answers Trusted answers to developer questions

The Word Break II problem in Python

Educative Answers Team

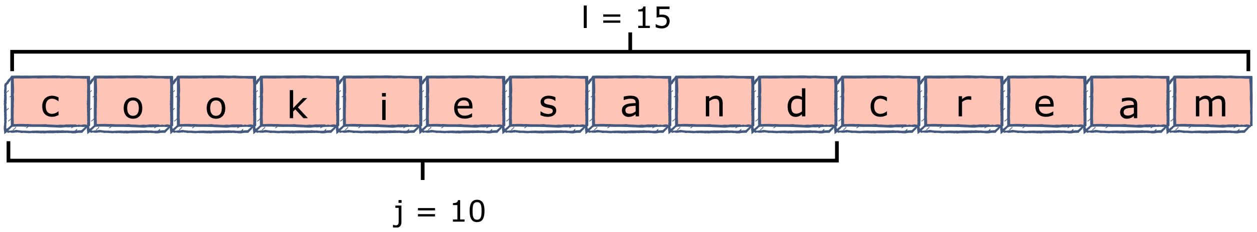
The Word Break II problem is about constructing all possible sentences, with words that are all present in the *dictionary*, by adding spaces in a *string*. A word in the *dictionary* can be used multiple times as well.

For instance, if the `dictionary` contains the words “cookie”, “cookies”, “and”, “sand”, “cream” – and the string `s` is “cookiesandcream”-- then the valid sentences are “cookie sand cream” and “cookies and cream”.

Algorithm

Dynamic Programming is used to solve the word break II problem; the idea is to store and re-use the partial results.

Let `result[j-1]` contain all the valid sentences that can be formed with a substring of string `s` of length j. Then the valid sentences, for `s` and length l, can be found out by checking if the substring, `s[j : l]`, is present in the dictionary and appending this word to all the sentences in `result [j-1]`.



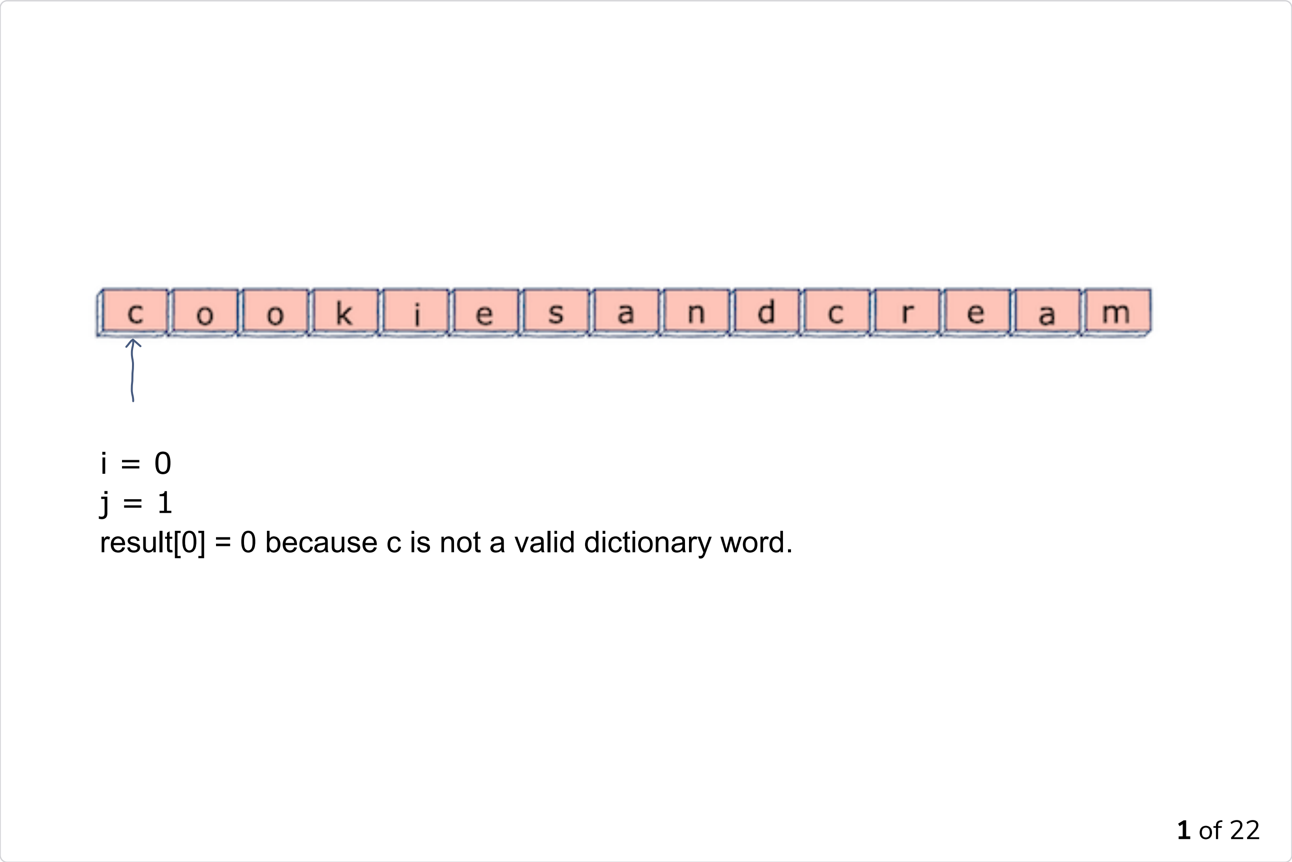
String s

`result[9]` = [“cookies and”, “cookie sand”]

Since “cream” is a valid dictionary word, append it to all the sentences that can be formed by “cookiesand”.

`result[14]` = [“cookies and cream”, “cookie sand cream”]

The animation below shows a few iterations of the algorithm in action.



i = 0
j = 1
result[0] = 0 because c is not a valid dictionary word.

1 of 22

Implementation

- `i` is set to `j-1` and, in every iteration, is incremented by 1 to check all the substrings ending at the index `j-1` (line 6).
- If a substring is a word in the dictionary, and the substring is not starting from index 0 (the start of `s`), then check if the substring starting from index 0 of `s`, that ends just before index `i`, forms a sentence. If it does, append the word to all the sentences in the `result[i-1]` (lines 16 - 24, line

append the word to all the sentences in the `result[i-1]` (lines 16 - 24, line 34) .

- If a substring is a word in the dictionary and the substring is starting from index 0 (the start of `s`), then append the word to an empty list and append this list to the `result` (lines 26 - 27).
- Else, append an empty list to the `result` to indicate that a valid sentence could not be formed by the substring of length j starting at index 0 (lines 31 - 32).
- In the end, check if `s` is a valid dictionary word. If so, append it to the `result` (lines 35 - 36).

```
s = "cookiesandcream"
dictionary = ["cookie", "cookies", "and", "sand", "cream"]
result = []
max_l = len(max(dictionary, key=len))
length = len(s) + 1
for j in range(1,length):
    i = j - 1
    flag = 0
    ans = []
    x = 0
    # Letting setting x to j - max_l optimization,
    # the code will work even if x is always set to 0
    if j > max_l:
        x = j - max_l
    while(i >= x):
        if s[i:j] in dictionary:
            if i > 0 and result[(i - 1)]:
                    # appending the word to all the valid sentences
                    # formed by the substring ending at i-1
                    temp = list((map(lambda x : x + " "+ s[i:j],\
                    result[(i - 1)])))
                    for elem in temp:
                        ans.append(elem)
                    flag = 2
            else:
                flag = 1
                result.append([s[i:j]])
        i=i-1
    # if the substring does not belong to the
    # dictionary append an empty list to result
    if flag == 0:
        result.append([])
    if flag == 2:
        result.append(ans)
if s in dictionary:
    result[len(s) - 1].append(s)
```

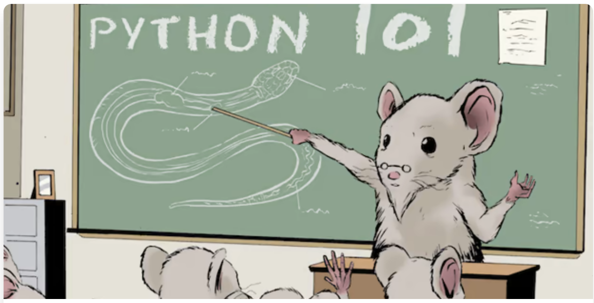
```
  result[len(s) - 1].append(s)
# Printing the result.
temp = ", result [{}]: "
for i in range(len(s)):
    print("s:", s[:(i+1)], temp.format(i), result[i])
# If result[len(s)-1] is empty then the string cannot be
# broken down into valid strings
print("Final answer for cookies and cream:", result[len(s) - 1])
```

Run

RELATED TAGS

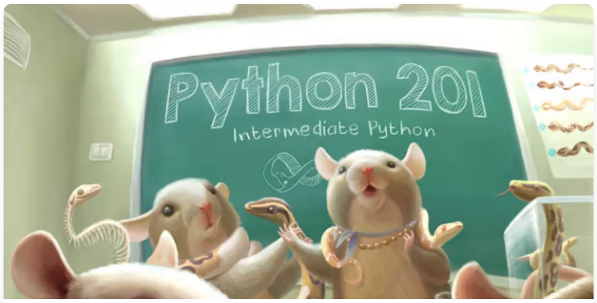
python

Related Courses



Michael Driscoll

Python 101: Interactively learn how to program with Python 3



Michael Driscoll

Python 201 - Interactively Learn Advanced Concepts in Python 3