A Tour of Matplotlib

From Bar Charts to XKCD-Style Plots

Susam Pal

PyCon UK 2019, Cardiff City Hall, Cardiff, UK

15 Sep 2019

About Me

Security architect at Walmart Labs.

Developing security services in Python and Go.

Author of a few open source projects:

- TeXMe: https://github.com/susam/texme
- GitPR: https://github.com/susam/gitpr
- Uncap: https://github.com/susam/uncap

Websites:

- https://twitter.com/susam
- https://github.com/susam
- https://susam.in/

Get Started

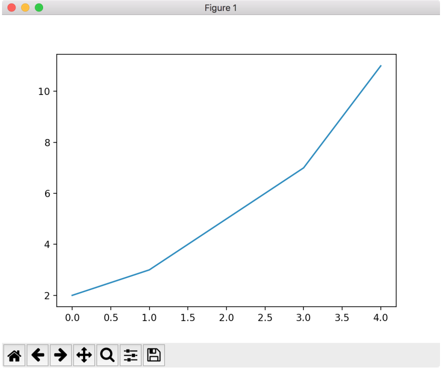
```
$ python3 -m venv myenv
$ source myenv/bin/activate
$ pip3 install matplotlib
```

Get Started

```
$ python3 -m venv myenv
$ source myenv/bin/activate
$ pip3 install matplotlib

$ python3
>>> import matplotlib.pyplot as plt
>>> plt.plot([2, 3, 5, 7, 11])
>>> plt.show()
```

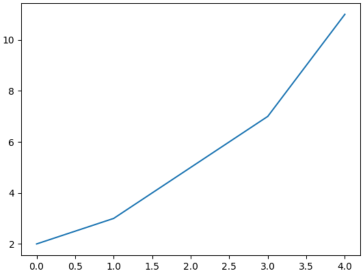
Get Started

```
$ python3 -m venv myenv
$ source myenv/bin/activate
$ pip3 install matplotlib

$ python3
>>> import matplotlib.pyplot as plt
>>> plt.plot([2, 3, 5, 7, 11])
>>> plt.show()
```



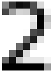
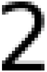
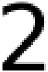
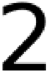
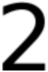
Save Plot to File

```
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11])
plt.savefig('out.png')
```



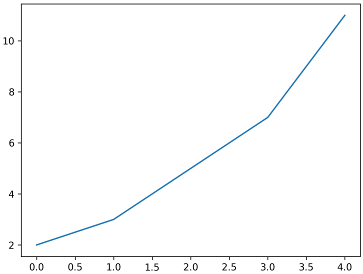
The default DPI (dots per inch) used to save the image is 100.

DPI: File Size vs. Quality Tradeoff

DPI	Inches	Pixels	KB	Quality
100	6.4” x 4.8”	640 x 480	15 KB	
200	6.4” x 4.8”	1280 x 960	34 KB	
300	6.4” x 4.8”	1920 x 1440	57 KB	
400	6.4” x 4.8”	2560 x 1920	78 KB	
500	6.4” x 4.8”	3200 x 2400	104 KB	

Save Plot to File: Set DPI to 300

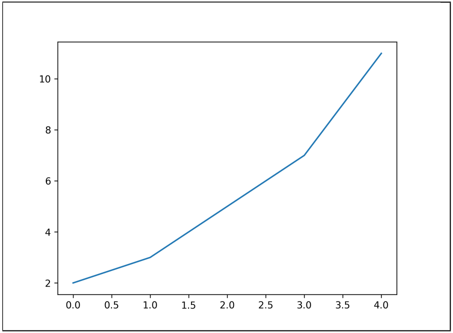
```python
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11])
plt.savefig('out.png', dpi=300)
```



DPI value of 300 provides a good tradeoff between quality and file size.

Default Bounding Box

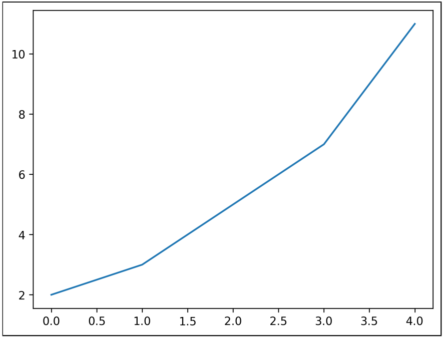
```
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11])
plt.savefig('out.png', dpi=300)
```



By default there is a lot of padding around the plot. The bounding box is large.

Tight Bounding Box

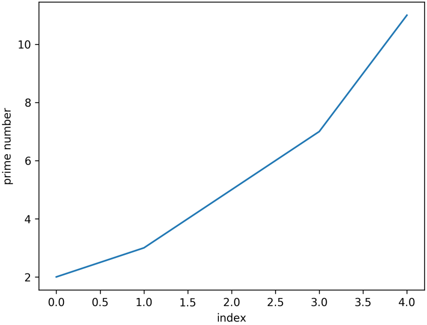
```
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11])
plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



The `bbox_inches='tight'` parameter creates a tight bounding box for the figure.

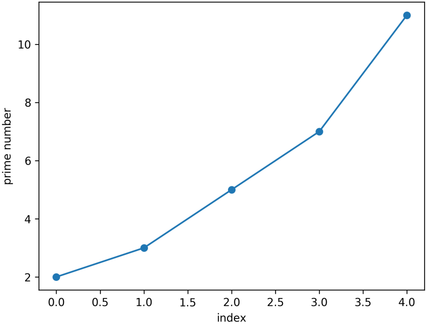
Labels

```
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11])

# Label x-axis and y-axis.
plt.xlabel('index')
plt.ylabel('prime number')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



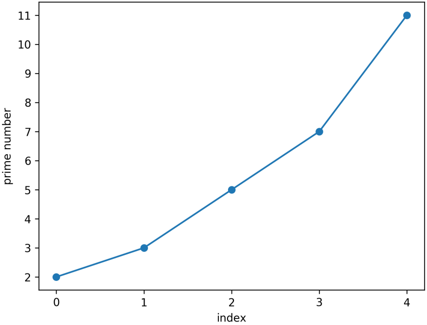
Circle Marker and Solid Line

```
import matplotlib.pyplot as plt

# Format string:
#   'o' for circle marker.
#   '-' for solid line.
plt.plot([2, 3, 5, 7, 11], 'o-')

plt.xlabel('index')
plt.ylabel('prime number')
plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



Integer Ticks

```
import matplotlib.pyplot as plt

plt.plot([2, 3, 5, 7, 11], 'o-')
plt.xlabel('index')
plt.ylabel('prime number')

# Place ticks at integer positions.
plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



Format String

Syntax:

`[marker][line][color]`

Format String

Syntax:

`[marker][line][color]`

Example:

`'o-k'`

Format String

Syntax:

`[marker][line][color]`

Example:

`'o-k'`

Explanation:

- `'o'` for circle marker
- `'-'` for solid line style
- `'k'` for black color

Format String

Syntax:

`[marker][line][color]`

Example:

`'o-k'`

Explanation:

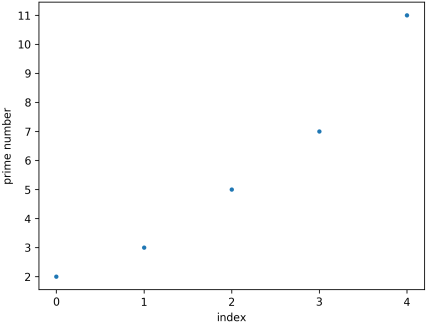
- `'o'` for circle marker
- `'-'` for solid line style
- `'k'` for black color

Usage:

```
plt.plot([2, 3, 5, 7, 11], 'o-k')
```

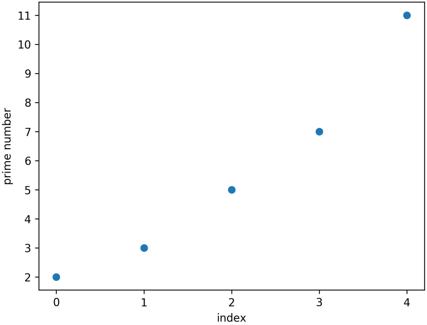
Point Marker

```
import matplotlib.pyplot as plt

# '.' for point marker
plt.plot([2, 3, 5, 7, 11], '.')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



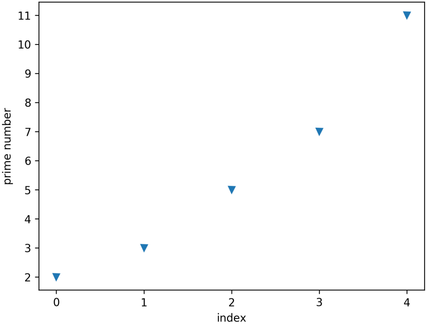
Circle Marker

```
import matplotlib.pyplot as plt

# 'o' for circle marker
plt.plot([2, 3, 5, 7, 11], 'o')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



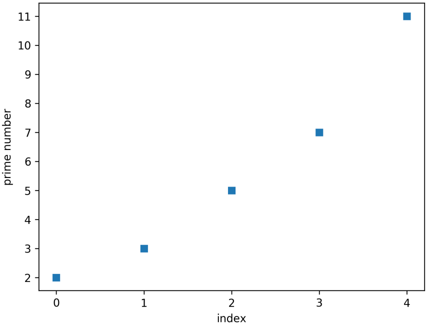
Triangle Down Marker

```
import matplotlib.pyplot as plt

# 'v' for triangle down marker
plt.plot([2, 3, 5, 7, 11], 'v')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```

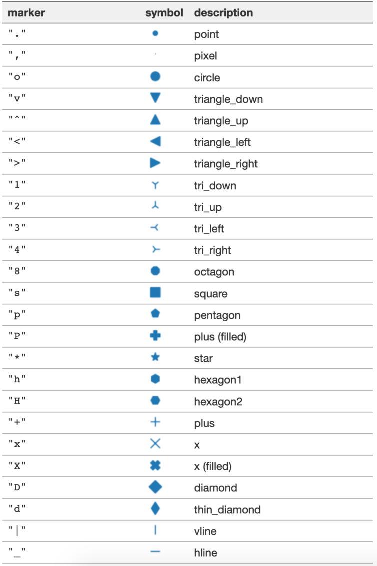


Square Marker

```
import matplotlib.pyplot as plt

# 's' for square marker
plt.plot([2, 3, 5, 7, 11], 's')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



Markers Reference

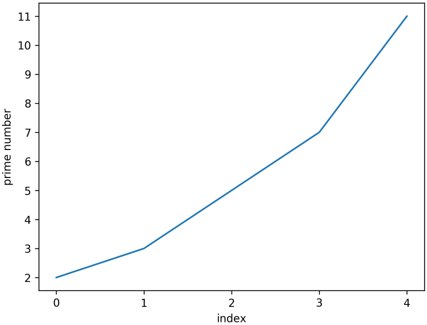


marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	⯃	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star
"h"	⬢	hexagon1
"H"	⬣	hexagon2
"+"	+	plus
"x"	×	x
"X"	✖	x (filled)
"D"	◆	diamond
"d"	♦	thin_diamond
"|"	|	vline
"_"	—	hline

Source:
https://matplotlib.org/api/markers_api.html

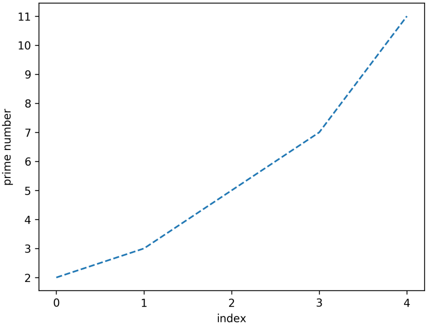
Solid Line Style

```
import matplotlib.pyplot as plt

# '-' for solid line style
plt.plot([2, 3, 5, 7, 11], '-')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



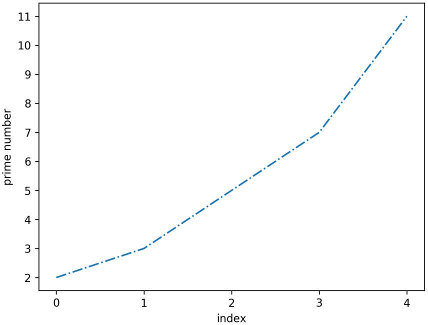
Dashed Line Style

```
import matplotlib.pyplot as plt

# '--' for dashed line style
plt.plot([2, 3, 5, 7, 11], '--')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



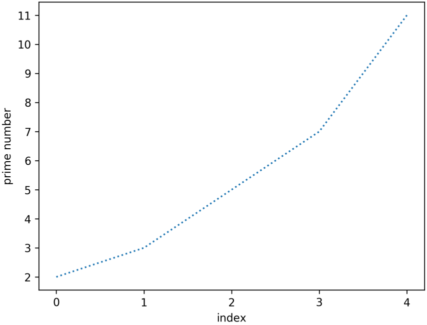
Dash-Dot Line Style

```
import matplotlib.pyplot as plt

# '-.' for dash dot line style
plt.plot([2, 3, 5, 7, 11], '-.')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```

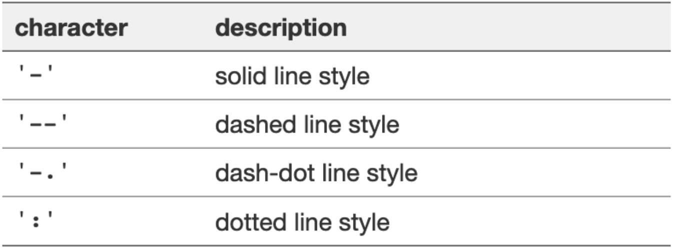


Dotted Line Style

```
import matplotlib.pyplot as plt

# ':' for dotted line style
plt.plot([2, 3, 5, 7, 11], ':')
plt.xlabel('index')
plt.ylabel('prime number')

plt.xticks(range(5))
plt.yticks(range(2, 12))

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



Line Styles Reference

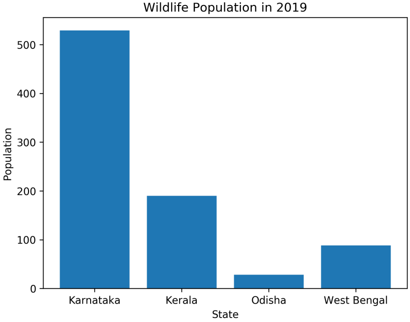


character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

Source: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

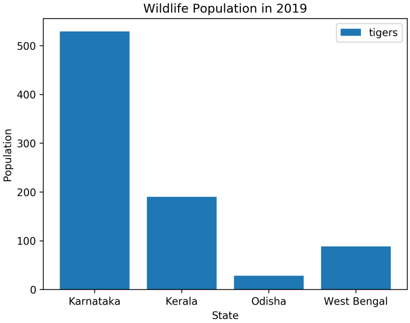
Bar Chart

```
import matplotlib.pyplot as plt

tigers = [529, 190, 28, 88]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

plt.bar(states, tigers)

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



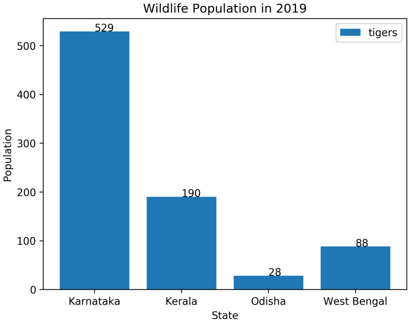
Bar Chart: Legend

```
import matplotlib.pyplot as plt

tigers = [529, 190, 28, 88]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

# Define a label to be used in legend.
plt.bar(states, tigers, label='tigers')

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')

# Place a legend.
plt.legend()

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



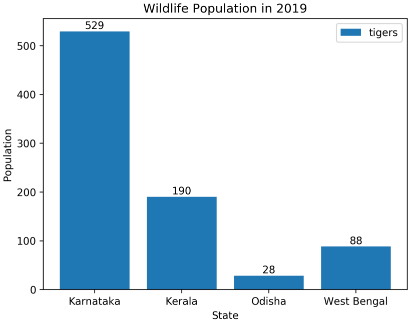
Bar Chart: Using plt.text() to Display Values

```
import matplotlib.pyplot as plt

tigers = [529, 190, 28, 88]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

plt.bar(states, tigers, label='tigers')

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')
plt.legend()

for index, value in enumerate(tigers):
    plt.text(index, value, value)

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



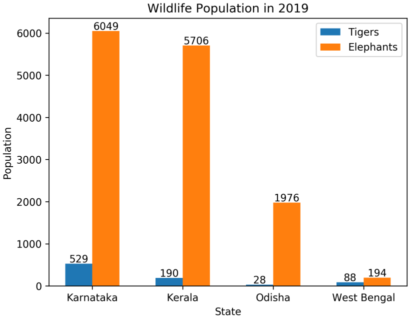
Bar Chart: Center the Values on the Bars

```
import matplotlib.pyplot as plt

tigers = [529, 190, 28, 88]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

plt.bar(states, tigers, label='tigers')

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')
plt.legend()

for index, value in enumerate(tigers):
    plt.text(index, value + 5, value, ha='center')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



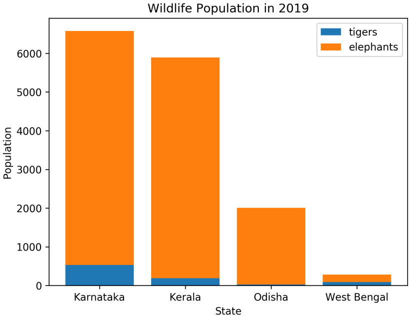
Grouped Bar Chart

```
import matplotlib.pyplot as plt
import numpy as np

tigers = [529, 190, 28, 88]
elephants = [6049, 5706, 1976, 194]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

indices = np.arange(len(states))
w = 0.3  # Width of each bar

plt.bar(indices - w / 2, tigers, w, label='Tigers')
plt.bar(indices + w / 2, elephants, w, label='Elephants')

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')
plt.legend()

plt.xticks(indices, states)

for index, value in enumerate(tigers):
    plt.text(index - w / 2, value + 30, value, ha='center')

for index, value in enumerate(elephants):
    plt.text(index + w / 2, value + 30, value, ha='center')

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



Stacked Bar Chart

```
import matplotlib.pyplot as plt

tigers = [529, 190, 28, 88]
elephants = [6049, 5706, 1976, 194]
states = ['Karnataka', 'Kerala',
          'Odisha', 'West Bengal']

plt.bar(states, tigers, label='tigers')
plt.bar(states, elephants, label='elephants',
        bottom=tigers)

plt.xlabel('State')
plt.ylabel('Population')
plt.title('Wildlife Population in 2019')
plt.legend()

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



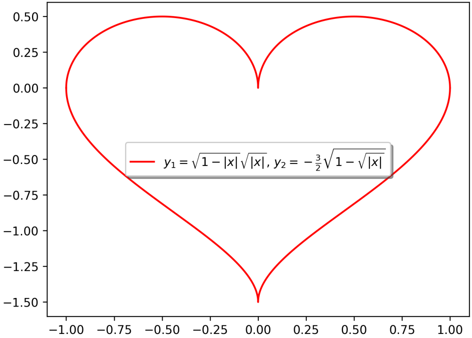
Heart Plot: The Equations

$$\begin{aligned} y_1 &= \sqrt{1 - |x|}\sqrt{|x|}, \\ y_2 &= -\frac{3}{2}\sqrt{1 - \sqrt{|x|}}. \end{aligned}$$

See https://github.com/susam/heart for more details.

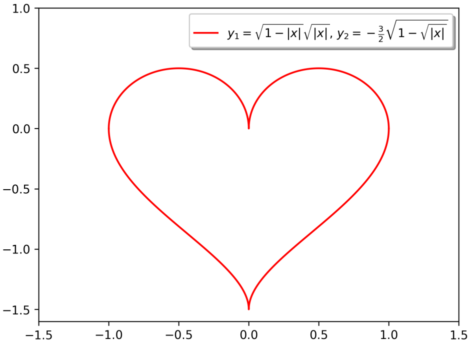
Heart Plot: Plot the Equations

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

label_for_legend = (
    r'$y_1 = \sqrt{1 - |x|} \sqrt{|x|}$, '
    r'$y_2 = -\frac{3}{2} \sqrt{1 - \sqrt{|x|}}$'
)

x = np.linspace(-1, 1, 10001)
y1 = np.sqrt(1 - np.abs(x)) * np.sqrt(np.abs(x))
y2 = (-3 / 2) * np.sqrt(1 - np.sqrt((np.abs(x))))

plt.plot(x, y1, 'r', label=label_for_legend)
plt.plot(x, y2, 'r')
plt.legend(shadow=True)

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



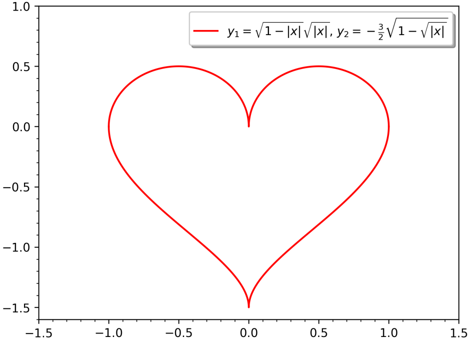
Heart Plot: X and Y Limits

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

label_for_legend = (
    r'$y_1 = \sqrt{1 - |x|} \sqrt{|x|}$, '
    r'$y_2 = -\frac{3}{2} \sqrt{1 - \sqrt{|x|}}$'
)

x = np.linspace(-1, 1, 10001)
y1 = np.sqrt(1 - np.abs(x)) * np.sqrt(np.abs(x))
y2 = (-3 / 2) * np.sqrt(1 - np.sqrt((np.abs(x))))

plt.plot(x, y1, 'r', label=label_for_legend)
plt.plot(x, y2, 'r')
plt.legend(shadow=True)

# Set wider x-limit and y-limit.
plt.xlim([-1.5, 1.5])
plt.ylim([-1.6, 1.0])

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



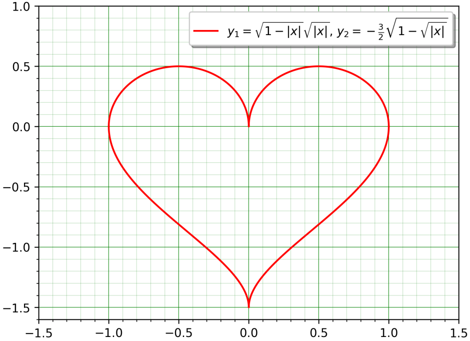
Heart Plot: Major and Minor Tick Locations

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

label_for_legend = (
    r'$y_1 = \sqrt{1 - |x|} \sqrt{|x|}$, '
    r'$y_2 = -\frac{3}{2} \sqrt{1 - \sqrt{|x|}}$'
)

x = np.linspace(-1, 1, 10001)
y1 = np.sqrt(1 - np.abs(x)) * np.sqrt(np.abs(x))
y2 = (-3 / 2) * np.sqrt(1 - np.sqrt((np.abs(x))))

plt.plot(x, y1, 'r', label=label_for_legend)
plt.plot(x, y2, 'r')
plt.legend(shadow=True)

# Set wider x-limit and y-limit.
plt.xlim([-1.5, 1.5])
plt.ylim([-1.6, 1.0])

ax = plt.gca()

# Set major and minor tick loations on the axes.
ax.xaxis.set_major_locator(mpl.ticker.MultipleLocator(0.5))
ax.xaxis.set_minor_locator(mpl.ticker.MultipleLocator(0.1))
ax.yaxis.set_major_locator(mpl.ticker.MultipleLocator(0.5))
ax.yaxis.set_minor_locator(mpl.ticker.MultipleLocator(0.1))

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



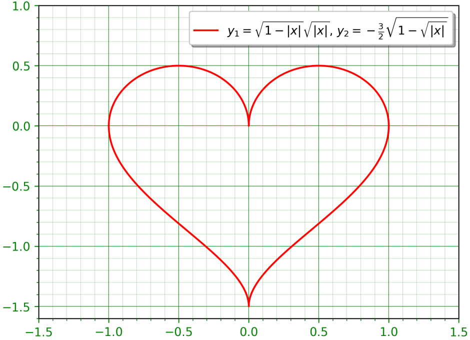
Heart Plot: Green Color Grid

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



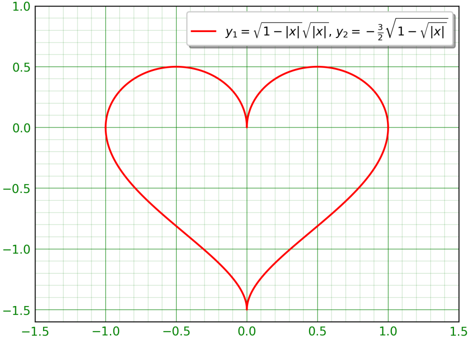
Heart Plot: Green Color Ticks

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

# Color the ticks and tick labels green.
plt.tick_params(which='major', colors='g')
plt.tick_params(which='minor', colors='g')

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



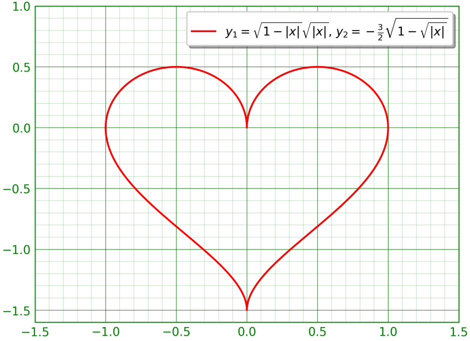
Heart Plot: Zero Tick Length

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

# Trim protruding ticks by setting tick length to 0.
plt.tick_params(which='major', colors='g', length=0)
plt.tick_params(which='minor', colors='g', length=0)

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



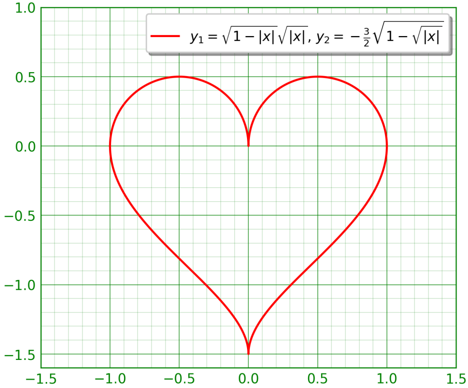
Heart Plot: Green Spines

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

# Trim protruding ticks by setting tick length to 0.
plt.tick_params(which='major', colors='g', length=0)
plt.tick_params(which='minor', colors='g', length=0)

# Color spines green.
ax.spines['bottom'].set_color('g')
ax.spines['top'].set_color('g')
ax.spines['right'].set_color('g')
ax.spines['left'].set_color('g')

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



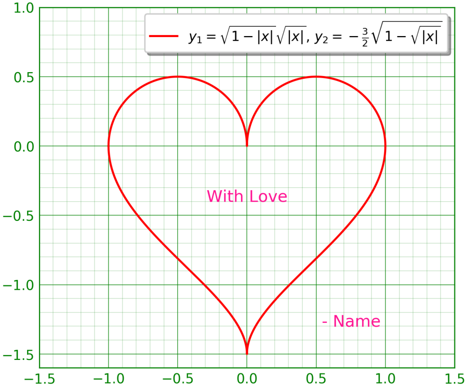
Heart Plot: Equal Aspect Ratio

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

# Trim protruding ticks by setting tick length to 0.
plt.tick_params(which='major', colors='g', length=0)
plt.tick_params(which='minor', colors='g', length=0)

# Color spines green.
ax.spines['bottom'].set_color('g')
ax.spines['top'].set_color('g')
ax.spines['right'].set_color('g')
ax.spines['left'].set_color('g')

# Use same scaling to plot units for both axes.
ax.set_aspect('equal')

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



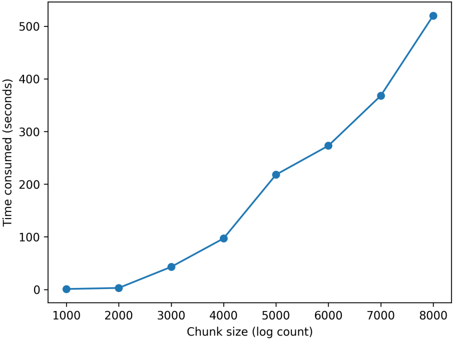
Heart Plot: Text Message

```
# Draw grid lines in green color.
plt.grid(which='major', color='g', linewidth='0.4')
plt.grid(which='minor', color='g', linewidth='0.1')

# Trim protruding ticks by setting tick length to 0.
plt.tick_params(which='major', colors='g', length=0)
plt.tick_params(which='minor', colors='g', length=0)

# Color spines green.
ax.spines['bottom'].set_color('g')
ax.spines['top'].set_color('g')
ax.spines['right'].set_color('g')
ax.spines['left'].set_color('g')

# Use same scaling to plot units for both axes.
ax.set_aspect('equal')

# Add a text message and sign the plot.
kwargs = {'size': 'large', 'color': 'deeppink'}
plt.text(0, -0.4, 'With Love', ha='center', **kwargs)
plt.text(0.54, -1.3, '- Name', **kwargs)

plt.savefig('out.png', dpi=300, bbox_inches='tight')
```



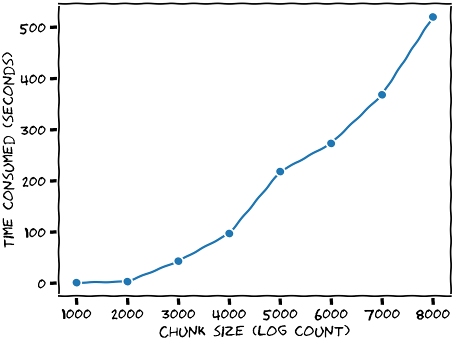
Regular Plot

```
import matplotlib.pyplot as plt

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



XKCD-Style Plot

```
import matplotlib.pyplot as plt

# Turn on XKCD sketch-style drawing.
plt.xkcd()

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



XKCD-Style Plot: Font: Humor Sans

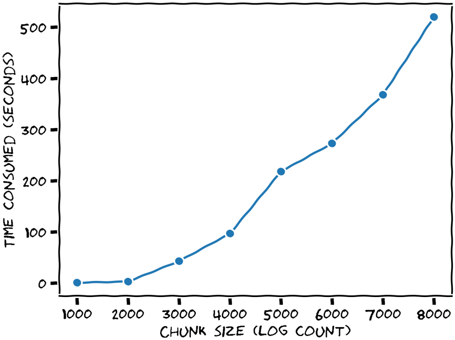
Install **Humor Sans** font for best results.

After installing the font, remove Matplotlib cache
to force rebuilding of font list.

```
# On macOS
brew tap homebrew/cask-fonts
brew cask install font-humor-sans
rm -rf ~/.matplotlib

# On Debian, Ubuntu, etc.
apt-get update
apt-get install fonts-humor-sans
rm -rf ~/.cache/matplotlib
```

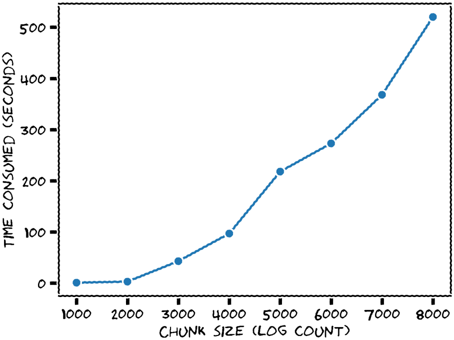
XKCD-Style Plot: Default Parameters

```
import matplotlib.pyplot as plt

plt.xkcd(scale=1, length=100,
         randomness=2)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



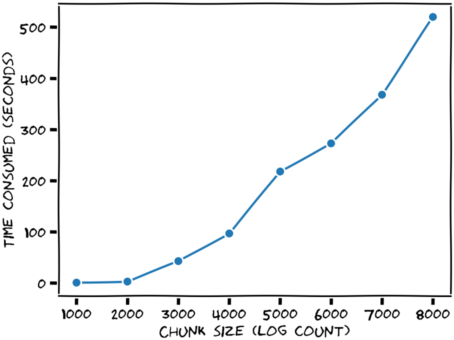
XKCD-Style Plot: `scale=5`

```
import matplotlib.pyplot as plt

plt.xkcd(scale=5)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



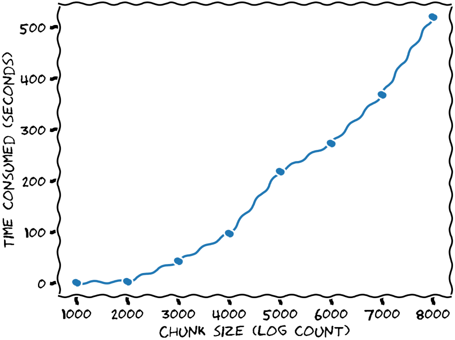
XKCD-Style Plot: `scale=0.2`

```
import matplotlib.pyplot as plt

plt.xkcd(scale=0.2)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



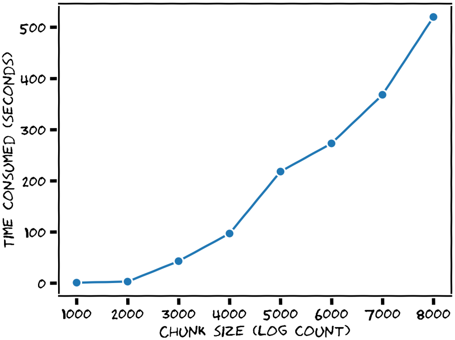
XKCD-Style Plot: `length=20`

```
import matplotlib.pyplot as plt

plt.xkcd(length=20)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



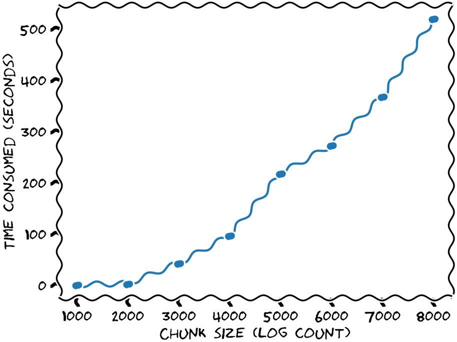
XKCD-Style Plot: `length=500`

```
import matplotlib.pyplot as plt

plt.xkcd(length=500)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



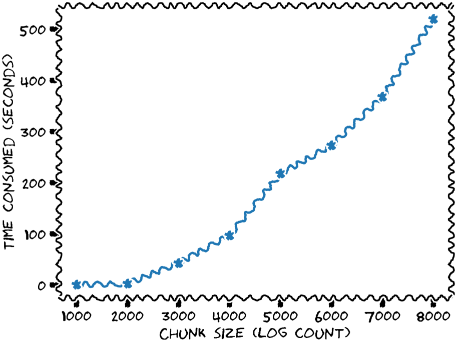
XKCD-Style Plot: `randomness=1`

```
import matplotlib.pyplot as plt

plt.xkcd(scale=10, randomness=1)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



XKCD-Style Plot: randomness=10

```
import matplotlib.pyplot as plt

plt.xkcd(scale=10, randomness=10)

logs = [1000, 2000, 3000, 4000,
        5000, 6000, 7000, 8000]

time = [1, 3, 43, 97, 218, 273,
        368, 520]

plt.plot(logs, time, 'o-')
plt.xlabel('Chunk size (log count)')
plt.ylabel('Time consumed (seconds)')

plt.savefig('out.png', dpi=300,
            bbox_inches='tight')
```



Thank You