Table of Contents

## 0.1 FunctionDef choose\_random\_item(items)

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

## 1.1 Overview

The choose\_random\_item function selects and returns a single string at random from a given non-empty list of strings.

## 1.2 parameters

* **items**: List[str] - A list of strings from which one item will be randomly selected. This list must not be empty.

## 1.3 Description

This function provides a safe way to choose a random element from a list of strings. The core logic is built around Python’s random.choice() method, which is designed to pick a single item uniformly at random from a non-empty sequence.

Before attempting to select an item, the function first performs a validation check: if not items:. This conditional statement verifies if the provided items list is empty. If it is, the function immediately raises a ValueError with the message “items must not be empty”. This proactive error handling prevents the IndexError that random.choice() would otherwise raise and provides a more descriptive error to the developer.

If the list is not empty, the function proceeds to execute return random.choice(items), which returns one of the strings from the list. Each string in the list has an equal probability of being chosen.

## 1.4 Usage Notes

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

**Output Example**: A single string from the input list. For an input of ['apple', 'banana', 'cherry'], a possible output is "banana".

## 1.5 Example

### 1.5.1 Example 1: Choosing from a valid list

import random  
from typing import List  
  
# Definition of the function  
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"]  
random\_fruit = choose\_random\_item(fruits)  
print(f"A random fruit: {random\_fruit}")

**Output:**

A random fruit: cherry

*(Note: The actual output will be one of the items from the fruits list, chosen randomly.)*

### 1.5.2 Example 2: Attempting to choose from an empty list

import random  
from typing import List  
  
# Definition of the function  
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 with an empty list  
empty\_list = []  
try:  
 choose\_random\_item(empty\_list)  
except ValueError as e:  
 print(e)

**Output:**

items must not be empty

## 1.6 FunctionDef shuffle\_copy(items)

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

## 2.1 Overview

The shuffle\_copy function returns a new, randomly shuffled copy of a given list, ensuring the original list remains unchanged.

## 2.2 parameters

* items: A list of integers (List[int]) that you want to create a shuffled copy of.

## 2.3 Description

This function provides a safe way to shuffle a list without modifying the original data structure, a concept known as non-mutating behavior. The logic proceeds in three simple steps:

1. A shallow copy of the input items list is created using copy = list(items). This is the crucial step that prevents mutation of the original list.
2. The random.shuffle() method is then called on the copy. This function shuffles the elements of the copy 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.

def shuffle\_copy(items: List[int]) -> List[int]:  
 """Return a shuffled copy of the given list without mutating the input.  
  
 Parameters:  
 items: A list of integers.  
  
 Returns:  
 A new list containing the same integers in random order.  
 """  
 copy = list(items)  
 random.shuffle(copy)  
 return copy

## 2.4 Usage Notes

* The primary benefit of this function is that it is non-mutating. The original list passed as the items argument will not be altered.
* This function depends on Python’s built-in random module. Ensure it is imported in your script (e.g., import random) before calling shuffle\_copy.
* While the type hint specifies List[int], the function’s logic will work correctly with lists containing other data types, such as strings, floats, or mixed types.

**Output Example**: A new list with the same elements as the input list, but in a randomized order. [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 after the function call.

import random  
  
# Assume shuffle\_copy is defined in the same scope  
  
# --- Example Usage ---  
original\_numbers = [1, 2, 3, 4, 5]  
print(f"Original List (before): {original\_numbers}")  
  
shuffled\_numbers = shuffle\_copy(original\_numbers)  
  
print(f"Original List (after): {original\_numbers}")  
print(f"Shuffled Copy: {shuffled\_numbers}")

**Output:**

(Note: The exact order of the “Shuffled Copy” will vary with each execution due to its random nature.)

Original List (before): [1, 2, 3, 4, 5]  
Original List (after): [1, 2, 3, 4, 5]  
Shuffled Copy: [3, 5, 1, 2, 4]