Below is a proposed software specification for the Python API functions. Each function is defined with its title, function name, unique ID (combining Workflow and Instruction IDs), purpose, signature (header and docstring), and the list of instructions it is used for.

**Function 1: Compare Probe Yield Summaries**  
**Function Name**: compare\_probe\_yield\_summaries  
**ID**: 4\_5  
**Purpose**:  
Compares the yield summaries from different probe configurations to evaluate how the probe size impacts yield performance.  
**Signature**:

def compare\_probe\_yield\_summaries(probe\_configs: list, yield\_summaries: pd.DataFrame) -> pd.DataFrame:

"""

Compare yield summaries across different probe configurations to evaluate yield performance.

Parameters:

probe\_configs (list): List of probe configuration identifiers.

yield\_summaries (pd.DataFrame): DataFrame containing yield summary data.

Returns:

pd.DataFrame: A DataFrame with the comparative analysis results.

"""

**Used For**:

* "Perform Comparative Analysis: Compare the yield summaries from different probe configurations to evaluate the impact of the probe size on yield performance." (Workflow 4, Instruction 5)

**Function 2: Compare Test House Yield Values**  
**Function Name**: compare\_test\_house\_yields  
**ID**: 5\_5  
**Purpose**:  
Analyzes and compares aggregated yield values across different test houses, highlighting any significant differences.  
**Signature**:

def compare\_test\_house\_yields(aggregated\_yields: pd.DataFrame, test\_houses: list, significance\_level: float = 0.05) -> pd.DataFrame:

"""

Compare aggregated yield values across test houses to identify significant differences.

Parameters:

aggregated\_yields (pd.DataFrame): DataFrame containing aggregated yield data for each test house.

test\_houses (list): List of test house identifiers.

significance\_level (float): Statistical significance threshold (default 0.05).

Returns:

pd.DataFrame: Comparative analysis results with highlighted differences.

"""

**Used For**:

* "Compare Performance: Analyze and compare the aggregated yield values across different test houses, highlighting any significant differences." (Workflow 5, Instruction 5)

**Function 3: Compare Configuration Yield Metrics**  
**Function Name**: compare\_configuration\_yield\_metrics  
**ID**: 15\_6  
**Purpose**:  
Performs a comparative analysis of yield metrics between various configurations to identify differences in performance.  
**Signature**:

def compare\_configuration\_yield\_metrics(yield\_metrics: pd.DataFrame, configurations: list) -> pd.DataFrame:

"""

Compare yield metrics across different configurations to identify performance differences.

Parameters:

yield\_metrics (pd.DataFrame): DataFrame containing yield metrics.

configurations (list): List of configuration identifiers.

Returns:

pd.DataFrame: A DataFrame summarizing the comparative analysis.

"""

**Used For**:

* "Compare Yield Performance - Perform a comparative analysis to examine differences in yield metrics between the various configurations." (Workflow 15, Instruction 6)

**Function 4: Generate Soft Bin Visualizations**  
**Function Name**: generate\_soft\_bin\_visualizations  
**ID**: 18\_5  
**Purpose**:  
Generates side-by-side visualizations or tables to compare soft bin distributions between two test program revisions.  
**Signature**:

def generate\_soft\_bin\_visualizations(soft\_bin\_rev1: pd.DataFrame, soft\_bin\_rev2: pd.DataFrame, viz\_type: str = 'side\_by\_side') -> None:

"""

Generate visualizations comparing soft bin distributions between two revisions.

Parameters:

soft\_bin\_rev1 (pd.DataFrame): DataFrame for soft bin distribution of revision 1.

soft\_bin\_rev2 (pd.DataFrame): DataFrame for soft bin distribution of revision 2.

viz\_type (str): Type of visualization (default is 'side\_by\_side').

Returns:

None

"""

**Used For**:

* "Compare Soft Bin Distributions: Generate side-by-side visualizations or tables comparing soft bin distributions between the two revisions." (Workflow 18, Instruction 5)

**Function 5: Compare E-test Distributions**  
**Function Name**: compare\_etest\_distributions  
**ID**: 23\_5  
**Purpose**:  
Compares the statistical distributions of E-test variations between high-yield and low-yield groups to identify significant differences.  
**Signature**:

def compare\_etest\_distributions(high\_yield\_data: pd.DataFrame, low\_yield\_data: pd.DataFrame, metric: str) -> pd.DataFrame:

"""

Compare statistical distributions of E-test variations between high-yield and low-yield groups.

Parameters:

high\_yield\_data (pd.DataFrame): DataFrame containing data for the high-yield group.

low\_yield\_data (pd.DataFrame): DataFrame containing data for the low-yield group.

metric (str): Specific E-test metric to compare.

Returns:

pd.DataFrame: A DataFrame with the comparative distribution analysis.

"""

**Used For**:

* "Perform Comparative Analysis: Compare the statistical distributions between high-yield and low-yield groups to identify any significant differences in E-test variations." (Workflow 23, Instruction 5)

**Function 6: Compare Handler and Load Board Yield Performance**  
**Function Name**: compare\_handler\_loadboard\_yield\_performance  
**ID**: 34\_5  
**Purpose**:  
Compares yield performance across different handler and load board groups to detect performance changes when the testing configuration changes.  
**Signature**:

def compare\_handler\_loadboard\_yield\_performance(yield\_data: pd.DataFrame, handler\_groups: list, load\_board\_groups: list) -> pd.DataFrame:

"""

Compare yield performance across handler and load board groups.

Parameters:

yield\_data (pd.DataFrame): DataFrame containing yield performance data.

handler\_groups (list): List of handler group identifiers.

load\_board\_groups (list): List of load board group identifiers.

Returns:

pd.DataFrame: A DataFrame summarizing comparative yield performance.

"""

**Used For**:

* "Comparative Yield Analysis - Compare the yield performance across different handler and load board groups to identify any performance changes." (Workflow 34, Instruction 5)

**Function 7: Compare Consecutive Lot Yields**  
**Function Name**: compare\_consecutive\_lot\_yields  
**ID**: 39\_5  
**Purpose**:  
Directly compares the calculated lot-level yields from two consecutive lots to identify differences and emerging trends.  
**Signature**:

def compare\_consecutive\_lot\_yields(lot\_yield1: float, lot\_yield2: float) -> dict:

"""

Compare lot-level yields from two consecutive lots.

Parameters:

lot\_yield1 (float): Yield value for the first lot.

lot\_yield2 (float): Yield value for the consecutive lot.

Returns:

dict: A dictionary with the comparison results and trend analysis.

"""

**Used For**:

* "Compare Yields: Directly compare the calculated lot-level yields from the two consecutive lots to identify differences and trends." (Workflow 39, Instruction 5)

**Function 8: Evaluate Yield Against Targets**  
**Function Name**: evaluate\_yield\_against\_targets  
**ID**: 50\_4  
**Purpose**:  
Evaluates computed yields against predefined target yield thresholds to identify deviations from the expected performance.  
**Signature**:

def evaluate\_yield\_against\_targets(yield\_data: pd.DataFrame, target\_thresholds: dict) -> pd.DataFrame:

"""

Evaluate computed yields against predefined target yield thresholds to identify deviations.

Parameters:

yield\_data (pd.DataFrame): DataFrame containing yield performance data.

target\_thresholds (dict): Dictionary mapping yield metrics to target thresholds.

Returns:

pd.DataFrame: A DataFrame detailing deviations from target yields.

"""

**Used For**:

* "Compare Against Yield Targets: Evaluate the computed yields against predefined target yield thresholds to identify any deviations." (Workflow 50, Instruction 4)

**Function 9: Check Aggregated Yield Against Target**  
**Function Name**: check\_aggregated\_yield\_against\_target  
**ID**: 63\_5  
**Purpose**:  
Checks whether the aggregated yield meets or exceeds a predefined target yield (e.g., 95%).  
**Signature**:

def check\_aggregated\_yield\_against\_target(aggregated\_yield: float, target\_yield: float = 95.0) -> bool:

"""

Check if the aggregated yield meets or exceeds the predefined target yield.

Parameters:

aggregated\_yield (float): The aggregated yield value.

target\_yield (float): The predefined target yield percentage (default is 95.0).

Returns:

bool: True if the aggregated yield meets/exceeds the target, False otherwise.

"""

**Used For**:

* "Compare Against Target Yield - Check whether the aggregated yield meets or exceeds the predefined target yield (e.g., 95%)." (Workflow 63, Instruction 5)

**Function 10: Compare Failure Metrics**  
**Function Name**: compare\_failure\_metrics  
**ID**: 9\_5  
**Purpose**:  
Organizes and compares failure metrics between logical and parametric test failures to determine differences and trends.  
**Signature**:

def compare\_failure\_metrics(logical\_failures: pd.DataFrame, parametric\_failures: pd.DataFrame) -> pd.DataFrame:

"""

Compare failure metrics between logical and parametric test failures.

Parameters:

logical\_failures (pd.DataFrame): DataFrame containing logical test failure data.

parametric\_failures (pd.DataFrame): DataFrame containing parametric test failure data.

Returns:

pd.DataFrame: A DataFrame with the side-by-side comparative analysis of failures.

"""

**Used For**:

* "Compare Failure Data - Organize and compare the failure metrics side-by-side to determine differences and trends between logical and parametric test failures." (Workflow 9, Instruction 5)

**Function 11: Compare Load Board Performance**  
**Function Name**: compare\_load\_board\_performance  
**ID**: 33\_7  
**Purpose**:  
Compares computed yield metrics across different load boards to identify any that consistently result in lower yield outcomes.  
**Signature**:

def compare\_load\_board\_performance(yield\_metrics: pd.DataFrame, load\_board\_ids: list) -> pd.DataFrame:

"""

Compare yield metrics across different load boards to identify lower yield outcomes.

Parameters:

yield\_metrics (pd.DataFrame): DataFrame containing yield metric data.

load\_board\_ids (list): List of load board identifiers.

Returns:

pd.DataFrame: A DataFrame summarizing the comparative analysis across load boards.

"""

**Used For**:

* "Compare Load Board Performance - Compare the computed yield metrics across load boards to identify any that consistently show lower yield outcomes." (Workflow 33, Instruction 7)

**Function 12: Assess Yield Control Limits**  
**Function Name**: assess\_yield\_control\_limits  
**ID**: 40\_6  
**Purpose**:  
Assesses whether the current lot's yield falls within the computed upper and lower control limits based on historical yield data.  
**Signature**:

def assess\_yield\_control\_limits(current\_yield: float, control\_limits: dict, historical\_yield: pd.DataFrame) -> bool:

"""

Assess if the current lot's yield is within the control limits derived from historical yield data.

Parameters:

current\_yield (float): Yield value for the current lot.

control\_limits (dict): Dictionary containing 'upper' and 'lower' control limits.

historical\_yield (pd.DataFrame): DataFrame with historical yield data for reference.

Returns:

bool: True if the current yield is within control limits, False otherwise.

"""

**Used For**:

* "Compare Current Yield with Control Limits: Assess whether the current lot's yield falls within the computed upper and lower control limits." (Workflow 40, Instruction 6)

This specification outlines 12 API functions that cover the range of comparative analyses and visualizations required for semiconductor chip test data analytics. Each function is designed to take the appropriate data inputs and return either analysis results or visualizations as needed.