Table of Contents

## 1 FunctionDef count\_vowels(text)

## 2 Function count\_vowels

**count\_vowels**: The function of count\_vowels is to count the total number of vowels within a given input string in a case-insensitive manner.

**parameters**: The parameters of this Function. · text: The input string to be scanned for vowels.

**Code Description**: The function initializes a set named vowels containing all lowercase and uppercase English vowels (‘a’, ‘e’, ‘i’, ‘o’, ‘u’, ‘A’, ‘E’, ‘I’, ‘O’, ‘U’). It then iterates through each character (ch) of the input text. For each character, it checks if the character is present in the vowels set. A generator expression yields the number 1 for every character that is a vowel. Finally, the built-in sum() function calculates the total of all the generated 1s, effectively counting the vowels, and returns this sum.

**Note**: The check for vowels is case-insensitive because the vowels set explicitly includes both uppercase and lowercase vowel characters. The function only considers ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ as vowels and does not account for other characters like ‘y’.

**Output Example**:

3

## 3 FunctionDef pairwise\_sum(numbers)

## 4 Function pairwise\_sum

**pairwise\_sum**: The function of pairwise\_sum is to compute the sum of an iterable of numbers using a numerically stable approach known as Kahan summation.

**parameters**: The parameters of this Function. · numbers: An iterable containing float or integer values to be summed.

**Code Description**: The function initializes a total and a compensation variable, both to 0.0. It then iterates through each value in the input numbers iterable.

Inside the loop, it first calculates a corrected value y by subtracting the compensation from the current value (which is cast to a float). This compensation term holds the error from the previous summation step. Next, it calculates a temporary sum t by adding the corrected value y to the running total. A new compensation value is then computed by finding the difference between the new sum t and the original total, and then subtracting y. This captures the low-order bits lost in the t = total + y operation. Finally, the total is updated to the value of t.

After the loop has processed all the numbers, the function returns the final total.

**Note**: This function implements the Kahan summation algorithm, which significantly reduces numerical error compared to a naive summation, especially when summing many floating-point numbers or numbers of widely different magnitudes. All input values are cast to float during the computation.

**Output Example**:

# Calling pairwise\_sum([0.1, 0.2, 0.3, 0.4, 0.5])  
0.1 + 0.2 + 0.3 + 0.4 + 0.5

1.5

## 5 FunctionDef split\_into\_chunks(text, size)

## 6 Function split\_into\_chunks

**split\_into\_chunks**: The function of split\_into\_chunks is to divide a string into a tuple of smaller, fixed-size substrings.

**parameters**: The parameters of this Function. · text: The input string that will be split into chunks. · size: An integer specifying the desired length for each chunk. This value must be positive.

**Code Description**: The function first validates the size parameter. If size is less than or equal to zero, it raises a ValueError with the message “size must be positive”. If size is valid, the function proceeds to split the text. It uses a generator expression within a tuple() constructor to create the chunks. The range() function generates starting indices for each chunk, beginning at 0 and incrementing by the size value until the end of the string is reached. For each starting index i, a slice of the string text[i : i + size] is created. This process continues until the entire string has been chunked. The resulting substrings are collected into a tuple and returned.

**Note**: The final substring in the returned tuple may be shorter than the specified size if the length of the input text is not an even multiple of size. A ValueError will be raised if the size parameter is not a positive number.

**Output Example**:

('hel', 'lo ', 'wor', 'ld')