**Report: Streaming Moving Anomaly Detection with Random Forest**

**Introduction:**

The Python script implements a streaming anomaly detection system using a Random Forest algorithm. The algorithm dynamically updates and adapts to incoming data points, identifying anomalies in real-time. The report aims to explain the structure of the code, its key components, and provide insights into its functionality.

**Code Overview:**

**1. Random Forest Implementation:**

- The `RCTree` class represents a tree node for a Random Cut Tree (RCT).

- `RCTreeNode` is a node in the RCT, and `RandomForest` manages a collection of these trees.

- The `update` method in `RCTree` adds a new node to the tree, ensuring a maximum size limit is maintained.

- The `codisp` method calculates the co-dispersion of a given point within the tree.

- `RandomForest` maintains multiple trees and can update them with new data points and calculate a collective anomaly score.

**2. StreamingAnomalyDetector:**

- This class initializes a Random Forest for anomaly detection.

- The `update` method adds new data points to the Random Forest.

- `detect\_anomaly` calculates the anomaly score for a given data point based on the collective response of the Random Forest.

**3. Data Generation:**

- The `generate\_data\_point` function generates synthetic data with a regular pattern, seasonal element, and random noise. This simulated data is used for testing the anomaly detection system.

**4. Example Usage and Visualization:**

- An instance of `StreamingAnomalyDetector` is created.

- The script sets up a real-time animated plot using `matplotlib`, where data points are continuously added, and anomalies are highlighted in red.

**Example Usage:**

- The script demonstrates the anomaly detection system on a simulated data stream.

- A streaming detector is initialized, and an animated plot is created to visualize the data stream and detected anomalies.