**Anatomy**

Anatomy of a Font. Visualise metrics.

Import the `anatomy` package:

```
#import "@preview/anatomy:0.1.1": metrics
```

**Display Metrics**

`metrics(72pt, "EB Garamond", display: "Typewriter")` will be rendered as follows:


Figure 1: The metrics of EB Garamond (Google Fonts)

Additionally, a closure using `metrics` dictionary as parameter can be used to layout additional elements below:

```
#metrics(54pt, "一點明體",
  display: "電傳打字機",
  use: metrics ⇒ table(
    columns: 2,
    ..metrics.pairs().flatten().map(x ⇒ [ #x ])
  )
)
```

It will generate:


| | |
|---|---|
| ascender | 47.51pt |
| cap-height | 38.6pt |
| x-height | 26.58pt |
| baseline | 0pt |
| descender | -6.49pt |

Figure 2: The metrics of I.Ming (table attached)

**Remark on Hybrid Typesetting**

To typeset CJK text, adopting font's ascender/descender as `top-edge`/`bottom-edge` makes more sense in some cases. As for most CJK fonts, the difference between ascender and descender heights will be exact `1em`.

Tested with `metrics(54pt, "Hiragino Mincho ProN", "テレタイプ端末")`:

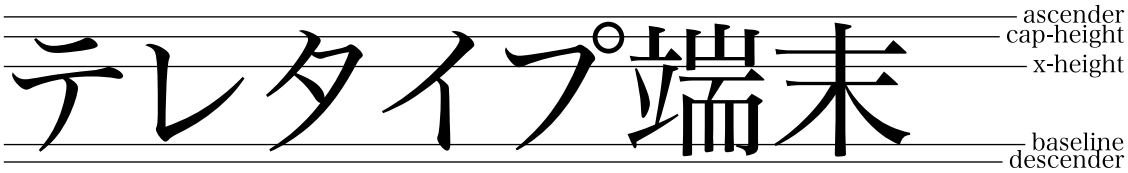

Figure 3: The metrics of Hiragino Mincho ProN

Since Typst will use metrics of the font which has the largest advance height amongst the list provided in `set text(font: ( … ))` to set up top and bottom edges of a line, leading might not work as expected in hybrid typesetting. This issue can be solved by passing the document to `metrics(use: metrics ⇒ { … })` like this:

```
#show: doc ⇒ metrics(font.size, font.main,
  // Retrieve the metrics of the CJK font
  use: metrics ⇒ {
    set text(
      font.size,
      font: ( font.latin, font.main ),
      features: ( "pkna", ),
      // Use CJK font's ascender/descender as top/bottom edges
      top-edge: metrics.ascender,
      bottom-edge: metrics.descender,
      // …
    )

    doc
  }
)
```