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 a single item will be randomly selected.

## 1.3 Description

This function provides a simple way to get a random element from a list. The core logic is built around Python’s standard random module.

First, the function validates the input items list. It checks if the list is empty using the condition if not items:. 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 check is crucial to prevent errors from the underlying random.choice function, which cannot operate on an empty sequence.

If the list is not empty, the function proceeds to call random.choice(items). This method from the random library takes a non-empty sequence as an argument and returns a single item chosen uniformly at random. The selected string is then returned as the result of the choose\_random\_item function.

# Internal logic for choosing an item  
import random  
# Assuming items = ['apple', 'banana', 'cherry']  
chosen\_item = random.choice(items)  
# chosen\_item will be one of 'apple', 'banana', or 'cherry'

## 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 script where this function is called.
* The selection is uniformly random, meaning every item in the list has an equal probability of being chosen.

**Output Example**: A single string from the input list. For an input list ['apple', 'banana', 'cherry'], a possible output 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."""  
 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 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 specific fruit chosen in the output will vary with each execution due to its random nature.)

## 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 without modifying the original input list.

## 2.2 parameters

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

## 2.3 Description

This function provides a non-destructive way to shuffle the elements of a list. The internal logic operates in three steps:

1. First, it creates a shallow copy of the input items list by calling list(items). This ensures that any subsequent modifications do not affect the original list provided by the user.
2. Next, it uses the random.shuffle() method to shuffle the newly created copy list. The random.shuffle() function modifies the sequence in-place, rearranging its elements into a random order.
3. Finally, the function returns the copy, which now contains the same elements as the original items list but in a new, randomized order.

# Internal logic example  
import random  
original\_list = [10, 20, 30]  
copy = list(original\_list) # copy is [10, 20, 30]  
random.shuffle(copy) # copy might become [20, 30, 10]  
# The function returns the 'copy'

## 2.4 Usage Notes

* This function is guaranteed not to mutate the original items list. It is a “safe” operation for shuffling.
* The function requires the random module to be imported in the script where it is used.
* Although the type hint specifies List[int], the function works correctly with lists containing elements of any data type (e.g., str, float, objects).

**Output Example**: The returned list will have the same length and elements as the input list. For an input of [1, 2, 3], a possible output is [3, 1, 2].

## 2.5 Example

import random  
  
# Define the function (assuming it's in the same file or imported)  
def shuffle\_copy(items: list) -> list:  
 copy = list(items)  
 random.shuffle(copy)  
 return copy  
  
# Example usage  
original\_numbers = [1, 2, 3, 4, 5, 6]  
print(f"Original list: {original\_numbers}")  
  
shuffled\_list = shuffle\_copy(original\_numbers)  
print(f"Shuffled copy: {shuffled\_list}")  
  
print(f"Original list after function call: {original\_numbers}")

**Output:**

Original list: [1, 2, 3, 4, 5, 6]  
Shuffled copy: [4, 1, 6, 3, 5, 2]  
Original list after function call: [1, 2, 3, 4, 5, 6]

*(Note: The order of elements in “Shuffled copy” will vary with each execution due to its random nature.)*