**Documentation**

**Introduction**  
 Palindromes are words, phrases, or sequences that read the same backward as forward. Examples include "Anna" and "wow." In text analysis, finding palindromes might be helpful in various linguistic and pattern recognition tasks.  
  
 Actually, this code is intended to run text analyses in search of the second most frequent palindrome across each fragment of text. It does so by splitting text into words, counting palindromes and their frequency, and finding the second most frequent palindrome.

**Function:** split\_text\_into\_words(text)

* **Description:** Splits any given text into its individual words. Goes through the text, looking at alphanumeric characters while discarding non-alphanumeric ones. Processes the text in lowercase for consistency.
* **Parameters:** text (str): Input text to be split into words.
* **Returns:** List[str]: List of words that will be extracted from the text given as an input.
* **Example:**

split\_text\_into\_words("Madam Anna and I will go kayaking.”

# Returns: [‘madam’, 'anna', 'and', 'i', ‘will’, ‘go’, ‘kayaking’]

* **Summary:**
* Converts input text to lower case.
* Iterates over each character in text.
* If character is alphanumeric, appends character to current word.
* If character is not alphanumeric, adds current word to list if it's not empty.
* Adds the last word to the list, if any remains after the loop.

**Function:** count\_palindromes(words)

* **Description:** Count the occurrences of palindromic words in a list. A palindrome is defined as a word that reads the same backward as forward and must have more than one character.
* **Parameters:** words (List[str]): A list of words to be analyzed for palindromes.
* **Returns:** Dict[str, int]: A dictionary where the keys are the palindromic words and the values are their counts.
* **Example:**

count\_palindromes(['hello', 'racecar', 'level', 'level'])

# Returns: {'racecar': 1, 'level': 2}

* **Summary:**
* Checks whether every word in the list is a palindrome and it has more than one character.
* Counts occurrences of each palindrome using a hash table.

**Function:** second\_frequent(text)

* **Description:** The second most frequent palindrome from the given text. In this function, the text preprocessed into words, then the count of each palindrome, and after that, the second most frequency one is found.
* **Parameters:** text (str): Text to parse.
* **Returns:** string: The second most frequent palindrome in the text, or a message that less than two palindromes I found.
* **Example:**

second\_most\_common\_palindrome ("level deed racecar deed")

# Returns: ‘level’

* **Summary:**
* Uses split\_text\_into\_words to extract words from the text.
* Uses count\_palindromes to count the palindromes in the list of words.
* Sorts the palindromes based on frequency in descending order, and alphabetically in case of a tie.
* Determines the highest frequency and identifies palindromes that are not the most frequent.
* Returns the first palindrome from the sorted list of second most frequent palindromes or indicates if no such palindrome exists.

**Function:** get\_texts\_from\_file(file\_path)

**Description:** Reads lines of text from a specified file and filters out empty lines. Each line, which is not empty, is considered as a separate piece of text.

* **Parameters:** file\_path (str): Path to the file containing the lines of text.
* **Returns:** List[str]: List of non-empty text lines read from the file.
* **Example:**

get\_texts\_from\_file("text.txt")

# Returns: ['First line of text', 'Second line of text', 'Third line of text']

* **Summary:**
* Opens the file and reads its contents.
* Strips leading and trailing whitespace from every line.
* Includes only non-empty lines in the resulting list.

**Function:** main()

**Description:** The main function runs the general logic of the script. The said script reads texts from a file, checks if there are at least three pieces of text, and then processes each piece for the second most common palindrome.

* **Parameters:** None.
* **Returns:** None.
* **Behavior:**
* Reads texts from the file text.txt.
* Checks if there are at least three pieces of text.
* It picks out of each piece of text, the second most common palindrome and prints out the result.