Table of Contents

## 1 FunctionDef num(a, b)

**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 added. · b: The second value to be added.

**Code Description**: The function num is defined to accept two parameters, a and b. It performs an addition operation on these two parameters using the + operator and returns the resulting sum.

**Note**: This function is designed for numeric types like integers or floats. If used with strings, it will perform concatenation instead of mathematical addition. Using incompatible types (e.g., an integer and a string) will raise a TypeError.

**Output Example**: Calling num(2, 3) returns 5. ## FunctionDef generate\_random\_integers(count, start, end) **generate\_random\_integers**: The function of generate\_random\_integers is to return a list of a specified quantity of pseudo-random integers within an inclusive range.

**parameters**: The parameters of this Function. · count: The number of integers to be generated. · start: The inclusive lower bound for the generated values, with a default of 0. · end: The inclusive upper bound for the generated values, with a default of 100.

**Code Description**: The function first validates the count parameter. If count is a negative number, it raises a ValueError. Next, it checks if the start value is greater than the end value. If it is, the function swaps them to ensure the range is always valid. Finally, it uses a list comprehension to generate a list of integers. This is done by calling the random.randint(start, end) function count times, which produces a random integer within the inclusive range of start and end for each iteration. The resulting list of random integers is then returned.

**Note**: This function requires the random module to be imported. The range defined by start and end is inclusive, meaning both start and end can appear in the output. The function automatically handles cases where the provided start value is greater than the end value by swapping them.

**Output Example**: A call to generate\_random\_integers(5, 1, 10) could return a list similar to [3, 9, 1, 10, 5]. ## FunctionDef choose\_random\_item(items) **choose\_random\_item**: The function of choose\_random\_item is to select and return a single random string from a non-empty list.

**parameters**: The parameters of this Function. · items: A list of strings to choose from.

**Code Description**: The function choose\_random\_item takes a single parameter, items, which is expected to be a list of strings. It first checks if the provided items list is empty. If it is empty, 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. An empty list will cause a ValueError to be raised.

**Output Example**: If items is ['red', 'green', 'blue'], a possible return value is 'green'. ## FunctionDef shuffle\_copy(items) **shuffle\_copy**: The function of shuffle\_copy is to return a shuffled copy of the given list without mutating the input.

**parameters**: The parameters of this Function. · items: A list of integers.

**Code Description**: The function first creates a shallow copy of the input list items and stores it in a new variable named copy. It then calls the random.shuffle() method, passing the copy to it. This shuffles the elements of the copy list in place, arranging them in a random order. Finally, the function returns the shuffled copy.

**Note**: This function depends on Python’s random module, which must be imported for the code to work. The original list passed as the items parameter remains unchanged.

**Output Example**: If items is [1, 2, 3, 4], a possible return value could be [3, 1, 4, 2].