Below is one possible software specification that defines a set of Python API functions covering the key types of calculations and analyses described by the instructions. (Note that many instructions share similar operations so each function is designed to be generic—accepting parameters (such as the data source, grouping keys, field names, etc.) to handle variations such as wafer‐level versus lot‐level, different grouping attributes, and additional filters.)

Each function specification includes:  
• **Title of the Function** (with a numbering)  
• **Function Name**  
• **ID** (a list of WorkflowID\_InstructionID pairs for which the function is used)  
• **Purpose** (a brief description)  
• **Signature** (the Python function header with a docstring outlining parameters and return value)  
• **Used For** (a list summarizing the original instructions that call for this function)

**1. Calculate Wafer-Level Yield**

**Function Name**: calculate\_wafer\_yield  
**ID**: 4\_3, 5\_3, 9\_3, 35\_2, 42\_3, 47\_3, 48\_3, 50\_2, 51\_3, 62\_3, 63\_3, 19\_3, 33\_3  
**Purpose**: Compute the yield for individual wafers (or wafer groups) as the ratio of passing dies to total dies. The function accepts an optional grouping key (e.g. probe configuration, test house, week, time segment) so that it can be reused in many contexts.  
**Signature**:

def calculate\_wafer\_yield(wafer\_data, group\_by=None, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Calculate wafer-level yield by computing the ratio of passing dies to total dies.

Parameters:

wafer\_data (DataFrame or list): The data set containing wafer test results.

group\_by (str, optional): Field name to group by (e.g., 'probe\_configuration', 'test\_house', 'week', 'time\_segment').

passing\_dies\_field (str): Field name indicating the number of passing dies.

total\_dies\_field (str): Field name indicating the total number of dies.

Returns:

Computed yield for each wafer or for each group if grouping is applied.

"""

**Used For**:  
– Calculate Wafer-Level Yields for different probe configurations (Workflow 4\_3)  
– Compute yields per test house (Workflow 5\_3)  
– Calculate wafer yields per week or time segment (Workflows 9\_3, 42\_3, 19\_3, 35\_2, 63\_3)  
– Evaluate yield before/after process adjustments (Workflow 48\_3)  
– Handle monthly or tester‐variation scenarios (Workflows 50\_2, 51\_3, 62\_3, 33\_3)

**2. Calculate Lot-Level Yield**

**Function Name**: calculate\_lot\_yield  
**ID**: 17\_4, 41\_2, 61\_3, 64\_3, 1\_3, 31\_2, 34\_3, 36\_3, 39\_3, 40\_3, 22\_3  
**Purpose**: Aggregate wafer-level yields (or directly use die counts) to compute the overall yield at the lot level.  
**Signature**:

def calculate\_lot\_yield(lot\_data, wafer\_yield\_field=None, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Calculate lot-level yield by aggregating wafer-level yields or by directly computing the ratio of passing dies to total dies.

Parameters:

lot\_data (DataFrame or list): The data set containing lot information.

wafer\_yield\_field (str, optional): Field name for pre-computed wafer yields (if available).

passing\_dies\_field (str): Field name for the count of passing dies.

total\_dies\_field (str): Field name for the count of total dies.

Returns:

The computed yield for each lot.

"""

**Used For**:  
– Compute overall lot yield by aggregating wafer results (Workflow 17\_4, 61\_3)  
– Lot yield for a fixed set of lots (Workflow 41\_2, 1\_3)  
– Yield performance by test house or for current quarter/lots (Workflows 64\_3, 34\_3, 36\_3, 39\_3, 40\_3, 22\_3, 31\_2)

**3. Calculate Failure Rate**

**Function Name**: calculate\_failure\_rate  
**ID**: 43\_4, 7\_4, 46\_3  
**Purpose**: Compute failure rates for a given test object (e.g. test program or specific bin) by dividing the count of failures by the total number of tests or dies.  
**Signature**:

def calculate\_failure\_rate(failure\_data, failure\_count\_field, total\_field, additional\_filter=None):

"""

Calculate the failure rate as the ratio of failure counts to the total number of tests or dies.

Parameters:

failure\_data (DataFrame or list): Data containing failure and total test/die counts.

failure\_count\_field (str): Field name for failure counts.

total\_field (str): Field name for total tests or dies.

additional\_filter (dict, optional): Criteria to filter data (e.g., specific bin or test program).

Returns:

Failure rate computed for the filtered data set.

"""

**Used For**:  
– Compute logical test failure rates (Workflow 43\_4, 7\_4)  
– Calculate bin-specific failure rates (e.g. bin 25 in Workflow 46\_3)

**4. Count Failing Dies**

**Function Name**: count\_failing\_dies  
**ID**: 79\_4  
**Purpose**: Count the total number of failing dies based on identified failing outcomes.  
**Signature**:

def count\_failing\_dies(test\_data, failing\_flag\_field='failing\_outcome'):

"""

Count the number of failing dies from the provided test data.

Parameters:

test\_data (DataFrame or list): Data set containing test results.

failing\_flag\_field (str): Field name that flags failing outcomes.

Returns:

The total count of failing dies.

"""

**Used For**:  
– Reporting the number of failing dies on the last tested wafer (Workflow 79\_4)

**5. Compute Statistical Summary**

**Function Name**: compute\_statistical\_summary  
**ID**: 12\_5, 17\_5, 23\_4, 28\_4, 29\_3, 49\_4, 51\_5, 57\_4, 62\_5  
**Purpose**: Calculate key summary statistics (e.g., mean, median, standard deviation, variance, range) for a given data set.  
**Signature**:

def compute\_statistical\_summary(data, fields, metrics=['mean', 'median', 'std', 'variance', 'range']):

"""

Compute statistical summary metrics for the specified fields in the data.

Parameters:

data (DataFrame or list): The input data.

fields (list): List of field names for which to compute statistics.

metrics (list): List of statistical metrics to compute.

Returns:

A dictionary or DataFrame of computed statistics.

"""

**Used For**:  
– Calculating E-test parameter distributions (Workflow 12\_5)  
– Summary statistics for lot yields (Workflow 17\_5)  
– Statistical measures for E-test measurements (Workflow 23\_4)  
– Performance metrics for tests (Workflow 28\_4)  
– Variance and test variance (Workflows 29\_3, 57\_4)  
– Statistical metrics across time or tester groups (Workflows 49\_4, 51\_5, 62\_5)

**6. Compute Control Limits**

**Function Name**: compute\_control\_limits  
**ID**: 40\_5  
**Purpose**: Determine control limits for current yield performance based on historical data (e.g. using mean ± 3\*standard deviation).  
**Signature**:

def compute\_control\_limits(historical\_yield\_data, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Compute control limits based on historical yield data.

Parameters:

historical\_yield\_data (DataFrame or list): Historical yield measurements.

passing\_dies\_field (str): Field name for passing dies.

total\_dies\_field (str): Field name for total dies.

Returns:

A tuple (lower\_limit, upper\_limit) representing the control limits.

"""

**Used For**:  
– Checking if the current lot’s yield is within control limits (Workflow 40\_5)

**7. Compute Variability Metrics**

**Function Name**: compute\_variability\_metrics  
**ID**: 66\_4  
**Purpose**: Quantify the variability of a set of parametric test measurements (e.g. by computing min, max, percentiles along with mean and standard deviation).  
**Signature**:

def compute\_variability\_metrics(data, measurement\_field, metrics=['mean', 'std', 'variance', 'min', 'max', 'percentiles']):

"""

Compute variability metrics for a specified measurement field.

Parameters:

data (DataFrame or list): Data set containing measurement values.

measurement\_field (str): Field name for the measurement.

metrics (list): List of statistical metrics to compute.

Returns:

A dictionary of variability metrics.

"""

**Used For**:  
– Reporting the variability of parametric test values (Workflow 66\_4)

**8. Perform Correlation Analysis**

**Function Name**: perform\_correlation\_analysis  
**ID**: 3\_5  
**Purpose**: Calculate statistical correlations between two or more variables (for example, between wafer map patterns and test program failures).  
**Signature**:

def perform\_correlation\_analysis(data, field\_x, field\_y, method='pearson'):

"""

Perform correlation analysis between two specified fields.

Parameters:

data (DataFrame or list): Data set containing the variables.

field\_x (str): The first variable.

field\_y (str): The second variable.

method (str): Correlation method (e.g., 'pearson', 'spearman').

Returns:

The correlation coefficient and p-value.

"""

**Used For**:  
– Correlating unusual wafer map patterns with specific test program failures (Workflow 3\_5)

**9. Quantify Failure Contributions**

**Function Name**: quantify\_failure\_contributions  
**ID**: 10\_4, 26\_5, 11\_3  
**Purpose**: Determine the contribution of specific test programs or test measurements to the overall failure count (by computing either percentages or counts).  
**Signature**:

def quantify\_failure\_contributions(data, failure\_field, total\_field, group\_by):

"""

Quantify the contribution of failures for each group (e.g., test program or measurement) by calculating failure percentages or counts.

Parameters:

data (DataFrame or list): The input data set.

failure\_field (str): Field name indicating the count of failures.

total\_field (str): Field name for total die count or tests.

group\_by (str): Field name used for grouping (e.g., 'test\_program', 'test\_measurement').

Returns:

A summary of failure contributions per group.

"""

**Used For**:  
– Quantifying failing die contributions per test program (Workflow 10\_4)  
– Quantifying measurement failure contributions (Workflow 26\_5)  
– Quantifying failure frequency of limit violations (Workflow 11\_3)

**10. Aggregate Failure Counts by Region**

**Function Name**: aggregate\_failure\_counts\_by\_region  
**ID**: 40\_5  
**Purpose**: Aggregate and optionally compute the percentage of failures per region on a wafer by comparing failure counts with the total die count in that region.  
**Signature**:

def aggregate\_failure\_counts\_by\_region(wafer\_data, region\_field, failure\_count\_field, total\_dies\_field):

"""

Aggregate failure counts per region and calculate the failure percentage.

Parameters:

wafer\_data (DataFrame or list): Data set containing wafer regions.

region\_field (str): Field name that indicates the wafer region.

failure\_count\_field (str): Field name for the number of failures.

total\_dies\_field (str): Field name for total die count.

Returns:

A summary table with failure counts and percentages per region.

"""

**Used For**:  
– Reporting total failures per region and failure percentages (Workflow 40\_5)

**11. Calculate Average Parametric Value**

**Function Name**: calculate\_average\_parametric\_value  
**ID**: 80\_5, 34\_6  
**Purpose**: Compute the average (mean) of a specified parametric test measurement—for example, for a failing test family across wafers or time periods.  
**Signature**:

def calculate\_average\_parametric\_value(data, measurement\_field):

"""

Calculate the average value of the specified parametric test measurement.

Parameters:

data (DataFrame or list): The input measurement data.

measurement\_field (str): Field name of the parametric test measurement.

Returns:

The computed average (mean) value.

"""

**Used For**:  
– Displaying the average parametric test value (Workflow 80\_5)  
– Computing average parametric values for failing test families (Workflow 34\_6)

**12. Calculate E-test Site Variation**

**Function Name**: calculate\_e\_test\_site\_variation  
**ID**: 30\_3  
**Purpose**: For each wafer, compute a variation metric (e.g. standard deviation or range) across different E-test sites to assess measurement variability.  
**Signature**:

def calculate\_e\_test\_site\_variation(wafer\_data, etest\_site\_field, measurement\_field, method='std'):

"""

Calculate a variation metric (e.g., standard deviation or range) for E-test measurements across different sites for each wafer.

Parameters:

wafer\_data (DataFrame or list): Data set containing E-test site measurements.

etest\_site\_field (str): Field name for E-test site identifiers.

measurement\_field (str): Field name for the measurement values.

method (str): The variation metric to use ('std' for standard deviation, 'range' for range, etc.).

Returns:

Variation metric for each wafer.

"""

**Used For**:  
– Analyzing correlation between E-test site variation and yield drop (Workflow 30\_3)

**13. Compute Yield Improvement**

**Function Name**: compute\_yield\_improvement  
**ID**: 38\_5  
**Purpose**: Compare lot yields before and after a retest (for example, retesting the worst wafer) and quantify the improvement using statistical measures.  
**Signature**:

def compute\_yield\_improvement(before\_data, after\_data, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Calculate the improvement in lot yield before and after a retest.

Parameters:

before\_data (DataFrame or list): Yield data before retest.

after\_data (DataFrame or list): Yield data after retest.

passing\_dies\_field (str): Field name for passing dies.

total\_dies\_field (str): Field name for total dies.

Returns:

A measure of yield improvement (e.g., difference or percentage change) along with relevant statistical significance.

"""

**Used For**:  
– Evaluating yield improvement from retesting (Workflow 38\_5)

**14. Calculate Weekly Yield**

**Function Name**: calculate\_weekly\_yield  
**ID**: 9\_3, 42\_3, 45\_3, 74\_3, 16\_3  
**Purpose**: Compute yield on a weekly basis by grouping wafer-level results by week and then calculating the ratio of passing dies to total dies for each week.  
**Signature**:

def calculate\_weekly\_yield(wafer\_data, week\_field, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Calculate weekly wafer-level yield by grouping data by week.

Parameters:

wafer\_data (DataFrame or list): The input wafer test data.

week\_field (str): Field name that indicates the week.

passing\_dies\_field (str): Field name for passing dies.

total\_dies\_field (str): Field name for total dies.

Returns:

Weekly yield values.

"""

**Used For**:  
– Yield calculation for wafers segmented by week (Workflows 9\_3, 42\_3, 45\_3, 74\_3, 16\_3)

**15. Calculate Wafer-Level Yield per Tester**

**Function Name**: calculate\_wafer\_yield\_per\_tester  
**ID**: 70\_4, 31\_4  
**Purpose**: Compute yield for each tester by grouping wafer-level yield data per tester (and possibly by week) to assess tester-specific performance differences.  
**Signature**:

def calculate\_wafer\_yield\_per\_tester(wafer\_data, tester\_field, passing\_dies\_field='passing\_dies', total\_dies\_field='total\_dies'):

"""

Calculate wafer-level yield for each tester.

Parameters:

wafer\_data (DataFrame or list): Data containing wafer yields.

tester\_field (str): Field name for tester identifiers.

passing\_dies\_field (str): Field name for passing dies.

total\_dies\_field (str): Field name for total dies.

Returns:

Yield metrics aggregated by tester.

"""

**Used For**:  
– Analyzing yield per tester over multiple weeks (Workflow 70\_4)  
– Comparing yield distributions between testers (Workflow 31\_4)

**16. Calculate Total Test Count**

**Function Name**: calculate\_total\_test\_count  
**ID**: 77\_4  
**Purpose**: Sum the total number of tests (or test events) executed in a given period, such as over a week.  
**Signature**:

def calculate\_total\_test\_count(test\_data, test\_count\_field):

"""

Calculate the total test count by summing the number of test events.

Parameters:

test\_data (DataFrame or list): Data containing test event counts.

test\_count\_field (str): Field name for the test count.

Returns:

The total test count.

"""

**Used For**:  
– Generating a report on the total number of tests executed last week (Workflow 77\_4)

**17. Normalize Failure Data**

**Function Name**: normalize\_failure\_data  
**ID**: 71\_4  
**Purpose**: Normalize raw failure counts to percentages by comparing each bin’s failure count with the total failures, to improve clarity of the failure distribution.  
**Signature**:

def normalize\_failure\_data(failure\_data, failure\_count\_field, total\_failures):

"""

Normalize failure counts to percentages.

Parameters:

failure\_data (DataFrame or list): Data containing failure counts per bin.

failure\_count\_field (str): Field name for failure counts.

total\_failures (int): Total failure count across all bins.

Returns:

Normalized failure percentages for each bin.

"""

**Used For**:  
– Normalizing failure data for clearer insights (Workflow 71\_4)

**18. Compute Subgroup Statistics**

**Function Name**: compute\_subgroup\_statistics  
**ID**: 16\_5  
**Purpose**: Calculate statistical metrics (such as failure rates and deviations) for specified subgroups across wafers to determine if yield issues are isolated to specific measurement subsets.  
**Signature**:

def compute\_subgroup\_statistics(data, subgroup\_field, metrics=['failure\_rate', 'deviation']):

"""

Compute statistical metrics for each subgroup in the data.

Parameters:

data (DataFrame or list): Input data containing subgroup identifiers.

subgroup\_field (str): Field name for subgroup classification.

metrics (list): List of statistical metrics to compute.

Returns:

A summary of statistics for each subgroup.

"""

**Used For**:  
– Analyzing yield issues by correlating test measurement subgroups (Workflow 16\_5)

**19. Compute E-test Site Means**

**Function Name**: compute\_e\_test\_site\_means  
**ID**: 34\_3  
**Purpose**: Aggregate E-test measurement values by test site (or by time period) and compute the mean to enable temporal correlation analysis.  
**Signature**:

def compute\_e\_test\_site\_means(data, site\_field, etest\_value\_field):

"""

Compute the mean of E-test measurements aggregated by test site.

Parameters:

data (DataFrame or list): Data containing E-test measurements.

site\_field (str): Field name for the test site.

etest\_value\_field (str): Field name for the E-test measurement values.

Returns:

Mean E-test values per site.

"""

**Used For**:  
– Evaluating temporal correlation between E-test site means and parametric values (Workflow 34\_3)

**20. Calculate Failure Percentage per Soft Bin**

**Function Name**: calculate\_failure\_percentage  
**ID**: 35\_3  
**Purpose**: For each wafer, compute the failure percentage for a soft bin by dividing the count of dies in that bin by the total die count.  
**Signature**:

def calculate\_failure\_percentage(wafer\_data, soft\_bin\_field, die\_count\_field, total\_dies\_field):

"""

Calculate the failure percentage for each soft bin on a wafer.

Parameters:

wafer\_data (DataFrame or list): Data containing die counts per soft bin.

soft\_bin\_field (str): Field name for the soft bin identifier.

die\_count\_field (str): Field name for the number of dies failing in the soft bin.

total\_dies\_field (str): Field name for the total die count.

Returns:

Failure percentage for each soft bin.

"""

**Used For**:  
– Finding wafers where over 5% of dies failed in a single soft bin (Workflow 35\_3)

This specification provides a modular set of API functions. Each function is parameterized so that it can be reused in various contexts (e.g. by specifying grouping keys or additional filters) to support the diverse semiconductor chip test data analytics instructions outlined in the provided JSON list.