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, uniformly random item from a given 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 provides a safe way to select a random element from a list. The core logic is divided into two main steps: validation and selection.

First, the function validates the input list items by checking if it is empty with the condition if not items:. If the list is empty, it is impossible to choose an item, so the function raises a ValueError with the message “items must not be empty”. This prevents potential errors from occurring in the selection process.

If the list is not empty, the function proceeds to the selection step. It utilizes the random.choice() method from Python’s random module to pick a single element from the items list. The random.choice() method ensures that each item in the list has an equal probability of being selected. The chosen string is then returned as the result.

# Internal logic example  
import random  
  
def choose\_random\_item(items: List[str]) -> str:  
 # 1. Validate that the list is not empty  
 if not items:  
 raise ValueError("items must not be empty")  
 # 2. Return a random choice from the list  
 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 random module. Ensure it is available in the environment.
* The selection is uniformly random, meaning every item in the provided list has an equal chance of being chosen.

**Output Example**: The function returns a single string that was present in the input items list. For an input of ['red', 'green', 'blue'], a possible return value is 'green'.

## 1.5 Example

# Example usage  
import random # This import is necessary for the function to work  
  
fruit\_list = ["apple", "banana", "cherry", "date"]  
random\_fruit = choose\_random\_item(fruit\_list)  
print(f"The randomly chosen fruit is: {random\_fruit}")  
  
# Example of what happens with an empty list  
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 chosen fruit in the first line of the output is random and will vary with each execution.)

## 1.6 FunctionDef shuffle\_copy(items)

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

## 2.1 Overview

The shuffle\_copy function creates and returns a new list containing the elements of the input list in a randomized order, without altering the original list.

## 2.2 parameters

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

## 2.3 Description

This function provides a safe way to get a shuffled version of a list while preserving the original. The process is as follows:

1. A shallow copy of the input items list is created using copy = list(items). This is a crucial step to prevent modification of the original list that was passed as an argument.
2. The random.shuffle() function is then called on the copy. This function shuffles the elements of the copy list in-place, rearranging them into a random sequence.
3. Finally, the function returns the modified copy, which is now a shuffled version of the original items list.

# Internal logic for an input of [1, 2, 3]  
original\_list = [1, 2, 3]  
copy\_of\_list = list(original\_list) # copy\_of\_list is now [1, 2, 3]  
# random.shuffle(copy\_of\_list) modifies it in-place, e.g., to [3, 1, 2]  
# The function then returns the shuffled copy\_of\_list

## 2.4 Usage Notes

* **Non-Mutating:** The primary feature of this function is that it does not change the input items list. It operates on a copy, leaving the original data intact.
* **Random Module Dependency:** This function depends on Python’s built-in random module. Ensure it is imported (e.g., import random) in the file where this function is used.
* **Shallow Copy:** The function creates a shallow copy. If the list contains mutable objects (like other lists or dictionaries), the objects themselves are not duplicated, only their references. For a list of simple types like integers, this behaves as a full, independent copy.

**Output Example**: A possible return value for an input of [1, 2, 3, 4, 5] could be:

[3, 5, 1, 4, 2]

## 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\_numbers = [10, 20, 30, 40, 50]  
shuffled\_numbers = shuffle\_copy(original\_numbers)  
  
print(f"Original List: {original\_numbers}")  
print(f"Shuffled Copy: {shuffled\_numbers}")  
  
# The original list remains unchanged  
assert original\_numbers == [10, 20, 30, 40, 50]

**Output:**

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

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