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 approach.

## 1.2 parameters

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

## 1.3 Description

This function calculates a Fibonacci number based on its index n. The Fibonacci sequence starts with 0 and 1, and each subsequent number is the sum of the two preceding ones (0, 1, 1, 2, 3, 5, …).

The function begins by validating the input. 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 0 and 1 respectively. These represent the first two numbers in the sequence, F₀ and F₁.

The core logic resides in a for loop that iterates n times. In each iteration, the values of a and b are updated to advance to the next number in the sequence. The current value of b is assigned to a, and the sum of the old a and b is assigned to the new b. This is achieved with the tuple assignment a, b = b, a + b.

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

For example, if n is 3: 1. Initialize a = 0, b = 1. 2. Loop 1: a becomes 1, b becomes 0 + 1 = 1. 3. Loop 2: a becomes 1, b becomes 1 + 1 = 2. 4. Loop 3: a becomes 2, b becomes 1 + 2 = 3. 5. The loop finishes, and the function returns the final value of a, which is 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, meaning fibonacci(0) returns the first element, 0.
* This iterative implementation is efficient in terms of memory usage and is suitable for calculating large Fibonacci numbers without risking stack overflow errors that can occur with naive recursive solutions.

**Output Example**: The function returns a single integer value.

34

## 1.5 Example

# Example usage of the fibonacci function  
# Calculate the 9th Fibonacci number (0-indexed)  
n\_index = 9  
result = fibonacci(n\_index)  
print(f"The Fibonacci number at index {n\_index} is: {result}")  
  
# Example with an edge case  
n\_index\_zero = 0  
result\_zero = fibonacci(n\_index\_zero)  
print(f"The Fibonacci number at index {n\_index\_zero} is: {result\_zero}")  
  
# Example that would raise an error  
try:  
 fibonacci(-1)  
except ValueError as e:  
 print(f"Error caught as expected: {e}")

**Output:**

The Fibonacci number at index 9 is: 34  
The Fibonacci number at index 0 is: 0  
Error caught as expected: n must be non-negative

## 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, returning a new dictionary.

## 2.2 parameters

* **mapping** (Dict[str, int]): A dictionary with string keys and integer values. The values in this dictionary must be unique to allow for a valid inversion.

## 2.3 Description

This function provides a safe way to invert a dictionary mapping strings to integers.

The function first performs a crucial validation step. It checks if all the values in the input mapping dictionary are unique. This is accomplished by comparing the number of values (len(mapping.values())) with the number of unique values (len(set(mapping.values()))). If these counts do not match, it indicates the presence of duplicate values, which would lead to data loss upon inversion (as multiple original keys would map to the same new key). In this case, the function raises a ValueError with a descriptive message.

If all values are unique, the function proceeds to create and return a new 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 where the original value becomes the new key and the original key becomes the new value.

# The core logic for inversion after validation  
# {value: key for key, value in mapping.items()}

## 2.4 Usage Notes

* The primary requirement for this function is that all values in the input mapping dictionary must be unique. If they are not, the function will raise a ValueError.
* The function is non-destructive; it does not modify the original dictionary. It returns a completely new dictionary instance.
* The key-value types are swapped. An input of Dict[str, int] will produce an output of Dict[int, str].

**Output Example**: A dictionary with integer keys and string values.

{  
 1: "one",  
 2: "two",  
 3: "three"  
}

## 2.5 Example

# Example 1: Successful inversion with a valid dictionary  
valid\_mapping = {'user\_one': 101, 'user\_two': 102, 'user\_three': 103}  
inverted\_mapping = invert\_dictionary(valid\_mapping)  
print(f"Original: {valid\_mapping}")  
print(f"Inverted: {inverted\_mapping}")  
  
# Example 2: Attempting to invert a dictionary with duplicate values  
invalid\_mapping = {'alpha': 1, 'beta': 2, 'gamma': 1}  
print(f"\nAttempting to invert: {invalid\_mapping}")  
try:  
 invert\_dictionary(invalid\_mapping)  
except ValueError as e:  
 print(f"Error: {e}")

**Output:**

Original: {'user\_one': 101, 'user\_two': 102, 'user\_three': 103}  
Inverted: {101: 'user\_one', 102: 'user\_two', 103: 'user\_three'}  
  
Attempting to invert: {'alpha': 1, 'beta': 2, 'gamma': 1}  
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 checks if a given string is a palindrome, ignoring letter casing and whitespace.

## 3.2 parameters

* text (str): The string to be evaluated.

## 3.3 Description

This function determines if a string reads the same forwards and backwards by first normalizing the input text.

The normalization process involves creating a new string, normalized, by iterating through each character (ch) of the input text. For each character, it is converted to lowercase using ch.lower(), and any character that is a whitespace (checked with ch.isspace()) is excluded. The resulting filtered and lowercased characters are joined together into the normalized string.

Finally, the function compares the normalized string to its reverse. The reversal is achieved using Python’s slice notation normalized[::-1]. If the normalized string is identical to its reversed version, the function returns True; otherwise, it returns False.

# Internal logic for normalization  
text = "Taco Cat"  
normalized = ''.join(ch.lower() for ch in text if not ch.isspace())  
# normalized becomes "tacocat"  
  
# Internal logic for comparison  
# "tacocat" == "tacocat"[::-1] -> "tacocat" == "tacocat" -> True  
return normalized == normalized[::-1]

## 3.4 Usage Notes

* The comparison is case-insensitive. For example, “Madam” and “madam” are treated identically.
* All whitespace characters (spaces, tabs, newlines) are removed before the check. For example, “taco cat” is evaluated as “tacocat”.
* Punctuation and symbols are **not** ignored and will be included in the palindrome check. For instance, “A man, a plan…” will return False because the punctuation breaks the symmetry.

**Output Example**: A boolean value indicating the result.

True

## 3.5 Example

# Example 1: A classic palindrome phrase 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  
word = "hello"  
result\_false = is\_palindrome(word)  
print(f"Is '{word}' a palindrome? {result\_false}")  
  
# Example 3: A word that is a palindrome but contains punctuation  
punctuated\_phrase = "Eva, can I see bees in a cave?"  
result\_punc = is\_palindrome(punctuated\_phrase)  
print(f"Is '{punctuated\_phrase}' a palindrome? {result\_punc}")

**Output:**

Is 'A man a plan a canal Panama' a palindrome? True  
Is 'hello' a palindrome? False  
Is 'Eva, can I see bees in a cave?' a palindrome? False