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 item chosen uniformly at random from a provided 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 simple way to get a random element from a sequence. The logic proceeds in two steps:

1. First, it performs a validation check: if not items:. This conditional statement evaluates whether the input list items is empty.
2. If the list is empty, the function immediately stops execution and raises a ValueError with the message “items must not be empty”. This prevents errors from the underlying random.choice function, which cannot operate on an empty sequence.
3. If the list is not empty, the function calls random.choice(items). This is a standard Python library function that takes a non-empty sequence and returns a randomly selected element. The selection is uniform, meaning every item has an equal chance of being picked.

The single, randomly chosen string is then returned as the result.

# Internal logic for choosing an item  
import random  
items = ["apple", "banana", "cherry"]  
# The function will return one of these three strings with equal probability.  
chosen\_item = random.choice(items)

## 1.4 Usage Notes

* This function requires that the random module be imported into the script’s scope.
* Providing an empty list as the items parameter will result in a ValueError. Always ensure the list contains at least one element before calling this function.
* The selection is uniformly random. Each item in the list has an equal probability of being chosen on any given call.

**Output Example**: The function returns a single string.

"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  
available\_colors = ["red", "green", "blue", "yellow", "purple"]  
random\_color = choose\_random\_item(available\_colors)  
print(f"The chosen color is: {random\_color}")  
  
# Example of what happens with an empty list  
try:  
 choose\_random\_item([])  
except ValueError as e:  
 print(f"Error caught: {e}")

**Output:**

The chosen color is: blue  
Error caught: items must not be empty

*(Note: The chosen color will vary with each execution as it is selected randomly.)*

## 1.6 FunctionDef shuffle\_copy

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

## 2.1 Overview

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

## 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 without altering the original data structure, a concept known as non-mutation. The process is straightforward and involves three main steps:

1. A shallow copy of the input items list is created using the list() constructor. This new list is assigned to the variable copy. This step is crucial as it ensures that any subsequent operations are performed on the new list, leaving the original items list untouched.
2. The random.shuffle() function is then called on the copy. This function, part of Python’s standard random module, rearranges the elements of the copy list into a random order. It performs the shuffle operation in-place, meaning it directly modifies the copy list.
3. Finally, the function returns the copy list, which now contains the same elements as the original list but in a new, randomized order.

# Step 1: Create a copy  
copy = list(items)  
# Step 2: Shuffle the copy in-place  
random.shuffle(copy)  
# Step 3: Return the shuffled copy  
return copy

## 2.4 Usage Notes

* **Non-Mutating**: The primary advantage of this function is that it does not modify (mutate) the input list. The original list you pass as an argument will remain in its original order after the function call.
* **Dependency**: This function relies on the random.shuffle() method, which requires Python’s random module to be imported into the script.
* **Type Generality**: Although the type hint specifies List[int], the function’s logic will work correctly with a list containing elements of any type (e.g., strings, floats, or mixed types).

**Output Example**: A new list with the same elements as the input list, but in a random sequence. For an input of [1, 2, 3, 4, 5], a possible output is:

[4, 1, 5, 3, 2]

## 2.5 Example

import random  
from typing import List  
  
# The function assumes the 'random' module is imported.  
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  
my\_numbers = [10, 20, 30, 40, 50]  
shuffled\_numbers = shuffle\_copy(my\_numbers)  
  
print("Original List (unchanged):")  
print(my\_numbers)  
  
print("\nShuffled Copy:")  
print(shuffled\_numbers)

**Output:**

Original List (unchanged):  
[10, 20, 30, 40, 50]  
  
Shuffled Copy:  
[30, 50, 10, 20, 40]

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