`openlane.common.metrics` API

Metrics Module

Classes and functions for dealing with Metrics based on the METRICS2.1 standard.

```
class openlane.common.metrics.MetricComparisonResult(metric_name: str,
    gold: Any, new: Any, delta: int | float | Decimal | None, delta_pct:
    int | float | Decimal | None, better: bool | None, critical: bool,
    significant_figures: int | None)
```

Bases: `object`

PARAMETERS:

- **metric_name** (*str*) – The name of the metric that has been compared
- **gold** (*Any*) – The "gold" value being compared against
- **new** (*Any*) – The new value being evaluated
- **delta** (*int* | *float* | *Decimal* | *None*) – `None` if and only if `before` - `after` is an invalid number. Evaluates to `after - before`.
- **delta_pct** (*int* | *float* | *Decimal* | *None*) – `None` if `delta` is None or before is zero. Otherwise, evaluates to `delta / before * 100`.
- **better** (*bool* | *None*) – Whether the change in the value is considered a good thing or not. `None` if `delta` is None or has no value set for `Metric.higher_is_better`.
- **critical** (*bool*) – Whether this change of value very likely results in a dead chip, i.e., an increase in DRC values, or an inexplicable change in the number of I/O pins.
- **significant_figures** (*int* | *None*)

```
class openlane.common.metrics.Metric(name: str, aggregator: Tuple[int |
    float | Decimal, Callable[[Iterable[int | float | Decimal]], int |
    float | Decimal]] | None = None, higher_is_better: bool | None =
    None, dont_aggregate: Iterable[str] | None = None, critical: bool =
    False)
```

Bases: `object`

An object storing data about a metric as defined in METRICS2.1.

PARAMETERS:

- **name** (*str*) – The string name of the metric.
- **aggregator** (*Tuple[int | float | Decimal, Callable[[Iterable[int | float | Decimal]], int | float | Decimal]] | None*) –

  A tuple of: - A starting value for an accumulator - A reduction function

  The aim is the ability to aggregate values from various sub-metrics, i.e., for the metric `timing__hold_vio__count`, the sub-metrics:

  - `timing__hold_vio__count__corner:A`
  - `timing__hold_vio__count__corner:B`

  Would be summed up to generate the value for `timing__hold_vio__count`.
- **higher_is_better** (*bool | None*) – At a high level, whether a higher numeric value for this metric is considered “good” (such as: better utilization) or “bad” (such as: more antenna violations.)
- **critical** (*bool*) – A critical metric is always watched for any change.
- **dont_aggregate** (*Iterable[str] | None*)

```
modified_name(modifiers: Mapping[str, str]) → str
```

PARAMETERS:

modifiers (*Mapping[str, str]*) – Modifiers of a metric (i.e. the elements postfixed to the metric in the format {key}:{value})

RETURNS:

The name with the modifiers added

RETURN TYPE:

str

```
compare(gold: Any, new: Any, significant_figures: int, modifiers:
```

```
Mapping[str, str] | None = None) → MetricComparisonResult
```

PARAMETERS:

- **gold** (*Any*) – The “gold-standard” value for this metric to compare against
- **new** (*Any*) – The new value for this metric being evaluated
- **modifier** – The modifiers that were parsed from the metric name (if applicable)- used to set the `metric_name` property of `MetricComparisonResult`.
- **significant_figures** (*int*)
- **modifiers** (*Mapping[str, str] | None*)

RETURNS:

The result of comparing two values for this metric.

RETURN TYPE:

MetricComparisonResult

```
openlane.common.metrics.parse_metric_modifiers(metric_name: str) →
    Tuple[str, Mapping[str, str]]
```

Parses a metric name into a base and modifiers as specified in the METRICS2.1 naming convention.

PARAMETERS:

metric_name (*str*) – The name of the metric as generated by a utility.

RETURNS:

A tuple of the base part as a string, then the modifiers as a key-value mapping.

RETURN TYPE:

Tuple[str, *Mapping*[str, str]]

```
openlane.common.metrics.aggregate_metrics(input: Mapping[str, Any],
    aggregator_by_metric: Mapping[str, Tuple[int | float | Decimal,
    Callable[[Iterable[int | float | Decimal]], int | float | Decimal]]
    | Metric] | None = None) → Dict[str, Any]
```

Takes a set of metrics generated according to the METRICS2.1 naming

convention.

PARAMETERS:

- **input** (*Mapping[str, Any]*) – A mapping of strings to values of metrics.
- **aggregator_by_metric** (*Mapping[str, Tuple[int | float | Decimal, Callable[[Iterable[int | float | Decimal]], int | float | Decimal]] | Metric] | None*) – A mapping of metric names to either: - A tuple of the initial accumulator and reducer to aggregate the values from all modifier metrics - A `Metric` class

RETURNS:

A tuple of the base part as a string, then the modifiers as a key-value mapping.

RETURN TYPE:

Dict[str, *Any*]

```
class openlane.common.metrics.MetricDiff(differences:
    Iterable[MetricComparisonResult])
```

Bases: `object`

Aggregates a number of `MetricComparisonResult` and allows a number of functions to be performed on them.

PARAMETERS:

differences (*List[MetricComparisonResult]*) – The metric comparison results.

```
class MetricStatistics(better: int = 0, worse: int = 0, critical: int
    = 0, unchanged: int = 0)
```

Bases: `object`

A glorified namespace encapsulating a number of statistics of `MetricDiff`.

Should be generated using `MetricDiff.stats()`.

PARAMETERS:

- **better** (*int*) – The number of datapoints that represent a positive change.
- **worse** (*int*) – The number of datapoints that represent a negative change.
- **critical** (*int*) – The number of changes for critical metrics.
- **unchanged** (*int*) – Values that are unchanged.

```
render_md(sort_by: Iterable[str] | None = None, table_verbosity:
    TableVerbosity = TableVerbosity.ALL) → str
```

PARAMETERS:

- **sort_by** (*Iterable[str] | None*) – A list of tuples corresponding to modifiers to sort metrics ascendingly by.
- **table_verbosity** (*TableVerbosity*) – The verbosity of the table: whether to include everything, just changes, only bad changes or only critical changes. Or just nothing.

RETURNS:

A table of the differences in Markdown format.

RETURN TYPE:

str

```
stats() → MetricStatistics
```

RETURNS:

A `MetricStatistics` object based on this aggregate.

RETURN TYPE:

MetricStatistics

```
classmethod from_metrics(gold: dict, new: dict, significant_figures:
    int, filter: ~openlane.common.misc.Filter =
    <openlane.common.misc.Filter object>) → MetricDiff
```

Creates a `MetricDiff` object from two sets of metrics.

PARAMETERS:

- **gold** (*dict*) – The “gold-standard” metrics to compare against
- **new** (*dict*) – The metrics being evaluated
- **filter** (*Filter*) – A `Filter` for the names of the metrics to include or exclude certain metrics.
- **significant_figures** (*int*)

RETURNS:

The aggregate of the differences between gold and good

RETURN TYPE:

MetricDiff