**Knowledge Transfer Document: Random Data Generator Utility**

**Objective**

This document outlines a utility designed to generate random data for use in testing or other applications. The utility provides methods to generate random strings and random numbers based on the desired length, allowing for dynamic data creation.

**Utility Overview**

* **Purpose:**
  + Dynamically generate random data for testing purposes.
  + Avoid hardcoding test data in scripts by generating it programmatically.
* **Methods:**
  + **Get\_random\_string**: Generates a random string of the specified length.
  + **Get\_random\_number**: Generates a random number with the specified number of digits.

**Implementation Details**

**Folder Structure**

The utility is assumed to be part of the following project folder structure:

bash

Copy code

Project/

├── Utilities/

│ └── random\_data\_generator.py # The utility script

├── Tests/

**Python Utility: random\_data\_generator.py**

python

Copy code

import random

import string

# Method 1: Get Random String

def get\_random\_string(length):

"""

Generate a random string of the specified length.

Args:

length (int): The desired length of the random string.

Returns:

str: Randomly generated string.

"""

if length <= 0:

raise ValueError("Length must be a positive integer.")

random\_string = ''.join(random.choices(string.ascii\_letters + string.digits, k=length))

return random\_string

# Method 2: Get Random Number

def get\_random\_number(length):

"""

Generate a random number with the specified number of digits.

Args:

length (int): The desired number of digits for the random number.

Returns:

str: Randomly generated number as a string to preserve leading zeros.

"""

if length <= 0:

raise ValueError("Length must be a positive integer.")

random\_number = ''.join(random.choices(string.digits, k=length))

return random\_number

# Example Usage

if \_\_name\_\_ == "\_\_main\_\_":

print("Random String:", get\_random\_string(10))

print("Random Number:", get\_random\_number(8))

**How to Use**

**Import the Utility**

To use the utility in your project, import the random\_data\_generator module:

python

Copy code

from Utilities.random\_data\_generator import get\_random\_string, get\_random\_number

**Call Methods**

1. **Get Random String**
   * Call the get\_random\_string method and provide the desired length as a parameter.
   * Example:

python

Copy code

random\_string = get\_random\_string(12)

print(random\_string) # Outputs a random string of 12 characters.

1. **Get Random Number**
   * Call the get\_random\_number method and provide the desired length as a parameter.
   * Example:

python

Copy code

random\_number = get\_random\_number(6)

print(random\_number) # Outputs a random 6-digit number.

**Execution Workflow**

1. **Input Parameters:**
   * **Get\_random\_string**: Accepts the length of the string to be generated.
   * **Get\_random\_number**: Accepts the number of digits for the random number.
2. **Output:**
   * Returns the generated random string or number as a string.

**Example Output**

1. **Random String:**  
   Input: get\_random\_string(8)  
   Output: aB3fX2pQ
2. **Random Number:**  
   Input: get\_random\_number(5)  
   Output: 39425

**Advantages**

* **Dynamic Data:** Removes the need for hardcoding data, allowing tests to be more flexible.
* **Versatility:** Useful for generating unique identifiers, test data, or placeholders in scripts.
* **Ease of Use:** Requires minimal input parameters and provides instant results.

**Best Practices**

1. **Validate Inputs:**  
   Ensure the length provided is a positive integer to avoid errors.
2. **Use in Test Scenarios:**  
   Leverage this utility for creating unique values in testing scenarios, such as random usernames, passwords, or identifiers.
3. **Avoid Overloading:**  
   Use realistic lengths for random data to ensure compatibility with test cases or databases.

**Conclusion**

The **Random Data Generator Utility** simplifies the creation of dynamic test data, making it a valuable addition to any testing framework. With its capability to generate both random strings and numbers, it ensures flexibility and ease of use.