Beginner's Python Cheat Sheet - Plotly

What is Plotly?

Data visualization involves exploring data through visual representations. Plotly helps you make visually appealing representations of the data you're working with. Plotly is particularly well suited for visualizations that will be presented online, because it supports interactive elements.

Plotly express lets you see a basic version of your plot with just a few lines of code. Once you know the plot works for your data, you can refine the style of your plot.

Installing Plotly

Plotly Express requires the pandas library.

Installing Plotly with `pip`

```
$ python -m pip install --user plotly
$ python -m pip install --user pandas
```

Line graphs, scatter plots, and bar graphs

To make a plot with Plotly Express, you specify the data and then create a fig object. The call to `fig.show()` *opens the plot in a new browser tab. You have a plot in just two lines of code!*

Making a line graph

Plotly generates JavaScript code to render the plot file. If you're curious to see the code, open your browser's inspector tool when the plot opens.

```
import plotly.express as px

# Define the data.
x_values = list(range(11))
squares = [x**2 for x in x_values]

# Visualize the data.
fig = px.line(x=x_values, y=squares)
fig.show()
```

Making a scatter plot

To make a scatter plot, change `line()` *to* `scatter()`*. This is the point of Plotly Express; you can easily see your data in a variety of ways before committing to a more specific styling options.*

```
fig = px.scatter(x=x_values, y=squares)
```

Line graphs, scatter plots, and bar graphs (cont.)

Making a bar graph

```
fig = px.bar(x=x_values, y=squares)
```

Initial customizations

The functions that generate plots also accept parameters that specify titles, labels, and other formatting directives for your visualizations.

Adding a title and axis labels

The title is a string. The `labels` *dictionary lets you specify which aspects of the plot will have custom labels.*

```
import plotly.express as px

# Define the data.
x_values = list(range(11))
squares = [x**2 for x in x_values]

# Visualize the data.
title = "Square Numbers"
labels = {'x': 'Value', 'y': 'Square of Value'}

fig = px.scatter(x=x_values, y=squares,
        title=title, labels=labels)
fig.show()
```

More customizations in the plotting call

Plotly Express was designed to give you as much control as possible, using as little code as possible. Here's a small example of how much can be customized within a single plotting call.

Most of these arguments can be single values, or sequences that match the size of the overall dataset.

```
import plotly.express as px

x_values = list(range(11))
squares = [x**2 for x in x_values]

title = "Square Numbers"
labels = {'x': 'Value', 'y': 'Square of Value'}

fig = px.scatter(
    x=x_values,
    y=squares,
    title=title,
    labels=labels,
    size=squares,
    color=squares,
    opacity=0.5,
    width=1200,
    height=800,
)

fig.show()
```

Further customizations

You can make a wide variety of further customizations to a plot using the update methods. For example, `update_layout()` *gives you control of many formatting options.*

Using `update_layout()`

Here the `update_layout()` *method is used to change the font sizes, and change the tick mark spacing on the x-axis.*

```
import plotly.express as px

x_values = list(range(11))
squares = [x**2 for x in x_values]

title = "Square Numbers"
labels = {'x': 'Value', 'y': 'Square of Value'}

fig = px.scatter(
    x=x_values,
    y=squares,
    ...
)

fig.update_layout(
    title_font_size=30,
    xaxis_title_font_size=24,
    xaxis_dtick=1,
    xaxis_tickfont_size=16,
    yaxis_title_font_size=24,
    yaxis_tickfont_size=16,
)

fig.show()
```

Plotly Express documentation

Plotly's documentation is extensive and well-organized. There's a lot of it, though, so it can be hard to know where to begin. Start with an overview of Plotly Express at plotly.com/python/plotly-express. This page itself is helpful; make sure you also click on the documentation for the kinds of plots you use most often. These lead to pages full of discussions and examples.

Also see the Python API reference for `plotly` *at plotly.com/python-api-reference. This is a reference page showing all the different kinds of plots you can make with Plotly. If you click on any of the links, you can see all the arguments that can be included in plotting calls.*

Using a predefined theme

A theme is a set of styles applied to a visualization in Plotly. Themes are implemented with templates.

Using a theme

```
import plotly.express as px

# Define the data.
x_values = list(range(11))
squares = [x**2 for x in x_values]

# Visualize the data.
title = "Square Numbers"
labels = {'x': 'Value', 'y': 'Square of Value'}

fig = px.scatter(x=x_values, y=squares,
        title=title, labels=labels,
        template='plotly_dark')
fig.show()
```

Viewing all available themes

```
>>> import plotly.io as pio
>>> pio.templates
Templates configuration
-----------------------
    Default template: 'plotly'
    Available templates:
        ['ggplot2', 'seaborn',...,
         'ygridoff', 'gridon', 'none']
```

Adding traces to a Plotly Express plot

In Plotly, a trace is a dataset that can be plotted on a chart. You can add traces to existing Plotly Express plots. Additional plots need to be specified using the graph_objects module.

Using `fig.add_trace()`

```
import plotly.express as px
import plotly.graph_objects as go

days = list(range(1, 10))
highs = [60, 63, 68, 70, 68, 70, 66, 62, 64]
lows = [51, 54, 53, 57, 54, 56, 52, 53, 49]

# Start by plotting low temperaturs.
fig = px.line(x=days, y=lows)

# Add a new trace for the high temperatures.
new_trace = go.Scatter(x=days, y=highs,
        mode='lines')
fig.add_trace(new_trace)

fig.show()
```

Using Subplots

It's often useful to have multiple plots share the same axes. This is done using the `subplots` module.

Adding subplots to a figure

To use the `subplots` module, make a figure to hold all the charts that will be made. Then use the `add_trace()` method to add each data series to the overall figure.

All individual plots need to be made using the graph_objects module.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

x_values = list(range(11))
squares = [x**2 for x in x_values]
cubes = [x**3 for x in x_values]

# Make two subplots, sharing a y-axis.
fig = make_subplots(rows=1, cols=2,
        shared_yaxes=True)

# Start by plotting the square numbers.
squares_trace = go.Scatter(x=x_values,
        y=squares)
fig.add_trace(squares_trace, row=1, col=1)

# Add a new trace for the cubes.
cubes_trace = go.Scatter(x=x_values, y=cubes)
fig.add_trace(cubes_trace, row=1, col=2)

title = "Squares and Cubes"
fig.update_layout(title_text=title)

fig.show()
```

Further documentation

After exploring the Plotly Express documenation, look at Styling Plotly Express Figures in Python, at plotly.com/python/styling-plotly-express. This explains all the ways you can style and format plots. After that, the Python Figure Reference (plotly.com/python/reference/index) will be much more useful. It shows you all the possible settings you can change, with examples for each.

Make sure you read about "magic underscores" in Plotly, at plotly.com/python/creating-and-updating-figures. They take a little getting used to, but once you're familiar with the syntax they make it much easier to specify exactly the settings you want to modify.

If you're using subplots, read Subplots in Python at plotly.com/python/subplots. Also look at Graph Objects in Python at plotly.com/python/graph-objects, which are used to make individual plots in a subplot.

Plotting global datasets

Plotly has a variety of mapping tools. For example, if you have a set of points represented by latitude and longitude, you can create a scatter plot of those points overlaying a map.

The scattergeo chart type

Here's a map showing the location of three of the higher peaks in North America. If you hover over each point, you'll see its location and the name of the mountain.

```
import plotly.express as px

# Points in (lat, lon) format.
peak_coords = [
    (63.069, -151.0063),
    (60.5671, -140.4055),
    (46.8529, -121.7604),
]

# Make matching lists of lats, lons,
#   and labels.
lats = [pc[0] for pc in peak_coords]
lons = [pc[1] for pc in peak_coords]

peak_names = [
    "Denali",
    "Mt Logan",
    "Mt Rainier"
]
elevations = [20_000, 18_000, 14_000]

# Generate initial map.
title = "Selected High Peaks"
fig = px.scatter_geo(
    lat=lats,
    lon=lons,
    title=title,
    projection="natural earth",
    text=peak_names,
    size=elevations,
    scope="north america",
)

# Customize formatting options.
fig.update_layout(titlefont_size=24)
fig.update_traces(
    textposition="middle right",
    textfont_size=18,
)

fig.show()
```