Table of Contents

## 0.1 FunctionDef fibonacci(n)

# 1 Function: fibonacci(n: int)

## 1.1 Overview

The fibonacci function computes the nth number in the Fibonacci sequence using an iterative method.

## 1.2 parameters

* n (int): The 0-indexed position in the Fibonacci sequence for which to compute the value.

## 1.3 Description

This function calculates the nth Fibonacci number, where the sequence starts with 0 and 1.

The function first validates the input n. If n is a negative number, it raises a ValueError because the Fibonacci sequence is not defined for negative indices.

It initializes two variables, a and b, to the first two numbers of the sequence, 0 and 1, respectively.

The core logic resides in a for loop that iterates n times. In each iteration, the function updates the values of a and b to move to the next number in the sequence. This is achieved through the tuple assignment a, b = b, a + b, where the current value of b becomes the new a, and the sum of the old a and b becomes the new b.

After the loop completes, the variable a holds the value of the nth Fibonacci number, which is then returned.

# Initialization  
a, b = 0, 1  
  
# For n = 3, the loop runs 3 times:  
# 1. a becomes 1, b becomes 0 + 1 = 1  
# 2. a becomes 1, b becomes 1 + 1 = 2  
# 3. a becomes 2, b becomes 1 + 2 = 3  
  
# The function returns a, which is 2.  
# fibonacci(3) = 2

## 1.4 Usage Notes

* The input n must be a non-negative integer. Providing a negative value will result in a ValueError.
* The function is 0-indexed. fibonacci(0) returns 0, fibonacci(1) returns 1, fibonacci(2) returns 1, and so on.
* This iterative implementation is efficient in terms of memory and performance for large values of n compared to a naive recursive approach.

**Output Example**: A single integer representing the Fibonacci number at the specified index.

55

## 1.5 Example

# Example usage  
# Calculate the 10th Fibonacci number (0-indexed)  
result = fibonacci(10)  
print(result)

**Output:**

55

## 1.6 FunctionDef invert\_dictionary(mapping)

# 2 Function: invert\_dictionary(mapping: Dict[str, int]) -> Dict[int, str]

## 2.1 Overview

The invert\_dictionary function inverts a given dictionary by swapping its keys and values, creating a new dictionary where the original values become the keys and the original keys become the values.

## 2.2 parameters

* **mapping** (Dict[str, int]): A dictionary mapping string keys to integer values. It is crucial that the values in this dictionary are unique.

## 2.3 Description

This function provides a safe way to invert a dictionary. The core logic involves two main steps:

1. **Uniqueness Validation**: Before performing the inversion, the function first validates that all values in the input mapping dictionary are unique. It does this by comparing the number of values with the number of unique values. The expression len(mapping.values()) != len(set(mapping.values())) evaluates to True if any duplicate values exist, as converting a list of values to a set removes duplicates.
2. **Inversion**: If the validation passes, the function proceeds to invert the dictionary. It uses a dictionary comprehension, {value: key for key, value in mapping.items()}, to iterate through each key-value pair of the original mapping. For each pair, it constructs a new entry in the returned dictionary, using the original value as the new key and the original key as the new value.

If the uniqueness check fails, the function raises a ValueError with the message “Values must be unique to invert dictionary” to prevent data loss that would occur if multiple keys were mapped to the same value in the inverted dictionary.

## 2.4 Usage Notes

* The primary constraint is that all values in the input mapping dictionary must be unique. The function will raise a ValueError if this condition is not met.
* The function returns a new dictionary and does not modify the original mapping dictionary in place.
* The type hints indicate an input of Dict[str, int] and an output of Dict[int, str], but the logic applies to any dictionary with hashable keys and unique, hashable values.

**Output Example**: A dictionary where the keys are integers and the values are strings.

{1: 'a', 2: 'b', 3: 'c'}

## 2.5 Example

from typing import Dict  
  
def invert\_dictionary(mapping: Dict[str, int]) -> Dict[int, str]:  
 """Invert a dictionary with unique values.  
  
 Parameters:  
 mapping: Dictionary mapping strings to integers with unique values.  
  
 Returns:  
 A new dictionary mapping integers to strings.  
 """  
 if len(mapping.values()) != len(set(mapping.values())):  
 raise ValueError("Values must be unique to invert dictionary")  
 return {value: key for key, value in mapping.items()}  
  
# --- Example 1: Successful Inversion ---  
# The input dictionary has unique values.  
original\_dict = {'apple': 10, 'banana': 20, 'cherry': 30}  
inverted\_dict = invert\_dictionary(original\_dict)  
print(f"Original dictionary: {original\_dict}")  
print(f"Inverted dictionary: {inverted\_dict}")  
  
# --- Example 2: Inversion Failure ---  
# The input dictionary has duplicate values (10).  
non\_unique\_dict = {'apple': 10, 'banana': 20, 'apricot': 10}  
print(f"\nAttempting to invert dictionary with non-unique values: {non\_unique\_dict}")  
try:  
 invert\_dictionary(non\_unique\_dict)  
except ValueError as e:  
 print(f"Caught expected error: {e}")

**Output:**

Original dictionary: {'apple': 10, 'banana': 20, 'cherry': 30}  
Inverted dictionary: {10: 'apple', 20: 'banana', 30: 'cherry'}  
  
Attempting to invert dictionary with non-unique values: {'apple': 10, 'banana': 20, 'apricot': 10}  
Caught expected error: Values must be unique to invert dictionary

## 2.6 FunctionDef is\_palindrome(text)

# 3 Function: is\_palindrome(text: str)

## 3.1 Overview

The is\_palindrome function determines if a given string is a palindrome, ignoring letter casing and whitespace characters.

## 3.2 parameters

* text (str): The string to be checked for palindrome properties.

## 3.3 Description

This function evaluates whether a string reads the same forwards and backwards after a normalization process. The core logic involves two main steps:

1. **Normalization**: The input text is processed to create a “normalized” version. This is achieved using a generator expression that iterates through each character of the input string.
   * Any character that is a whitespace (identified by ch.isspace()) is completely removed.
   * All remaining alphabetic characters are converted to their lowercase equivalents using ch.lower().
   * These processed characters are then joined together to form a new, contiguous string.

* normalized = ''.join(ch.lower() for ch in text if not ch.isspace())

1. **Comparison**: The normalized string is then compared to its own reverse. The reversal is accomplished using slice notation [::-1], which creates a reversed copy of the string.

* return normalized == normalized[::-1]

If the normalized string and its reversed version are identical, the function returns True; otherwise, it returns False.

## 3.4 Usage Notes

* **Case-Insensitive**: The comparison is case-insensitive. For example, “Racecar” and “racecar” are both treated as valid palindromes.
* **Whitespace Ignored**: All forms of whitespace (spaces, tabs, newlines, etc.) are removed from the string before the check is performed.
* **Punctuation and Symbols**: The function does not remove punctuation, numbers, or symbols. These characters are included in the palindrome check. For example, is\_palindrome("madam, I'm adam") would return False because the comma and apostrophe are not ignored.

**Output Example**: The function returns a boolean value.

True

## 3.5 Example

# Example 1: A classic palindrome with mixed casing and spaces  
phrase = "A man a plan a canal Panama"  
result = is\_palindrome(phrase)  
print(f"Is '{phrase}' a palindrome? {result}")  
  
# Example 2: A simple non-palindrome  
word = "hello"  
result\_non\_palindrome = is\_palindrome(word)  
print(f"Is '{word}' a palindrome? {result\_non\_palindrome}")

**Output:**

Is 'A man a plan a canal Panama' a palindrome? True  
Is 'hello' a palindrome? False