Most frequently used words in a text

Write a function that, given a string of text (possibly with punctuation and line-breaks), returns an array of the top-3 most occurring words, in descending order of the number of occurrences.

Assumptions:

- A word is a string of letters (A to Z) optionally containing one or more apostrophes (`'`) in ASCII.
- Apostrophes can appear at the start, middle or end of a word (`'abc`, `abc'`, `'abc'`, `ab'c` are all valid)
- Any other characters (e.g. `#`, `\`, `/`, `.` ...) are not part of a word and should be treated as whitespace.
- Matches should be case-insensitive, and the words in the result should be lowercased.
- Ties may be broken arbitrarily.
- If a text contains fewer than three unique words, then either the top-2 or top-1 words should be returned, or an empty array if a text contains no words.

Examples:

```
"In a village of La Mancha, the name of which I have no desir
e to call to
mind, there lived not long since one of those gentlemen that
keep a lance
in the lance-rack, an old buckler, a lean hack, and a greyhou
nd for
coursing. An olla of rather more beef than mutton, a salad on
most
```

```
nights, scraps on Saturdays, lentils on Fridays, and a pigeon
or so extra
on Sundays, made away with three-quarters of his income."

--> ["a", "of", "on"]


"e e e e DDD ddd DdD: ddd ddd aa aA Aa, bb cc cC e e e"

--> ["e", "ddd", "aa"]


"  //wont won't won't"

--> ["won't", "wont"]
```

```
import re
from collections import Counter

def top_3_words(text):
    # Convert text to lowercase and use regex to find words
    words = re.findall(r"\b[a-zA-Z']+\b", text.lower())

    # Count the frequency of each word
    freq = Counter(words)

    # Get the top 3 most common words
    top_3 = freq.most_common(3)

    # Return only the words, not the counts
    return [word for word, count in top_3]
```

```
# Test cases
print(top_3_words("Hello, hello, hello. How low? How low? How lo
print(top_3_words("The quick brown fox jumps over the lazy dog."
```

Regular expression

converts the text to lowercase and uses regex to find words

regex pattern matces words which contains alphabetic characters and apostraphes \b is sandwiched between [a-zA-Z’] denotes the word boundaries

text.lower() simply just converts entire text to lower case

Counter from collections - creates a counter object that counts frequency of each word in the list ‘words’

freq.most_common(3) - returns 3 most common words and their counts as a list of tuples

[word for word, count in top_3]

this list comprehension denotes that its a new list

word is the expression, so whats being added to the list

for word, count in top_3

basically unpacks the variable of word and count in top_3

```
def top_3_words(text):
    # Helper function to strip punctuation
    def strip_punctuation(word):
        return ''.join(char for char in word if char.isalnum() o

    # Convert text to lowercase and split into words
    words = text.lower().split()
```

```
    # Process words to remove punctuation and filter out empty w
    words = [strip_punctuation(word) for word in words if strip_
    //A way to read this is, for every word in words, it will re
    //We know that an if conditions is a boolean operation, can
    //What strip_punctuation does is simply returns a word witho
    //or an empty string (false) after presumably processing an
    //The resulting items that passes the filter, the expressior
    //And thats what defines what each element in the resulting
    //[word for word in words if strip_punctuation(word)] would
    //With punctuated words


    # Count the frequency of each word
    freq = {}
    for word in words:
        if word in freq:
            freq[word] += 1
        else:
            freq[word] = 1

    # Sort words by frequency in descending order and get the to
    sorted_words = sorted(freq.items(), key=lambda item: item[1]

    # Return only the words, not the counts
    return [word for word, count in sorted_words[:3]]

# Test cases
print(top_3_words("Hello, hello, hello. How low? How low? How lo
print(top_3_words("The quick brown fox jumps over the lazy dog."
```

A function is created to strip any punctuation, by iterating every character in word if the characters are alphanumeric or is an ‘

then the return simply adds the words back together again

The word is then split into words, then converted into lower case

```
words = [strip_punctuation(word) for word in words if strip_punc
```

as words are already split, a function combs through the list of words in the array, removing the punctuations

```
freq = {}
   for word in words:
       if word in freq:
           freq[word] += 1
       else:
           freq[word] = 1
```

a set is created, iterates through words, for every word in words, if its found in freq, it increments the frequency of that word

remember that dictionary in python will automatically add a new key value pair if you try to access a key that doesnt exist

```
sorted_words = sorted(freq.items(), key=lambda item: item[1], re
```

freq.items() a method that returns view object that displays list of a dictionary key, value tuple pair

key=lambda item: item[1] - key function used for sorting,

lambda function that takes item (key, value) tuple and returns second element of the tuple (frequency in this case) so items are sorted based on their frequency - this is so that frequency, so the value is sorted.

reverse=True sorts items in descending order

lambda item defines an anonymous function that takes one argument which we're calling item, item[1] access the second elmeent of item which is the frequency of the word

For example, suppose you have a list of tuples like `[(word1, frequency1), (word2, frequency2), ...]`. Using `key=lambda item: item[1]` with `sorted()` will sort this list based on the frequencies (`frequency1`, `frequency2`, etc.).