- Shin’s Lab -

Python for Data Visualization

Python for Data Visualization

-Chapter.2 Line Plot -

Python for Data Visualization

-Chapter.2 Line Plot -

2-04. Line Filling

1. fill_between Basic Usage
2. where Argument
3. interpolate Argument

1. fill_between Basic Usage

matplotlib.pyplot.fill_between

matplotlib.pyplot.**fill_between**(*x*, *y1*, *y2=0*, *where=None*, *interpolate=False*, *step=None*, ***, *data=None*, ***kwargs*) [source]

Fill the area between two horizontal curves.

The curves are defined by the points (*x*, *y1*) and (*x*, *y2*). This creates one or multiple polygons describing the filled area.

You may exclude some horizontal sections from filling using *where*.

By default, the edges connect the given points directly. Use *step* if the filling should be a step function, i.e. constant in between *x*.

matplotlib.pyplot.fill_betweenx

matplotlib.pyplot.**fill_betweenx**(*y*, *x1*, *x2=0*, *where=None*, *step=None*, *interpolate=False*, ***, *data=None*, ***kwargs*) [source]

Fill the area between two vertical curves.

The curves are defined by the points (*y*, *x1*) and (*y*, *x2*). This creates one or multiple polygons describing the filled area.

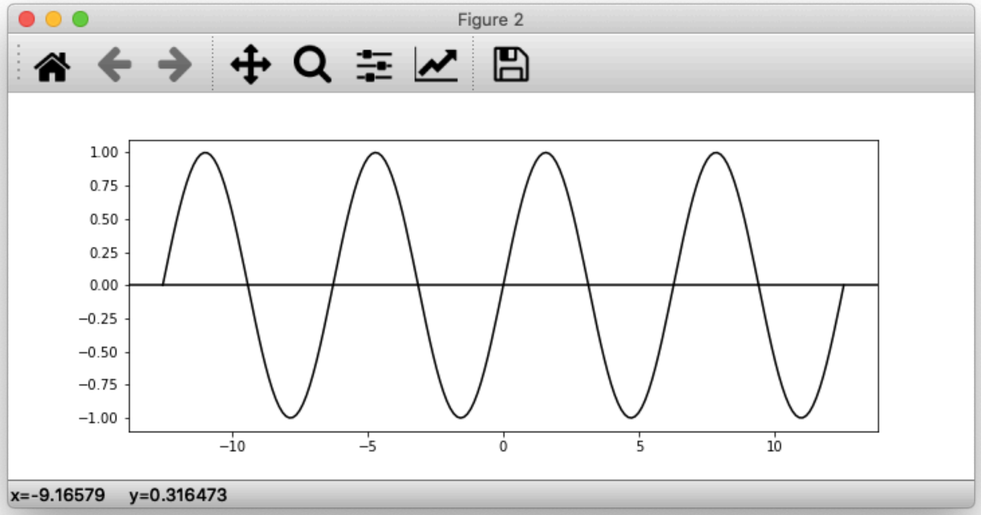
You may exclude some vertical sections from filling using *where*.

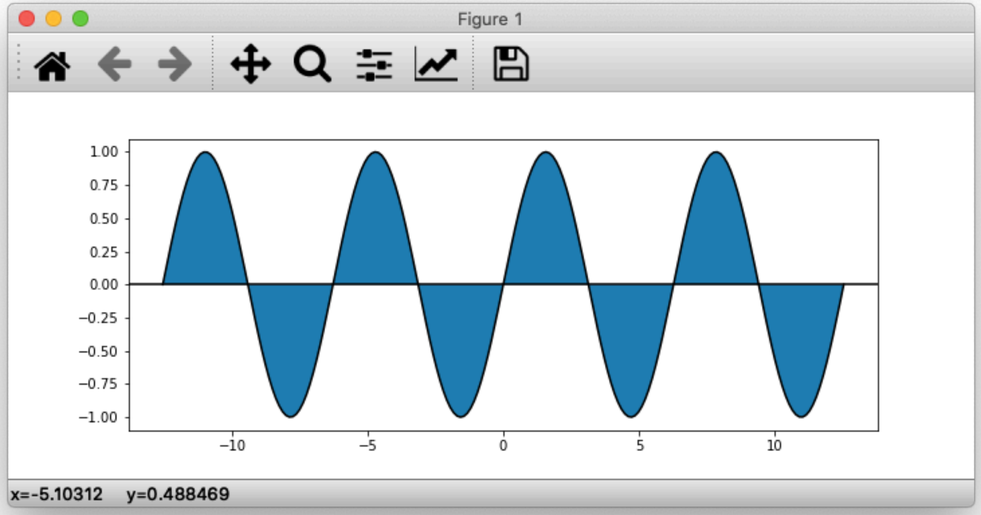
By default, the edges connect the given points directly. Use *step* if the filling should be a step function, i.e. constant in between *y*.

https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.fill_between.html#matplotlib.pyplot.fill_between
https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.fill_betweenx.html#matplotlib.pyplot.fill_betweenx

1. fill_between Basic Usage

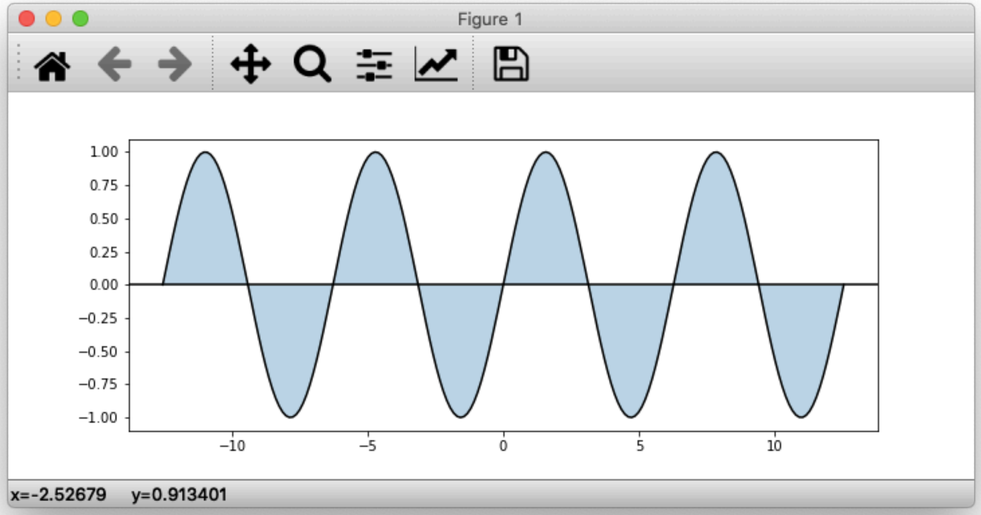
```
import matplotlib.pyplot as plt
import numpy as np

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

fig, ax = plt.subplots(figsize=(10, 7))
ax.plot(t, sin,
        color='black')
ax.axhline(0,
           color='black')

---------------------------------
ax.fill_between(t, sin)
```





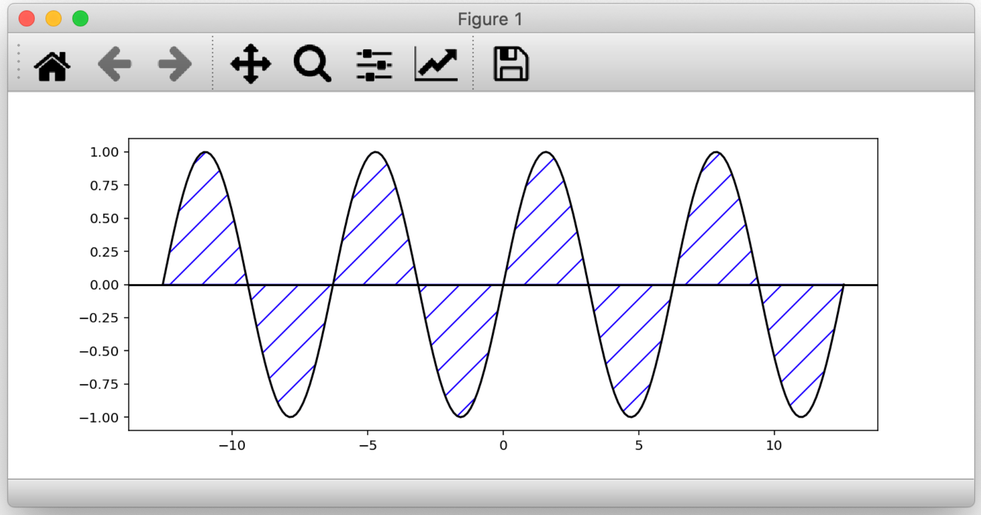
1. fill_between Basic Usage

```
ax.fill_between(t, sin,
                alpha=0.3)
```



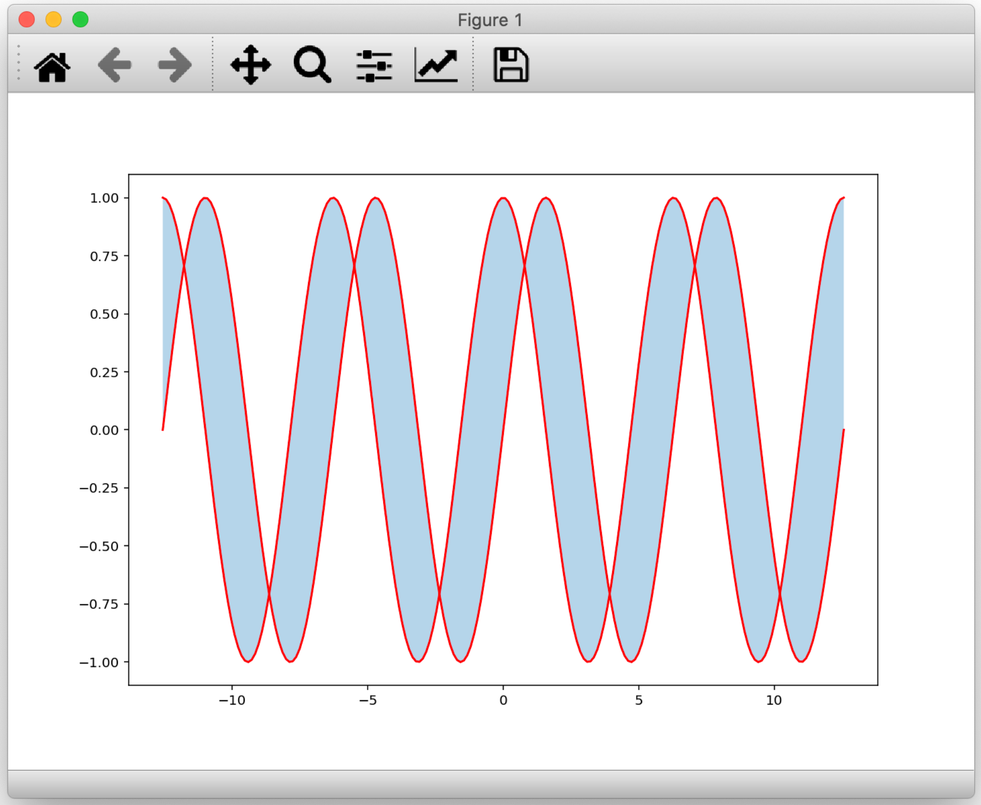

```
ax.fill_between(t, sin,
                facecolor='white',
                hatch='/',
                edgecolor='b')
```

hatch	{'/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}



1. fill_between Basic Usage

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

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

fig, ax = plt.subplots(figsize=(10, 7))
ax.plot(t, sin,
        color='r')
ax.plot(t, cos,
        color='r')

ax.fill_between(t, sin, cos,
                alpha=0.3)
```



2. where Argument

matplotlib.pyplot.fill_between

matplotlib.pyplot.fill_between(*x*, *y1*, *y2=0*, *where=None*, *interpolate=False*, *step=None*, ***, *data=None*, ***kwargs*) [source]

Fill the area between two horizontal curves.

The curves are defined by the points (*x*, *y1*) and (*x*, *y2*). This creates one or multiple polygons describing the filled area.

You may exclude some horizontal sections from filling using *where*.

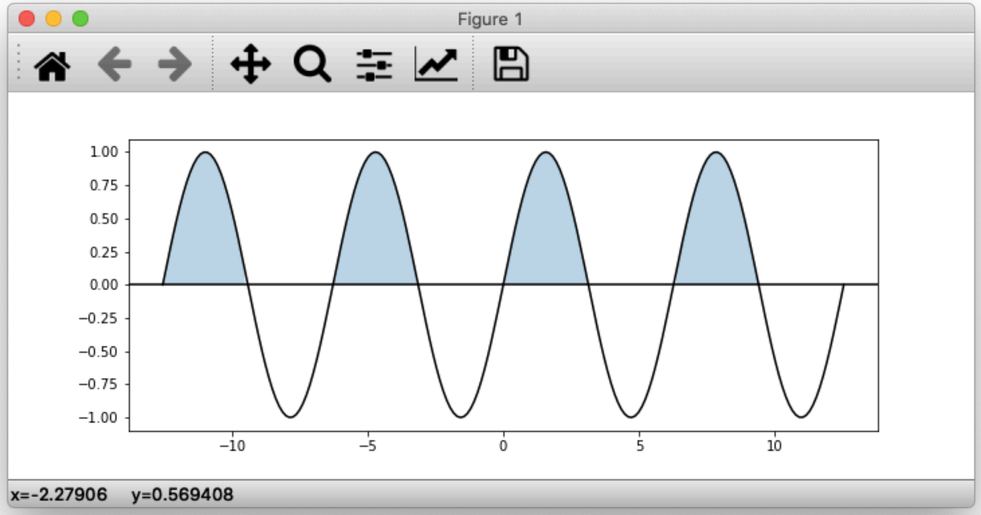
By default, the edges connect the given points directly. Use *step* if the filling should be a step function, i.e. constant in between *x*.

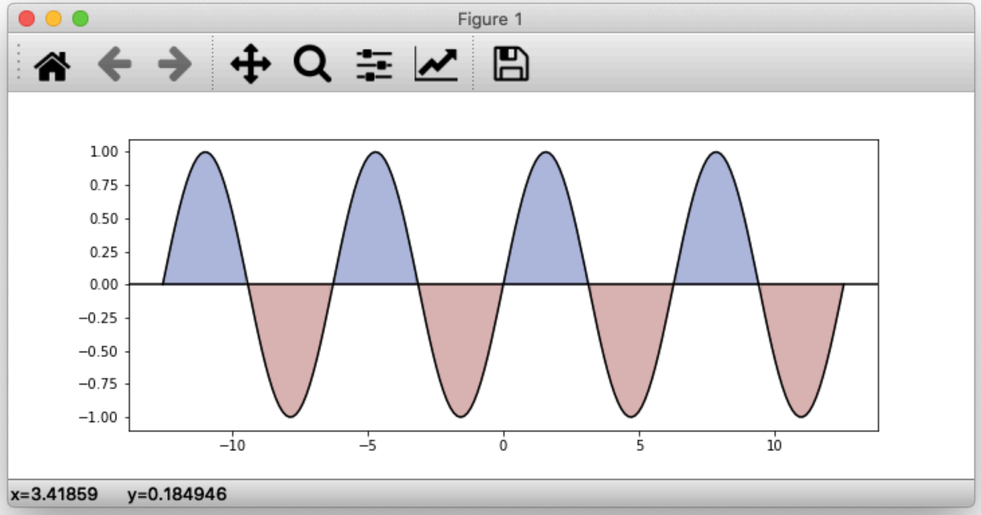
where : array of bool (length N), optional

Define *where* to exclude some horizontal regions from being filled. The filled regions are defined by the coordinates `x[where]`. More precisely, fill between `x[i]` and `x[i+1]` if `where[i] and where[i+1]`. Note that this definition implies that an isolated *True* value between two *False* values in *where* will not result in filling. Both sides of the *True* position remain unfilled due to the adjacent *False* values.

2. where Argument

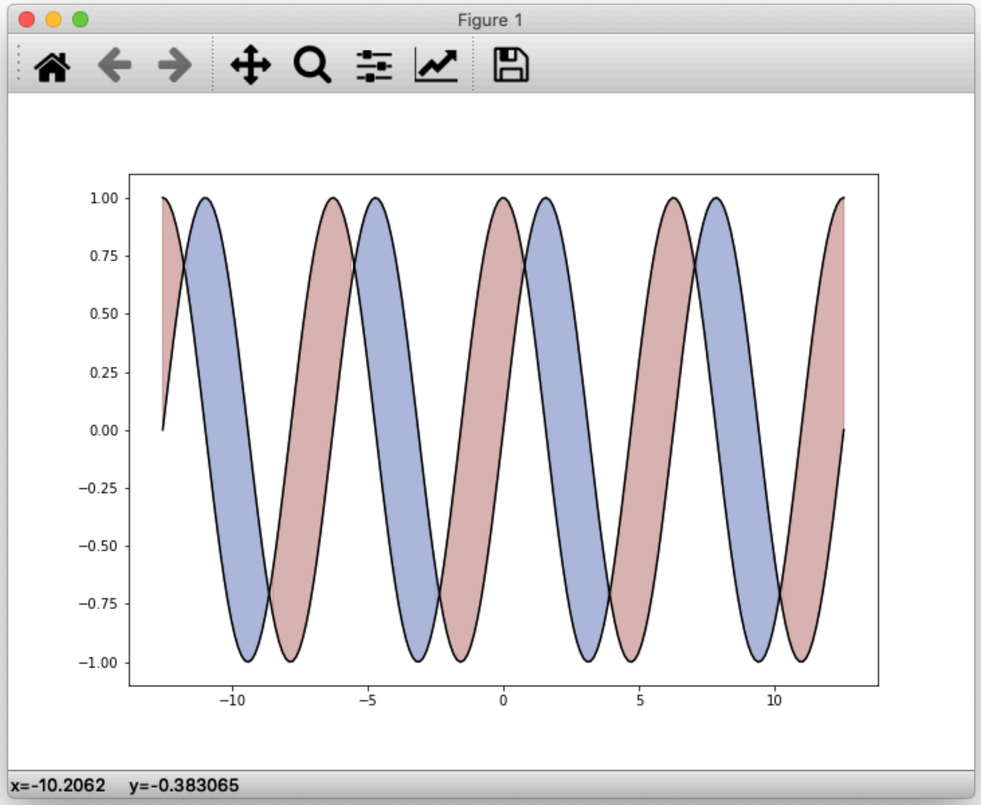
```
import matplotlib.pyplot as plt
import numpy as np

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

sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(t, sin,
        color='black')
ax.axhline(0,
           color='black')

ax.fill_between(t, sin,
                alpha=0.3,
                where=sin>=0)
-------------------------------------------
ax.fill_between(t, sin,
                color='darkred',
                alpha=0.3,
                where=sin<0)
```





2. where 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)
cos = np.cos(t)

fig, ax = plt.subplots(figsize=(10, 7))
ax.plot(t, sin,
        color='black')
ax.plot(t, cos,
        color='black')


ax.fill_between(t, sin, cos,
                color='darkblue',
                alpha=0.3,
                where=sin>=cos)


ax.fill_between(t, sin, cos,
                color='darkred',
                alpha=0.3,
                where=sin<cos)
```



3. interpolate Argument

matplotlib.pyplot.fill_between

matplotlib.pyplot.fill_between(*x*, *y1*, *y2=0*, *where=None*, *interpolate=False*, *step=None*, *, *data=None*, **kwargs*) [source]

Fill the area between two horizontal curves.

The curves are defined by the points (*x*, *y1*) and (*x*, *y2*). This creates one or multiple polygons describing the filled area.

You may exclude some horizontal sections from filling using *where*.

By default, the edges connect the given points directly. Use *step* if the filling should be a step function, i.e. constant in between *x*.

interpolate : bool, default: False

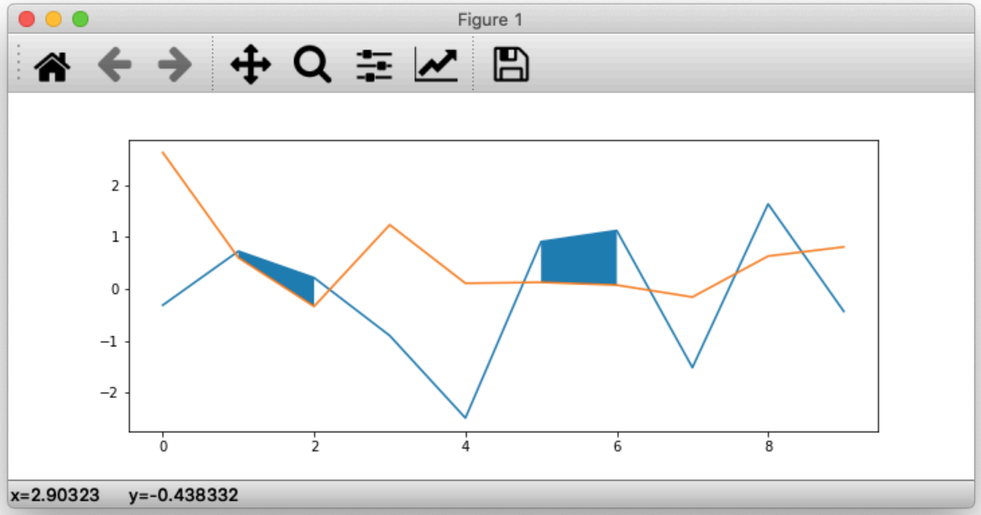
This option is only relevant if *where* is used and the two curves are crossing each other.

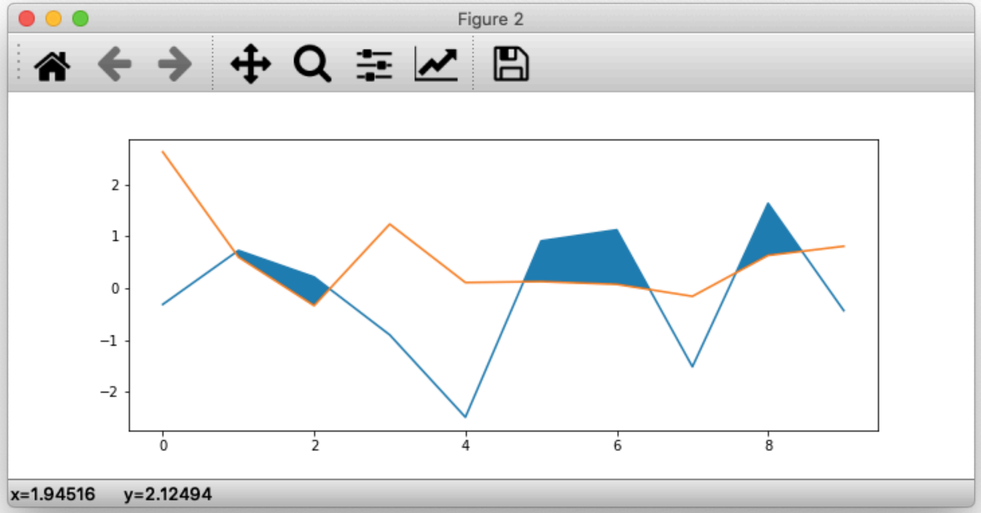
Semantically, *where* is often used for *x1* > *x2* or similar. By default, the nodes of the polygon defining the filled region will only be placed at the positions in the *y* array. Such a polygon cannot describe the above semantics close to the intersection. The y-sections containing the intersection are simply clipped.

Setting *interpolate* to *True* will calculate the actual intersection point and extend the filled region up to this point.

3. interpolate Argument

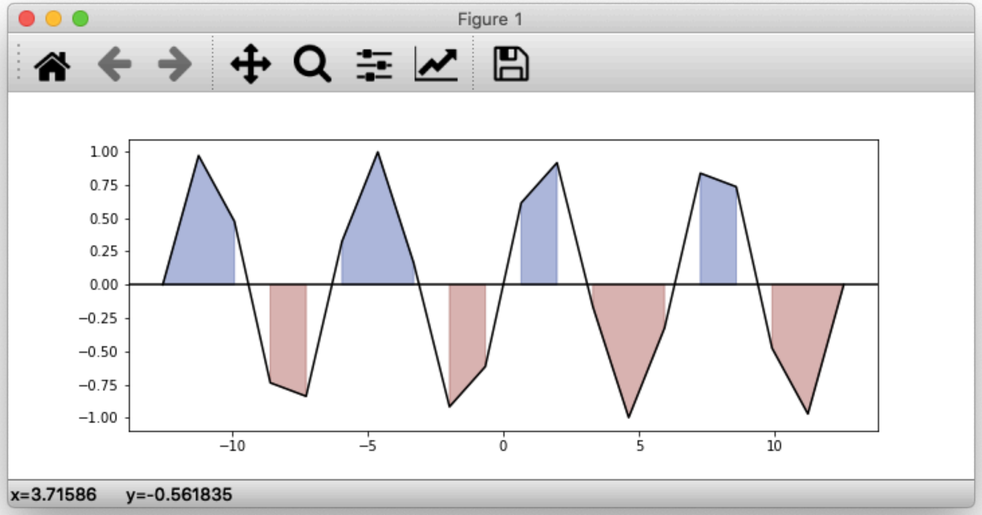
```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(6)

n_data = 10
data_idx = np.arange(0, n_data)
noise1 = np.random.normal(0, 1, (n_data,))
noise2 = np.random.normal(0, 1, (n_data,))

fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(data_idx, noise1)
ax.plot(data_idx, noise2)

ax.fill_between(data_idx, noise1, noise2,
                where=noise1>=noise2)

------------------------------------------------------------
ax.fill_between(data_idx, noise1, noise2,
                where=noise1>=noise2,
                interpolate=True)
```



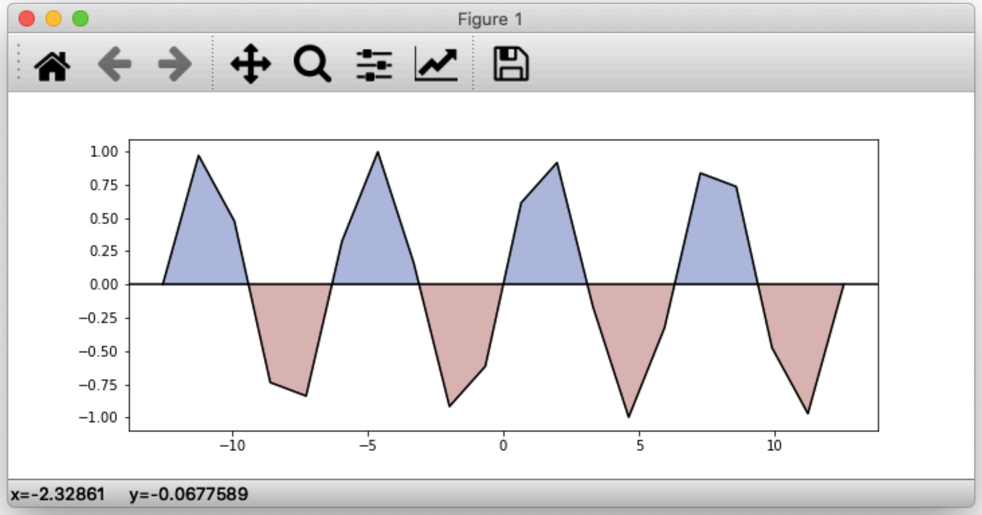


3. interpolate Argument

```
import matplotlib.pyplot as plt
import numpy as np
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 20)
sin = np.sin(t)
cos = np.cos(t)
fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(t, sin,
        color='black')
ax.axhline(0,
           color='black')
ax.fill_between(t, sin,
                color='darkblue',
                alpha=0.3,
                where=sin>=0)
ax.fill_between(t, sin,
                color='darkred',
                alpha=0.3,
                where=sin<0)
```




```
ax.fill_between(t, sin,
                color='darkblue',
                alpha=0.3,
                where=sin>=0,
                interpolate=True)
ax.fill_between(t, sin,
                color='darkred',
                alpha=0.3,
                where=sin<0,
                interpolate=True)
```



Python for Data Visualization

-Chapter.2 Line Plot -

2-04. Line Filling

1. fill_between Basic Usage
2. where Argument
3. interpolate Argument