**Knowledge Transfer Document: Date Utility Functions**

**Objective**

This document outlines utility functions designed to calculate various dates relative to the current system date. These functions include methods to get today's date, yesterday's date, tomorrow's date, six months past, and six months future dates. The date format is customizable, as provided by the user.

**Utility Overview**

* **Purpose:**
  + Provides a set of functions to generate dates relative to the current system date.
  + Supports custom date formatting for all outputs.
* **Supported Methods:**
  + **Get\_today\_date**: Returns today's date in the specified format.
  + **Get\_yesterday\_date**: Returns yesterday's date in the specified format.
  + **Get\_tomorrow\_date**: Returns tomorrow's date in the specified format.
  + **Get\_six\_months\_past\_date**: Returns the date six months in the past in the specified format.
  + **Get\_six\_months\_future\_date**: Returns the date six months in the future in the specified format.

**Implementation Details**

**Folder Structure**

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

bash

Copy code

Project/

├── Utilities/

│ └── date\_utilities.py # The utility script

├── Tests/

**Python Utility: date\_utilities.py**

Code paste

### ****How to Use****

#### ****Import the Utility****

**Execution Workflow**

1. **Input Parameters:**
   * **date\_format**: The desired date format (e.g., %d/%m/%Y, %m-%d-%Y).
2. **Output:**
   * Returns the calculated date in the specified format as a string.

**Advantages**

* **Customizable Formatting:** Supports a variety of date formats to suit diverse needs.
* **Dynamic Date Calculations:** Simplifies date-based operations for testing or automation.
* **Reusable Functions:** Can be easily integrated into different projects.

**Best Practices**

1. **Validate Formats:**  
   Ensure the date\_format parameter is correctly specified using Python's strftime directives.
2. **Use Standardized Formats:**  
   For consistency, use a single format across all scripts unless specific scenarios require otherwise.
3. **Error Handling:**  
   Handle invalid format inputs gracefully to avoid runtime errors.

**Conclusion**

The **Date Utility Functions** provide an efficient way to handle date calculations dynamically. With support for various formats and ease of integration, they are a valuable tool for any automation or testing framework.