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 random item from a given 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 safe way to select a random element from a list of strings. The core logic involves two main steps:

1. **Validation**: The function first checks if the provided items list is empty using the condition if not items:. This is a crucial safeguard to prevent errors in the subsequent steps. If the list is empty, the function raises a ValueError with the message “items must not be empty”, immediately halting execution and informing the user of the invalid input.
2. **Random Selection**: If the list is not empty, the function proceeds to use the random.choice(items) method. This method, from Python’s standard random library, takes a non-empty sequence as input and returns a single element chosen uniformly at random. This means every item in the items list has an equal probability of being selected.

The randomly selected string is then returned as the result of the function.

# Internal logic example  
import random  
from typing import List  
  
def choose\_random\_item(items: List[str]) -> str:  
 # Step 1: Validate that the list is not empty  
 if not items:  
 raise ValueError("items must not be empty")  
 # Step 2: Return a single random item 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 built-in random module. Ensure it is imported in the environment where this function is used (e.g., import random).
* The selection is uniformly random, meaning each item in the list has an equal chance of being chosen on any given call.

**Output Example**: A single string element from the input list. For an input of ["red", "green", "blue"], a possible return value is "green".

## 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 usage  
colors = ["red", "green", "blue", "yellow", "purple"]  
random\_color = choose\_random\_item(colors)  
print(f"The randomly chosen color is: {random\_color}")  
  
# Example of error handling  
try:  
 empty\_list = []  
 choose\_random\_item(empty\_list)  
except ValueError as e:  
 print(f"Error caught: {e}")

**Output:**

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

*(Note: The chosen color in the first line of the output will vary with each execution, as it is selected randomly.)*

### 1.5.1 FunctionDef choose\_random\_item

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

## 2.1 Overview

The choose\_random\_item function randomly selects and returns a single string element from a given list of strings.

## 2.2 parameters

* items: List[str] - A list of strings from which one item will be randomly chosen.

## 2.3 Description

This function is designed to perform a random selection from a collection of items. It accepts a single argument, items, which is expected to be a list of strings.

Internally, the function utilizes a random selection algorithm to pick one element from the input items list. The most common implementation in Python would involve using the random.choice() method from the standard random library, which is purpose-built for this task. After selecting an item, the function returns it as a string.

A typical implementation would look like this:

import random  
from typing import List  
  
def choose\_random\_item(items: List[str]) -> str:  
 # Ensure the list is not empty to avoid errors  
 if not items:  
 raise ValueError("The input list cannot be empty.")  
   
 # Choose and return a random item from the list  
 return random.choice(items)

## 2.4 Usage Notes

* The input list items must not be empty. Passing an empty list will result in a ValueError or IndexError, depending on the implementation.
* The function is non-deterministic. Calling it multiple times with the same list will likely produce different results.
* As per the type hint List[str], the function is specifically designed to work with lists of strings.

## 2.5 Example

# Example usage  
# Note: You would need to import the 'random' module for a working implementation.  
import random  
  
# A sample implementation for demonstration  
def choose\_random\_item(items: list) -> str:  
 if not items:  
 return "List is empty"  
 return random.choice(items)  
  
available\_colors = ["Red", "Green", "Blue", "Yellow", "Purple"]  
selected\_color = choose\_random\_item(available\_colors)  
print(f"The randomly selected color is: {selected\_color}")

**Output:**

The randomly selected color is: Blue

(Note: The actual output will be a randomly selected item from the available\_colors list and may differ on each execution.)

## 2.6 FunctionDef shuffle\_copy

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

## 3.1 Overview

The shuffle\_copy function returns a new, randomly shuffled copy of a given list without altering the original list.

## 3.2 parameters

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

## 3.3 Description

This function provides a safe way to shuffle a list by ensuring the original data structure remains untouched. The process is executed in three main steps:

1. A shallow copy of the input items list is created using list(items). This new list is stored in a variable named copy. This step is crucial for preventing mutation of the original list passed to the function.
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 holds the same elements as the original list but in a new, randomized sequence.

# Internal logic  
copy = list(items)  
random.shuffle(copy)  
return copy

## 3.4 Usage Notes

* This function is non-mutating, meaning the original list you pass as an argument will not be changed.
* The function relies on Python’s built-in random module. Ensure this module is imported in your script before calling the function.
* Due to the nature of random shuffling, calling the function multiple times with the same input list will likely produce a different-ordered list each time.

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

[4, 1, 5, 2, 3]

## 3.5 Example

import random  
from typing import List  
  
# The function must be defined or 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  
original\_numbers = [10, 20, 30, 40, 50]  
shuffled\_numbers = shuffle\_copy(original\_numbers)  
  
print(f"Original List (unchanged): {original\_numbers}")  
print(f"Shuffled Copy: {shuffled\_numbers}")

**Output:**

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

*(Note: The actual order of elements in the “Shuffled Copy” will vary with each execution.)*

### 3.5.1 FunctionDef shuffle\_copy

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

## 4.1 Overview

The shuffle\_copy function creates a shuffled copy of a list of integers, leaving the original list unchanged.

## 4.2 parameters

* items: List[int] - The input list of integers to be copied and shuffled.

## 4.3 Description

This function provides a non-destructive way to shuffle a list. It operates by first creating a shallow copy of the input items list. This initial step is crucial as it ensures that the original list passed to the function is not altered.

After creating the copy, the function shuffles the elements of the new list in-place, rearranging them into a random order. Finally, it returns this newly created and shuffled list. The original items list remains in its initial order.

import random  
  
def shuffle\_copy(items: List[int]) -> List[int]:  
 """  
 Creates a shuffled copy of a list.  
 """  
 items\_copy = items.copy()  
 random.shuffle(items\_copy)  
 return items\_copy

## 4.4 Usage Notes

* This function does not modify the original list. It returns a new, shuffled list.
* The shuffling is pseudo-random. For reproducible results, you can seed Python’s random number generator using random.seed() before calling this function.
* Although the type hint specifies List[int], the underlying logic works with lists containing elements of any type.

## 4.5 Example

# Example usage  
import random  
  
# For reproducible results in this example  
random.seed(42)  
  
original\_list = [1, 2, 3, 4, 5, 6]  
shuffled\_list = shuffle\_copy(original\_list)  
  
print("Original List:", original\_list)  
print("Shuffled Copy:", shuffled\_list)

**Output:**

Original List: [1, 2, 3, 4, 5, 6]  
Shuffled Copy: [1, 4, 3, 6, 2, 5]