Basic Plotting with matplotlib

You can show matplotlib figures directly in the notebook by using the `%matplotlib notebook` and `%matplotlib inline` magic commands.

`%matplotlib notebook` provides an interactive environment.

In [ ]:
```
%matplotlib notebook
```

In [ ]:
```
import matplotlib as mpl
mpl.get_backend()
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.plot?
```

In [ ]:
```
# because the default is the line style '-',
# nothing will be shown if we only pass in one point (3,2)
plt.plot(3, 2)
```

In [ ]:
```
# we can pass in '.' to plt.plot to indicate that we want
# the point (3,2) to be indicated with a marker '.'
plt.plot(3, 2, '.')
```

Let's see how to make a plot without using the scripting layer.

In [ ]:
```
# First let's set the backend without using mpl.use() from the scripting layer
from matplotlib.backends.backend_agg import FigureCanvasAgg
from matplotlib.figure import Figure

# create a new figure
fig = Figure()

# associate fig with the backend
canvas = FigureCanvasAgg(fig)

# add a subplot to the fig
ax = fig.add_subplot(111)

# plot the point (3,2)
ax.plot(3, 2, '.')

# save the figure to test.png
# you can see this figure in your Jupyter workspace afterwards by going to
# https://hub.coursera-notebooks.org/
canvas.print_png('test.png')
```

We can use html cell magic to display the image.

In [ ]:
```
%%html
<img src='test.png' />
```

In [ ]:
```
# create a new figure
plt.figure()

# plot the point (3,2) using the circle marker
plt.plot(3, 2, 'o')

# get the current axes
ax = plt.gca()

# Set axis properties [xmin, xmax, ymin, ymax]
ax.axis([0,6,0,10])
```

In [ ]:
```
# create a new figure
plt.figure()

# plot the point (1.5, 1.5) using the circle marker
plt.plot(1.5, 1.5, 'o')
# plot the point (2, 2) using the circle marker
plt.plot(2, 2, 'o')
# plot the point (2.5, 2.5) using the circle marker
plt.plot(2.5, 2.5, 'o')
```

In [ ]:
```
# get current axes
ax = plt.gca()
# get all the child objects the axes contains
ax.get_children()
```

Scatterplots

In [ ]:
```
import numpy as np

x = np.array([1,2,3,4,5,6,7,8])
y = x

plt.figure()
plt.scatter(x, y) # similar to plt.plot(x, y, '.'), but the underlying child obje
```

In [ ]:
```
import numpy as np

x = np.array([1,2,3,4,5,6,7,8])
y = x

# create a list of colors for each point to have
# ['green', 'green', 'green', 'green', 'green', 'green', 'green', 'red']
colors = ['green']*(len(x)-1)
colors.append('red')

plt.figure()

# plot the point with size 100 and chosen colors
plt.scatter(x, y, s=100, c=colors)
```

In [ ]:
```
# convert the two lists into a list of pairwise tuples
zip_generator = zip([1,2,3,4,5], [6,7,8,9,10])

print(list(zip_generator))
# the above prints:
# [(1, 6), (2, 7), (3, 8), (4, 9), (5, 10)]

zip_generator = zip([1,2,3,4,5], [6,7,8,9,10])
# The single star * unpacks a collection into positional arguments
print(*zip_generator)
# the above prints:
# (1, 6) (2, 7) (3, 8) (4, 9) (5, 10)
```

In [ ]:
```
# use zip to convert 5 tuples with 2 elements each to 2 tuples with 5 elements ea
print(list(zip((1, 6), (2, 7), (3, 8), (4, 9), (5, 10))))
# the above prints:
# [(1, 2, 3, 4, 5), (6, 7, 8, 9, 10)]


zip_generator = zip([1,2,3,4,5], [6,7,8,9,10])
# let's turn the data back into 2 lists
x, y = zip(*zip_generator) # This is like calling zip((1, 6), (2, 7), (3, 8), (4,
print(x)
print(y)
# the above prints:
# (1, 2, 3, 4, 5)
# (6, 7, 8, 9, 10)
```

In [ ]:
```
plt.figure()
# plot a data series 'Tall students' in red using the first two elements of x and
plt.scatter(x[:2], y[:2], s=100, c='red', label='Tall students')
# plot a second data series 'Short students' in blue using the last three element
plt.scatter(x[2:], y[2:], s=100, c='blue', label='Short students')
```

In [ ]:
```
# add a label to the x axis
plt.xlabel('The number of times the child kicked a ball')
# add a label to the y axis
plt.ylabel('The grade of the student')
# add a title
plt.title('Relationship between ball kicking and grades')
```

In [ ]:
```
# add a legend (uses the labels from plt.scatter)
plt.legend()
```

In [ ]:
```
# add the legend to loc=4 (the lower right hand corner), also gets rid of the fra
plt.legend(loc=4, frameon=False, title='Legend')
```

In [ ]:
```
# get children from current axes (the legend is the second to last item in this l
plt.gca().get_children()
```

In [ ]:
```
# get the legend from the current axes
legend = plt.gca().get_children()[-2]
```

In [ ]:
```
# you can use get_children to navigate through the child artists
legend.get_children()[0].get_children()[1].get_children()[0].get_children()
```

In [ ]:
```
# import the artist class from matplotlib
from matplotlib.artist import Artist

def rec_gc(art, depth=0):
    if isinstance(art, Artist):
        # increase the depth for pretty printing
        print("  " * depth + str(art))
        for child in art.get_children():
            rec_gc(child, depth+2)

# Call this function on the legend artist to see what the legend is made up of
rec_gc(plt.legend())
```

Line Plots

In [ ]:
```
import numpy as np

linear_data = np.array([1,2,3,4,5,6,7,8])
exponential_data = linear_data**2

plt.figure()
# plot the linear data and the exponential data
plt.plot(linear_data, '-o', exponential_data, '-o')
```

In [ ]:
```
# plot another series with a dashed red line
plt.plot([22,44,55], '--r')
```

In [ ]:
```
plt.xlabel('Some data')
plt.ylabel('Some other data')
plt.title('A title')
# add a legend with legend entries (because we didn't have labels when we plotted
plt.legend(['Baseline', 'Competition', 'Us'])
```

In [ ]:
```
# fill the area between the linear data and exponential data
plt.gca().fill_between(range(len(linear_data)),
                       linear_data, exponential_data,
                       facecolor='blue',
                       alpha=0.25)
```

Let's try working with dates!

In [ ]:
```
plt.figure()

observation_dates = np.arange('2017-01-01', '2017-01-09', dtype='datetime64[D]')

plt.plot(observation_dates, linear_data, '-o',  observation_dates, exponential_da
```

Let's try using pandas

In [ ]:
```
import pandas as pd

plt.figure()
observation_dates = np.arange('2017-01-01', '2017-01-09', dtype='datetime64[D]')
observation_dates = map(pd.to_datetime, observation_dates) # trying to plot a map
plt.plot(observation_dates, linear_data, '-o',  observation_dates, exponential_da
```

In [ ]:
```
plt.figure()
observation_dates = np.arange('2017-01-01', '2017-01-09', dtype='datetime64[D]')
observation_dates = list(map(pd.to_datetime, observation_dates)) # convert the ma
plt.plot(observation_dates, linear_data, '-o',  observation_dates, exponential_da
```

In [ ]:
```
x = plt.gca().xaxis

# rotate the tick labels for the x axis
for item in x.get_ticklabels():
    item.set_rotation(45)
```

In [ ]:
```
# adjust the subplot so the text doesn't run off the image
plt.subplots_adjust(bottom=0.25)
```

In [ ]:
```
ax = plt.gca()
ax.set_xlabel('Date')
ax.set_ylabel('Units')
ax.set_title('Exponential vs. Linear performance')
```

In [ ]:
```
# you can add mathematical expressions in any text element
ax.set_title("Exponential ($x^2$) vs. Linear ($x$) performance")
```

Bar Charts

In [ ]:
```
plt.figure()
xvals = range(len(linear_data))
plt.bar(xvals, linear_data, width = 0.3)
```

In [ ]:
```
new_xvals = []

# plot another set of bars, adjusting the new xvals to make up for the first set o
for item in xvals:
    new_xvals.append(item+0.3)

plt.bar(new_xvals, exponential_data, width = 0.3 ,color='red')
```

In [ ]:
```
from random import randint
linear_err = [randint(0,15) for x in range(len(linear_data))]

# This will plot a new set of bars with errorbars using the list of random error
plt.bar(xvals, linear_data, width = 0.3, yerr=linear_err)
```

In [ ]:
```
# stacked bar charts are also possible
plt.figure()
xvals = range(len(linear_data))
plt.bar(xvals, linear_data, width = 0.3, color='b')
plt.bar(xvals, exponential_data, width = 0.3, bottom=linear_data, color='r')
```

In [ ]:
```
# or use barh for horizontal bar charts
plt.figure()
xvals = range(len(linear_data))
plt.barh(xvals, linear_data, height = 0.3, color='b')
plt.barh(xvals, exponential_data, height = 0.3, left=linear_data, color='r')
```