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, uniformly random item from a given list of strings.

## 1.2 parameters

* items (List[str]): A non-empty list of strings from which to choose a random item.

## 1.3 Description

This function provides a safe way to select a random element from a list. It first performs a validation check to ensure the input list items is not empty. If the list is empty, the function raises a ValueError with the message "items must not be empty" to prevent runtime errors from downstream operations.

If the list contains one or more elements, the function then utilizes the random.choice() method from Python’s random module. This method selects a single item from the sequence at random, where each item has an equal probability of being chosen. The selected string is then returned as the result.

# Internally, the function works like this:  
import random  
  
def choose\_random\_item(items: List[str]) -> str:  
 # 1. Check if the list is empty  
 if not items:  
 # 2. Raise an error if it is  
 raise ValueError("items must not be empty")  
 # 3. If not empty, use random.choice to select and return an item  
 return random.choice(items)

## 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 environment where this function is used.
* The type hint specifies the input should be a list of strings (List[str]). While random.choice works on any non-empty sequence, this function’s explicit contract is for a list of strings.

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

## 1.5 Example

# This example requires the 'random' module to be imported.  
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)  
  
# --- Usage Example 1: Choosing from a populated list ---  
options = ["red", "green", "blue", "yellow"]  
random\_color = choose\_random\_item(options)  
print(f"The chosen color is: {random\_color}")  
  
# --- Usage Example 2: Handling an empty list ---  
empty\_list = []  
try:  
 choose\_random\_item(empty\_list)  
except ValueError as e:  
 print(f"Caught expected error: {e}")

**Output:**

The chosen color is: blue  
Caught expected error: items must not be empty

*(Note: The actual color chosen in the first part 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 new list containing the same elements as the input list but in a randomized order, without modifying the original list.

## 2.2 parameters

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

## 2.3 Description

This function provides a safe way to shuffle a list without causing side effects. The logic proceeds in three steps:

1. A shallow copy of the input items list is created using copy = list(items). This is the key step that ensures the original list passed to the function remains unchanged.
2. The standard library’s random.shuffle() function is then called on the copy. This function shuffles the elements of the list *in-place*, rearranging the copy into a random permutation.
3. Finally, the function returns the modified copy, which is now a shuffled version of the original list.

# The function depends on Python's built-in 'random' module.  
import random  
  
# Example of internal logic  
original\_items = [10, 20, 30, 40]  
copy\_of\_items = list(original\_items) # copy\_of\_items is now [10, 20, 30, 40]  
random.shuffle(copy\_of\_items) # copy\_of\_items might now be [30, 10, 40, 20]  
# The function would then return copy\_of\_items

## 2.4 Usage Notes

* **Non-mutating:** The primary feature of this function is that it does not alter the original input list. It operates on a copy.
* **Dependency:** This function requires the random module to be imported in the script where it is defined.
* **Randomness:** The shuffle is pseudo-random. For reproducible results during testing, you can set the seed for the random number generator using random.seed() before calling this function.

**Output Example**: A new list object containing the same elements as the input list, but in a random order.

[4, 1, 5, 2, 3]

## 2.5 Example

import random  
  
def shuffle\_copy(items: list) -> list:  
 """Return a shuffled copy of the given list without mutating the input."""  
 copy = list(items)  
 random.shuffle(copy)  
 return copy  
  
# Example usage  
original\_list = [1, 2, 3, 4, 5, 6]  
print(f"Original list (before): {original\_list}")  
  
shuffled\_list = shuffle\_copy(original\_list)  
print(f"Shuffled list (new): {shuffled\_list}")  
print(f"Original list (after): {original\_list}")

**Output:**

Original list (before): [1, 2, 3, 4, 5, 6]  
Shuffled list (new): [4, 1, 6, 3, 5, 2]  
Original list (after): [1, 2, 3, 4, 5, 6]

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