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, uniformly random string from a provided list of strings.

## 1.2 parameters

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

## 1.3 Description

This function is designed to safely select one random element from a list. It first performs a validation check to ensure the input list items is not empty. If the list is found to be empty using the if not items: condition, the function immediately stops execution and raises a ValueError with the message “items must not be empty”. This prevents errors that would occur when trying to choose from an empty collection.

If the list is valid (i.e., contains at least one item), the function then utilizes random.choice(items) to perform the selection. The random.choice() method, from Python’s random module, returns a single item chosen uniformly at random from the input sequence.

# Internal logic for a non-empty list  
return random.choice(items)

## 1.4 Usage Notes

* The input list items must contain at least one element. 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 scope where choose\_random\_item is defined.
* Each item in the list has an equal probability of being selected.

**Output Example**: For an input of ['red', 'green', 'blue'], a possible output is 'green'. The actual output will be one of the strings from the list, chosen randomly.

## 1.5 Example

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 1: Choosing from a populated list  
options = ["apple", "banana", "cherry", "date"]  
random\_selection = choose\_random\_item(options)  
print(f"Randomly selected item: {random\_selection}")  
  
# Example 2: Attempting to choose from an empty list  
try:  
 empty\_list = []  
 choose\_random\_item(empty\_list)  
except ValueError as e:  
 print(f"Caught expected error: {e}")

**Output:**

Randomly selected item: cherry  
Caught expected error: items must not be empty

*(Note: The selected item in the first line is random and may differ on each execution.)*

## 1.6 FunctionDef shuffle\_copy

# 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: List[int] - The list of integers to be shuffled.

## 2.3 Description

This function provides a safe way to shuffle a list without altering the original data structure. The core logic involves three main steps:

1. A shallow copy of the input items list is created using copy = list(items). This is a critical step that allocates new memory for a new list, preventing any modifications to the original list passed to the function.
2. The random.shuffle(copy) function is then called. This function, from Python’s standard random module, shuffles the elements of the copy list *in-place*. It rearranges the sequence of elements into a random permutation.
3. Finally, the function returns the copy, which now contains the same elements as the original items list but in a new, random order.

# The function relies on the 'random' module  
import random  
  
# Step 1: Create a copy  
original = [1, 2, 3]  
copy = list(original) # copy is now [1, 2, 3], but a separate object  
  
# Step 2: Shuffle the copy in-place  
random.shuffle(copy) # copy might now be [3, 1, 2]  
  
# Step 3: Return the shuffled copy  
# The function would return [3, 1, 2] while 'original' remains [1, 2, 3]

## 2.4 Usage Notes

* The primary benefit of this function is that it is non-mutating. The original list you pass as an argument will not be modified.
* This function requires the random module to be imported into the script where it is used (i.e., import random).
* Although the type hint specifies List[int], the function will work correctly with lists containing elements of any type (e.g., strings, floats, or other objects) as random.shuffle is generic.

**Output Example**: The function returns a new list. For an input of [1, 2, 3], a possible return value could be [2, 3, 1]. The order is random and will likely be different on each execution.

## 2.5 Example

import random  
from typing import List  
  
def shuffle\_copy(items: List[int]) -> List[int]:  
 """Return a shuffled copy of the given list without mutating the input."""  
 copy = list(items)  
 random.shuffle(copy)  
 return copy  
  
# Example usage  
original\_list = [10, 20, 30, 40, 50]  
print(f"Original list before shuffling: {original\_list}")  
  
shuffled\_list = shuffle\_copy(original\_list)  
print(f"Shuffled list (new): {shuffled\_list}")  
print(f"Original list after shuffling: {original\_list}")

**Output:**

Original list before shuffling: [10, 20, 30, 40, 50]  
Shuffled list (new): [30, 50, 10, 20, 40]  
Original list after shuffling: [10, 20, 30, 40, 50]