Setup

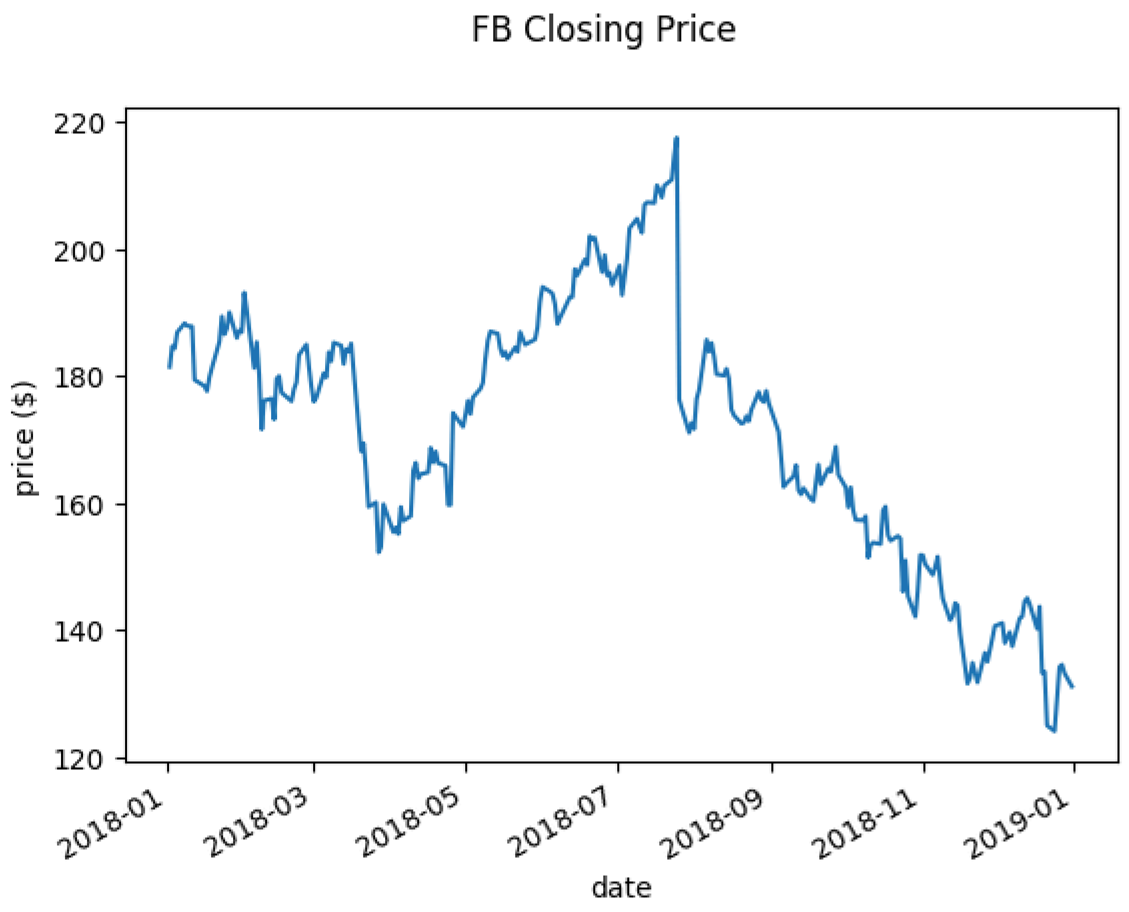
In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

fb = pd.read_csv(
    '/content/fb_stock_prices_2018.csv', index_col='date', parse_dates=True
)
```

Title and Axis Labels

- plt.suptitle() adds a title to plots and subplots
- plt.title() adds a title to a single plot. Note if you use subplots, it will only put the title on the last subplot, so you will need to use plt.suptitle()
- plt.xlabel() labels the x-axis
- plt.ylabel() labels the y-axis

In [ ]:
```
fb.close.plot()
plt.suptitle('FB Closing Price')
plt.xlabel('date')
plt.ylabel('price ($)')
```

Out[ ]:
```
Text(0, 0.5, 'price ($)')
```

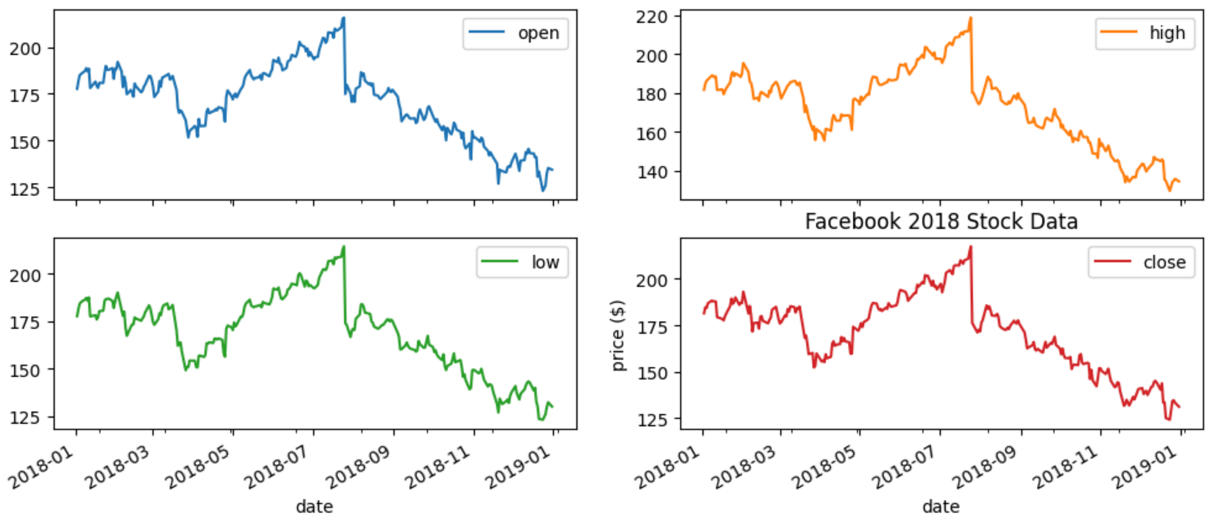


plt.suptitle() vs. plt.title()

Check out what happens when we call plt.title() with subplots:

```
In [ ]: fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
        plt.title('Facebook 2018 Stock Data')
        plt.xlabel('date')
        plt.ylabel('price ($)')
```

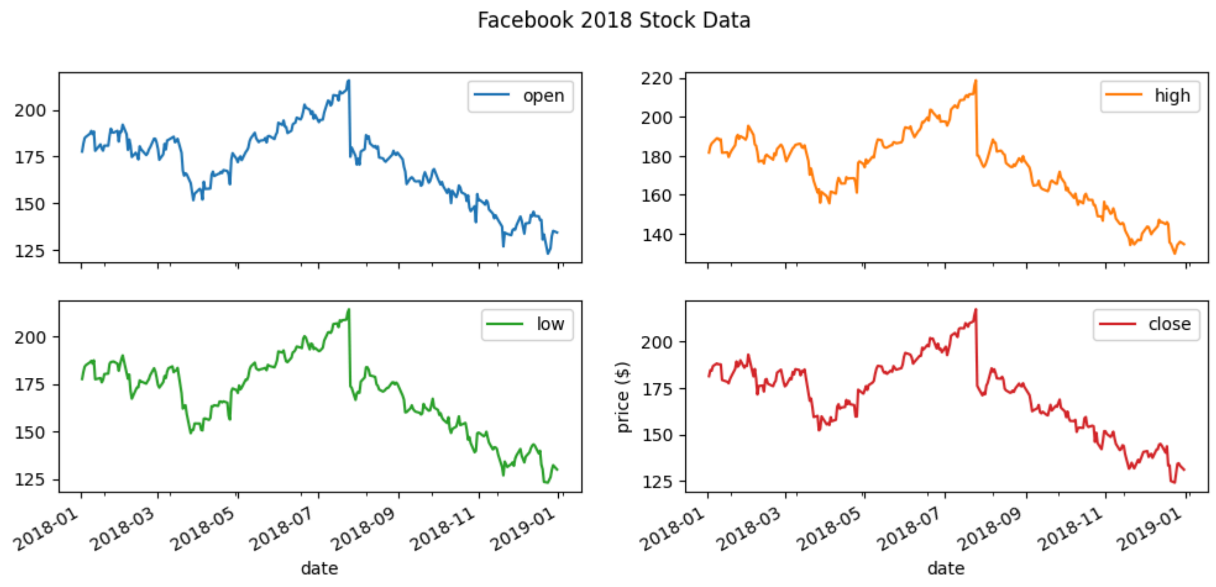
```
Out[ ]: Text(0, 0.5, 'price ($)')
```



Simply getting into the habit of using plt.suptitle() instead of plt.title() will save you this confusion:

```
In [ ]: fb.iloc[:,:4].plot(subplots=True, layout=(2, 2), figsize=(12, 5))
        plt.suptitle('Facebook 2018 Stock Data')
        plt.xlabel('date')
        plt.ylabel('price ($)')
```

```
Out[ ]: Text(0, 0.5, 'price ($)')
```

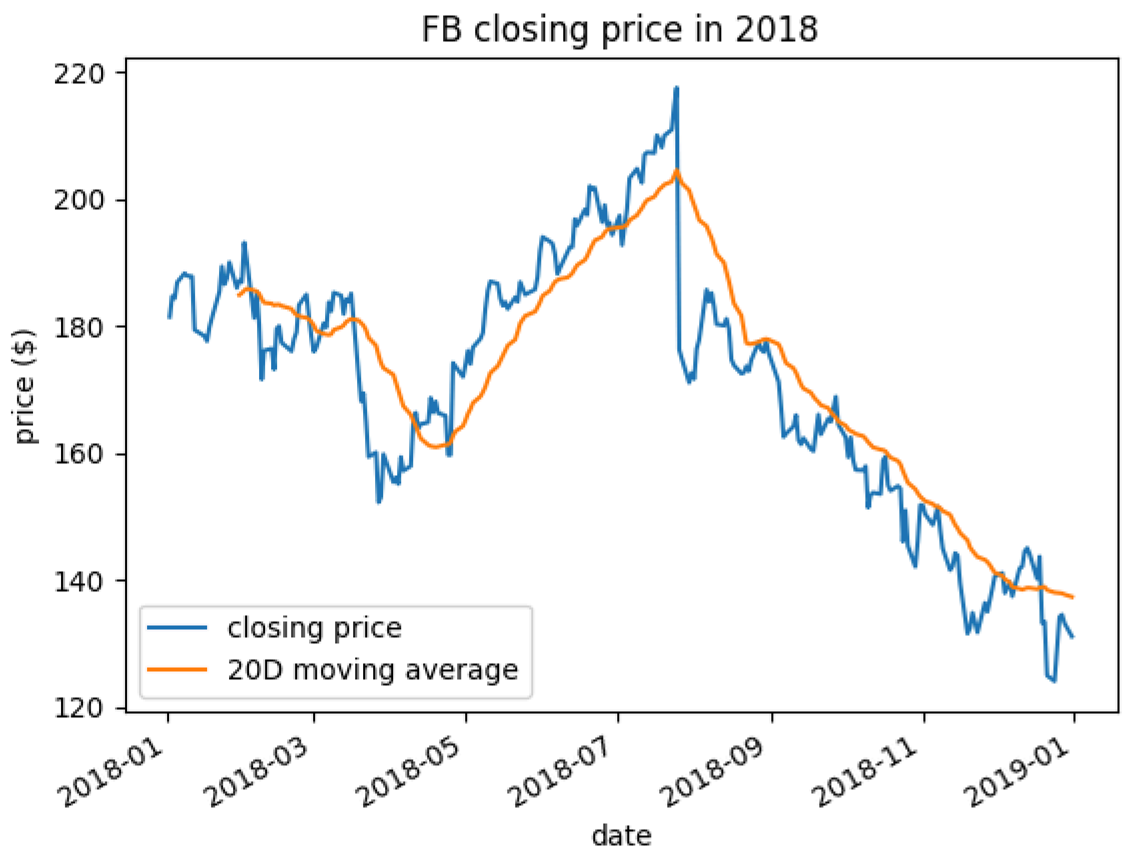


Legends

plt.legend() adds a legend to the plot. We can specify where to place it with the loc parameter:

```
In [ ]: fb.assign(
            ma=lambda x: x.close.rolling(20).mean()
        ).plot(
            y=['close', 'ma'],
            title='FB closing price in 2018',
            label=['closing price', '20D moving average']
        )
        plt.legend(loc='lower left')
        plt.ylabel('price ($)')
```

```
Out[ ]: Text(0, 0.5, 'price ($)')
```



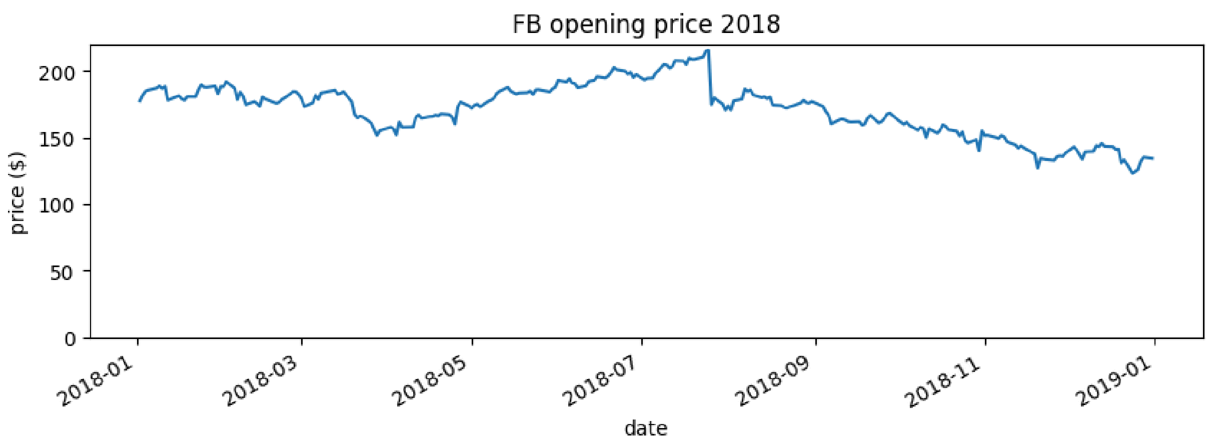
Formatting Axes

Specifying axis limits

plt.xlim() and plt.ylim() can be used to specify the minimum and maximum values for the axis. Passing None will have matplotlib determine the limit.

In [ ]:
```
fb.open.plot(figsize=(10, 3), title='FB opening price 2018')
plt.ylim(0, None)
plt.ylabel('price ($)')
```

Out[ ]:
```
Text(0, 0.5, 'price ($)')
```

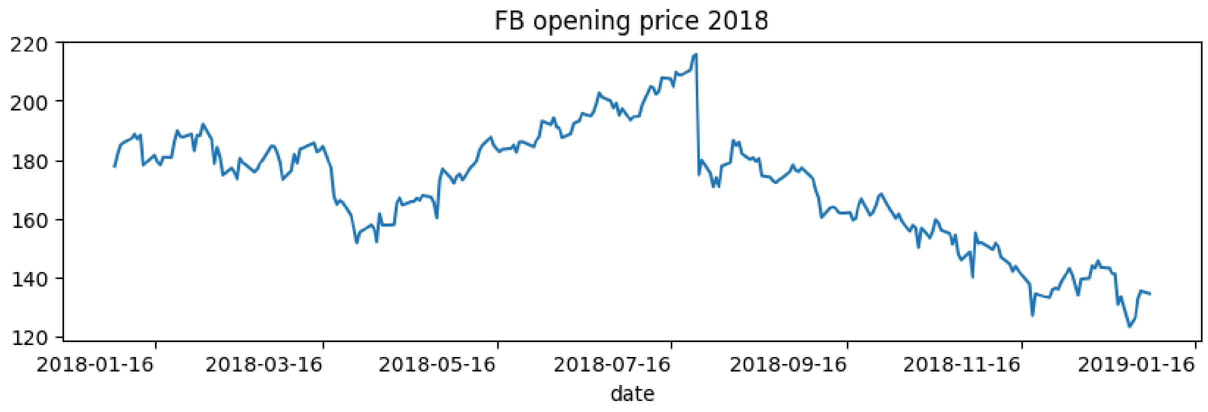


Formatting the Axis Ticks

We can use plt.xticks() and plt.yticks() to provide tick labels and specify, which ticks to show. Here, we show every other month:

In [ ]:
```
import calendar

fb.open.plot(figsize=(10, 3), rot=0, title='FB opening price 2018')
locs, labels = plt.xticks()
plt.xticks(locs + 15 , calendar.month_abbr[1:12])
plt.ylabel('price ($)')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-30-68db9cefce64> in <cell line: 5>()
      3 fb.open.plot(figsize=(10, 3), rot=0, title='FB opening price 2018')
      4 locs, labels = plt.xticks()
----> 5 plt.xticks(locs + 15 , calendar.month_abbr[1:12])
      6 plt.ylabel('price ($)')

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py in xticks(ticks, label
s, minor, **kwargs)
   1891             l._internal_update(kwargs)
   1892     else:
-> 1893         labels = ax.set_xticklabels(labels, minor=minor, **kwargs)
   1894 
   1895     return locs, labels

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_base.py in wrapper(self, *a
rgs, **kwargs)
     72 
     73         def wrapper(self, *args, **kwargs):
---> 74             return get_method(self)(*args, **kwargs)
     75 
     76         wrapper.__module__ = owner.__module__

/usr/local/lib/python3.10/dist-packages/matplotlib/_api/deprecation.py in wrapper(*a
rgs, **kwargs)
    295                 f"for the old name will be dropped %(removal)s.")
    296             kwargs[new] = kwargs.pop(old)
--> 297         return func(*args, **kwargs)
    298 
    299     # wrapper() must keep the same documented signature as func(): if we

/usr/local/lib/python3.10/dist-packages/matplotlib/axis.py in set_ticklabels(self, l
abels, minor, fontdict, **kwargs)
   1967             # remove all tick labels, so only error for > 0 labels
   1968             if len(locator.locs) != len(labels) and len(labels) != 0:
-> 1969                 raise ValueError(
   1970                     "The number of FixedLocator locations"
   1971                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (7), usually from a call to set_tic
ks, does not match the number of labels (11).
```



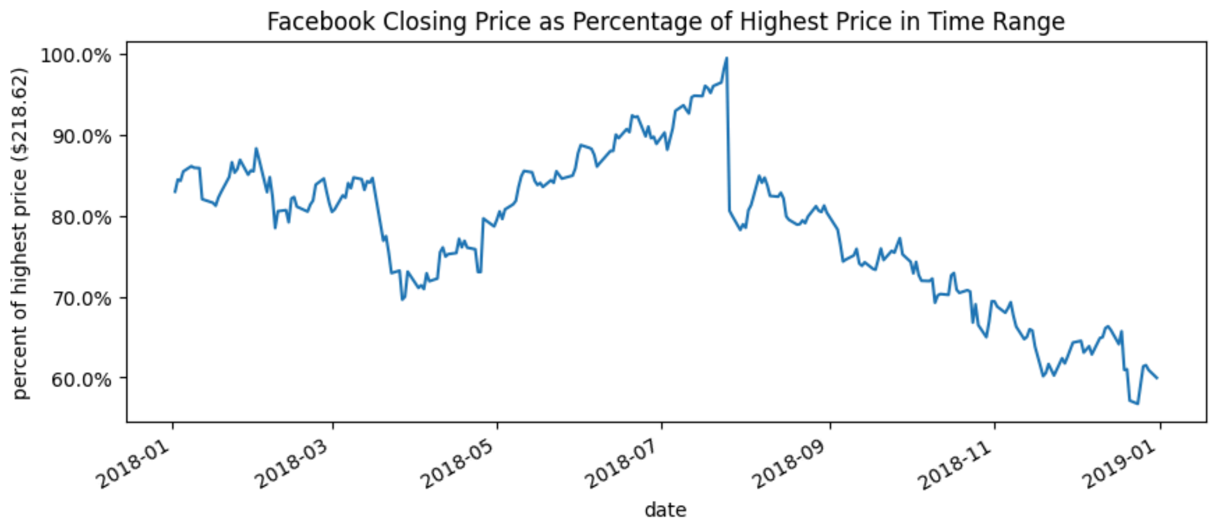
Using ticker

PercentFormatter

We can use ticker.PercentFormatter and specify the denominator (xmax) to use when calculating the percentages. This gets passed to the set_major_formatter() method of the xaxis or yaxis on the Axes .

In [ ]:
```
import matplotlib.ticker as ticker

ax = fb.close.plot(
    figsize=(10, 4),
    title='Facebook Closing Price as Percentage of Highest Price in Time Range'
)
ax.yaxis.set_major_formatter(
    ticker.PercentFormatter(xmax=fb.high.max())
)
ax.set_yticks([
    fb.high.max()*pct for pct in np.linspace(0.6, 1, num=5)
]) # show round percentages only (60%, 80%, etc.)
ax.set_ylabel(f'percent of highest price (${fb.high.max()})')
```

Out[ ]:
```
Text(0, 0.5, 'percent of highest price ($218.62)')
```

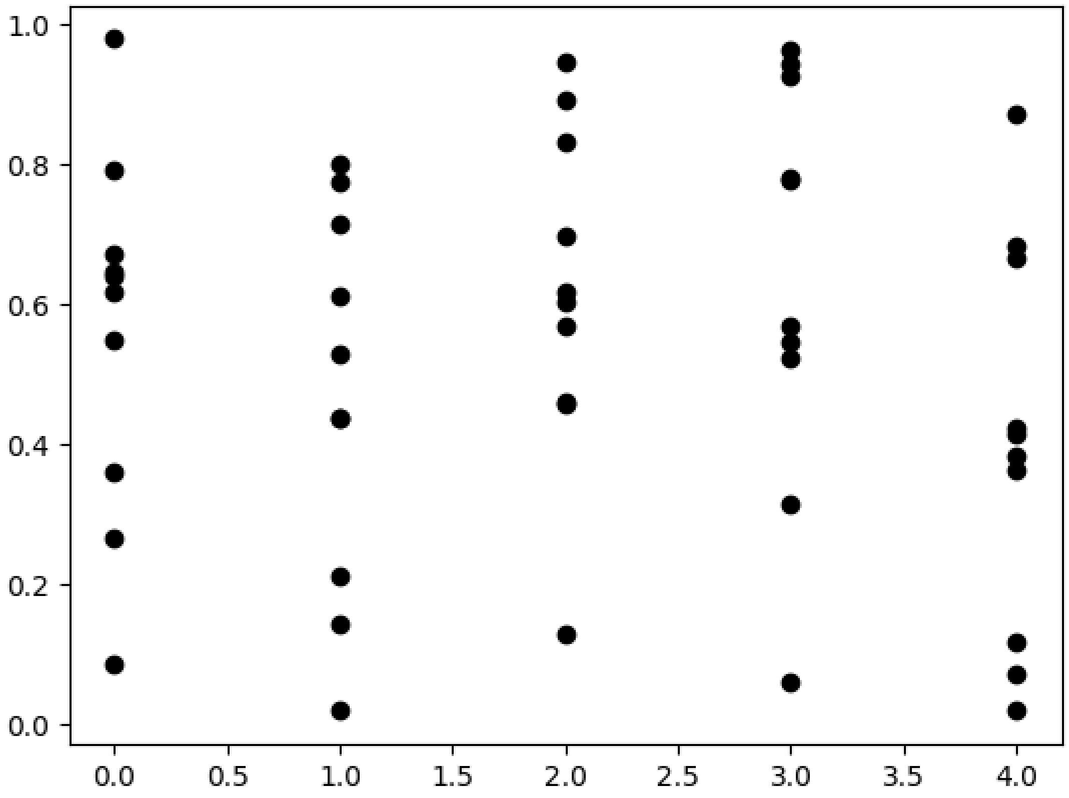


MultipleLocator

Say we have the following data. The points only take on integer values for x .

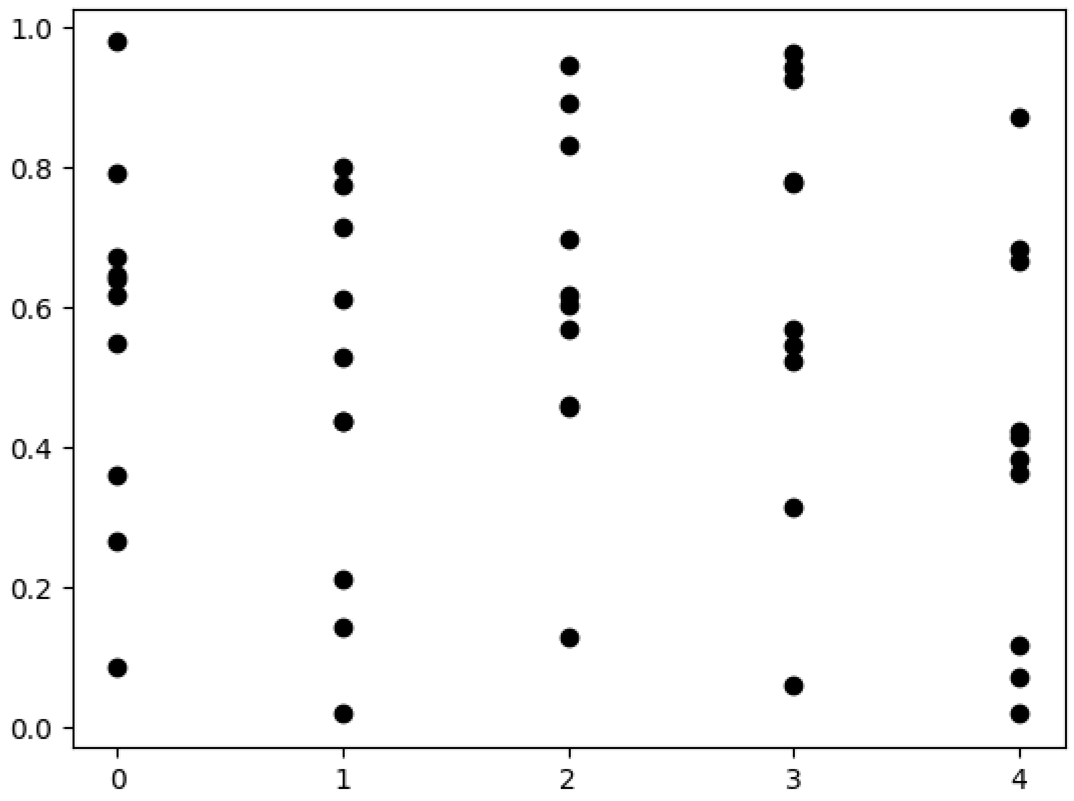
In [ ]:
```
fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7cd2387364a0>]
```



If we don't want to show decimal values on the x-axis, we can use the MultipleLocator . This will give ticks for all multiples of a number specified with the base parameter. To get integer values, we use base=1 :

In [ ]:
```
fig, ax = plt.subplots(1, 1)
np.random.seed(0)
ax.plot(np.tile(np.arange(0, 5), 10), np.random.rand(50), 'ko')
ax.get_xaxis().set_major_locator(
    ticker.MultipleLocator(base=1)
)
```


1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4