Table of Contents

## 1 FunctionDef fibonacci(n)

The function of **fibonacci** is to compute the nth Fibonacci number using an iterative approach.

**parameters**: The parameters of this Function. · n: An integer representing the 0-indexed position in the Fibonacci sequence for which to compute the value.

**Code Description**: The function begins by checking if the input n is a negative number. If it is, a ValueError is raised, as the Fibonacci sequence is not defined for negative indices.

Next, two variables, a and b, are initialized to 0 and 1 respectively. These represent the first two numbers in the Fibonacci sequence.

The code then enters a for loop that iterates n times. In each iteration, the values of a and b are updated. The current value of b is assigned to a, and the sum of the previous a and b is assigned to b. This process effectively steps through the Fibonacci sequence one number at a time.

After the loop has completed n iterations, the function returns the final value of a, which holds the nth Fibonacci number.

**Note**: This function calculates the sequence starting from the 0th index. For example, fibonacci(0) returns 0, and fibonacci(1) returns 1. The input n must be a non-negative integer.

**Output Example**:

# Calling fibonacci(8)  
fibonacci(8)  
  
# Returns  
21

## 2 FunctionDef invert\_dictionary(mapping)

The function of **invert\_dictionary** is to swap the keys and values of a dictionary, raising an error if the original dictionary’s values are not unique.

**parameters**: The parameters of this Function. · mapping: A dictionary with string keys and integer values. The values in this dictionary must be unique.

**Code Description**: The function first validates the input mapping. It compares the number of values in the dictionary (len(mapping.values())) with the number of unique values (len(set(mapping.values()))). If these two counts are not equal, it indicates that there are duplicate values in the input dictionary. In this case, a ValueError is raised with the message “Values must be unique to invert dictionary”.

If all values are unique, the function proceeds to create and return a new dictionary using a dictionary comprehension. It iterates through each key-value pair of the input mapping and constructs a new dictionary where the original value becomes the new key and the original key becomes the new value.

**Note**: This function will fail and raise a ValueError if the input dictionary contains any duplicate values, as this would result in key collisions in the inverted dictionary.

**Output Example**:

# Given the input:  
mapping = {'apple': 1, 'banana': 2, 'cherry': 3}  
  
# Calling invert\_dictionary(mapping) returns:  
{1: 'apple', 2: 'banana', 3: 'cherry'}

## 3 FunctionDef is\_palindrome(text)

The function of **is\_palindrome** is to determine if a given string is a palindrome, meaning it reads the same forwards and backwards, while disregarding letter casing and spaces.

**parameters**: The parameters of this Function. · text: The input string that will be checked for the palindrome property.

**Code Description**: The function first normalizes the input text. It does this by creating a new string called normalized. This new string is built by iterating through each character (ch) of the input text. For each character, it checks if it is a space using ch.isspace(). If the character is not a space, it is converted to lowercase and included in the normalized string. All characters that are spaces are excluded.

Finally, the function compares the normalized string with its reversed version. The reversal is achieved using the slice notation [::-1]. The function returns True if the normalized string is identical to its reverse, and False otherwise.

**Note**: This function specifically ignores spaces and letter casing. Punctuation and other whitespace characters (like tabs or newlines) are not ignored and will be included in the palindrome check.

**Output Example**:

is\_palindrome("A man a plan a canal Panama")

Output:

True