- Shin’s Lab -

Python for Data Visualization

Python for Data Visualization

-Chapter.2 Line Plot -

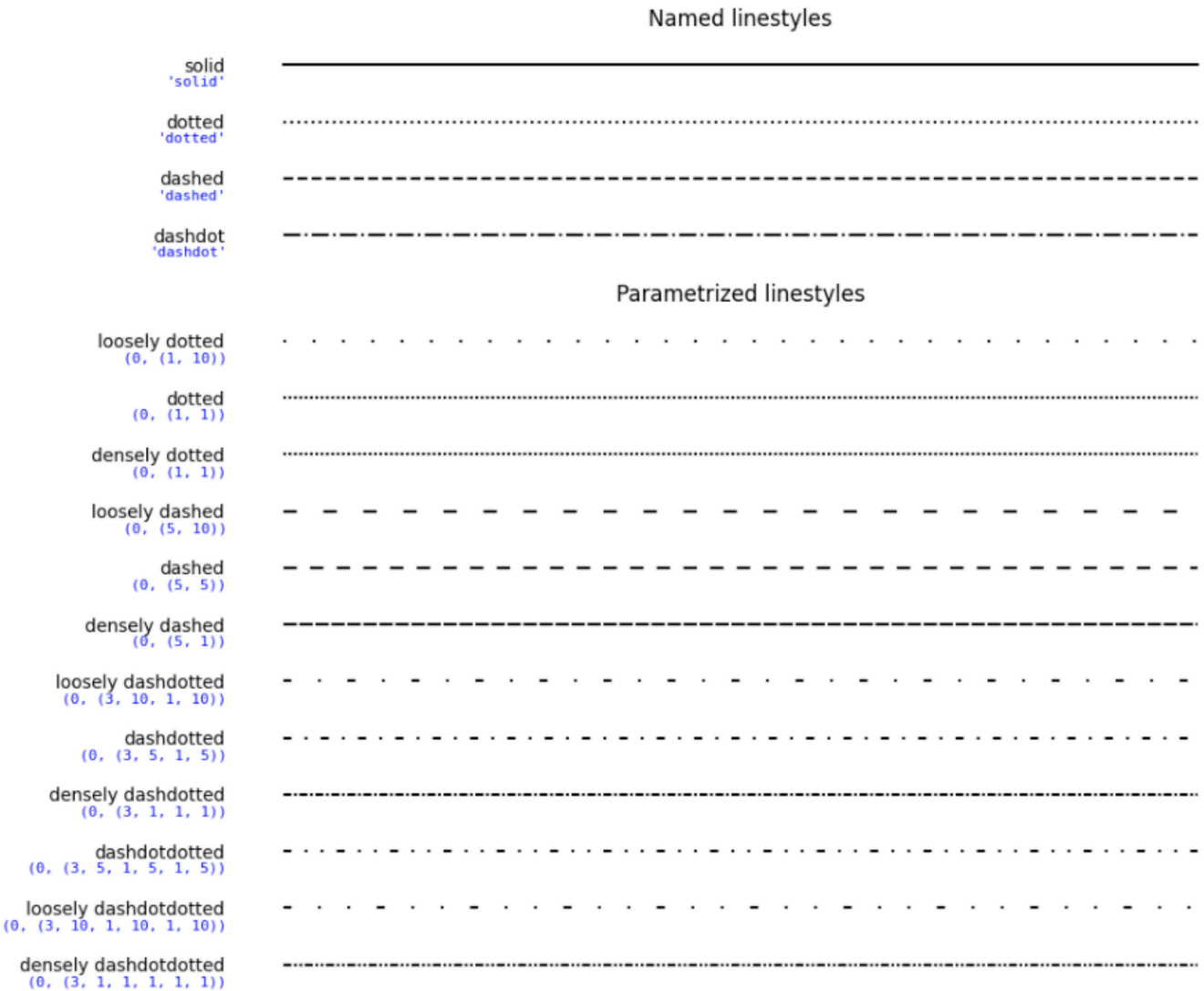
Python for Data Visualization

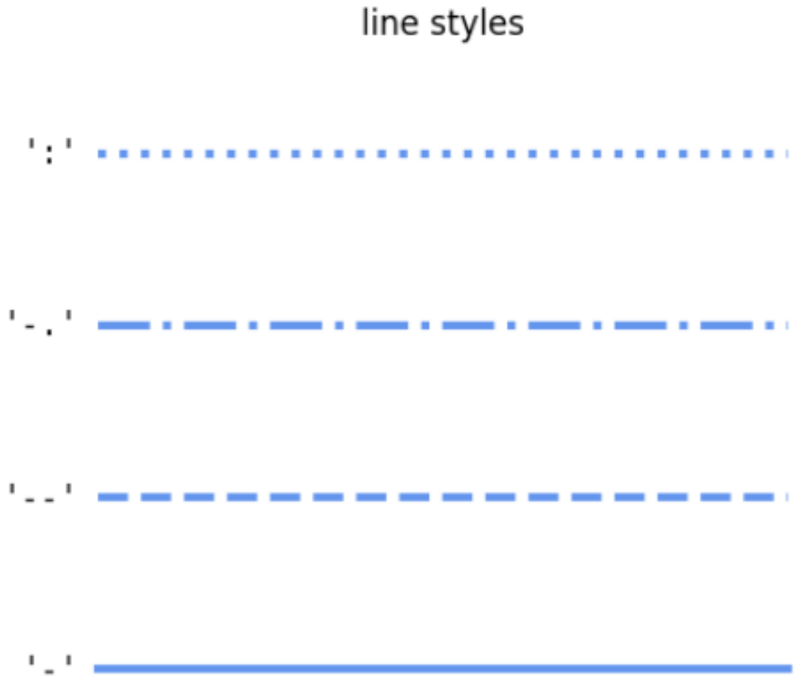
-Chapter.2 Line Plot -

2-03. Line Styles and Markers

1. Line Styles
2. Markers
3. Customizing Markers
4. Line Styles/Markers with Legend
5. fmt Argument

1. Line Styles



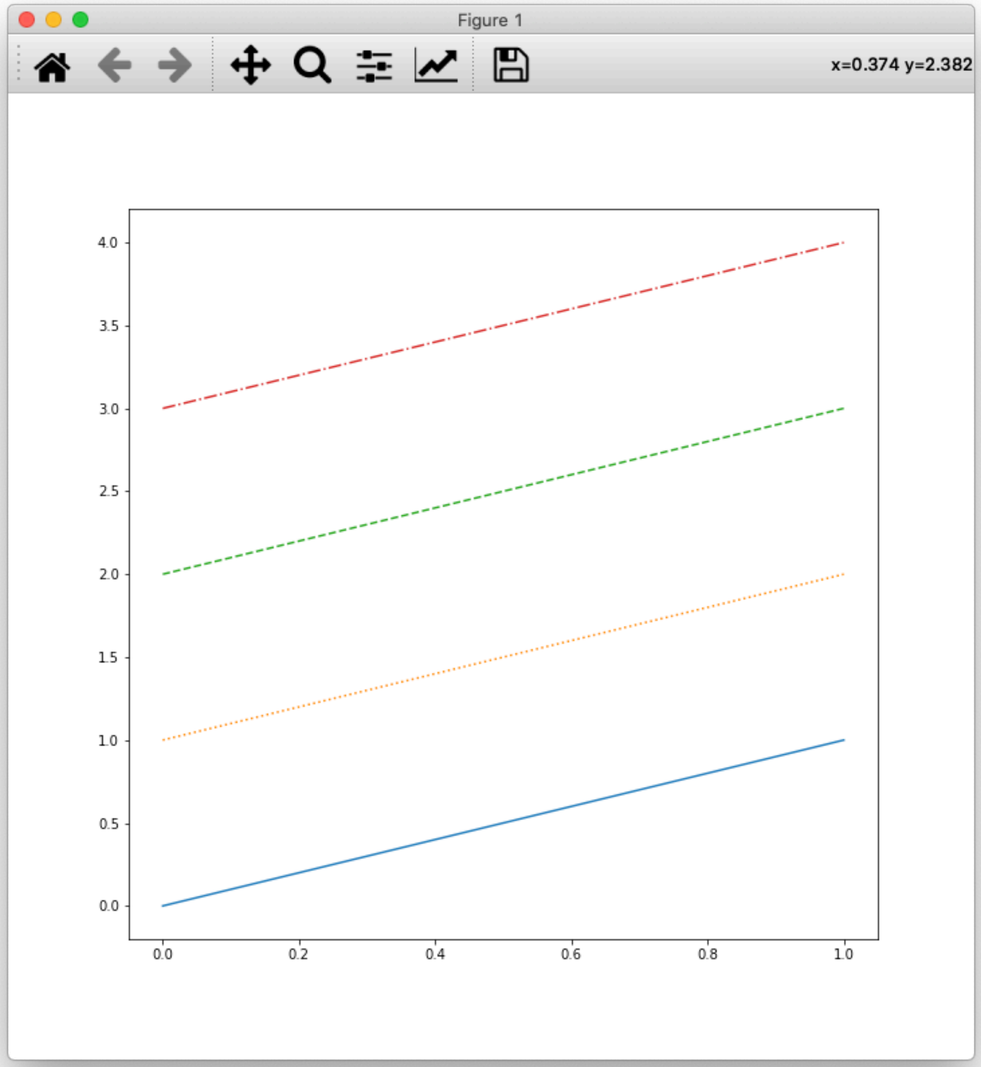


https://matplotlib.org/3.2.1/gallery/lines_bars_and_markers/linestyles.html

https://matplotlib.org/gallery/lines_bars_and_markers/line_styles_reference.html

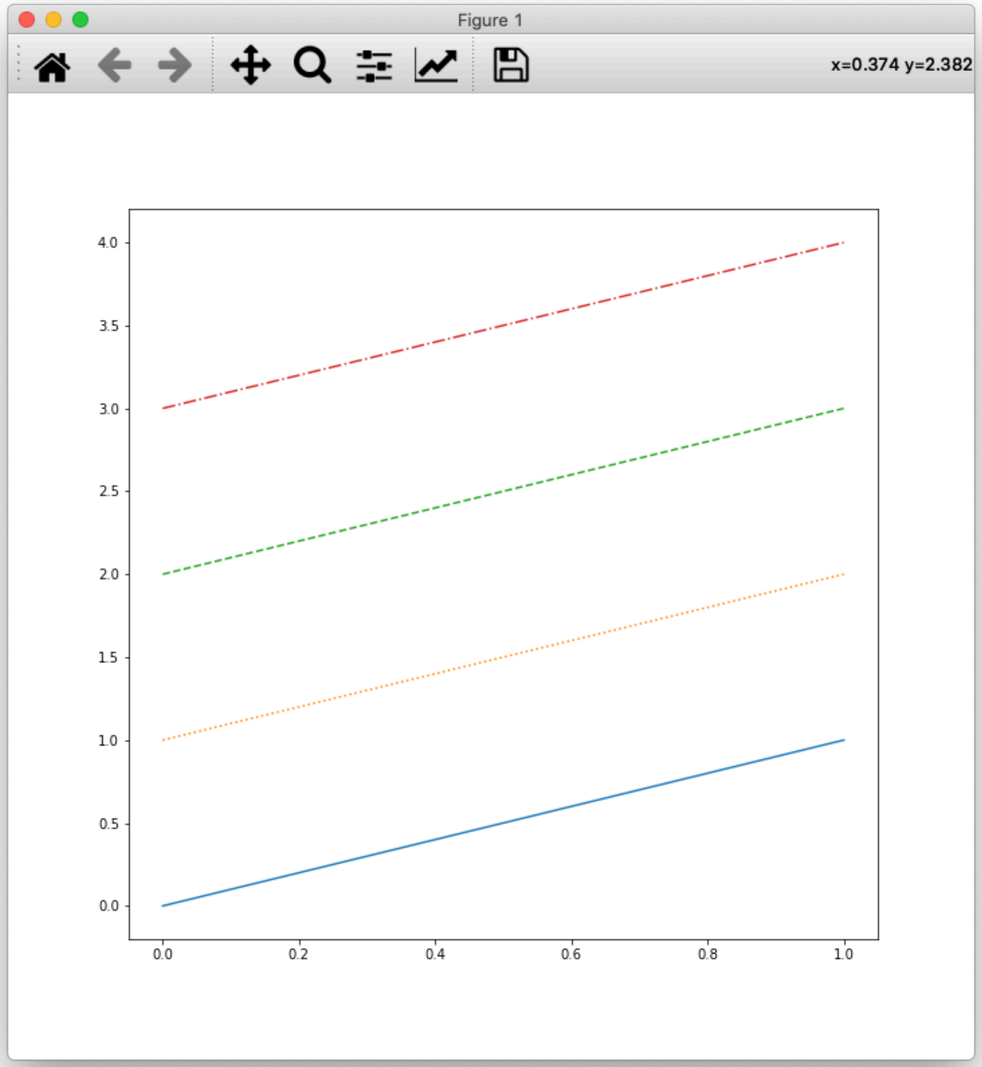
1. Line Styles

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

x_data = np.array([0, 1])
y_data = x_data

fig, ax = plt.subplots(figsize=(10, 10))

ax.plot(x_data, y_data)

ax.plot(x_data, y_data+1,
        linestyle='dotted')

ax.plot(x_data, y_data+2,
        linestyle='dashed')

ax.plot(x_data, y_data+3,
        linestyle='dashdot')
```



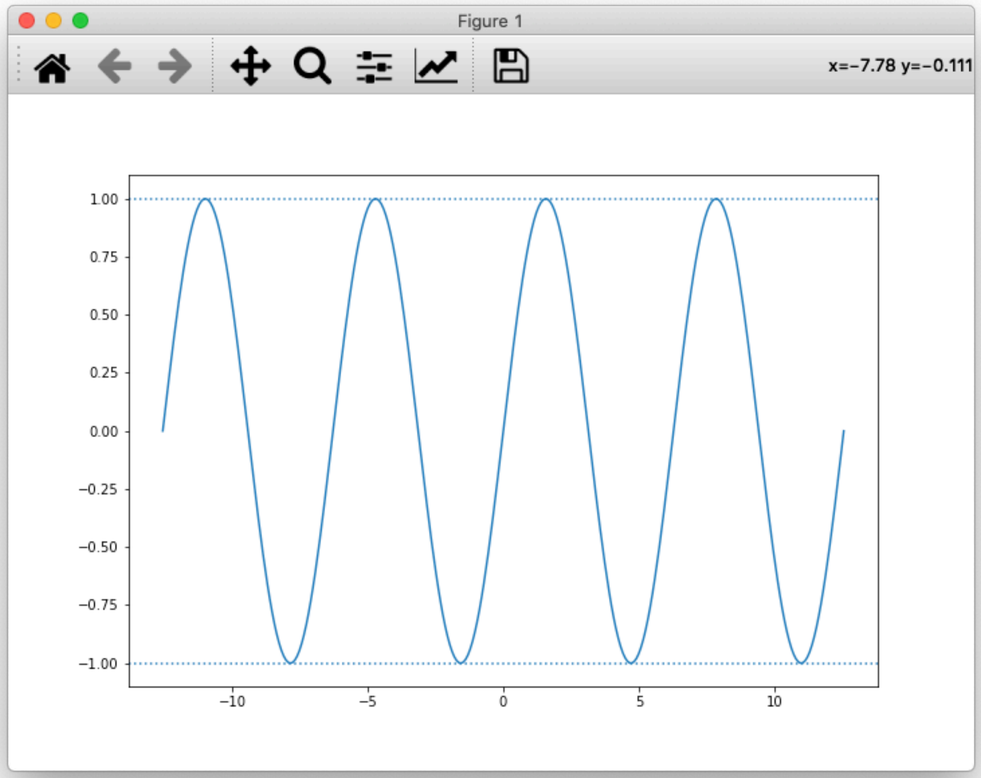
1. Line Styles

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

x_data = np.array([0, 1])
y_data = x_data

fig, ax = plt.subplots(figsize=(10, 10))

ax.plot(x_data, y_data)


ax.plot(x_data, y_data+1,
        linestyle=':')


ax.plot(x_data, y_data+2,
        linestyle='--')


ax.plot(x_data, y_data+3,
        linestyle='-.')
```



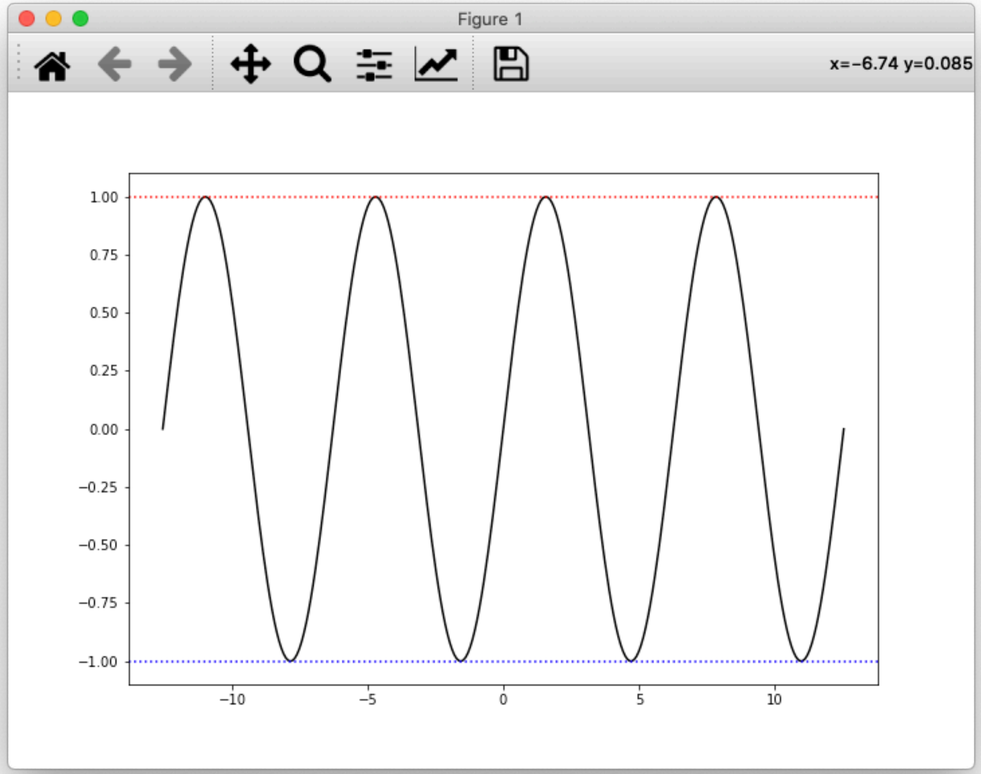
1. Line Styles

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin)

ax.axhline(y=1,
           linestyle=':')

ax.axhline(y=-1,
           linestyle=':')
```



1. Line Styles

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin,
        color='black')

ax.axhline(y=1,
           linestyle=':',
           color='red')

ax.axhline(y=-1,
           linestyle=':',
           color='blue')
```



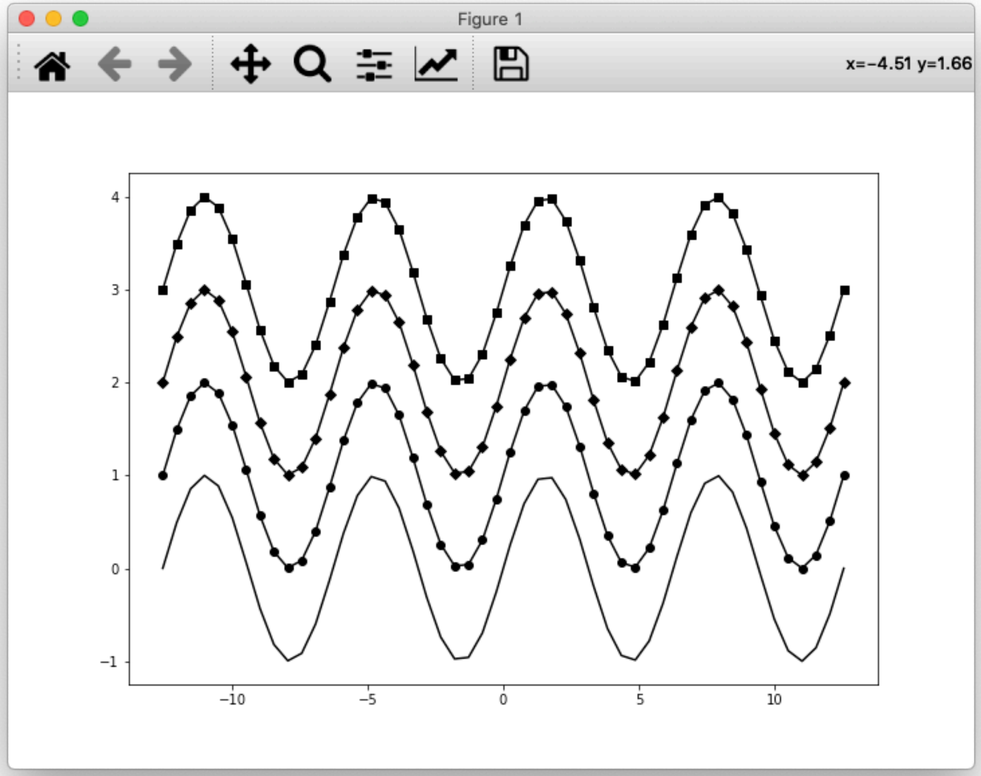
2. Markers

marker	symbol	description
`"."`		point
`","`		pixel
`"o"`		circle
`"v"`		triangle_down
`"^"`		triangle_up
`"<"`		triangle_left
`">"`		triangle_right
`"1"`		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
`0 (TICKLEFT)`		tickleft
`1 (TICKRIGHT)`		tickright
`2 (TICKUP)`		tickup
`3 (TICKDOWN)`		tickdown
`4 (CARETLEFT)`		caretleft
`5 (CARETRIGHT)`		caretright
`6 (CARETUP)`		caretup
`7 (CARETDOWN)`		caretdown
`8 (CARETLEFTBASE)`		caretleft (centered at base)
`9 (CARETRIGHTBASE)`		caretright (centered at base)
`10 (CARETUPBASE)`		caretup (centered at base)
`11 (CARETDOWNBASE)`		caretdown (centered at base)
`"None", " " or ""`		nothing
`'$...$'`	*f*	Render the string using mathtext. E.g `"$f$"` for marker showing the letter `f`.

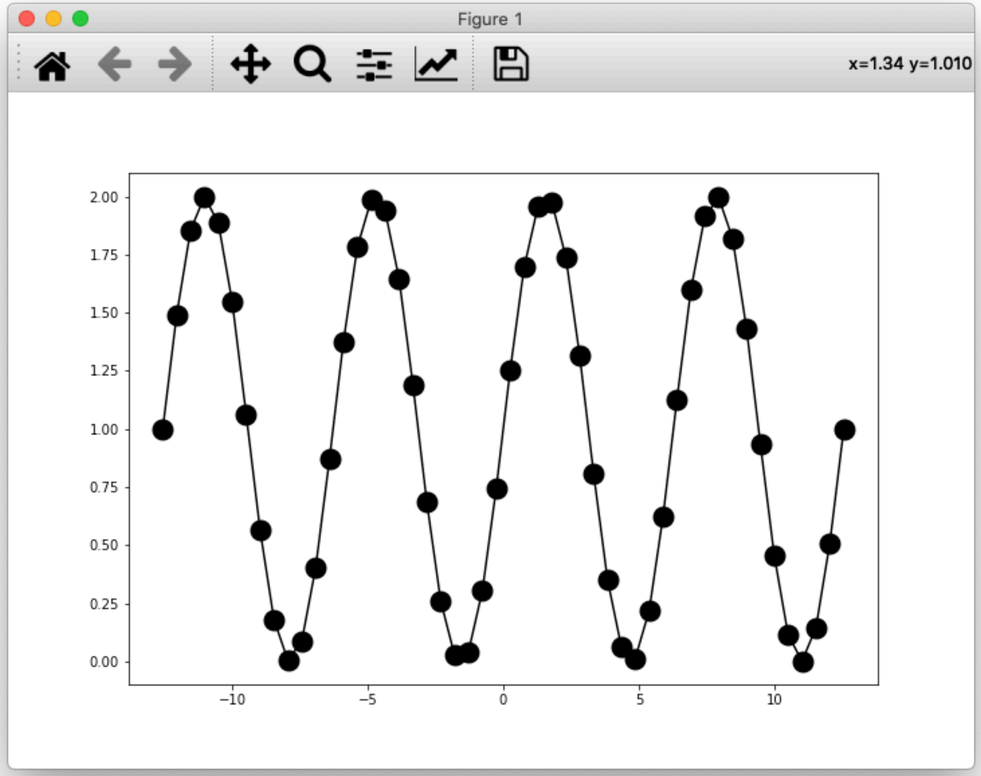
2. Markers

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin,
        color='black')

ax.plot(t, sin+1,
        marker='o',
        color='black')

ax.plot(t, sin+2,
        marker='D',
        color='black')

ax.plot(t, sin+3,
        marker='s',
        color='black')
```



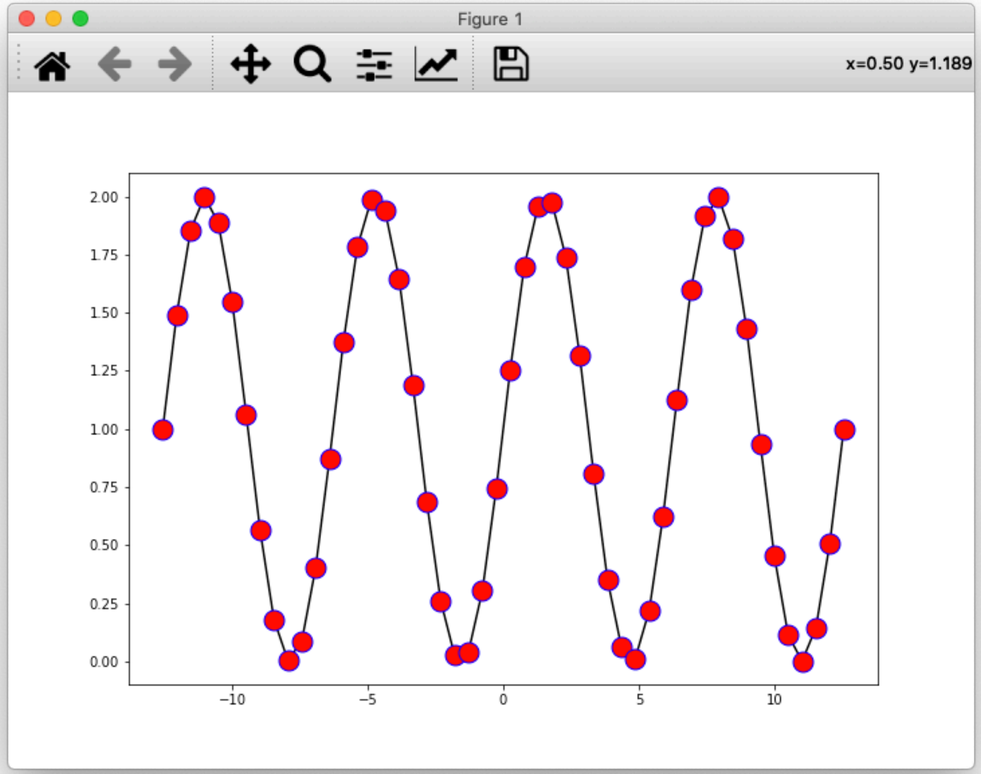
3. Customizing Markers

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15)
```



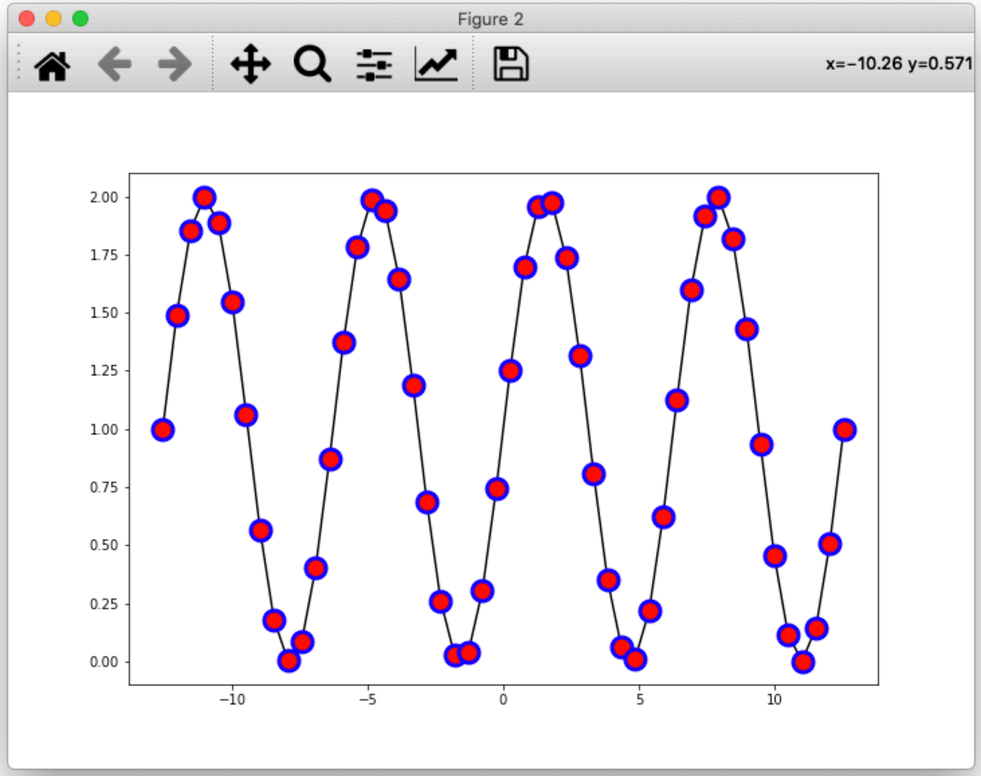
3. Customizing Markers

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15,
        markerfacecolor='r',
        markeredgecolor='b')
```



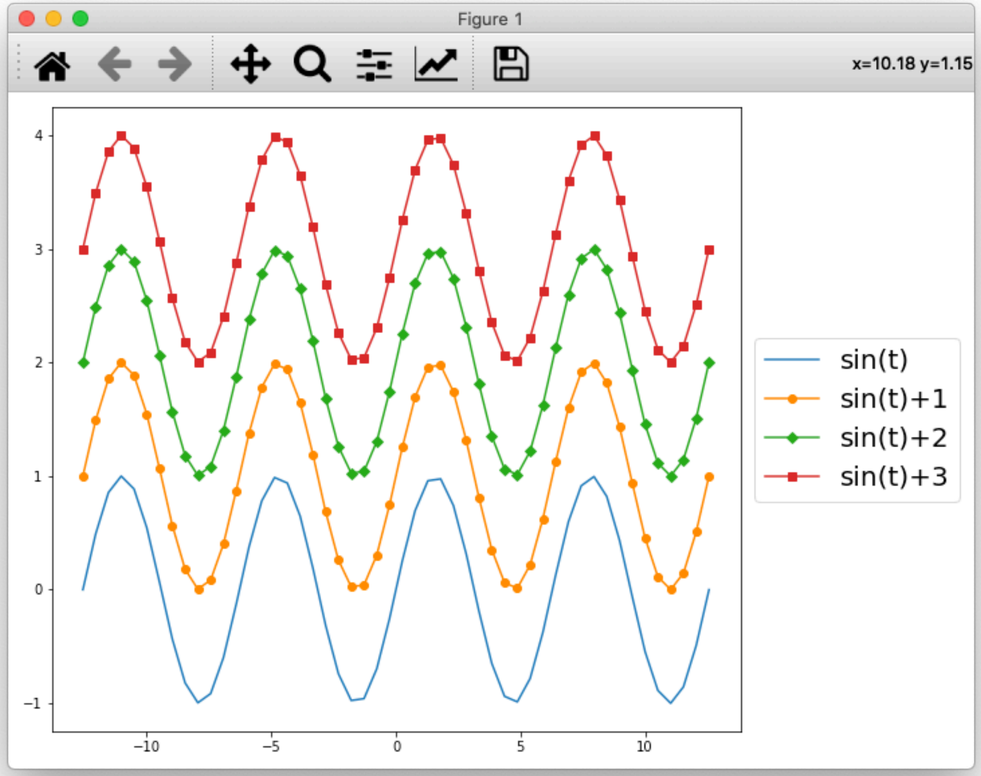
3. Customizing Markers

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15,
        markerfacecolor='r',
        markeredgecolor='b',
        markeredgewidth=3)
```



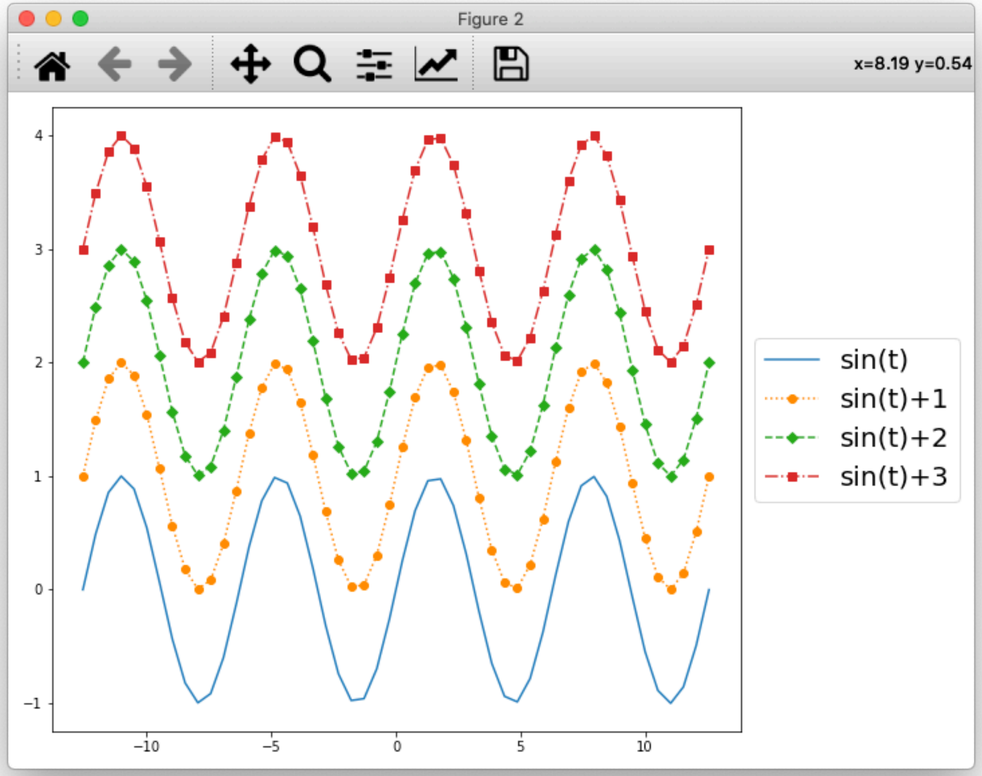
4. Line Styles/Markers with Legend

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 50)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin,
        label='sin(t)')

ax.plot(t, sin+1,
        marker='o',
        label='sin(t)+1')

ax.plot(t, sin+2,
        marker='D',
        label='sin(t)+2')

ax.plot(t, sin+3,
        marker='s',
        label='sin(t)+3')


ax.legend(loc='center left',
          bbox_to_anchor=(1, 0.5),
          fontsize=20)

fig.tight_layout()
```



4. Line Styles/Markers with Legend

```
ax.plot(t, sin,
        label='sin(t)')

ax.plot(t, sin+1,
        marker='o',
        label='sin(t)+1',
        linestyle=':')

ax.plot(t, sin+2,
        marker='D',
        label='sin(t)+2',
        linestyle='--')

ax.plot(t, sin+3,
        marker='s',
        label='sin(t)+3',
        linestyle='-.')

ax.legend(loc='center left',
          bbox_to_anchor=(1, 0.5),
          fontsize=20)
fig.tight_layout()
```



5. fmt Argument

Format Strings

A format string consists of a part for color, marker and line:

```
fmt = '[marker][line][color]'
```

Each of them is optional. If not provided, the value from the style cycle is used. Exception: If `line` is given, but no `marker`, the data will be a line without markers.

Other combinations such as `[color][marker][line]` are also supported, but note that their parsing may be ambiguous.

Markers

character	description
`'.'`	point marker
`','`	pixel marker
`'o'`	circle marker
`'v'`	triangle_down marker
`'^'`	triangle_up marker
`'<'`	triangle_left marker
`'>'`	triangle_right marker
`'1'`	tri_down marker
`'2'`	tri_up marker
`'3'`	tri_left marker
`'4'`	tri_right marker
`'s'`	square marker
`'p'`	pentagon marker
`'*'`	star marker
`'h'`	hexagon1 marker
`'H'`	hexagon2 marker
`'+'`	plus marker
`'x'`	x marker
`'D'`	diamond marker
`'d'`	thin_diamond marker
`'|'`	vline marker
`'_'`	hline marker

Line Styles

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

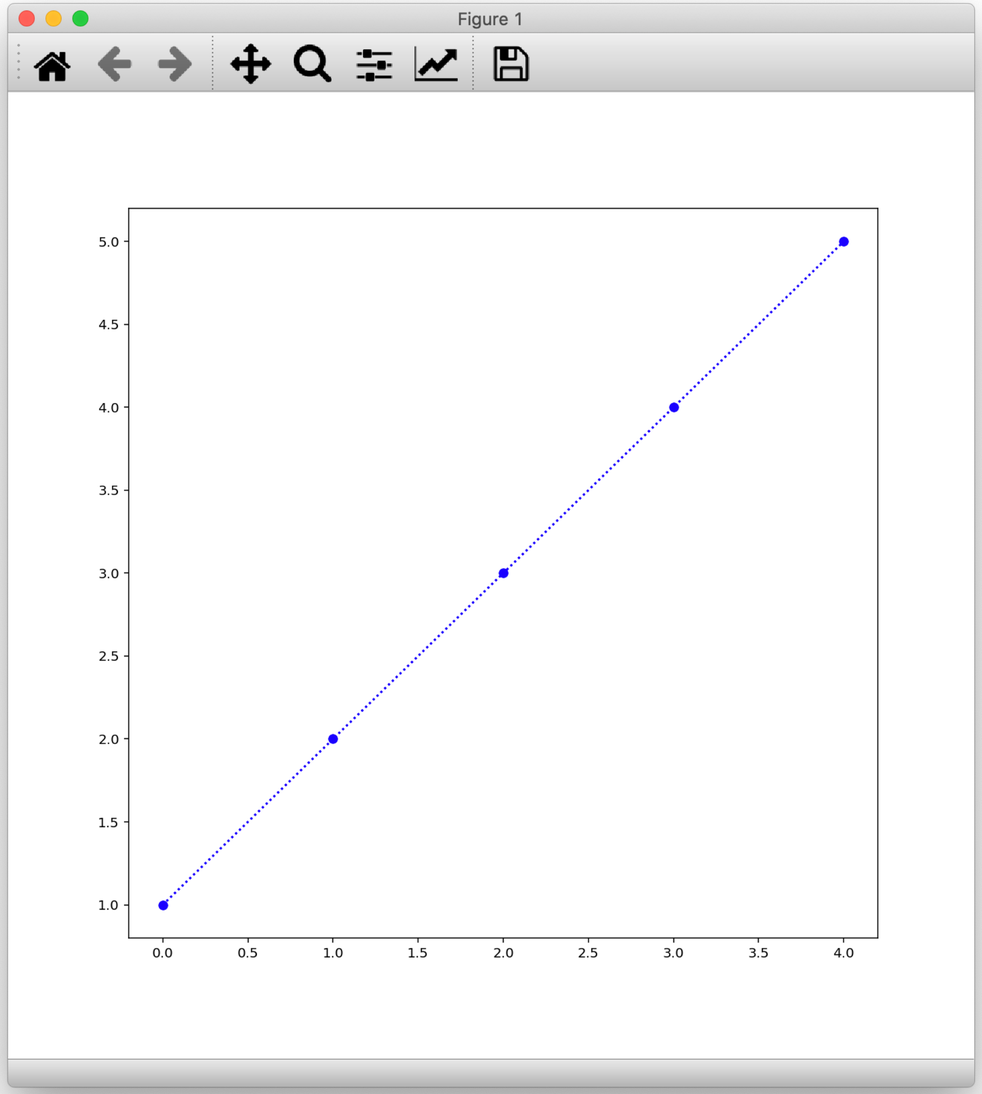
Colors

The supported color abbreviations are the single letter codes

character	color
`'b'`	blue
`'g'`	green
`'r'`	red
`'c'`	cyan
`'m'`	magenta
`'y'`	yellow
`'k'`	black
`'w'`	white

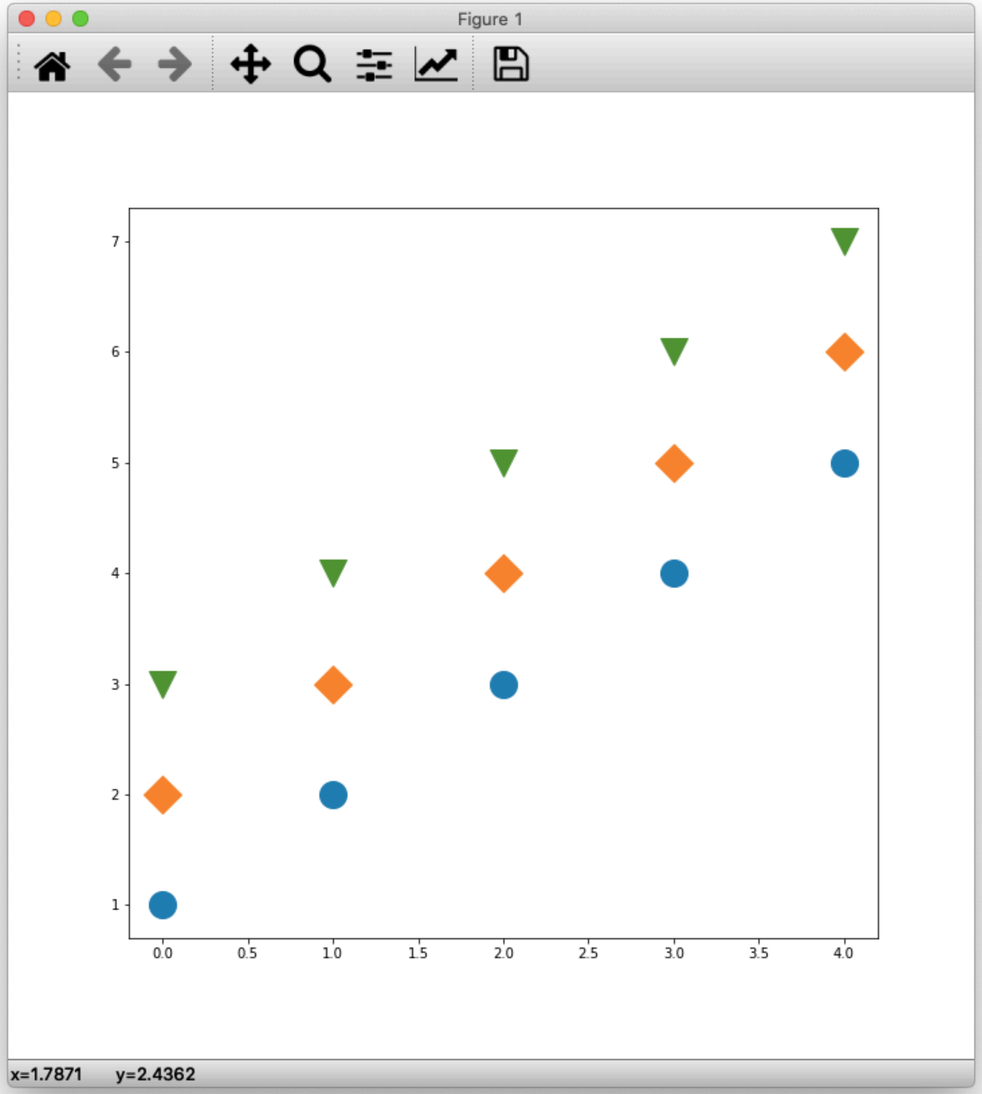
5. fmt Argument

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

x_data = np.array([1, 2, 3, 4, 5])

fig, ax = plt.subplots(figsize=(10, 10))


ax.plot(x_data,
        linestyle=':',
        marker='o',
        color='b')


ax.plot(x_data, ':ob')
```



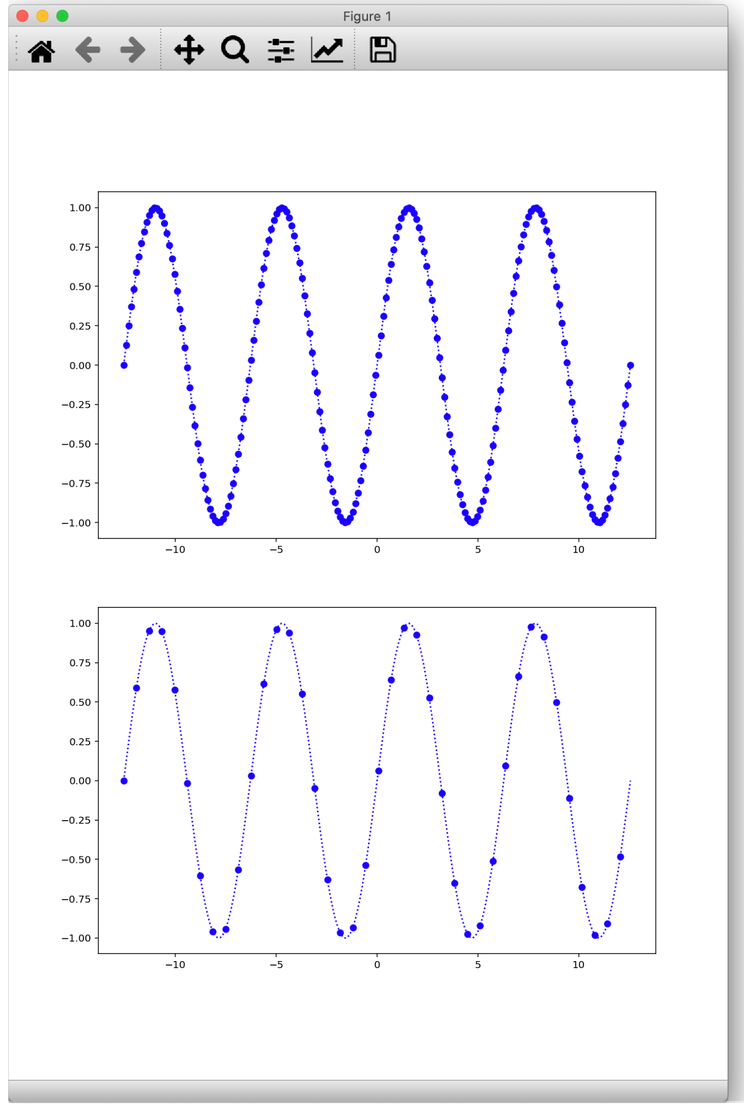
5. fmt Argument

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

x_data = np.array([1, 2, 3, 4, 5])

fig, ax = plt.subplots(figsize=(10, 10))


ax.plot(x_data, 'o',
        markersize=20)

ax.plot(x_data+1, 'D',
        markersize=20)

ax.plot(x_data+2, 'v',
        markersize=20)
```



5. fmt Argument

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

PI = np.pi
t = np.linspace(-4*PI, 4*PI, 200)
sin = np.sin(t)

t_mark = t[::5]
sin_mark = np.sin(t_mark)

fig, axes = plt.subplots(2, 1, figsize=(10, 20))
axes[0].plot(t, sin,
             'bo:')

axes[1].plot(t, sin,
             'b:')
axes[1].plot(t_mark, sin_mark,
             'bo')
```



Python for Data Visualization

-Chapter.2 Line Plot -

2-03. Line Styles and Markers

1. Line Styles
2. Markers
3. Customizing Markers
4. Line Styles/Markers with Legend
5. fmt Argument