Table of Contents

## 0.1 FunctionDef choose\_random\_item

# 1 Function: choose\_random\_item(items: List[str])

## 1.1 Overview

The choose\_random\_item function selects and returns a single random item from a given list of strings.

## 1.2 parameters

* **items** (List[str]): A list of strings from which to choose a single random item. This list must not be empty.

## 1.3 Description

This function provides a simple way to get a random element from a list of strings. The core logic is handled in two main steps:

1. **Input Validation**: The function first checks if the provided items list is empty using the if not items: condition. If the list is empty, it immediately raises a ValueError with the message “items must not be empty”. This prevents the program from attempting to choose an item from a non-existent collection, which would otherwise cause an error.
2. **Random Selection**: If the list is not empty, the function utilizes the random.choice(items) method from Python’s built-in random library. This method takes a non-empty sequence (in this case, the items list) and returns a single element chosen uniformly at random. “Uniformly” means that every item in the list has an equal probability of being selected.

The selected string is then returned as the result of the function call.

## 1.4 Usage Notes

* This function requires the random module to be imported in the script where it is defined.
* Providing an empty list as the items parameter will result in a ValueError. Always ensure the list contains at least one element before calling this function.
* The selection is uniformly random, meaning each item in the list has an equal chance of being chosen on any given call.

**Output Example**: A single string from the input list. For a list ['apple', 'banana', 'cherry'], a possible return value is 'banana'.

## 1.5 Example

import random  
from typing import List  
  
def choose\_random\_item(items: List[str]) -> str:  
 """Choose a single random item from a non-empty sequence.  
  
 Parameters:  
 items: A list of strings to choose from.  
  
 Returns:  
 A single string chosen uniformly at random.  
 """  
 if not items:  
 raise ValueError("items must not be empty")  
 return random.choice(items)  
  
# Example usage  
fruits = ["apple", "banana", "cherry", "date", "elderberry"]  
random\_fruit = choose\_random\_item(fruits)  
print(f"The randomly chosen fruit is: {random\_fruit}")  
  
# Example of what happens with an empty list  
try:  
 choose\_random\_item([])  
except ValueError as e:  
 print(f"Error: {e}")

**Output:**

The randomly chosen fruit is: cherry  
Error: items must not be empty

(Note: The chosen fruit in the first line of the output will vary with each execution.)

## 1.6 FunctionDef shuffle\_copy

# 2 Function: shuffle\_copy(items: List[int])

## 2.1 Overview

The shuffle\_copy function returns a shuffled copy of a given list of integers without modifying the original list.

## 2.2 parameters

* items: List[int] - A list of integers to be copied and shuffled.

## 2.3 Description

This function provides a safe, non-destructive way to shuffle a list by operating on a copy rather than the original data. The logic proceeds as follows:

1. A shallow copy of the input items list is created using the expression copy = list(items). This step is crucial as it ensures that the original list passed to the function remains unchanged.
2. The random.shuffle() function is then called on the copy. This standard Python function shuffles the elements of the list in-place, rearranging them into a random order.
3. Finally, the function returns the modified copy, which now contains the same elements as the original items list but in a new, random sequence.

This approach is useful when you need to preserve the original order of a list for later use while also needing a randomized version of it.

## 2.4 Usage Notes

* The primary feature of this function is that it is non-mutating; the input list items is not altered.
* This function requires the random module to be imported (e.g., import random) in the script where it is used.
* The order of elements in the returned list is non-deterministic due to the nature of the random shuffling algorithm.
* The function creates a shallow copy. For a list of simple types like integers, this is perfectly safe.

**Output Example**: A possible return value for an input of [10, 20, 30, 40].

[30, 10, 40, 20]

## 2.5 Example

The following example demonstrates the use of shuffle\_copy and verifies that the original list is preserved after the function call. Note that the random module must be imported.

import random  
from typing import List  
  
# Assume shuffle\_copy is defined as in the source code  
  
# --- Example Usage ---  
original\_numbers = [1, 2, 3, 4, 5, 6, 7]  
print(f"Original list before calling function: {original\_numbers}")  
  
# Get a shuffled copy of the list  
shuffled\_version = shuffle\_copy(original\_numbers)  
  
print(f"Original list after calling function: {original\_numbers}")  
print(f"Returned shuffled list: {shuffled\_version}")

**Output:**

Original list before calling function: [1, 2, 3, 4, 5, 6, 7]  
Original list after calling function: [1, 2, 3, 4, 5, 6, 7]  
Returned shuffled list: [4, 1, 7, 5, 2, 6, 3]

*(Note: The order of elements in the Returned shuffled list will vary with each execution.)*