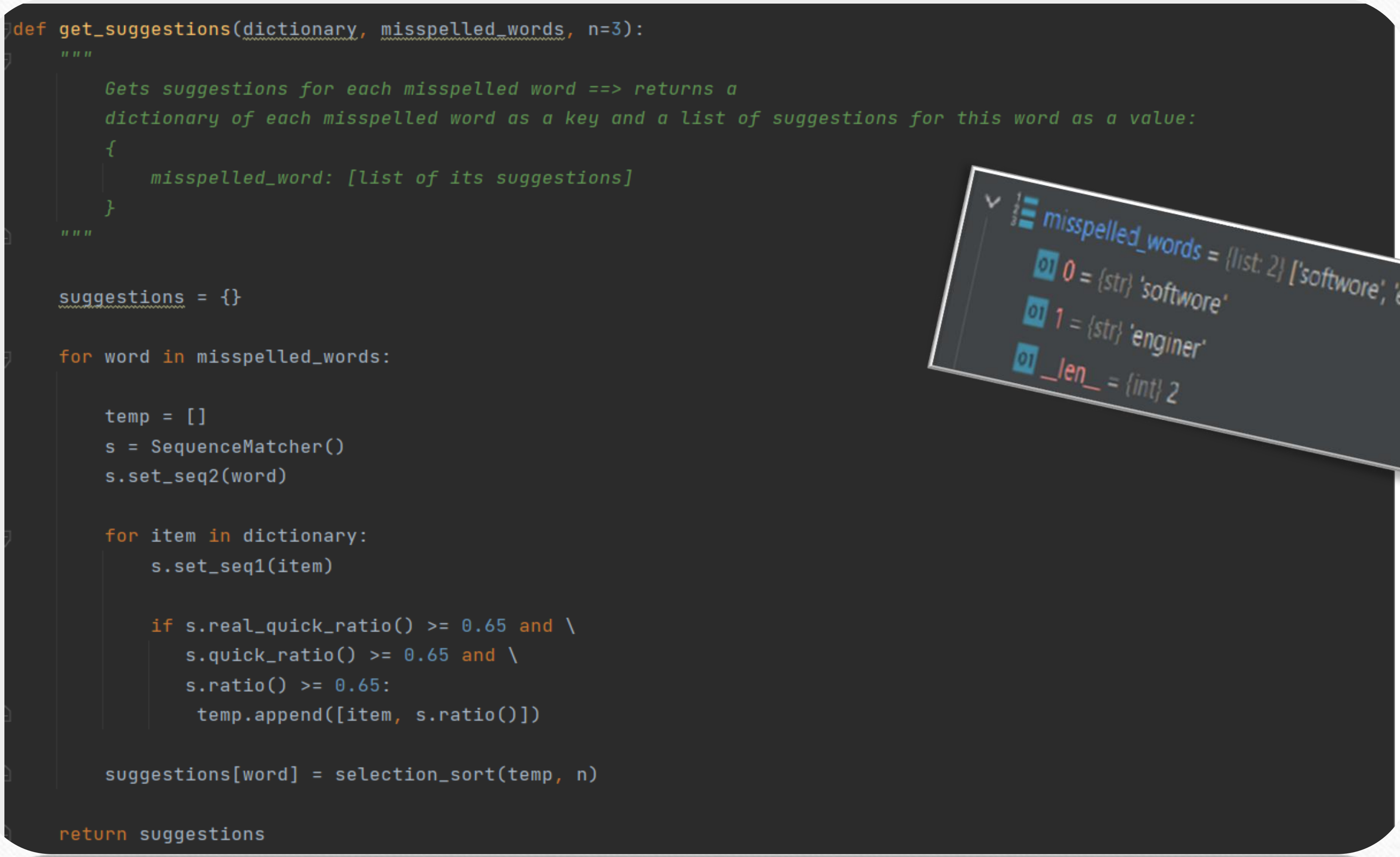
3. Find Suggestions

A. Get Suggestions

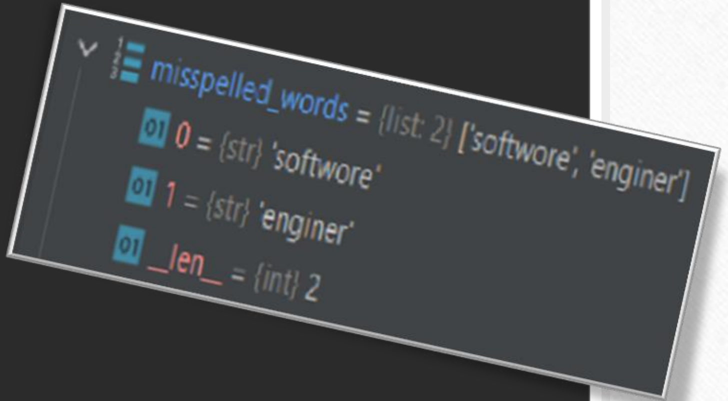
B. Sort the suggested words

A. Get Suggestions

A. Get Suggestions

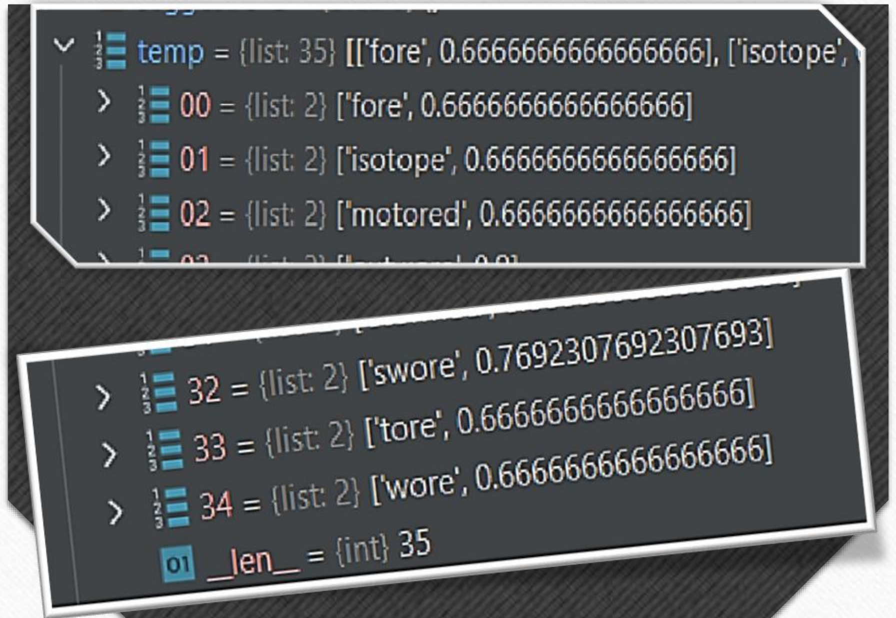


```python
def get_suggestions(dictionary, misspelled_words, n=3):
    """
        Gets suggestions for each misspelled word ==> returns a
        dictionary of each misspelled word as a key and a list of suggestions for this word as a value:
        {
            misspelled_word: [list of its suggestions]
        }
    """

    suggestions = {}

    for word in misspelled_words:

        temp = []
        s = SequenceMatcher()
        s.set_seq2(word)

        for item in dictionary:
            s.set_seq1(item)

            if s.real_quick_ratio() >= 0.65 and \
               s.quick_ratio() >= 0.65 and \
               s.ratio() >= 0.65:
                temp.append([item, s.ratio()])

        suggestions[word] = selection_sort(temp, n)

    return suggestions
```

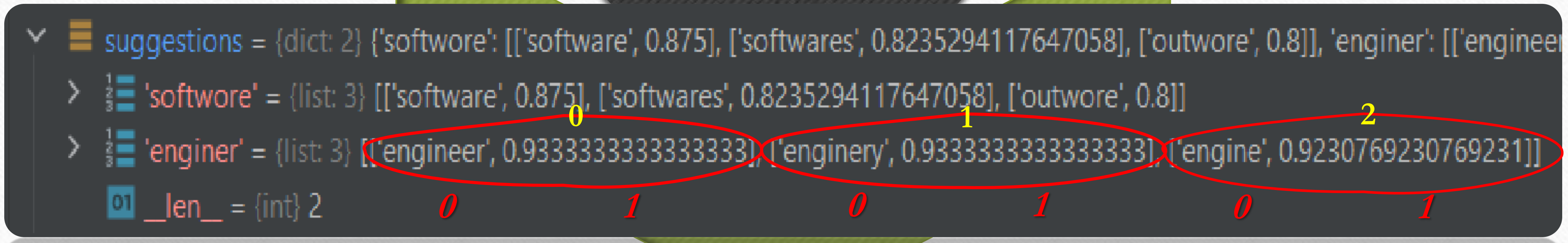


```
misspelled_words = {list: 2} ['softwore', 'enginer']
    0 = {str} 'softwore'
    1 = {str} 'enginer'
    __len__ = {int} 2
```

Suggestions for **“softwore”**



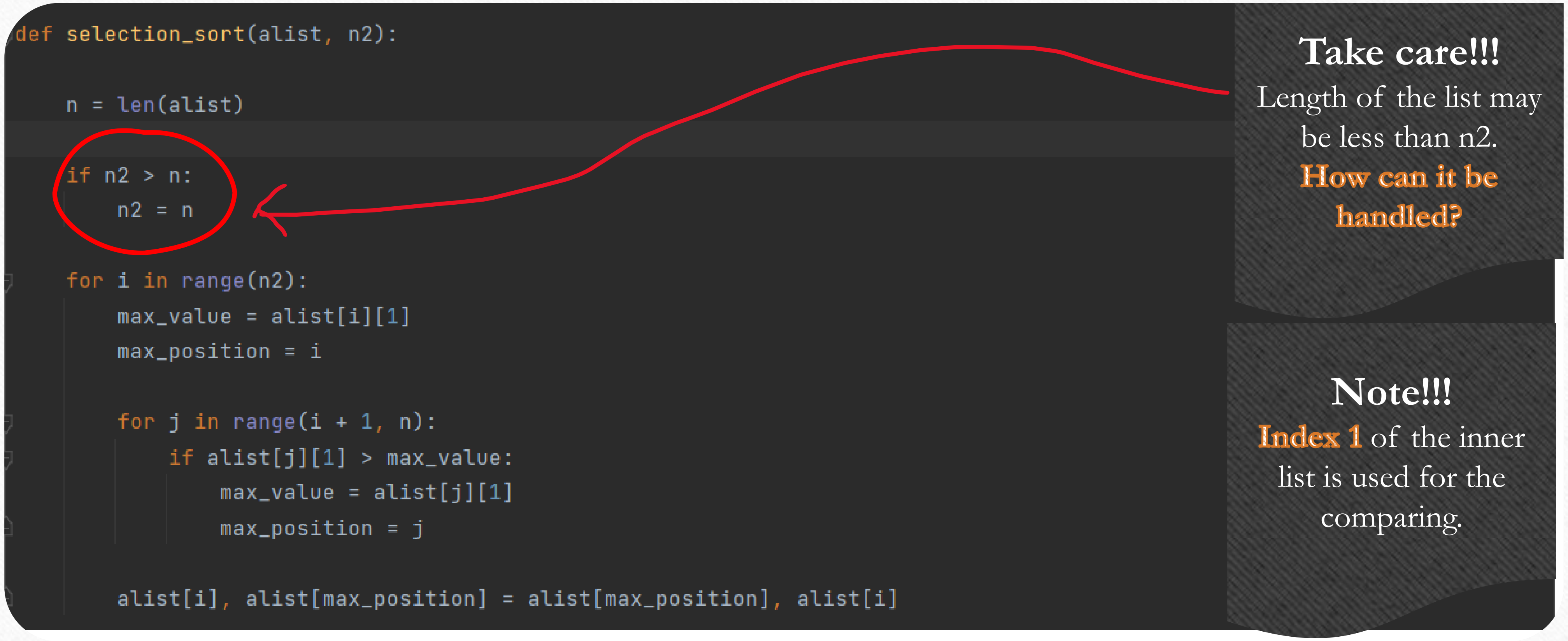
```
temp = {list: 35} [['fore', 0.6666666666666666], ['isotope',
  00 = {list: 2} ['fore', 0.6666666666666666]
  01 = {list: 2} ['isotope', 0.6666666666666666]
  02 = {list: 2} ['motored', 0.6666666666666666]
  32 = {list: 2} ['swore', 0.7692307692307693]
  33 = {list: 2} ['tore', 0.6666666666666666]
  34 = {list: 2} ['wore', 0.6666666666666666]
  __len__ = {int} 35
```



```
suggestions = {dict: 2} {'softwore': [['software', 0.875], ['softwares', 0.8235294117647058], ['outwore', 0.8]], 'enginer': [['engineer
  'softwore' = {list: 3} [['software', 0.875], ['softwares', 0.8235294117647058], ['outwore', 0.8]]
  'enginer' = {list: 3} [['engineer', 0.9333333333333333], ['enginery', 0.9333333333333333], ['engine', 0.9230769230769231]]
  __len__ = {int} 2
```

B. Sort the suggested words

B. Sort the suggested words



```
def selection_sort(alist, n2):

    n = len(alist)

    if n2 > n:
        n2 = n

    for i in range(n2):
        max_value = alist[i][1]
        max_position = i

        for j in range(i + 1, n):
            if alist[j][1] > max_value:
                max_value = alist[j][1]
                max_position = j

        alist[i], alist[max_position] = alist[max_position], alist[i]
```

Take care!!!
Length of the list may be less than n2.
How can it be handled?

Note!!!
Index 1 of the inner list is used for the comparing.

4. Print Results

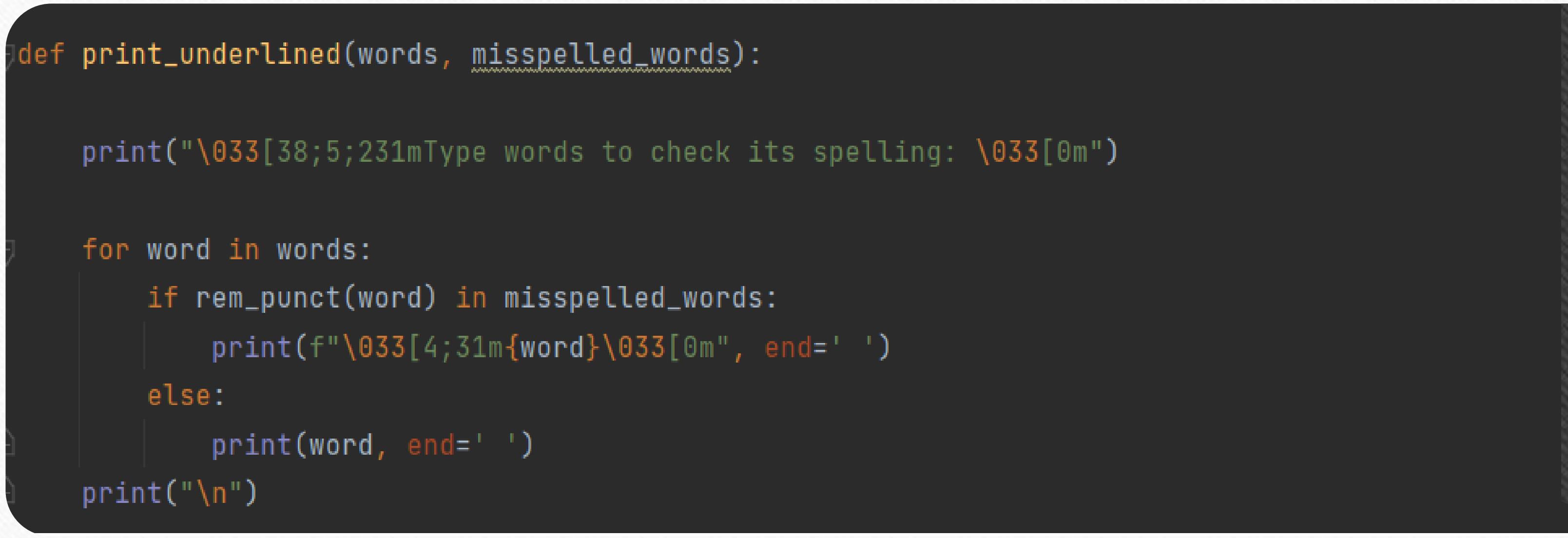
1

Mark the wrong words.

2

Print the suggestions.

A. Mark the wrong words

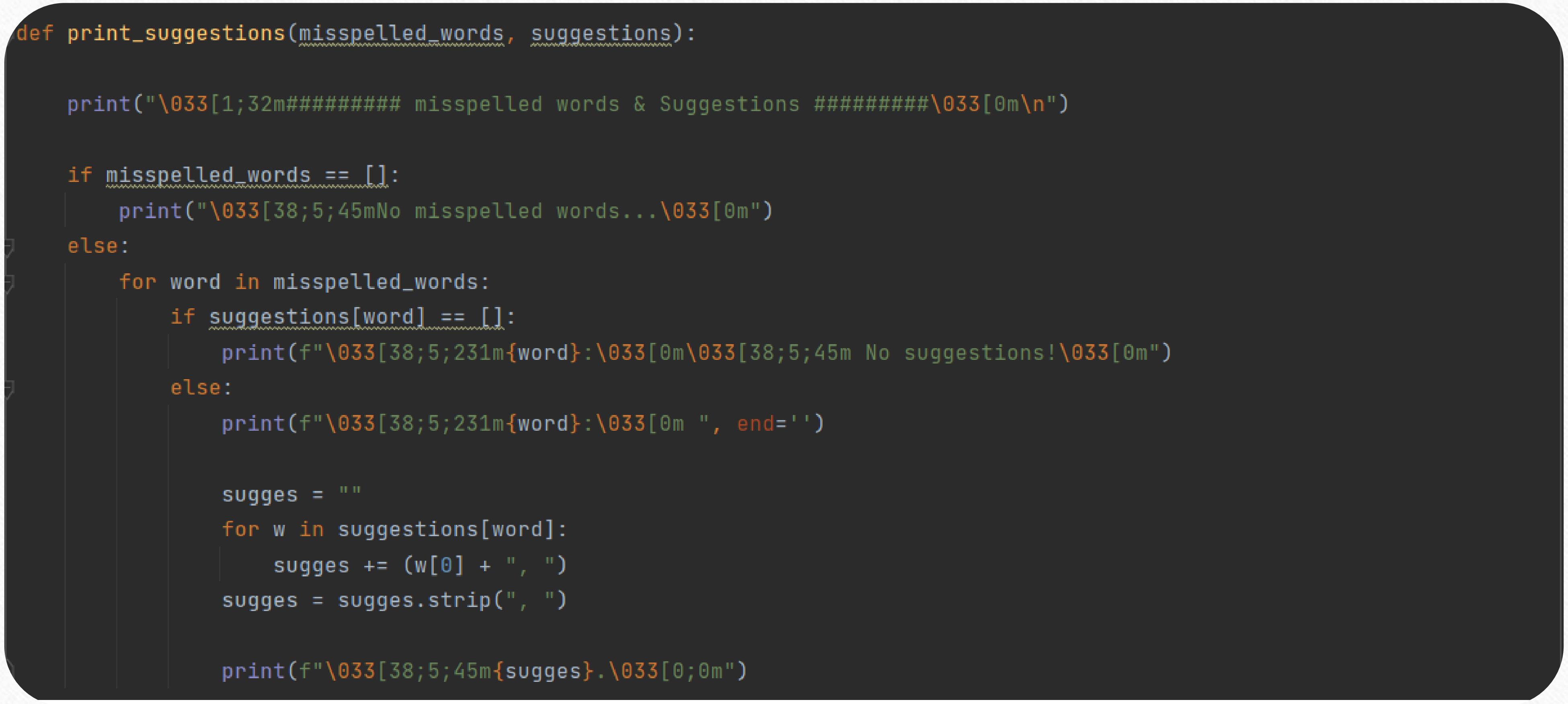


```
def print_underlined(words, misspelled_words):

    print("\033[38;5;231mType words to check its spelling: \033[0m")

    for word in words:
        if rem_punct(word) in misspelled_words:
            print(f"\033[4;31m{word}\033[0m", end=' ')
        else:
            print(word, end=' ')
    print("\n")
```

Help!
the following link contains more info. about coloring console.

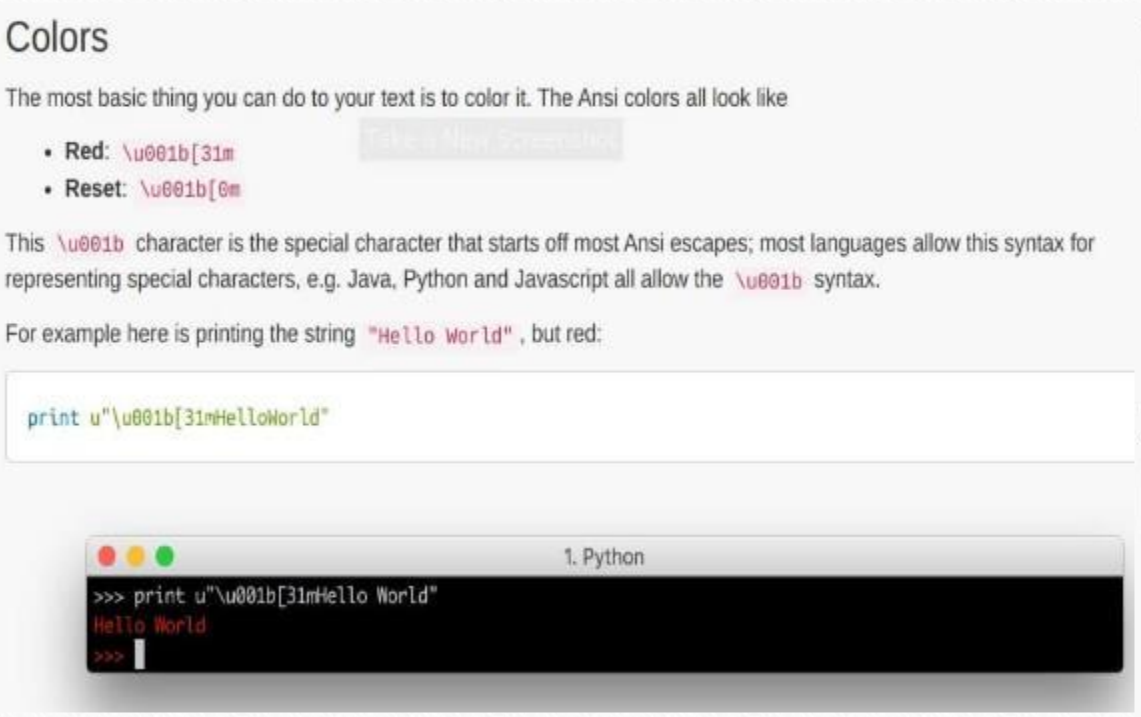
Build your own Command Line with ANSI escape codes (lihaoyi.com)

B. Print suggestions



```
def print_suggestions(misspelled_words, suggestions):

    print("\033[1;32m######### misspelled words & Suggestions #########\033[0m\n")

    if misspelled_words == []:
        print("\033[38;5;45mNo misspelled words...\033[0m")
    else:
        for word in misspelled_words:
            if suggestions[word] == []:
                print(f"\033[38;5;231m{word}:\033[0m\033[38;5;45m No suggestions!\033[0m")
            else:
                print(f"\033[38;5;231m{word}:\033[0m ", end='')

                sugges = ""
                for w in suggestions[word]:
                    sugges += (w[0] + ", ")
                sugges = sugges.strip(", ")

                print(f"\033[38;5;45m{sugges}.\033[0;0m")
```

How coloring?



Colors

The most basic thing you can do to your text is to color it. The Ansi colors all look like

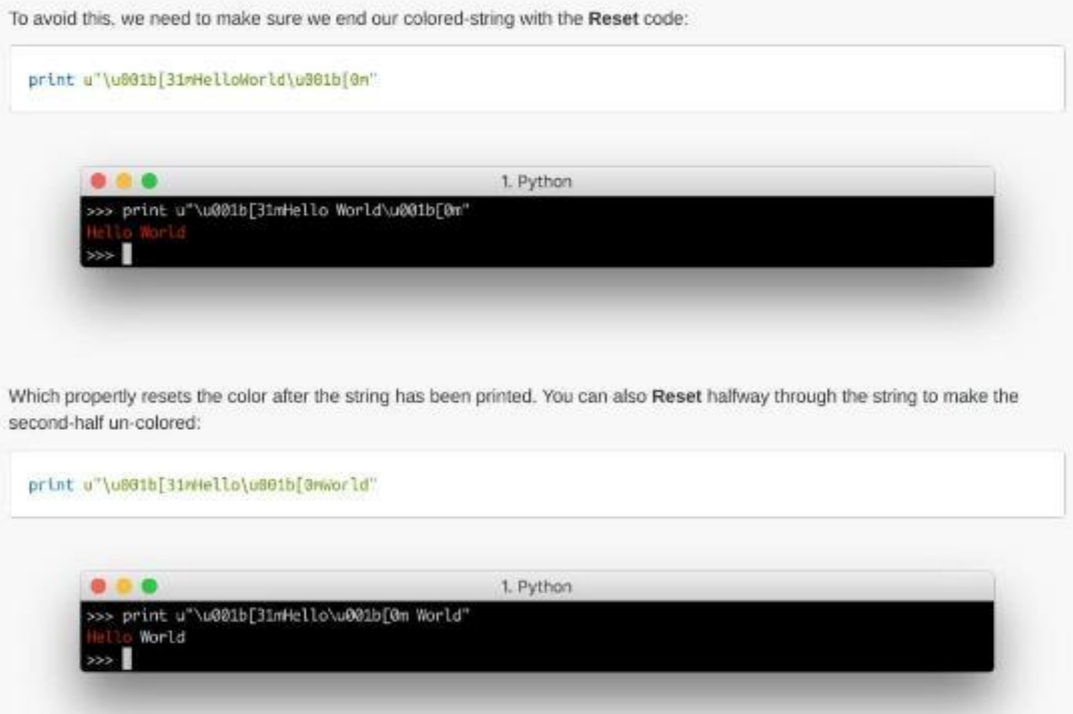
- **Red**: `\u001b[31m`
- **Reset**: `\u001b[0m`

This `\u001b` character is the special character that starts off most Ansi escapes; most languages allow this syntax for representing special characters, e.g. Java, Python and Javascript all allow the `\u001b` syntax.

For example here is printing the string `"Hello World"`, but red:

```
print u"\u001b[31mHelloWorld"
```

```
1. Python
>>> print u"\u001b[31mHello World"
Hello World
>>>
```



To avoid this, we need to make sure we end our colored-string with the **Reset** code:

```
print u"\u001b[31mHelloWorld\u001b[0m"
```

```
1. Python
>>> print u"\u001b[31mHello World\u001b[0m"
Hello World
>>>
```

Which properly resets the color after the string has been printed. You can also **Reset** halfway through the string to make the second-half un-colored:

```
print u"\u001b[31mHello\u001b[0mWorld"
```

```
1. Python
>>> print u"\u001b[31mHello\u001b[0m World"
Hello World
>>>
```

Build your own Command Line with ANSI escape codes (lihaoyi.com)

Team Members

- | Mohamed Abohend
- | Mohamed Shams
- | Yousef Khaled
- | Omar Negm
- | Hazem Shahawy
- | Mohamed Adel
- | Yousef Hamad
- | Amr Elkafrawy
- | Ahmed Elsberbawy
- | Omar Reda
- | Omar Osama

- | Omar Elmezayn
- | Mohamed Elelemy
- | Shereen Nasr
- | Mohamed Gamal
- | Eman Khayal
- | Ibrahim Elawa
- | Moaz Ramdan
- | Mohamed Saeed
- | Maged Mohamed
- | Ahmed Shaaban

https://github.com/yousefkhaled2000/SpellChecker/

Prof. **Mohamed Aita**
Eng. **Mohamed Elkomy**