Table of Contents

## 0.1 FunctionDef choose\_random\_item

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

## 1.1 Overview

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

## 1.2 parameters

* **items** List[str]: A list of strings to choose from. This list must not be empty.

## 1.3 Description

This function provides a safe way to select a single item uniformly at random from a list.

The function first validates the input items list. It checks if the list is empty using the if not items: condition. If the list is found to be empty, the function immediately stops execution and raises a ValueError with the message “items must not be empty”. This prevents potential errors from attempting to choose an item from an empty collection.

If the items list is not empty, the function proceeds to use the random.choice() method. This standard library method takes a non-empty sequence as an argument and returns one of its elements, with each element having an equal probability of being selected. The chosen string is then returned as the output of the choose\_random\_item function.

## 1.4 Usage Notes

* The input list items must not be empty. Providing an empty list will result in a ValueError.
* This function depends on Python’s built-in random module. Ensure it is imported in the execution environment.
* The selection is uniformly random, meaning every item in the input list has an equal chance of being chosen.

**Output Example**: If the input list is ['red', 'green', 'blue'], a possible return value is 'green'.

## 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"]  
random\_fruit = choose\_random\_item(fruits)  
print(f"The randomly chosen fruit is: {random\_fruit}")  
  
# Example of error handling  
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 actual fruit chosen 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 new list containing the same elements as the input list but in a randomized order, without altering the original list.

## 2.2 parameters

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

## 2.3 Description

This function provides a safe way to shuffle a list by operating on a copy rather than the original data. The process is as follows:

1. A shallow copy of the input items list is created using copy = list(items). This is a crucial step to prevent mutation of the original list provided by the caller.
2. The random.shuffle() function is then called on this copy. The random.shuffle() method shuffles the sequence of the list *in-place*, rearranging the order of its elements randomly.
3. Finally, the function returns the modified copy, which is now a shuffled version of the original items list.

# Internal logic  
# Assumes 'random' module is imported  
items = [1, 2, 3]  
copy = list(items) # copy is [1, 2, 3], items is [1, 2, 3]  
random.shuffle(copy) # copy might become [3, 1, 2], items is still [1, 2, 3]  
# return copy

## 2.4 Usage Notes

* **Non-Mutating:** The primary feature of this function is that it does not modify the original list passed as an argument. It returns a new, shuffled list.
* **Dependency:** This function requires the random module to be imported in the script where it is defined.
* **Randomness:** The order of elements in the returned list is random. Calling the function multiple times with the same input list will likely result in different outputs.
* **Data Integrity:** The returned list will contain the exact same elements as the input list, only in a different order. No elements are added or removed.

**Output Example**: A possible return value for an input of [1, 2, 3, 4, 5].

[4, 1, 5, 3, 2]

## 2.5 Example

The following example demonstrates how to use shuffle\_copy and confirms that the original list remains unchanged. Note that the random module must be imported for this function to work.

import random  
from typing import List  
  
# Definition of the function as provided  
def shuffle\_copy(items: List[int]) -> List[int]:  
 copy = list(items)  
 random.shuffle(copy)  
 return copy  
  
# Example usage  
original\_numbers = [10, 20, 30, 40, 50]  
shuffled\_numbers = shuffle\_copy(original\_numbers)  
  
print(f"Original List: {original\_numbers}")  
print(f"Shuffled Copy: {shuffled\_numbers}")  
  
# Verify the original list is unchanged  
print(f"Original list is the same: {original\_numbers == [10, 20, 30, 40, 50]}")

**Output:**

Original List: [10, 20, 30, 40, 50]  
Shuffled Copy: [40, 10, 50, 20, 30]  
Original list is the same: True

*(Note: The order of elements in “Shuffled Copy” will vary with each execution.)*