Table of Contents

## 1 FunctionDef num(a, b)

## 2 Function: num

The function of **num** is to calculate and return the sum of two input values.

**parameters**: The parameters of this Function. · a: The first value to be used in the addition. · b: The second value to be used in the addition.

**Code Description**: The function num is defined to accept two arguments, a and b. It performs a single operation: adding a and b together using the + operator. The result of this addition is then immediately returned by the function.

**Note**: This function will work with any two types that support the addition (+) operator, such as integers, floats, or strings (for concatenation). If incompatible types are provided, a TypeError will be raised.

**Output Example**:

# Calling num(2, 3) would return:  
5

## 3 FunctionDef generate\_random\_integers(count, start, end)

## 4 Function: generate\_random\_integers

The function of **generate\_random\_integers** is to return a list of a specified number of pseudo-random integers within a defined inclusive range.

**parameters**: The parameters of this Function. · count: An integer specifying the number of random integers to generate. · start: An optional integer representing the inclusive lower bound for the random values, defaulting to 0. · end: An optional integer representing the inclusive upper bound for the random values, defaulting to 100.

**Code Description**: The function begins by checking if the count parameter is a negative number. If it is, a ValueError is raised with the message “count must be non-negative”. Next, it checks if the start value is greater than the end value. If this condition is true, the function swaps the values of start and end to ensure the range is always valid. Finally, it uses a list comprehension to generate the random numbers. It iterates count times, and in each iteration, it calls random.randint(start, end) to produce a single pseudo-random integer within the inclusive range of [start, end]. The function then returns the complete list of generated integers.

**Note**: This function depends on Python’s built-in random module. If the start parameter is provided with a value greater than the end parameter, the function will automatically correct the range by swapping them instead of raising an error. The function will raise a ValueError if a negative value is provided for the count parameter.

**Output Example**: A call to generate\_random\_integers(5, 1, 10) might return a list similar to the following (the actual values will vary with each execution):

[3, 9, 1, 7, 5]

## 5 FunctionDef choose\_random\_item(items)

## 6 Function: choose\_random\_item

The function of **choose\_random\_item** is to choose and return a single random string from a non-empty list of strings.

**parameters**: The parameters of this Function. · items: A list of strings from which a random item will be selected.

**Code Description**: The function first checks if the provided items list is empty. If it is, the function raises a ValueError with the message “items must not be empty”. If the list is not empty, the function uses random.choice() to select a single element uniformly at random from the items list and returns that element.

**Note**: This function requires the random module to be imported. It will raise a ValueError if an empty list is passed as an argument.

**Output Example**:

# Given the input list: ['red', 'green', 'blue']  
# A possible return value could be:  
"green"

## 7 FunctionDef shuffle\_copy(items)

## 8 Function: shuffle\_copy

The function of **shuffle\_copy** is to return a new list containing the elements of the input list in a randomized order, without altering the original list.

**parameters**: The parameters of this Function. · items: A list of integers that will be copied and shuffled.

**Code Description**: The function begins by creating a shallow copy of the input list items and assigning it to a new variable named copy. This is done to prevent modification of the original list. It then uses the random.shuffle() method to rearrange the elements of the copy list into a random order in-place. Finally, the function returns the shuffled copy list.

**Note**: This function requires the random module to be imported to work correctly. The original items list passed to the function will remain unchanged.

**Output Example**:

# Given an input list like [1, 2, 3, 4, 5]  
# A possible return value could be:  
[3, 5, 1, 4, 2]