Every Boilermaker Engineer Codes: 101
Entry-Level Programming in Python

Lecture 10

Dr. John H. Cole
<jhcole@purdue.edu>

PURDUE
UNIVERSITY®

COLLEGE OF ENGINEERING

Spring 2021

Table of Contents

What's pip?

- pip is Python's standard package installer
- pip is a recursive acronym for "Pip Installs Packages"
- most python installations include pip by default

Check that you have pip installed

Terminal

```
$ python -m pip --version
pip 20.0.2 from ... (python 3.8)
```

Success pip is already installed!

Installing pip

If pip is not already installed, you will see something like this

Terminal

```
$ python -m pip --version
/some/path/python: No module named pip
```

To install pip, run the following command in a terminal.

Terminal

```
$ python -m ensurepip --user
```

Updating pip

- update pip before installing packages

Terminal

```
$ python -m pip install --upgrade pip
Collecting pip
  Downloading pip-20.2.4-py2.py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 6.4 MB/s
Installing collected packages: pip
Successfully installed pip-20.2.4
```

Using pip

- use the following to run pip as a module from python
- ensures pip installs for the correct Python version

Terminal

```
$ python -m pip install --user some_package
```

- pip can be run directly from the terminal
- you will see this suggested a lot online
- don't do this

Terminal

```
$ pip install some_package
```

Using pip to Install matplotlib 1

- we use the matplotlib package to make plots in python
- use an interactive session to see if it is installed

Terminal

```
>>> import matplotlib
>>> print(matplotlib.__version__)
3.3.3
>>>
```

Terminal

```
>>> import matplotlib
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'matplotlib'
>>>
```

Using pip to Install matplotlib 2

- use the following command to install the `matplotlib` package

Terminal

```
$ python -m pip install --user matplotlib
```

Using pip to Install matplotlib 3

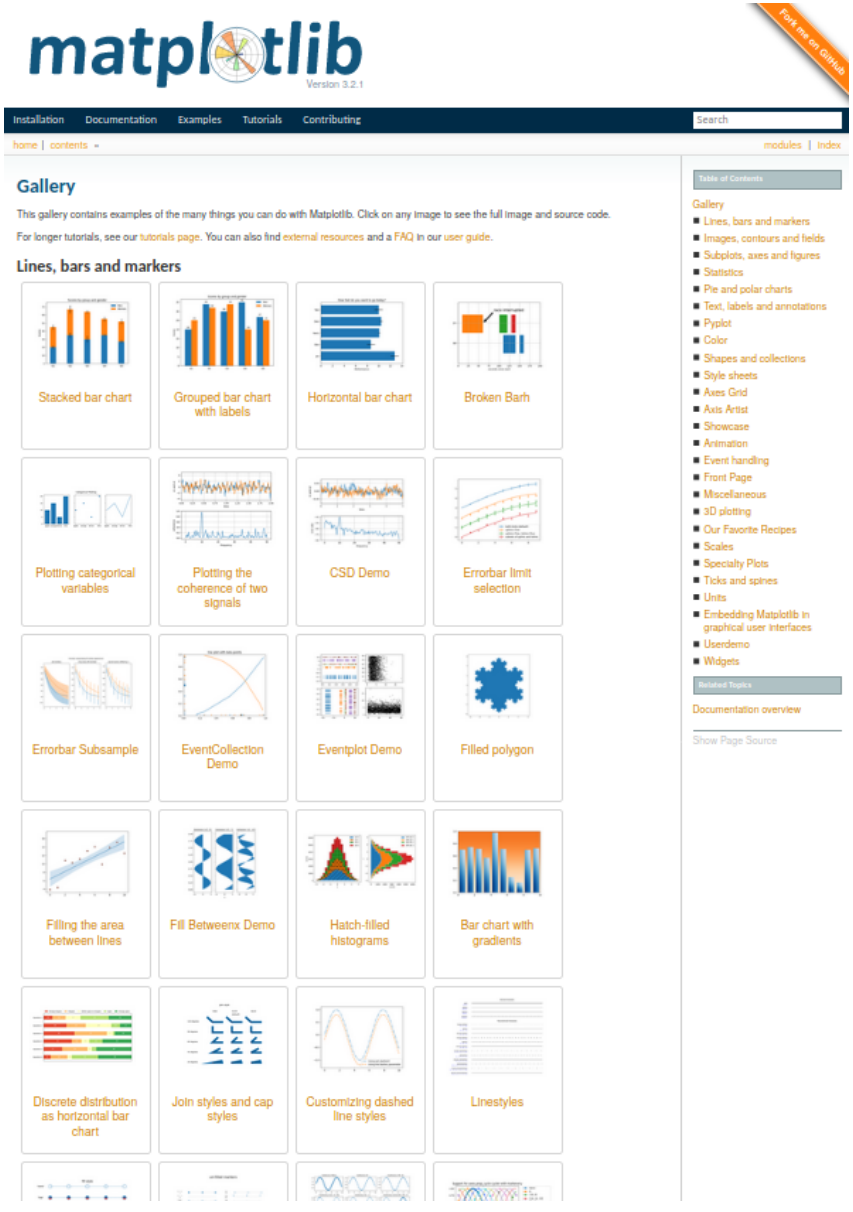
Terminal

```
$ python -m pip install --user matplotlib
Collecting matplotlib
  Downloading matplotlib-3.3.3-cp38-cp38-manylinux1_x86_64.whl (11.6 MB)
     |████████████████████████████████| 11.6 MB 497 kB/s
Collecting pillow>=6.2.0
  Downloading Pillow-8.0.1-cp38-cp38-manylinux1_x86_64.whl (2.2 MB)
     |████████████████████████████████| 2.2 MB 15.1 MB/s
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
     |████████████████████████████████| 67 kB 7.3 MB/s
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting python-dateutil>=2.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
     |████████████████████████████████| 227 kB 21.6 MB/s
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp38-cp38-manylinux1_x86_64.whl (1.2 MB)
     |████████████████████████████████| 1.2 MB 25.9 MB/s
Collecting numpy>=1.15
  Downloading numpy-1.19.4-cp38-cp38-manylinux2010_x86_64.whl (14.5 MB)
     |████████████████████████████████| 14.5 MB 13.1 MB/s
Collecting six
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: pillow, pyparsing, six, cycler, python-dateutil, kiwisolver, numpy,
matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.3.3 numpy-1.19.4 pillow-8.0.1
pyparsing-2.4.7 python-dateutil-2.8.1 six-1.15.0
```

What's matplotlib?

`matplotlib`

- a comprehensive library for visualizations in Python
  - static
  - animated
  - interactive
- check out some examples at: matplotlib.org/gallery



What's pyplot?

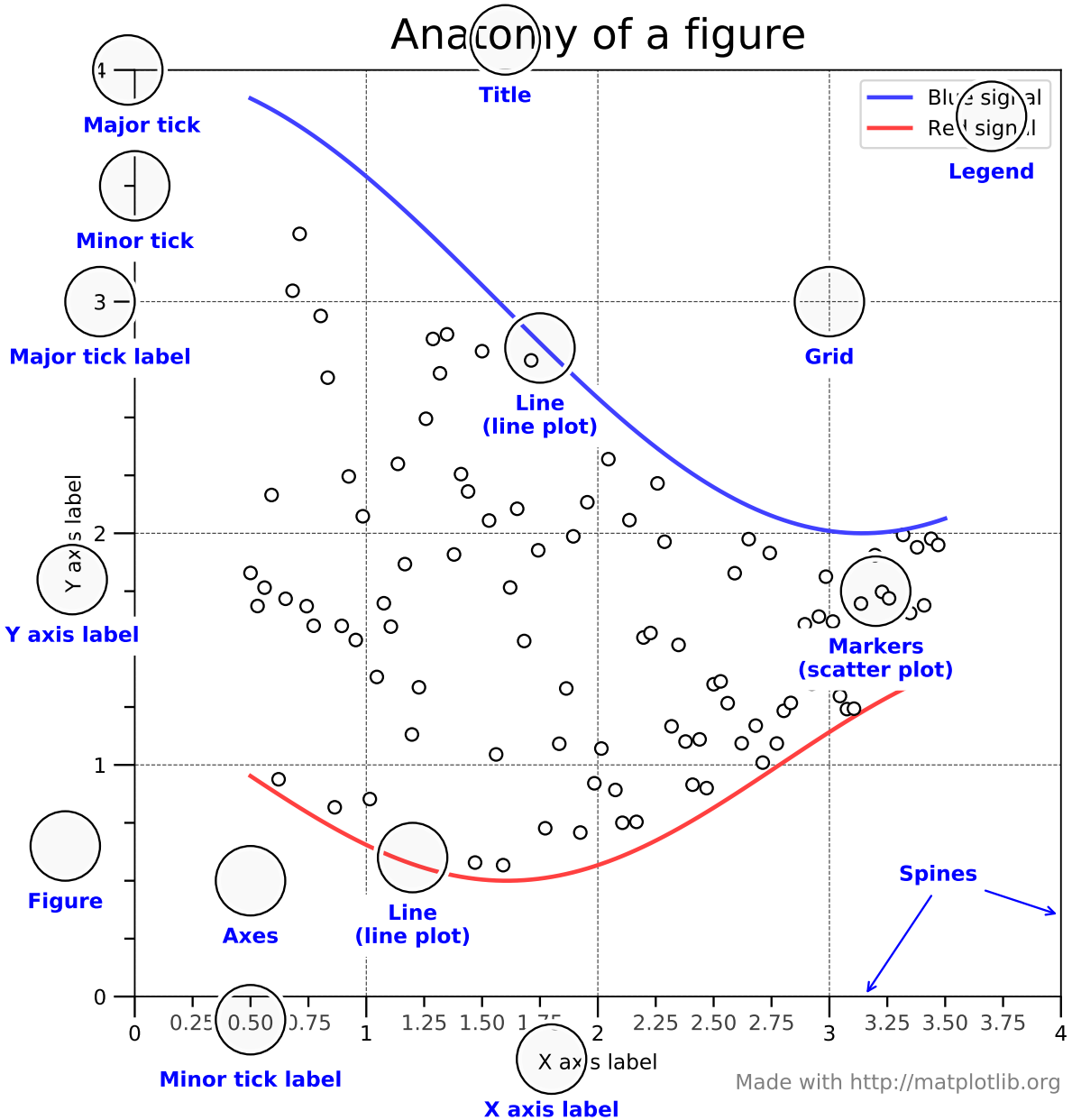
`pyplot`

- a submodule of `matplotlib`
- provides the primary interface to `matplotlib`
- often alias `pyplot` to `plt`

Terminal

```
>>> import matplotlib.pyplot as plt
```

Figure Anatomy



- everything is contained in a figure object
- the figure object contains one or more axes objects (the data space)
- this figure has a single axes object

fig, ax = plt.subplots()

`fig, ax = plt.subplots()`

- creates a figure with a single axes object
- we will use these to control the figure

Terminal

```
>>> fig, ax = plt.subplots()
```

`fig` is the figure object

`ax` is the axes object

ax.plot(x, y)

`ax.plot(x, y)` create a line plot by connecting the points (x, y).

Terminal

```
>>> ax.plot([1,2,3,4], [1,2,5,3])
[<matplotlib.lines.Line2D object at 0x7f3232da18b0>]
```

- nothing is drawn on the screen yet

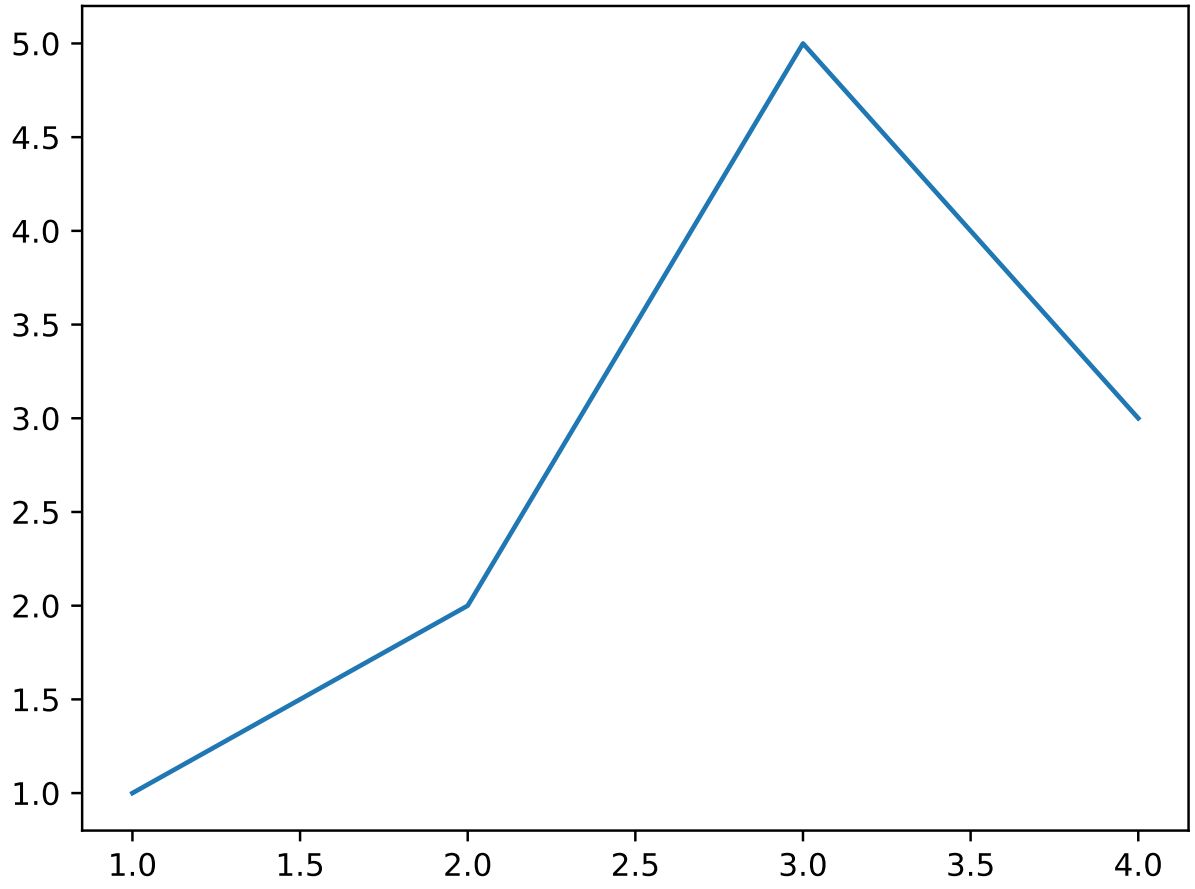
fig.show()

`fig.show()` display the figure

Terminal

```
>>> fig.show()
```

- only use this method in interactive mode



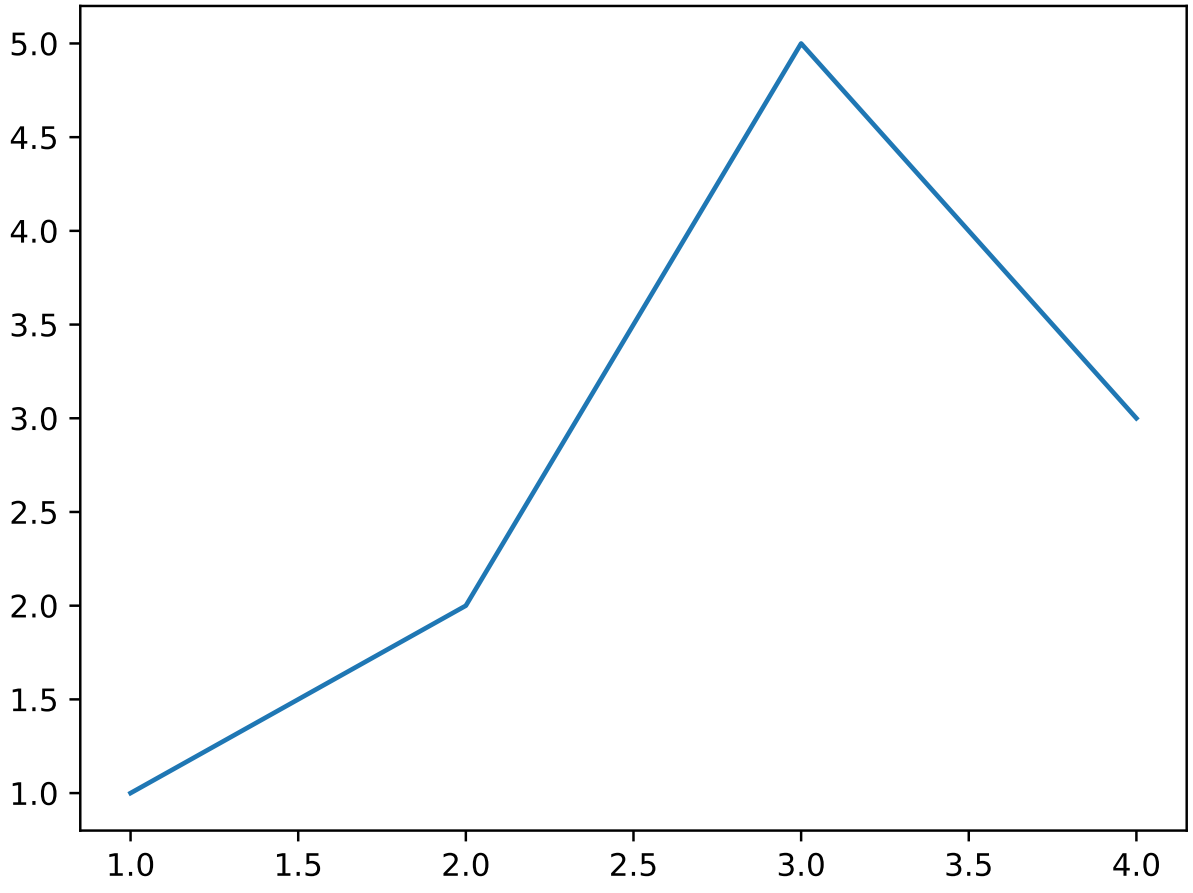
plt.show()

`plt.show()` display the figure and block until it is closed

Terminal

```
>>> plt.show()
```

- use this method in your programs

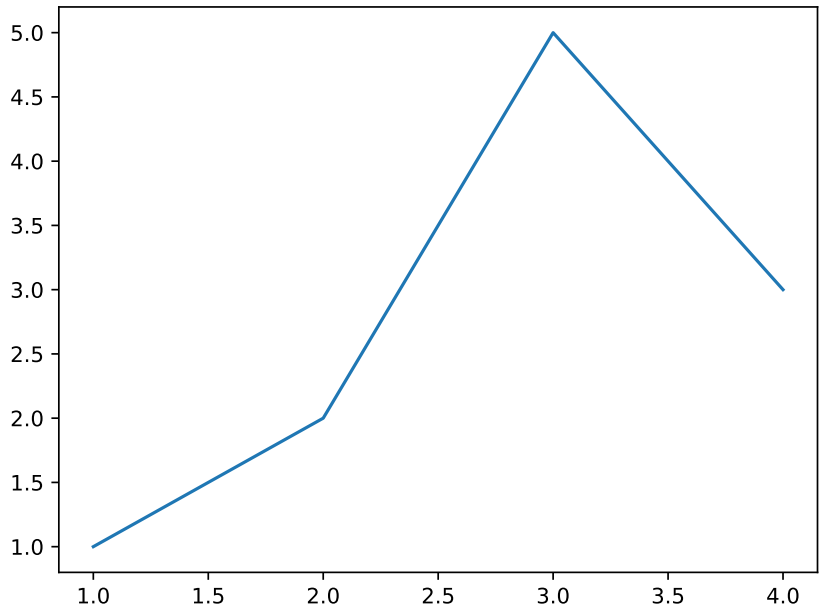


A Simple Plot

We can do the same thing in a .py file.

Editor - plot_1.py

```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots()
4 x = [1, 2, 3, 4]
5 y = [1, 2, 5, 3]
6 ax.plot(x, y)
7 plt.show()
```

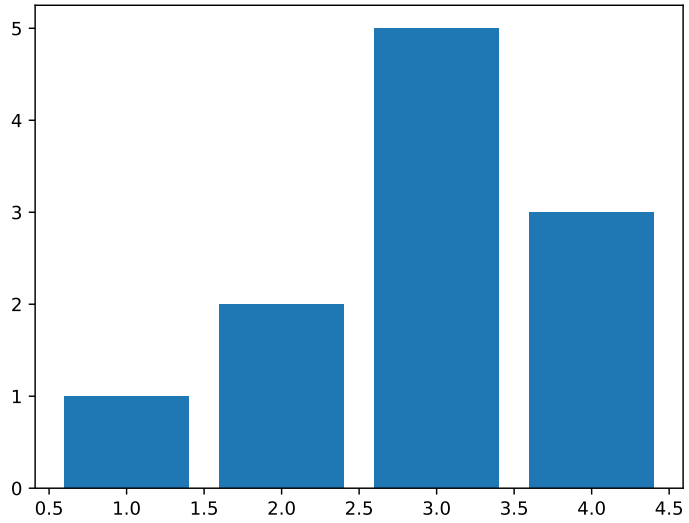


ax.bar()

`ax.bar()` plot a bar chart

Editor - bar_1.py

```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots()
4 x = [1, 2, 3, 4]
5 y = [1, 2, 5, 3]
6 ax.bar(x, y)
7
8 plt.show()
```

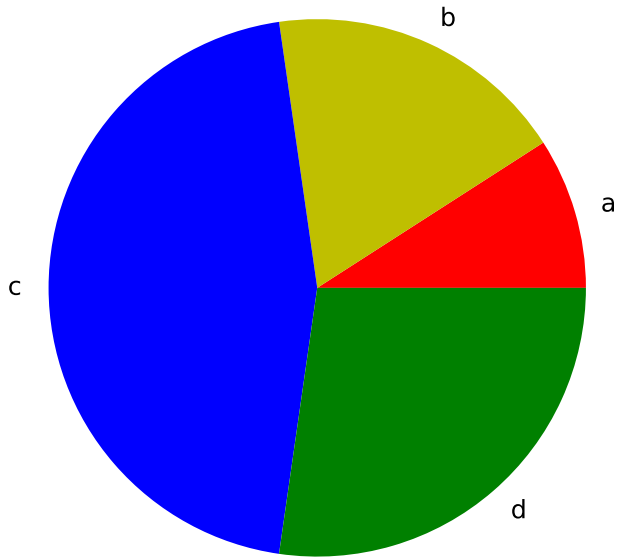


ax.pie()

`ax.pie()` plot a pie chart

Editor - pie_1.py

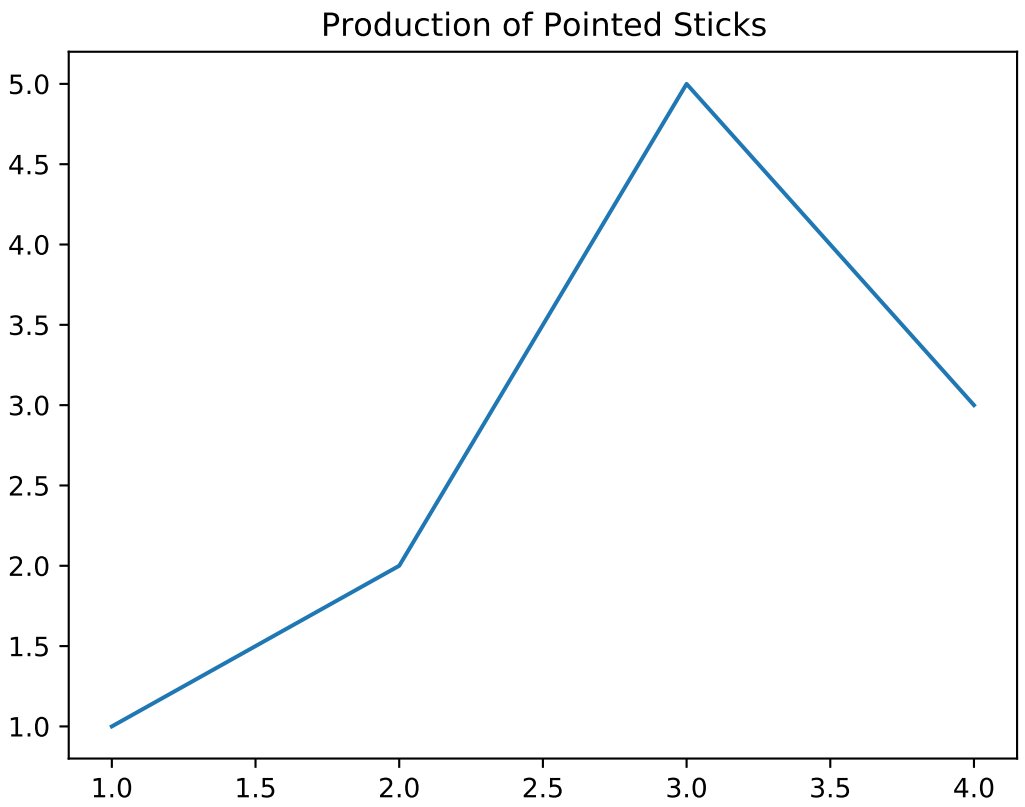
```
1 import matplotlib.pyplot as plt
2
3 fig, ax = plt.subplots()
4 y = [1, 2, 5, 3]
5 c = ('r','y','b','g')
6 l = ('a','b','c','d')
7 ax.pie(y, colors=c, labels=l)
8 plt.show()
```



set_title()

`ax.set_title(s)` title axes with string s

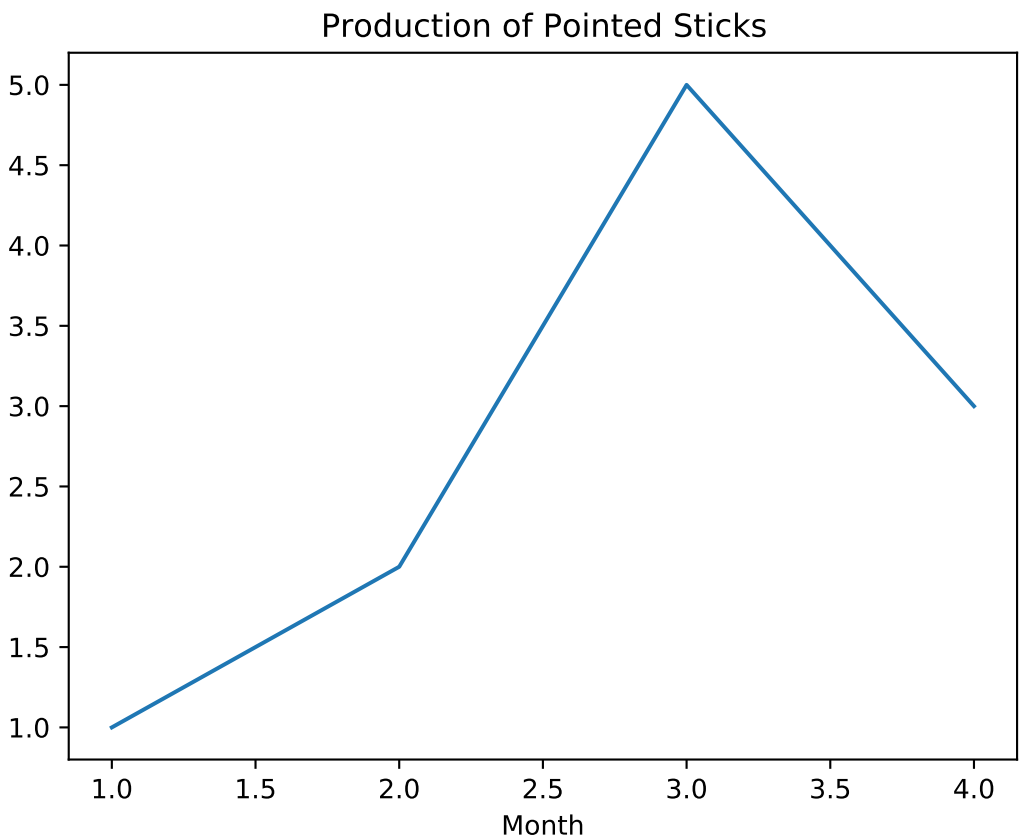
```
1 ax.set_title('Production of Pointed Sticks')
```



set_xlabel()

`ax.set_xlabel(s)` label x-axis with string s

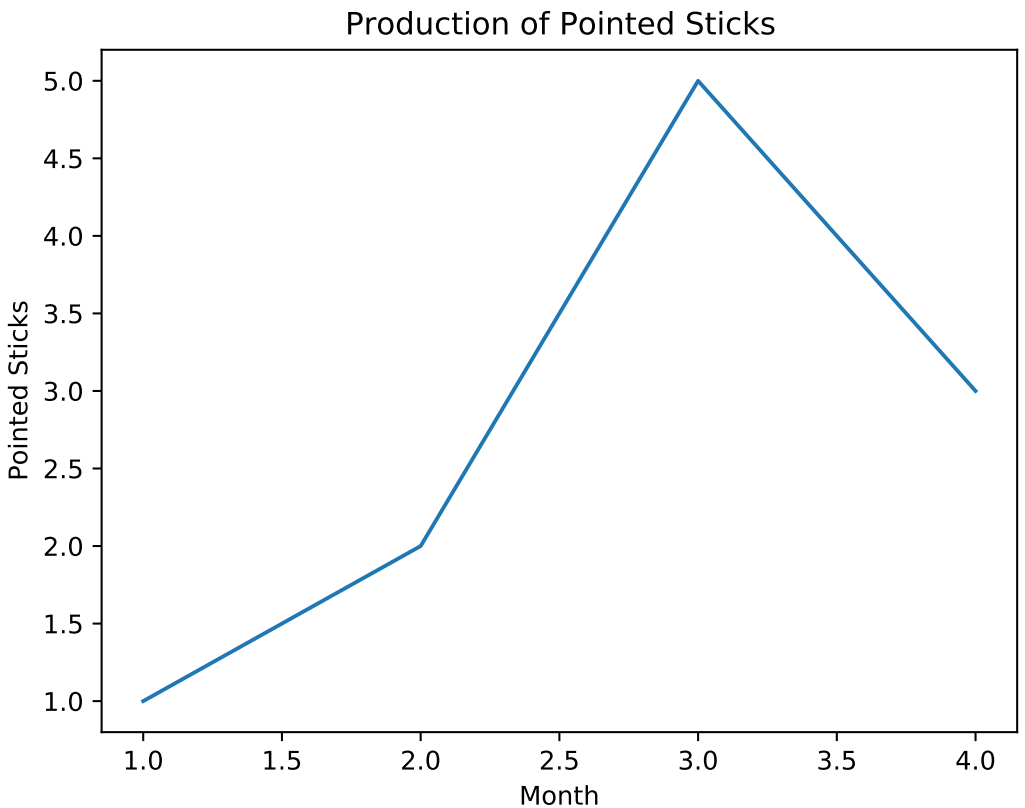
```
1 ax.set_xlabel('Month')
```



set_ylabel()

`ax.set_ylabel(s)` label y-axis with string s

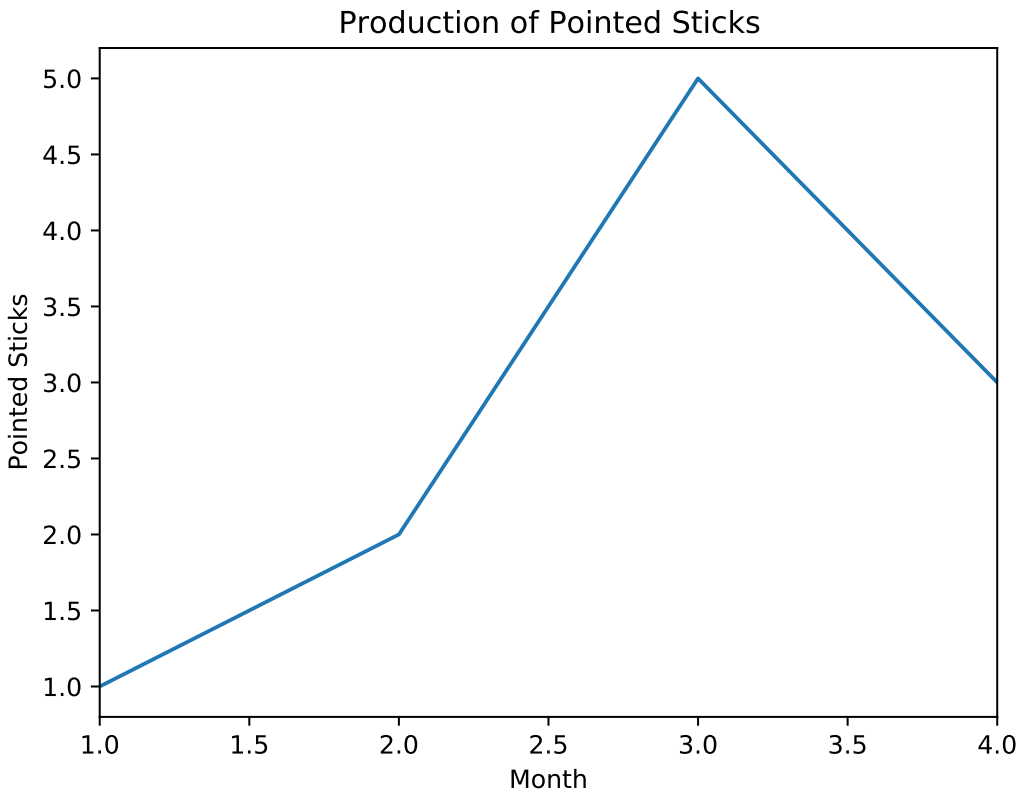
```
1 ax.set_ylabel('Pointed Sticks')
```



set_xlim()

`ax.set_xlim(left, right)` set the `left` and `right` x-axis limits

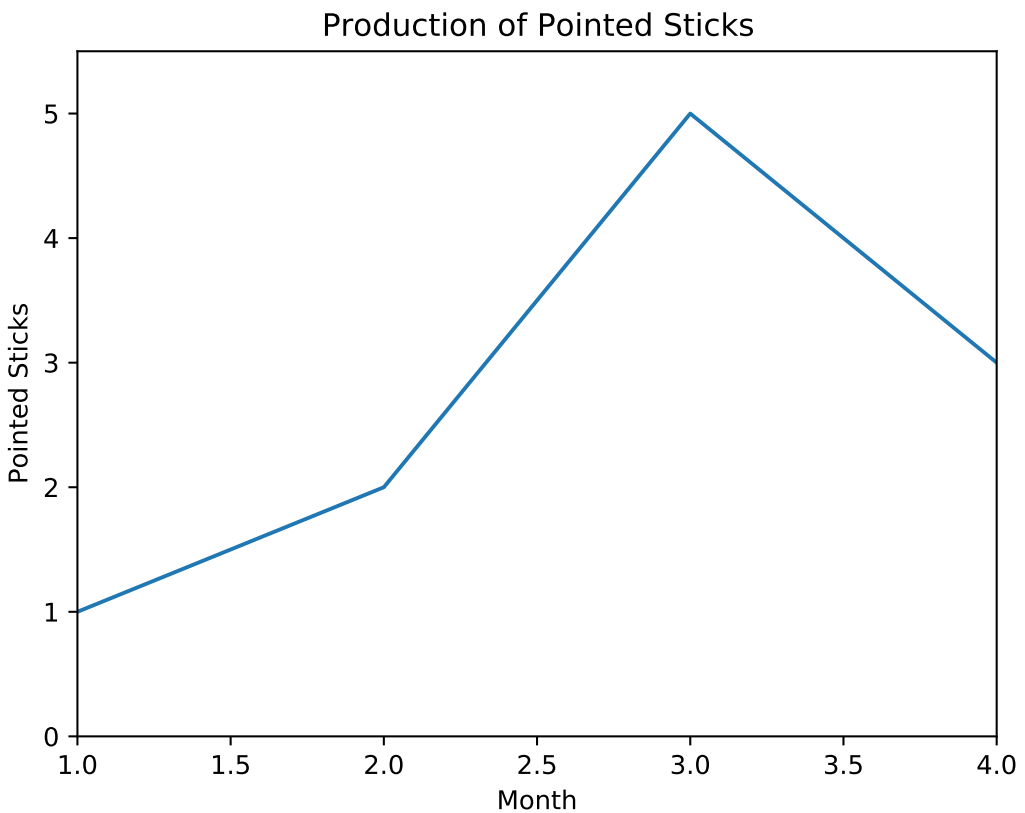
```
1 ax.set_xlim(1,4)
```



set_ylim()

`ax.set_ylim(bottom, top)` set the `bottom` and `top` y-axis limits

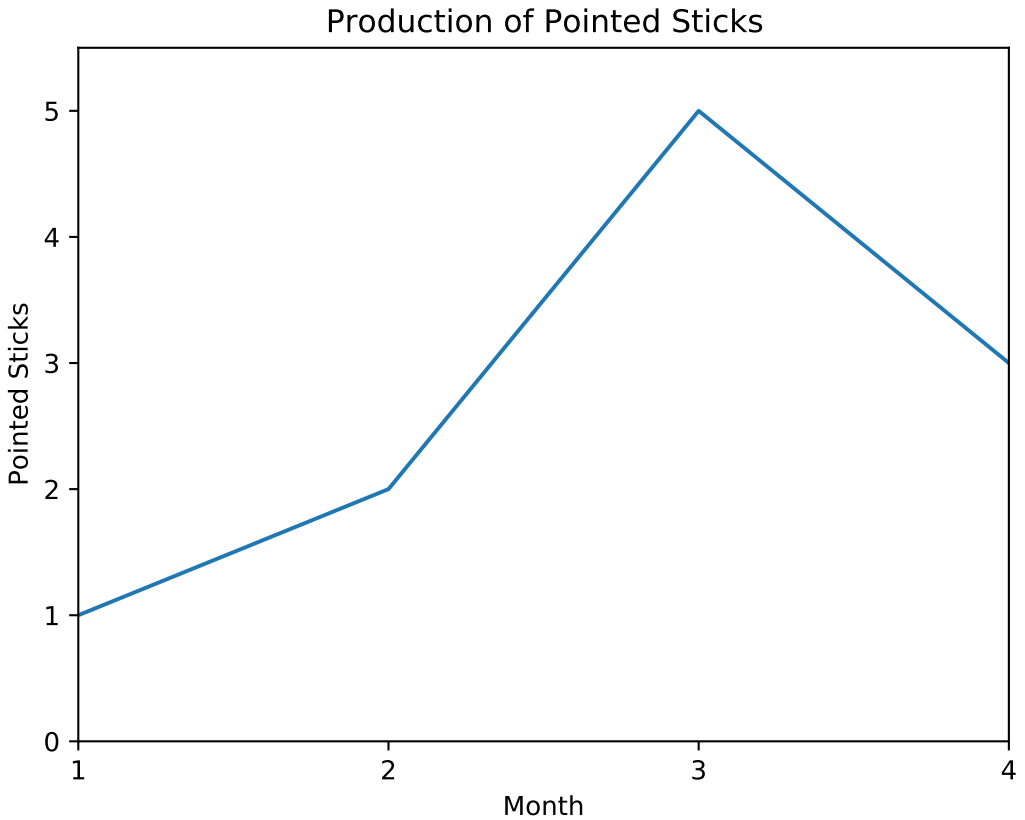
```
1 ax.set_ylim(0,5.5)
```



set_xticks()

`ax.set_xticks(ticks)` specify the x-axis tick locations

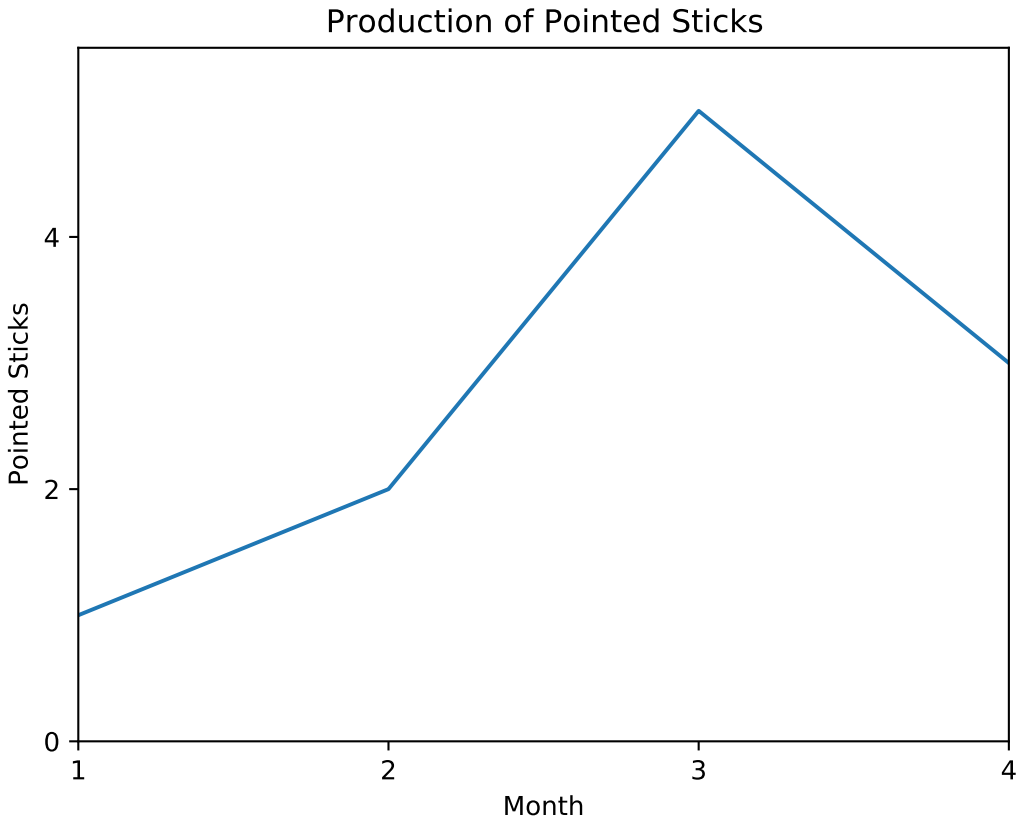
```
1 ax.set_xticks([1,2,3,4])
```



set_yticks()

`ax.set_yticks(ticks)` specify the y-axis tick locations

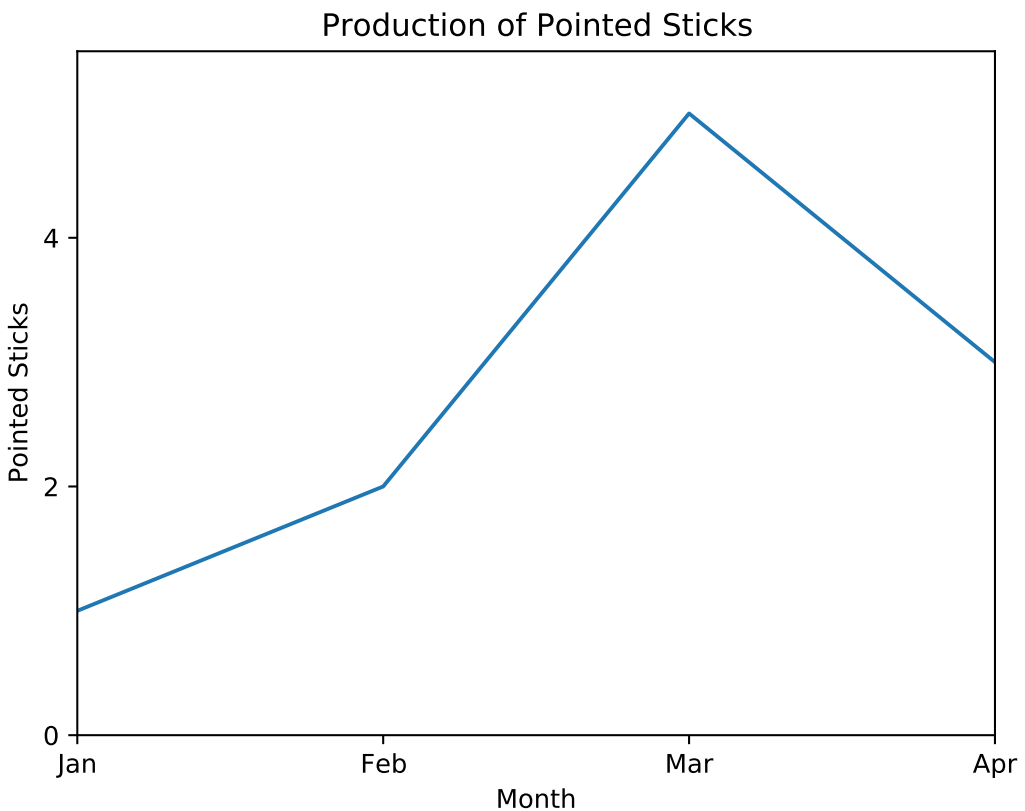
```
1 ax.set_yticks([0,2,4])
```



set_xticklabels()

`ax.set_xticklabels(labels)` specify the x-axis tick labels

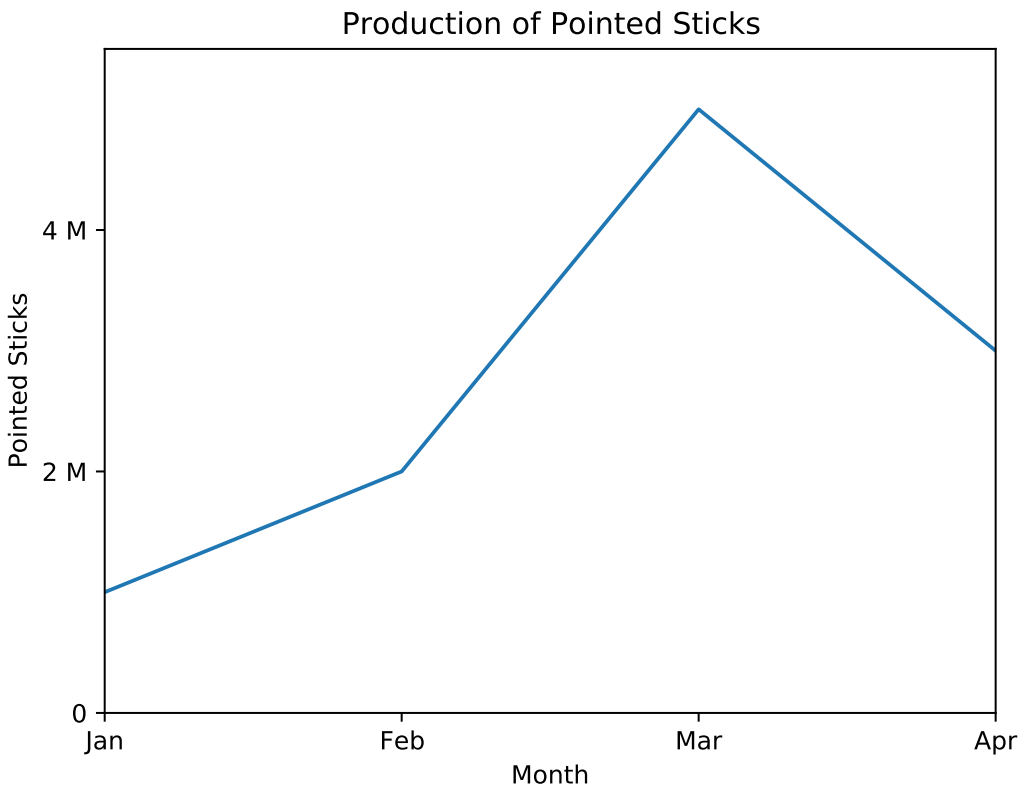
```
1 xlabels = ['Jan','Feb','Mar','Apr']
2 ax.set_xticklabels(xlabels)
```



set_yticklabels()

`ax.set_yticklabels(labels)` specify the y-axis tick labels

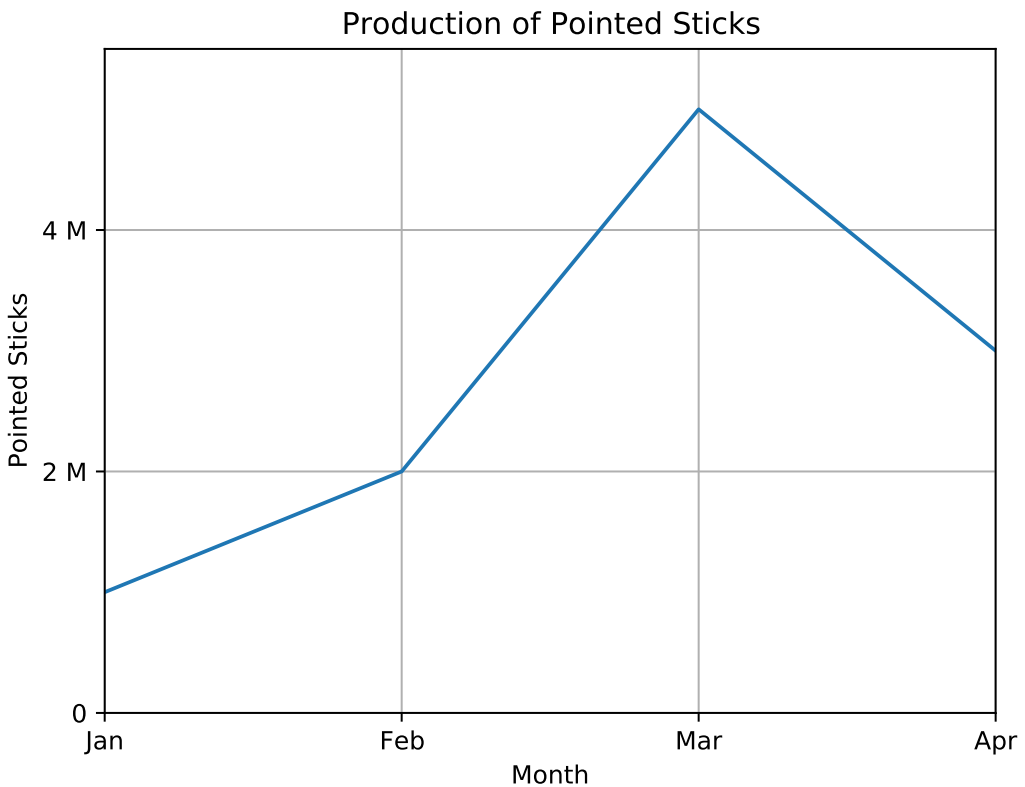
```
1 ylabels = ['0','2 M','4 M']
2 ax.set_yticklabels(ylabels)
```



grid()

`grid()` toggles grid lines

```
1 ax.grid()
```



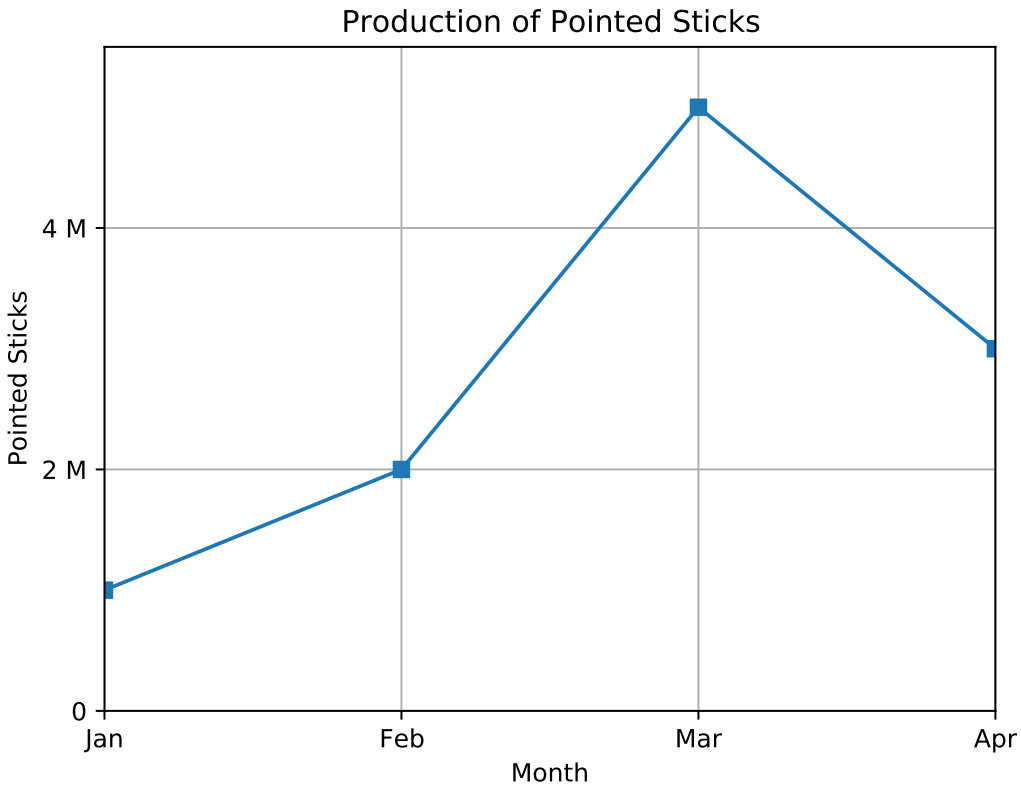
Markers

`plot([x], y, marker=m)` set marker symbol

```
1 ax.plot(x, y, marker='s')
```

Common Markers

Markers	Description
`'.'`	point
`'o'`	circle
`'s'`	square
`'*'`	star
`'x'`	x



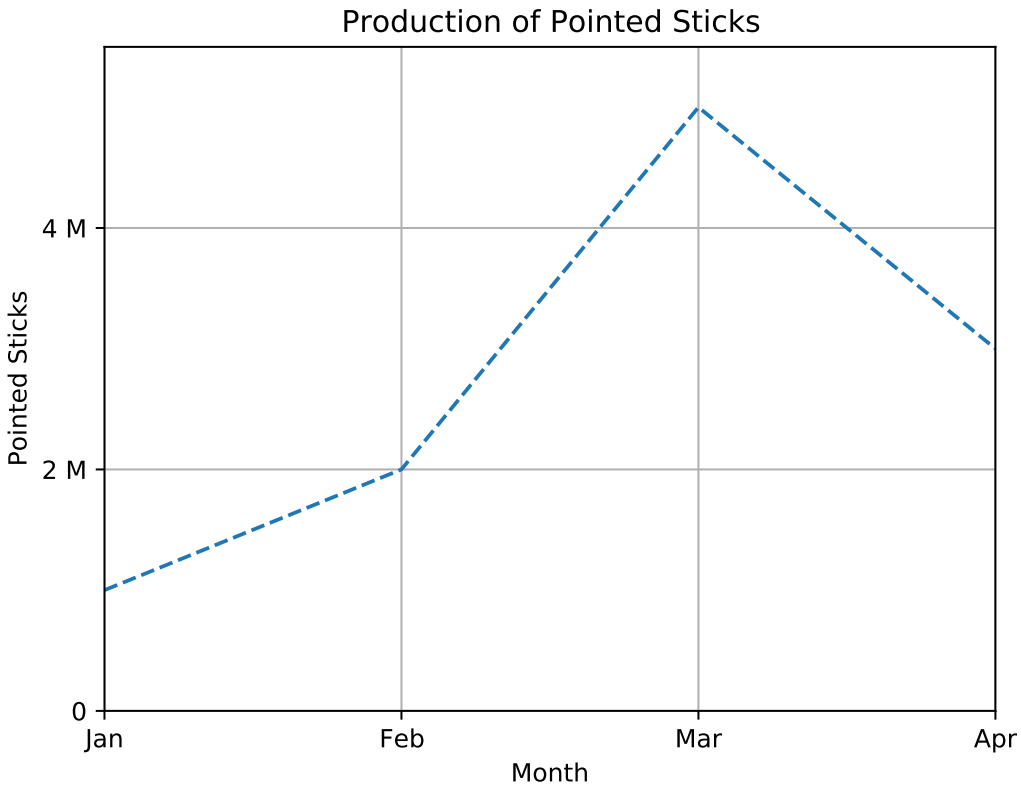
For more see: matplotlib.org/api/markers_api.html

Linestyle

`plot([x], y, linestyle=s)` set the linestyle

```
1 ax.plot(x, y, linestyle='dashed')
```

Line Styles	Description
`'solid'`	solid line
`'dotted'`	dotted line
`'dashed'`	dashed line
`'dashdot'`	dash-dotted line



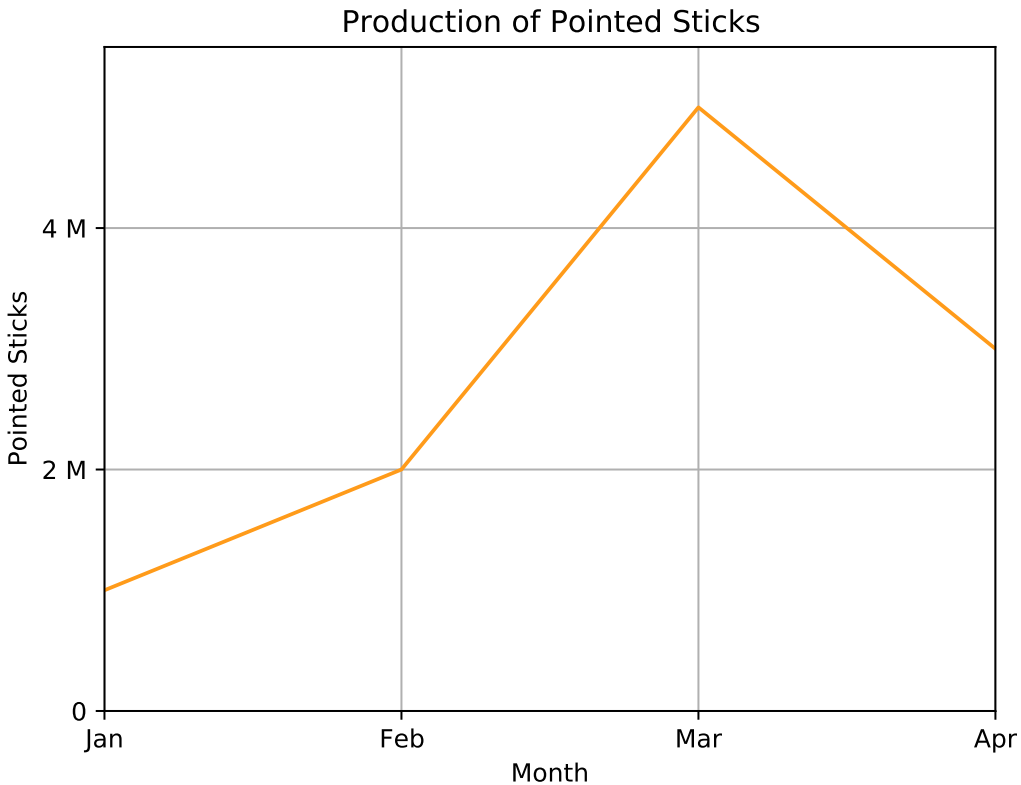
For more see: matplotlib.org/gallery/lines_bars_and_markers/linestyles.html

Color

`plot([x], y, color=c)` set the line color

```
1 ax.plot(x, y, color='#FF9B1A')
```

Named Colors	Color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

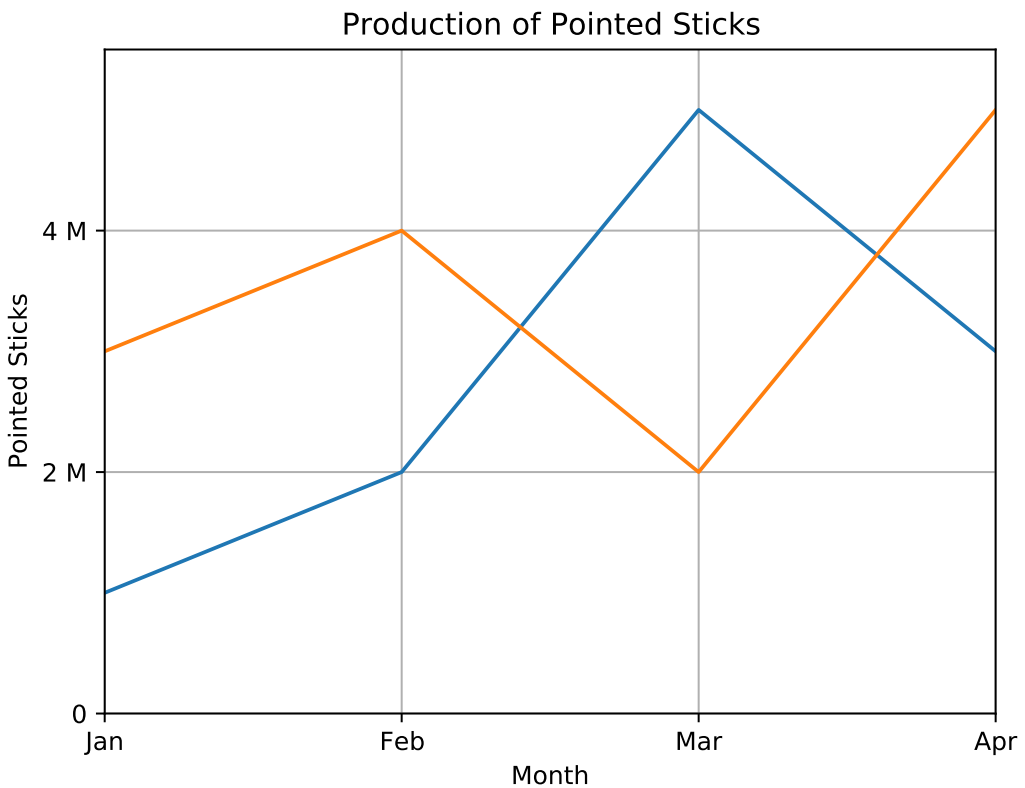


For more colors, see: matplotlib.org/3.2.1/gallery/color/named_colors.html

Multiple Lines

`plot([x], y)` call `plot()` again to add more lines

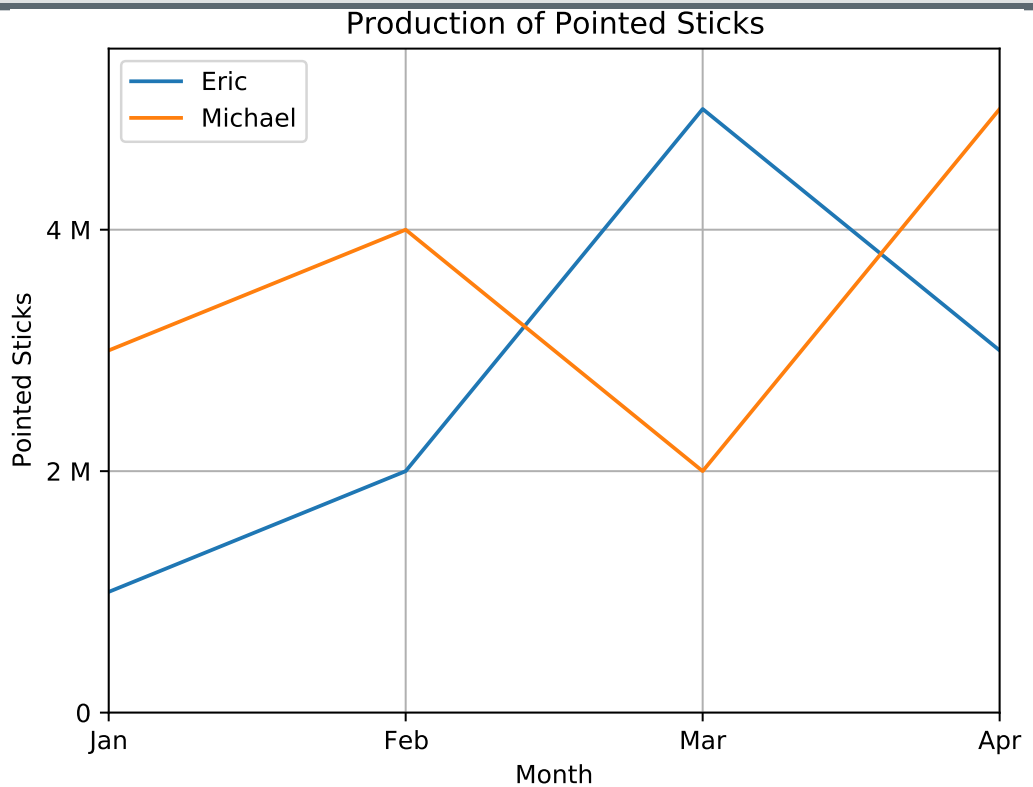
```
1 ax.plot(x, y1)
2 ax.plot(x, y2)
```



Legends

`legend()` add a legend using labels from plot()

```
1 ax.plot(x, y1, label='Eric')
2 ax.plot(x, y2, label='Michael')
3 ax.legend()
```



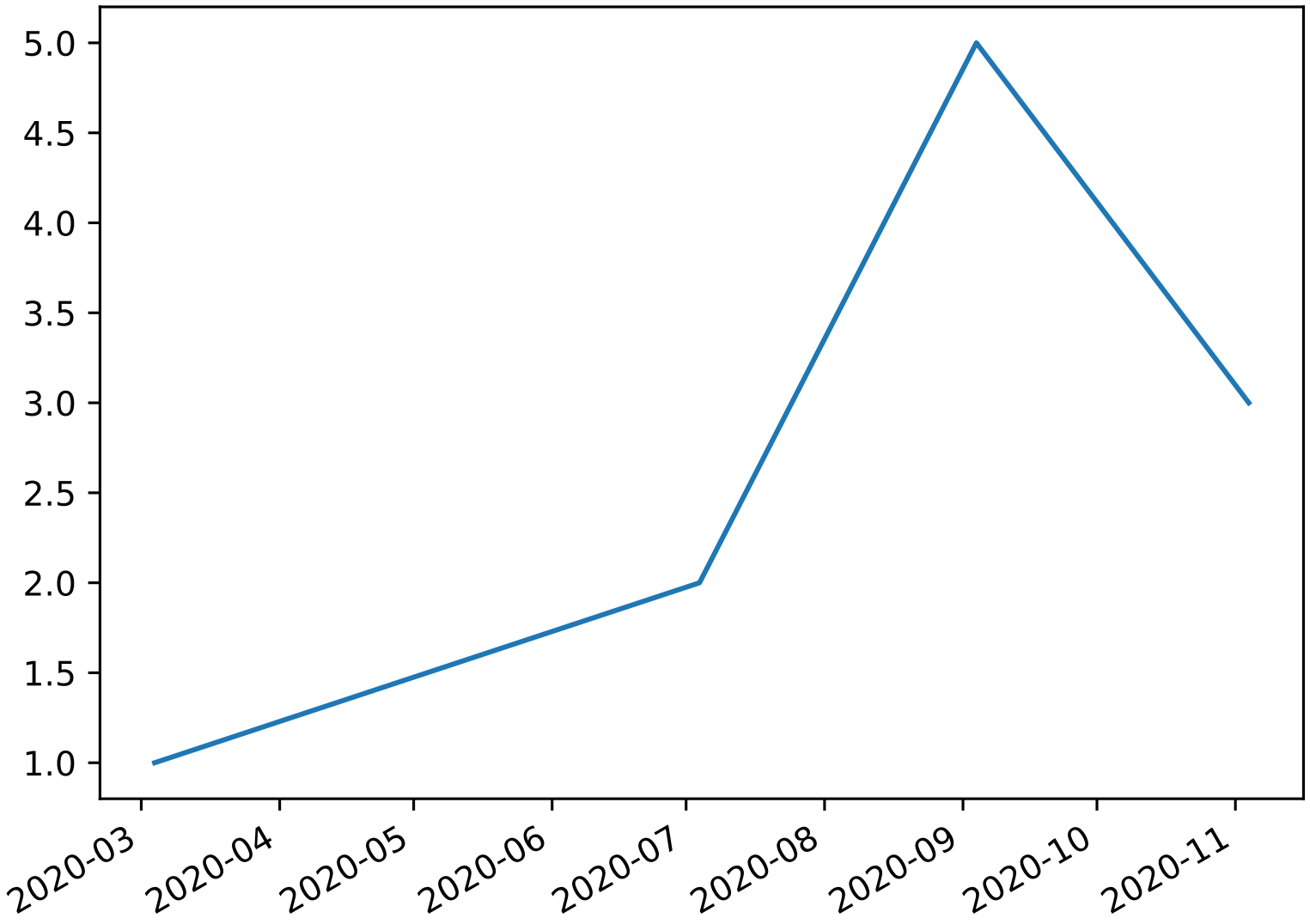
Dates

`datetime.date(y,m,d)` returns a datetime object from year `y`, month `m`, and day `d`

`fig.autofmt_xdate()` rotate and align x labels like dates

```
import datetime
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
dates=['2020-03-04', '2020-07-04',
       '2020-09-04', '2020-11-04']
x = []
for date in dates:
    y, m, d = date.split('-')
    dt = datetime.date(int(y), int(m), int(d))
    x.append(dt)
ax.plot(x, [1, 2, 5, 3])
fig.autofmt_xdate()
```

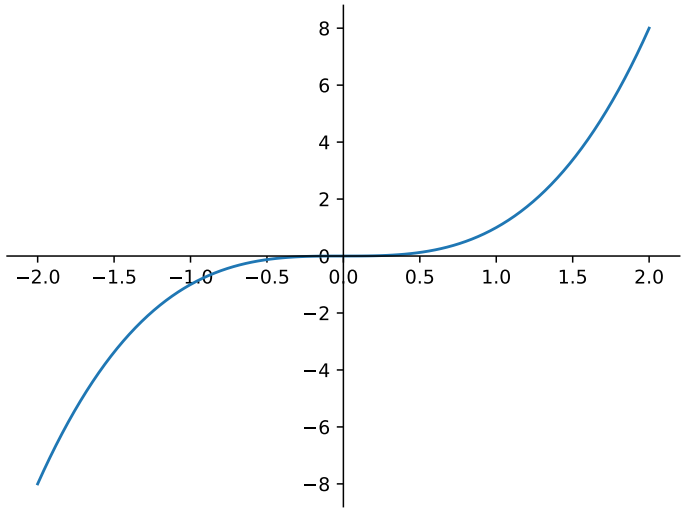
Dates – continued



Spines

```
1  import matplotlib.pyplot as plt
2  fig, ax = plt.subplots()
3  N = 50
4  u = range(-N, N+1)
5  x, y = [], []
6  for n in u:
7    x.append(2*(n/N))
8    y.append((2*(n/N))**3)
9  ax.plot(x, y)
10 for spine in ['top', 'right']:
11   ax.spines[spine].set_visible(False)
12 for spine in ['bottom', 'left']:
13   ax.spines[spine].set_position('zero')
14 plt.show()
```

`ax.spines` a list of spines ('top', 'bottom', 'left', 'right')




Thanks for watching!