Python Figure Reference

The pages linked in the sidebar together form the exhaustive reference for all of the attributes in the core figure data structure (/python/figure-structure/) that the `plotly` library operates on. They are automatically-generated from the machine-readable Plotly.js schema reference (https://raw.githubusercontent.com/plotly/plotly.js/master/dist/plot-schema.json).

How to use these Reference pages

Figures are represented as trees with named nodes called "attributes". The root node of the tree has three top-level attributes: `data`, `layout` and `frames`. Attributes are referred to in text and in this reference by their full "path" i.e. the dot-delimited concatenation of their parents. For example `"layout.width"` refers to the attribute whose key is `"width"` inside a dict which is the value associated with a key `"layout"` at the root of the figure. If one of the parents is a list rather than a dict, a set of brackets is inserted in the path when referring to the attribute in the abstract, e.g. `"layout.annotations[].text"`. Finally, as explained below, the top-level "data" attribute defines a list of typed objects called "traces" with the schema dependent upon the type, and these attributes' paths are listed in this reference as `"data[type=scatter].name"`. When manipulating a `plotly.graph_objects.Figure` object (/python/creating-and-updating-figures/), attributes can be set either directly using Python object attributes e.g. `fig.layout.title.font.family="Open Sans"` or using update methods and "magic underscores" (/python/creating-and-updating-figures/#magic-underscore-notation) e.g. `fig.update_layout(title_font_family="Open Sans")`

When building a figure, it is *not necessary to populate every attribute* of every object. At render-time, the JavaScript layer will compute default values for each required unspecified attribute, depending upon the ones that are specified, as documented in this page. An example of this would be `layout.xaxis.range`, which may be specified explicitly, but if not will be computed based on the range of `x` values for every trace linked to that axis. The JavaScript layer will ignore unknown attributes or malformed values, although the `plotly.graph_objects` module provides Python-side validation for attribute values. Note also that if the `layout.template` key is present (as it is by default) (/python/templates/) then default values will be drawn first from the contents of the template and only if missing from there will the JavaScript layer infer further defaults. The built-in template can be disabled by setting `layout.template="none"`.