- Shin’s Lab -

Python for Data Visualization

Python for Data Visualization

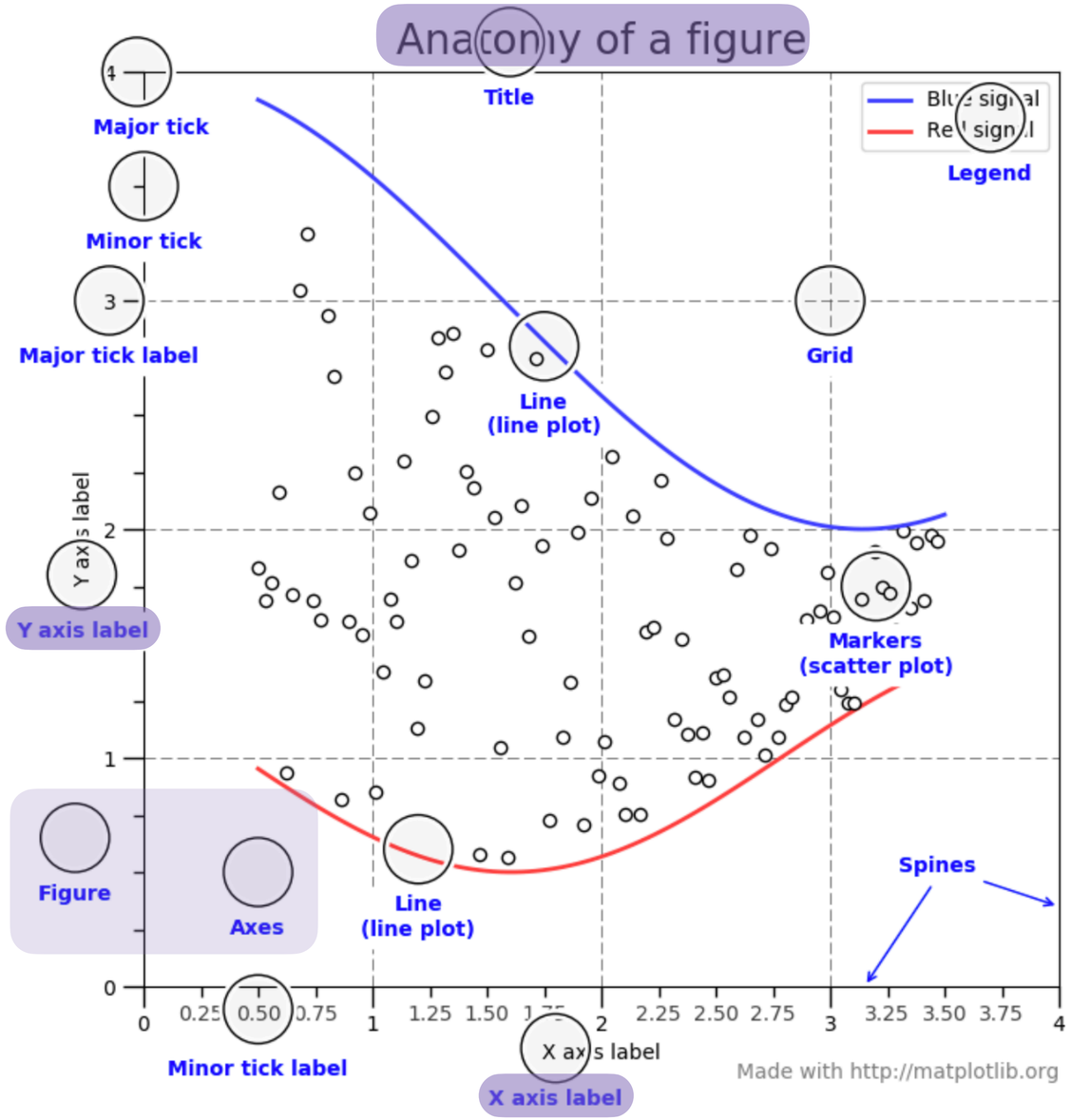
-Chapter.1 Matplotlib Anatomy -

Python for Data Visualization

-Chapter.1 Matplotlib Anatomy -

1-03. Titles, Labels and Font Dict

1. Title APIs
2. fig.suptitle and ax.set_title
3. ax.set_xlabel and ax.set_ylabel
4. Text Alignment
5. Title Alignment
6. Text Properties
7. Font Dictionary
8. Title Exercise


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Legend
Grid
Line
(line plot)
Y axis label
Markers
(scatter plot)
Figure
Axes
Line
(line plot)
Spines
Minor tick label
X axis label
Made with http://matplotlib.org

1. Title APIs(fig.suptitle)

matplotlib.pyplot.suptitle

`matplotlib.pyplot.suptitle(t, **kwargs)` [source]

Add a centered title to the figure.

Parameters:

t : str

The title text.

x : float, default 0.5

The x location of the text in figure coordinates.

y : float, default 0.98

The y location of the text in figure coordinates.

horizontalalignment, ha : {'center', 'left', right'}, default: 'center'

The horizontal alignment of the text relative to (*x*, *y*).

verticalalignment, va : {'top', 'center', 'bottom', 'baseline'}, default: 'top'

The vertical alignment of the text relative to (*x*, *y*).

fontsize, size : default: `rcParams["figure.titlesize"]` (default: `'large'`)

The font size of the text. See `Text.set_size` for possible values.

fontweight, weight : default: `rcParams["figure.titleweight"]` (default: `'normal'`)

The font weight of the text. See `Text.set_weight` for possible values.

Other Parameters:

fontproperties : None or dict, optional

A dict of font properties. If *fontproperties* is given the default values for font size and weight are taken from the `FontProperties` defaults. `rcParams["figure.titlesize"]` (default: `'large'`) and `rcParams["figure.titleweight"]` (default: `'normal'`) are ignored in this case.

****kwargs**

Additional kwargs are `matplotlib.text.Text` properties.

1. Title APIs(ax.set_title)

matplotlib.axes.Axes.set_title

`Axes.set_title(self, label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)` [source]

Set a title for the axes.

Set one of the three available axes titles. The available titles are positioned above the axes in the center, flush with the left edge, and flush with the right edge.

Parameters:

label : str
Text to use for the title

fontdict : dict
A dictionary controlling the appearance of the title text, the default *fontdict* is:

```
{'fontsize': rcParams['axes.titlesize'],
 'fontweight': rcParams['axes.titleweight'],
 'color': rcParams['axes.titlecolor'],
 'verticalalignment': 'baseline',
 'horizontalalignment': loc}
```

loc : {'center', 'left', 'right'}, default: `rcParams["axes.titlelocation"]` (default: `'center'`)
Which title to set.

y : float, default: `rcParams["axes.titley"]` (default: `None`)
Vertical axes loation for the title (1.0 is the top). If None (the default), y is determined automatically to avoid decorators on the axes.

pad : float, default: `rcParams["axes.titlepad"]` (default: `6.0`)
The offset of the title from the top of the axes, in points.

Returns:

`Text`
The matplotlib text instance representing the title

Other Parameters:

****kwargs** : `Text` properties
Other keyword arguments are text properties, see `Text` for a list of valid text properties.

https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.set_title.html

1. Title APIs(ax.set_xlabel, ax.set_ylabel)

matplotlib.axes.Axes.set_xlabel

`Axes.set_xlabel(self, xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the x-axis.

Parameters:

xlabel : str
The label text.

labelpad : float, default: None
Spacing in points from the axes bounding box including ticks and tick labels.

loc : {'left', 'center', 'right'}, default: `rcParams["xaxis.labellocation"]` (default: `'center'`)
The label position. This is a high-level alternative for passing parameters *x* and *horizontalalignment*.

Other Parameters:

****kwargs** : `Text` properties
`Text` properties control the appearance of the label.

matplotlib.axes.Axes.set_ylabel

`Axes.set_ylabel(self, ylabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the y-axis.

1. Title APIs(Text Properties)

class matplotlib.text.Text(*x=0*, *y=0*, *text=''*, *color=None*, *verticalalignment='baseline'*, *horizontalalignment='left'*, *multialignment=None*, *fontproperties=None*, *rotation=None*, *linespacing=None*, *rotation_mode=None*, *usetex=None*, *wrap=False*, ***kwargs*) [source]

Bases: matplotlib.artist.Artist

Handle storing and drawing of text in window or data coordinates.

Create a Text instance at *x*, *y* with string *text*.

Valid keyword arguments are:

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color
contains	unknown
figure	Figure
fontfamily or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
fontproperties or font or font_properties	font_manager.FontProperties or str or pathlib.Path
fontstretch or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
fontstyle or style	{'normal', 'italic', 'oblique'}
fontvariant or variant	{'normal', 'small-caps'}
fontweight or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}
gid	str
horizontalalignment or ha	{'center', 'right', 'left'}
in_layout	bool
label	object
linespacing	float (multiple of font size)
multialignment or ma	{'left', 'right', 'center'}
path_effects	AbstractPathEffect
picker	None or bool or callable
position	(float, float)
rasterized	bool or None
rotation	float or {'vertical', 'horizontal'}
rotation_mode	{None, 'default', 'anchor'}
sketch_params	(scale: float, length: float, randomness: float)
snap	bool or None
text	object
transform	Transform
url	str
usetex	bool or None
verticalalignment or va	{'center', 'top', 'bottom', 'baseline', 'center_baseline'}
visible	bool
wrap	bool
x	float
y	float
zorder	float

2. fig.suptitle and ax.set_title(Document)

matplotlib.pyplot.suptitle

matplotlib.pyplot.suptitle(*t*, ***kwargs*) [source]

Add a centered title to the figure.

matplotlib.axes.Axes.set_title

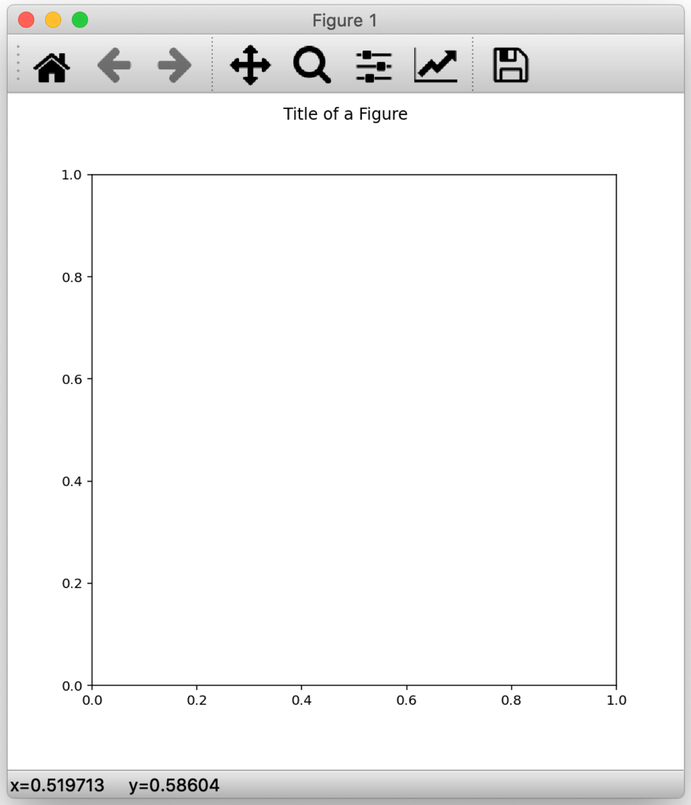
Axes.set_title(*self*, *label*, *fontdict=None*, *loc=None*, *pad=None*, ***, *y=None*, ***kwargs*) [source]

Set a title for the axes.

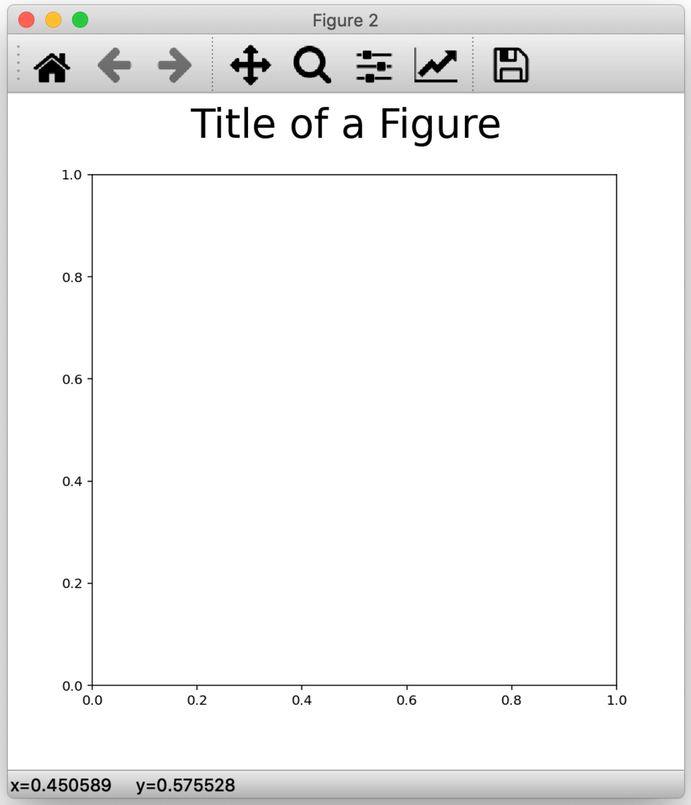
Set one of the three available axes titles. The available titles are positioned above the axes in the center, flush with the left edge, and flush with the right edge.

2. fig.suptitle and ax.set_title(Title of Figure)

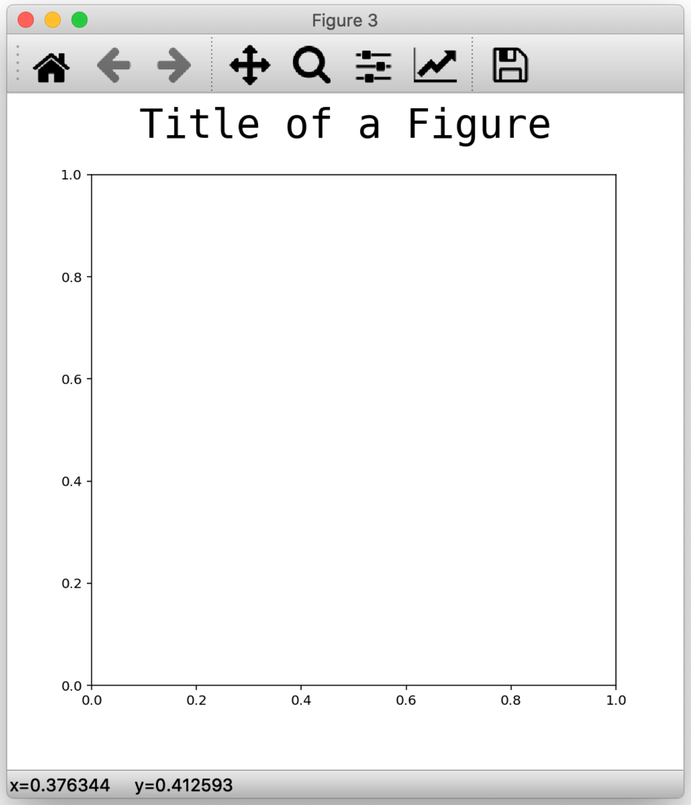
```
import matplotlib.pyplot as plt
figsize = (7, 7)
fig, ax = plt.subplots(figsize=figsize)
fig.suptitle("Title of a Figure")
```




```
fig.suptitle("Title of a Figure",
             fontsize=30)
```




```
fig.suptitle("Title of a Figure",
             fontsize=30,
             fontfamily='monospace')
```

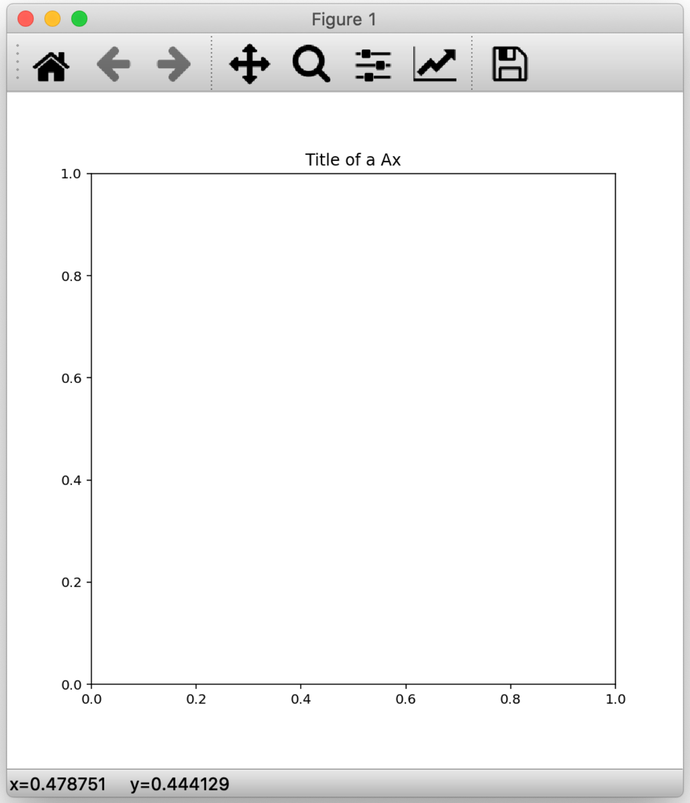


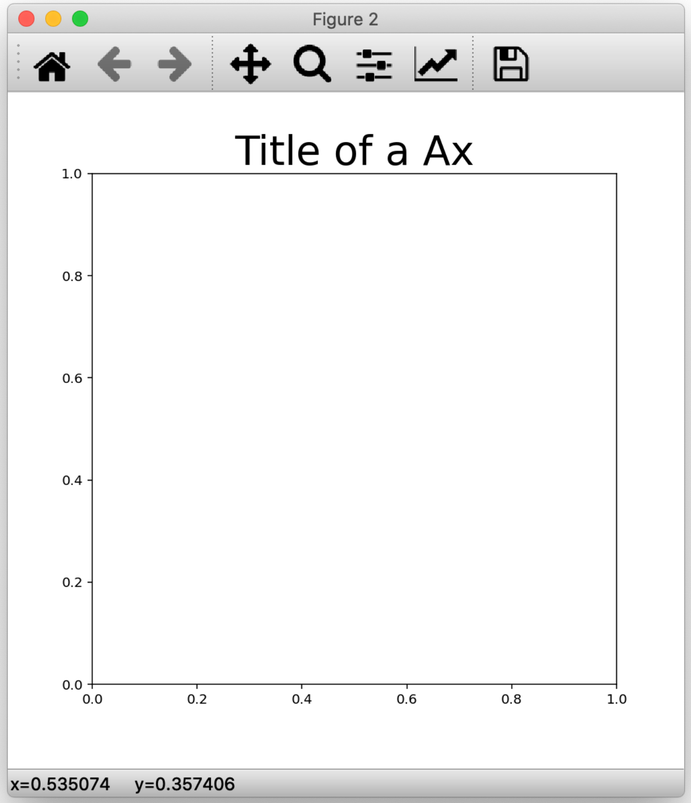
2. fig.suptitle and ax.set_title(Title of Ax)

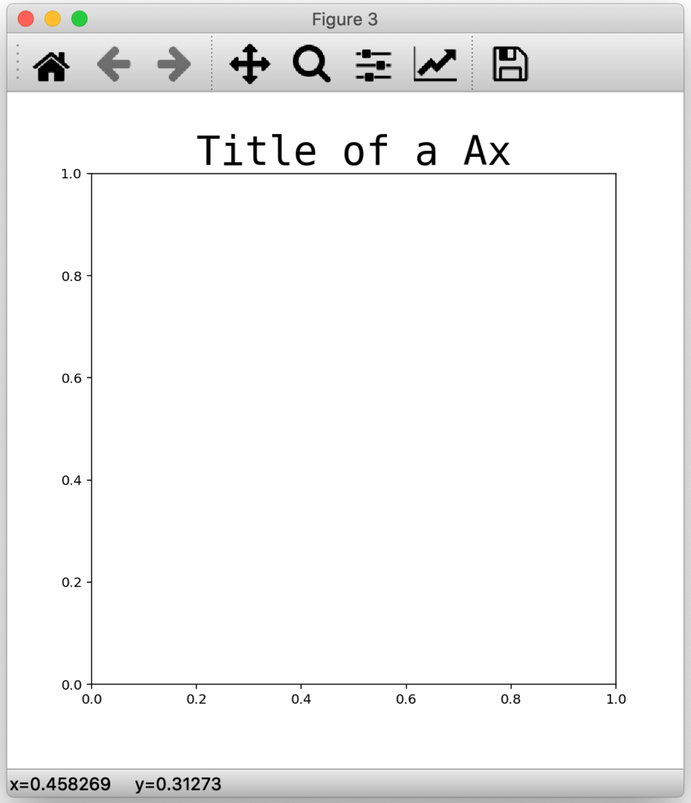
```
import matplotlib.pyplot as plt
figsize = (7, 7)
fig, ax = plt.subplots(figsize=figsize)
ax.set_title("Title of a Ax")
```

```
ax.set_title("Title of a Ax",
             fontsize=30)
```

```
ax.set_title("Title of a Ax",
             fontsize=30,
             fontfamily='monospace')
```

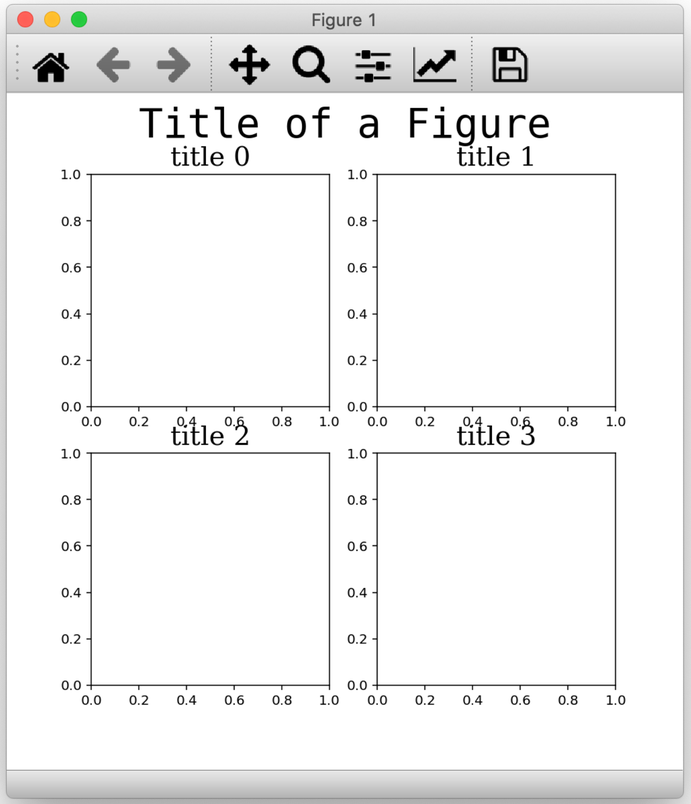






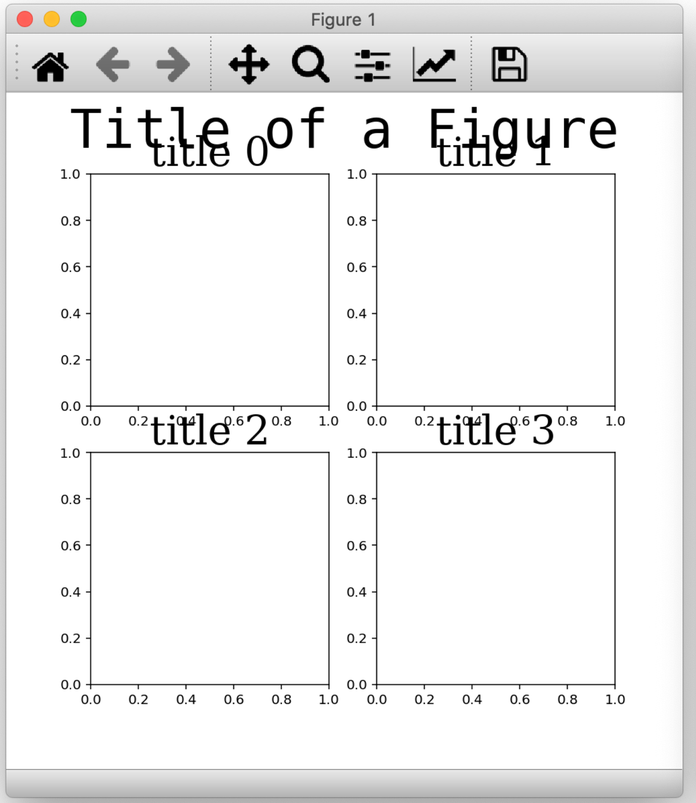
2. fig.suptitle and ax.set_title(with Many Axes)

```
import matplotlib.pyplot as plt

figsize = (7, 7)
ax_title_list = ['title ' + str(i) for i in range(4)]

fig, axes = plt.subplots(2, 2, figsize=figsize)
fig.suptitle("Title of a Figure",
             fontsize=30,
             fontfamily='monospace')

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(ax_title_list[ax_idx],
                 fontsize=20,
                 fontfamily='serif')
```

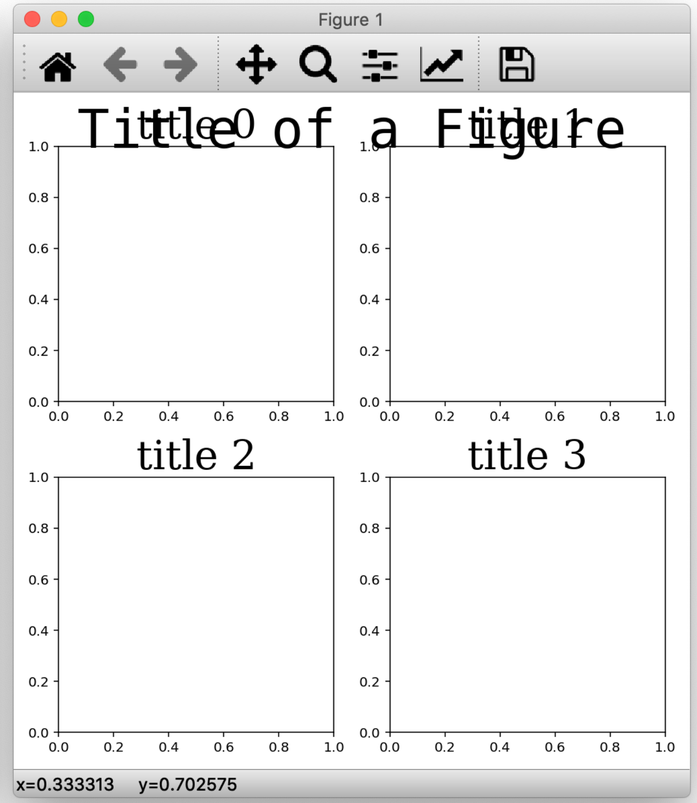


2. fig.suptitle and ax.set_title(with fig.tight_layout)

```
import matplotlib.pyplot as plt
figsize = (7, 7)
ax_title_list = ['title ' + str(i) for i in range(4)]

fig, axes = plt.subplots(2, 2, figsize=figsize)
fig.suptitle("Title of a Figure",
             fontsize=40,
             fontfamily='monospace')

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(ax_title_list[ax_idx],
                 fontsize=30,
                 fontfamily='serif')

fig.tight_layout()
```



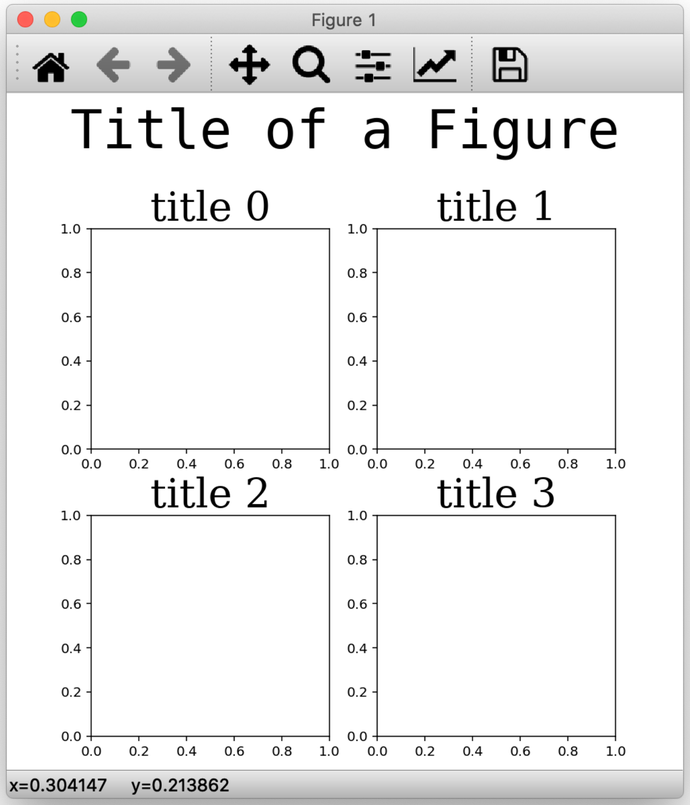
before tight_layout()



after tight_layout()

2. fig.suptitle and ax.set_title(with fig.subplots_adjust)

```
import matplotlib.pyplot as plt
figsize = (7, 7)
ax_title_list = ['title ' + str(i) for i in range(4)]

fig, axes = plt.subplots(2, 2, figsize=figsize)
fig.suptitle("Title of a Figure",
             fontsize=40,
             fontfamily='monospace')

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(ax_title_list[ax_idx],
                 fontsize=30,
                 fontfamily='serif')

fig.subplots_adjust(bottom=0.05, top=0.8,
                    hspace=0.3)
```



3. ax.set_xlabel and ax.set_ylabel(Document)

matplotlib.axes.Axes.set_xlabel

`Axes.set_xlabel(self, xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the x-axis.

Parameters:

xlabel : str

The label text.

labelpad : float, default: None

Spacing in points from the axes bounding box including ticks and tick labels.

loc : {'left', 'center', 'right'}, default: `rcParams["xaxis.labellocation"]` (default: `'center'`)

The label position. This is a high-level alternative for passing parameters *x* and *horizontalalignment*.

Other Parameters:

****kwargs** : `Text` properties

`Text` properties control the appearance of the label.

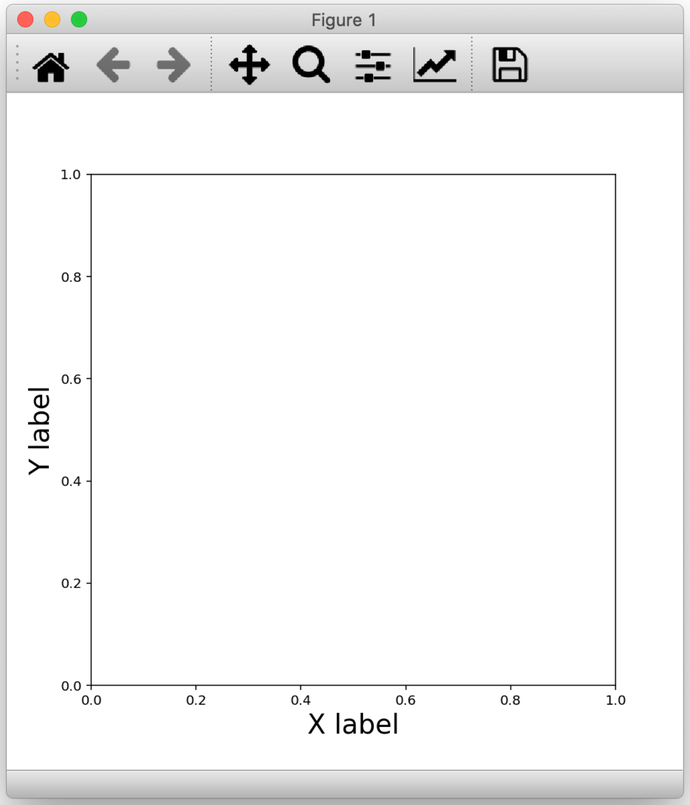
matplotlib.axes.Axes.set_ylabel

`Axes.set_ylabel(self, ylabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the y-axis.

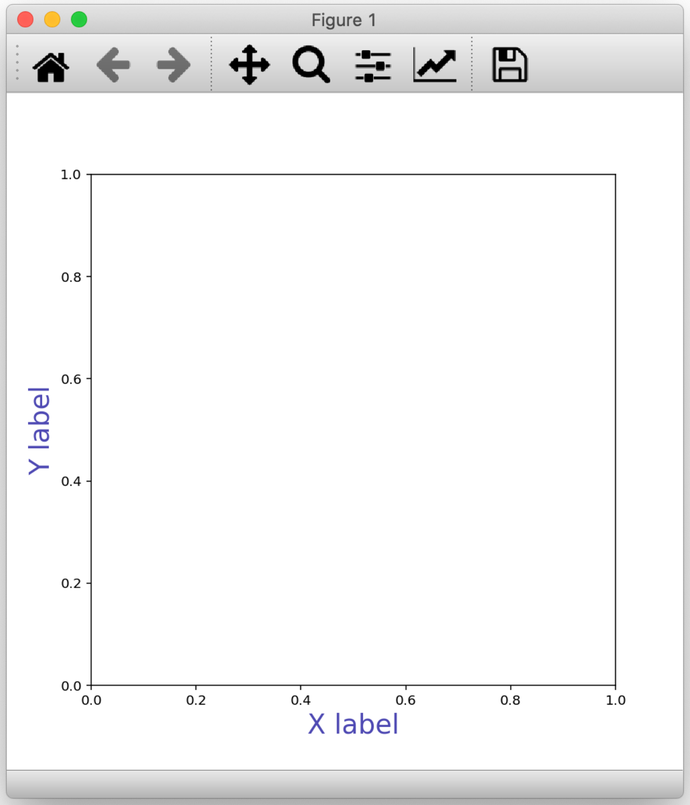
3. ax.set_xlabel and ax.set_ylabel(Usage)

```
import matplotlib.pyplot as plt

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

ax.set_xlabel("X label",
              fontsize=20)
ax.set_ylabel("Y label",
              fontsize=20)
```



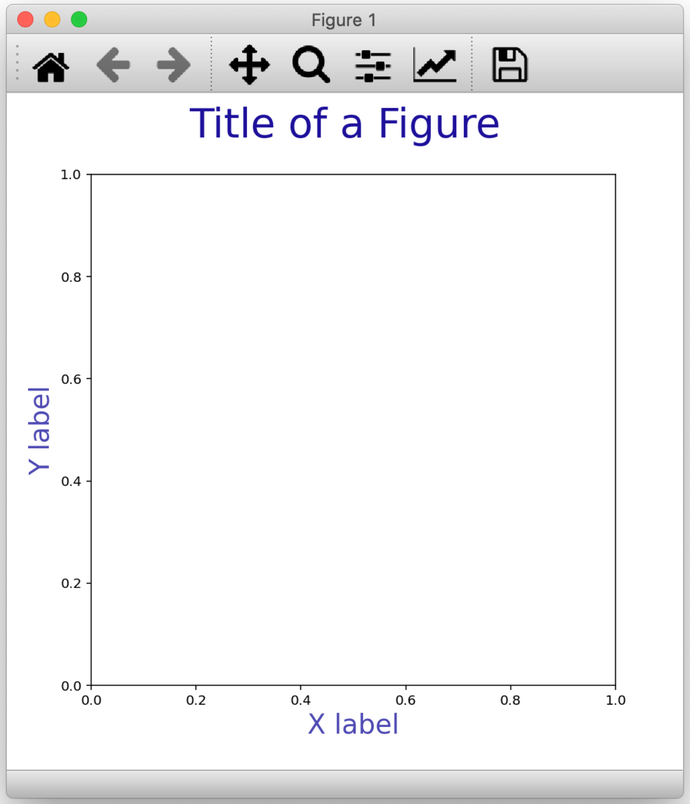
3. ax.set_xlabel and ax.set_ylabel(Arguments)

```
import matplotlib.pyplot as plt

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

ax.set_xlabel("X label",
              fontsize=20,
              color='darkblue',
              alpha=0.7)
ax.set_ylabel("Y label",
              fontsize=20,
              color='darkblue',
              alpha=0.7)
```

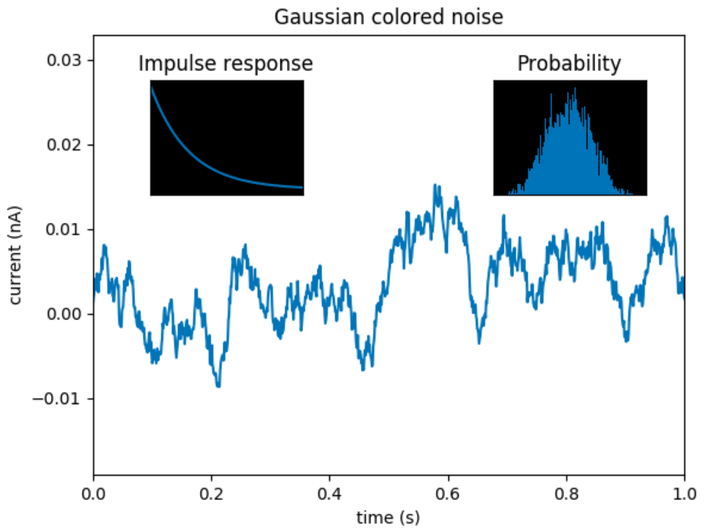


3. ax.set_xlabel and ax.set_ylabel(with Title)

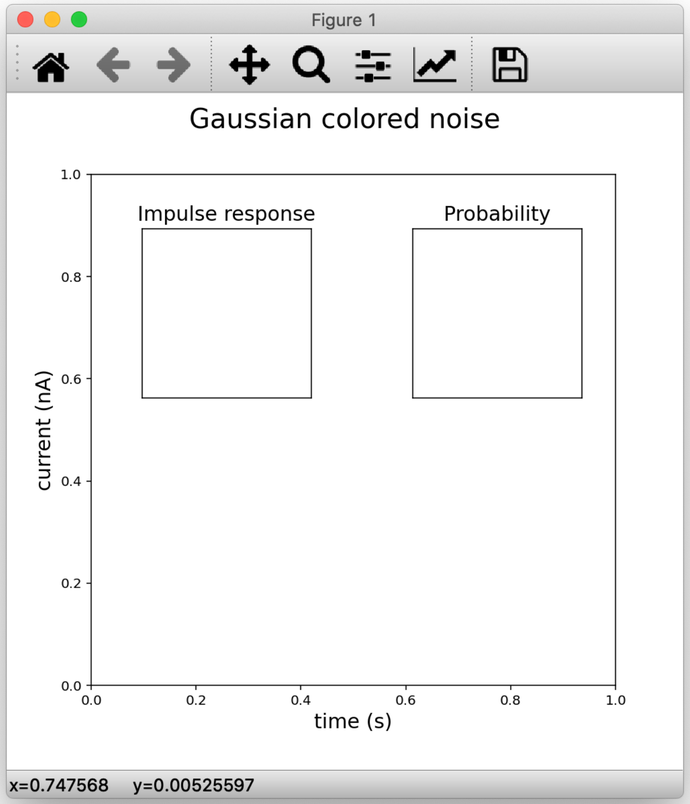
```
import matplotlib.pyplot as plt

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

fig.suptitle("Title of a Figure",
             fontsize=30,
             color='darkblue',
             alpha=0.9)

ax.set_xlabel("X label",
              fontsize=20,
              color='darkblue',
              alpha=0.7)

ax.set_ylabel("Y label",
              fontsize=20,
              color='darkblue',
              alpha=0.7)
```



3. ax.set_xlabel and ax.set_ylabel(Exercise)




```
import matplotlib.pyplot as plt

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

ax_impulse = fig.add_axes([0.2, 0.55, 0.25, 0.25])
ax_prob = fig.add_axes([0.6, 0.55, 0.25, 0.25])

ax_impulse.get_xaxis().set_visible(False)
ax_impulse.get_yaxis().set_visible(False)
ax_prob.get_xaxis().set_visible(False)
ax_prob.get_yaxis().set_visible(False)

fig.suptitle("Gaussian colored noise",
             fontsize=20)
ax.set_xlabel("time (s)",
              fontsize=15)
ax.set_ylabel("current (nA)",
              fontsize=15)

ax_impulse.set_title("Impulse response",
                     fontsize=15)
ax_prob.set_title("Probability",
                  fontsize=15)
```



4. Text Alignment

matplotlib.pyplot.suptitle

`matplotlib.pyplot.suptitle(t, **kwargs)` [source]

Add a centered title to the figure.

Parameters:

t : str

The title text.

x : float, default 0.5

The x location of the text in figure coordinates.

y : float, default 0.98

The y location of the text in figure coordinates.

horizontalalignment, ha : {'center', 'left', right'}, default: 'center'

The horizontal alignment of the text relative to (*x*, *y*).

verticalalignment, va : {'top', 'center', 'bottom', 'baseline'}, default: 'top'

The vertical alignment of the text relative to (*x*, *y*).

fontsize, size : default: `rcParams["figure.titlesize"]` (default: `'large'`)

The font size of the text. See `Text.set_size` for possible values.

fontweight, weight : default: `rcParams["figure.titleweight"]` (default: `'normal'`)

The font weight of the text. See `Text.set_weight` for possible values.

Other Parameters:

fontproperties : None or dict, optional

A dict of font properties. If *fontproperties* is given the default values for font size and weight are taken from the `FontProperties` defaults. `rcParams["figure.titlesize"]` (default: `'large'`) and `rcParams["figure.titleweight"]` (default: `'normal'`) are ignored in this case.

****kwargs**

Additional kwargs are `matplotlib.text.Text` properties.

4. Text Alignment(ax.text)

matplotlib.axes.Axes.text

`Axes.text(self, x, y, s, fontdict=None, **kwargs)` [source]

Add text to the axes.

Add the text *s* to the axes at location *x*, *y* in data coordinates.

Parameters:

x, y : float

The position to place the text. By default, this is in data coordinates. The coordinate system can be changed using the *transform* parameter.

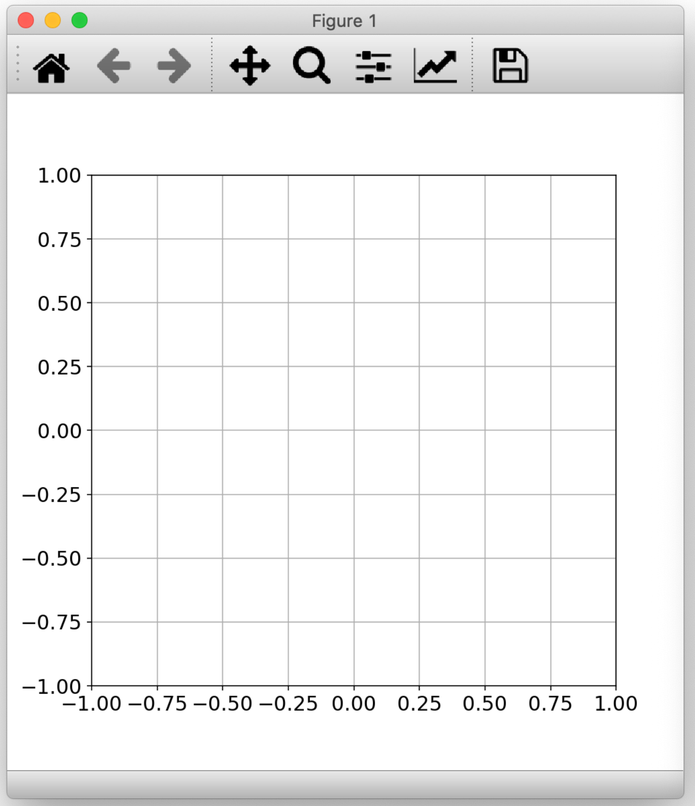
s : str

The text.

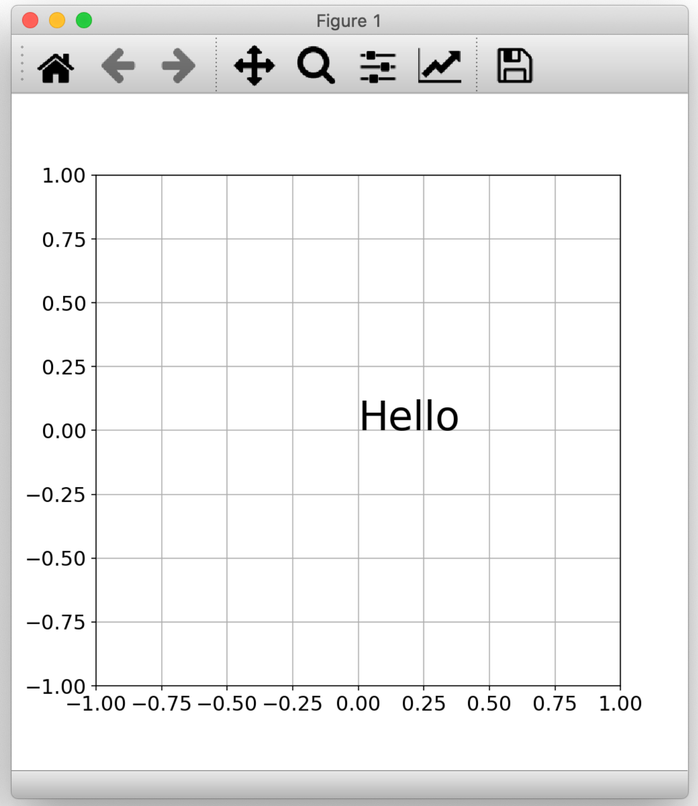
`horizontalalignment` or ha	{'center', 'right', 'left'}
`verticalalignment` or va	{'center', 'top', 'bottom', 'baseline', 'center_baseline'}

4. Text Alignment(ax.text)

```
import matplotlib.pyplot as plt

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

ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])

ax.grid()
ax.tick_params(axis='both',
               labelsize=15)


ax.text(x=0, y=0,
        s="Hello",
        fontsize=30)
```



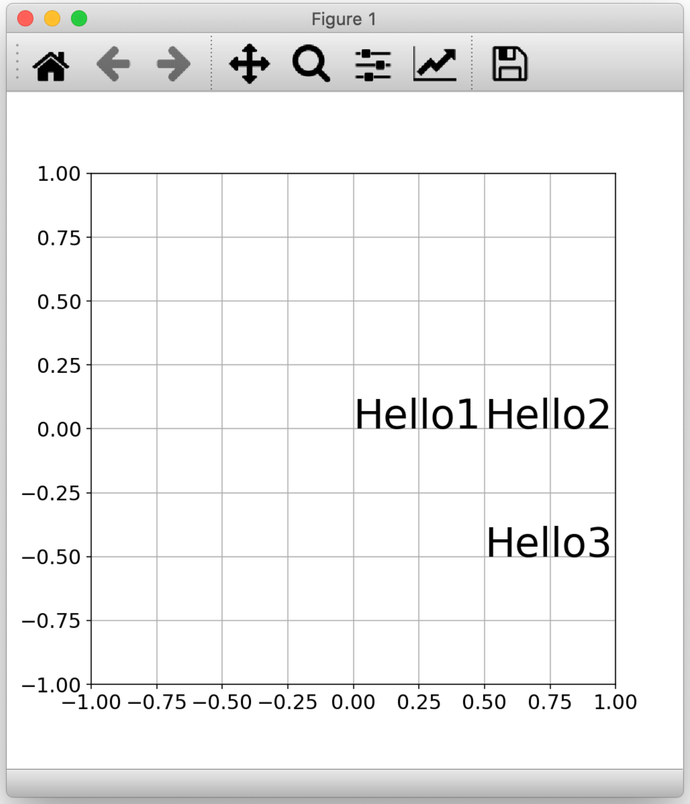
before ax.text()



after ax.text()

4. Text Alignment(x, y location)

```
import matplotlib.pyplot as plt

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

ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])

ax.grid()
ax.tick_params(axis='both',
               labelsize=15)

ax.text(x=0, y=0,
        s="Hello1",
        fontsize=30)

ax.text(x=0.5, y=0,
        s="Hello2",
        fontsize=30)

ax.text(x=0.5, y=-0.5,
        s="Hello3",
        fontsize=30)
```



4. Text Alignment(Alignments Arguments)

Parameters:

t : str

The title text.

x : float, default 0.5

The x location of the text in figure coordinates.

y : float, default 0.98

The y location of the text in figure coordinates.

horizontalalignment, ha : {'center', 'left', right'}, default: 'center'

The horizontal alignment of the text relative to (*x*, *y*).

verticalalignment, va : {'top', 'center', 'bottom', 'baseline'}, default: 'top'

The vertical alignment of the text relative to (*x*, *y*).

fontsize, size : default: `rcParams["figure.titlesize"]` (default: `'large'`)

The font size of the text. See `Text.set_size` for possible values.

fontweight, weight : default: `rcParams["figure.titleweight"]` (default: `'normal'`)

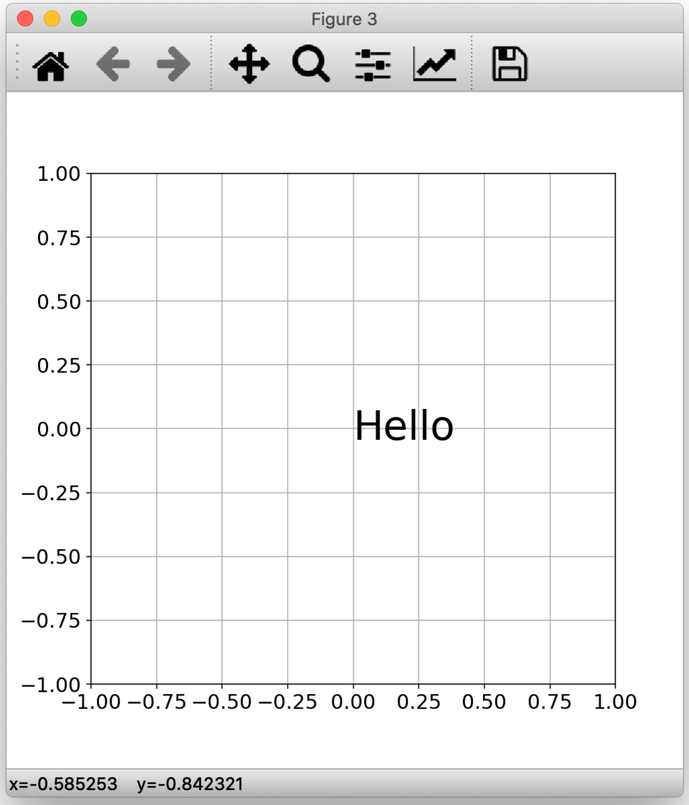
The font weight of the text. See `Text.set_weight` for possible values.

horizontal alignment
- center
- left
- right

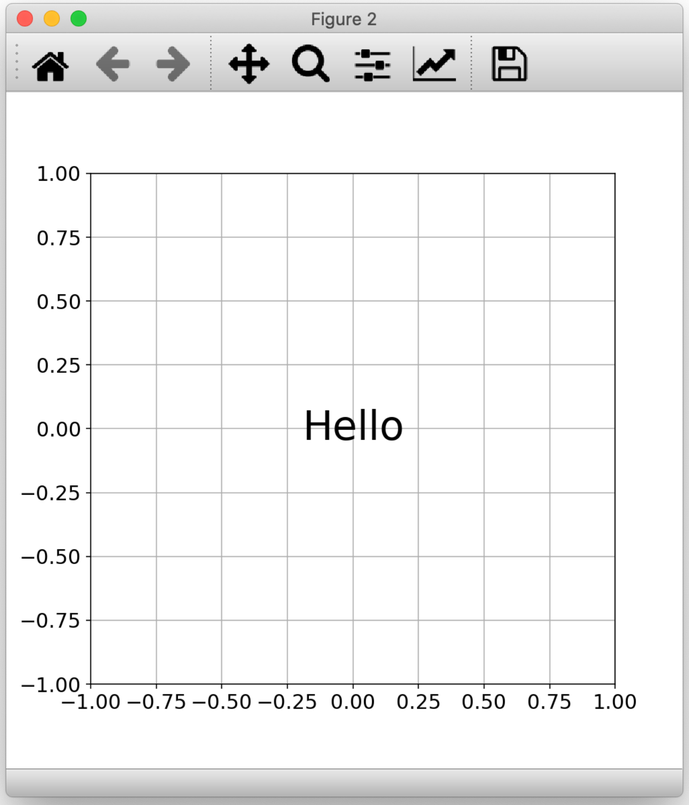
vertical alignment
- center
- top
- bottom

4. Text Alignment(ha Argument)

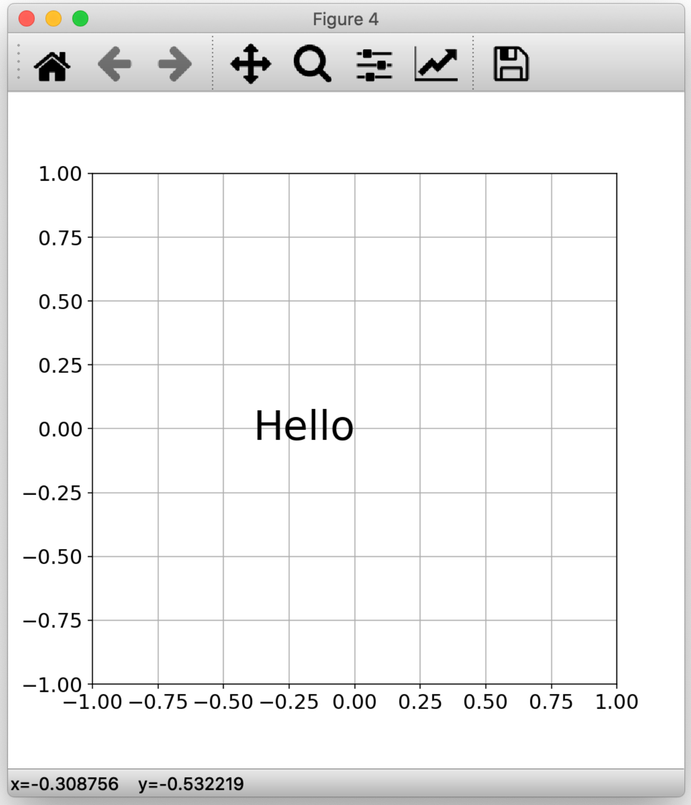
```
ax.text(x=0, y=0,
        va='center', ha='left',
        s="Hello",
        fontsize=30)
```




```
ax.text(x=0, y=0,
        va='center', ha='center',
        s="Hello",
        fontsize=30)
```

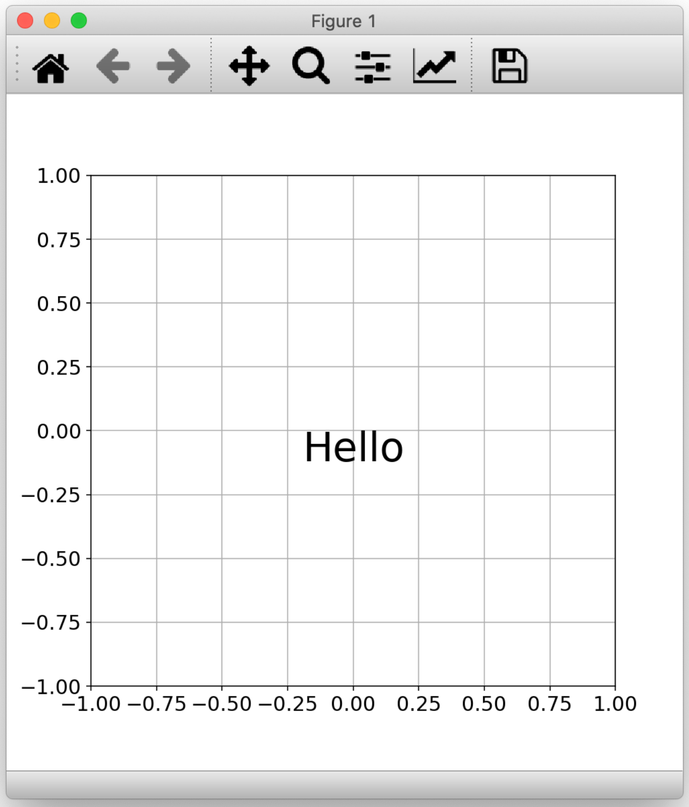



```
ax.text(x=0, y=0,
        va='center', ha='right',
        s="Hello",
        fontsize=30)
```

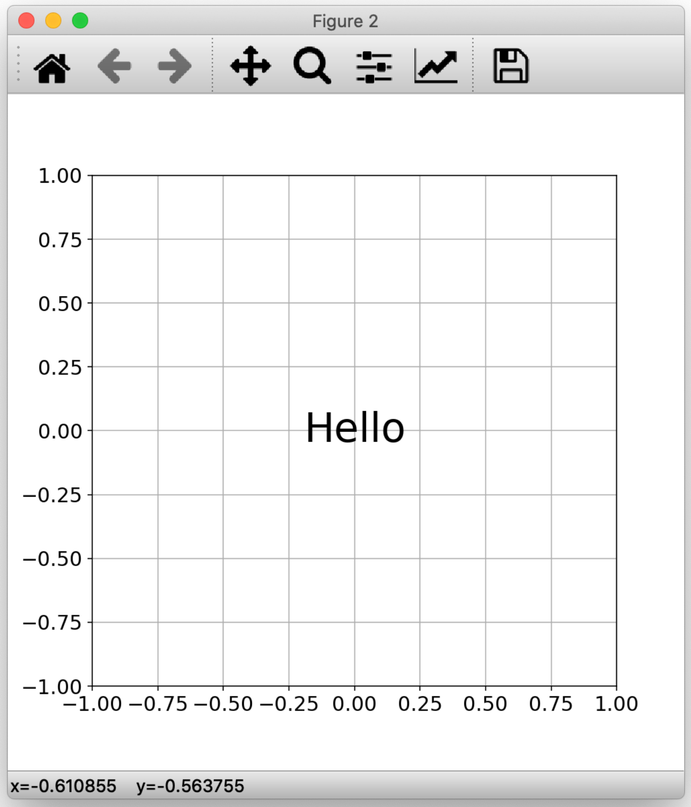


4. Text Alignment(va Alignment)

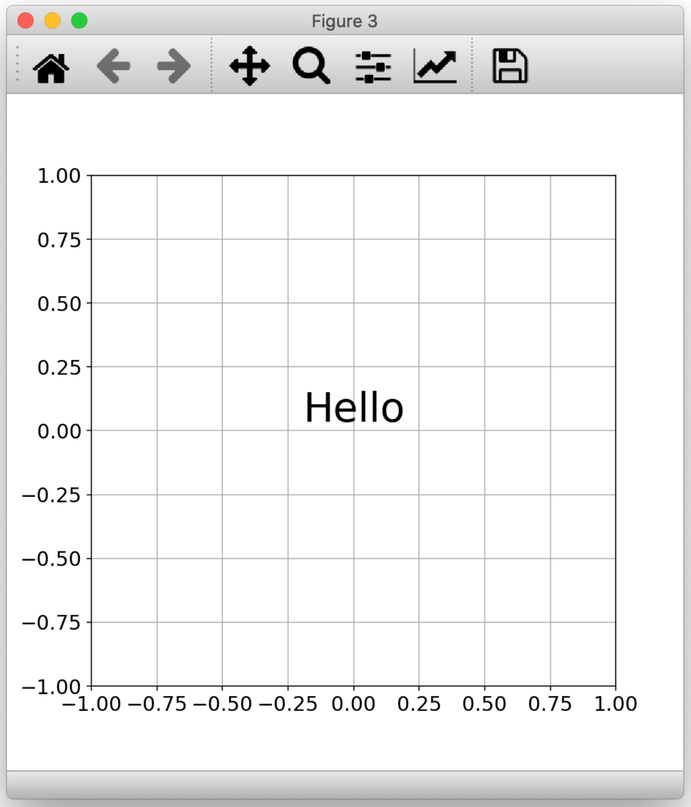
```
ax.text(x=0, y=0,
        va='top', ha='center',
        s="Hello",
        fontsize=30)
```




```
ax.text(x=0, y=0,
        va='center', ha='center',
        s="Hello",
        fontsize=30)
```

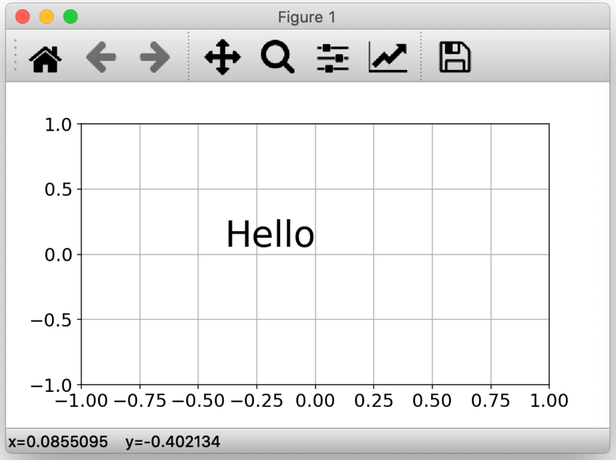


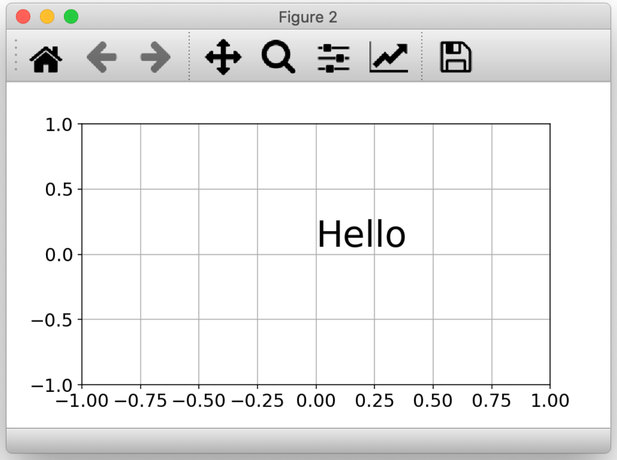

```
ax.text(x=0, y=0,
        va='bottom', ha='center',
        s="Hello",
        fontsize=30)
```



4. Text Alignment(ha, va Alignment)

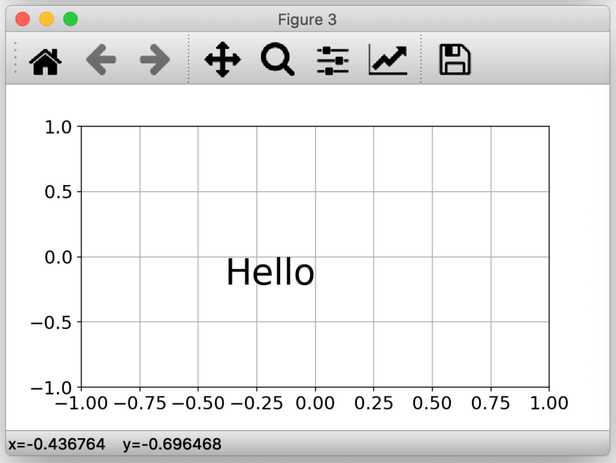
```
ax.text(x=0, y=0,
        va='bottom', ha='right',
        s="Hello",
        fontsize=30)
```

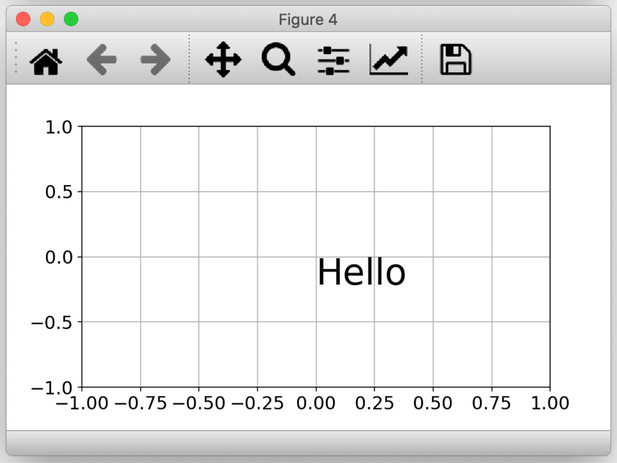





```
ax.text(x=0, y=0,
        va='bottom', ha='left',
        s="Hello",
        fontsize=30)
```

```
ax.text(x=0, y=0,
        va='top', ha='right',
        s="Hello",
        fontsize=30)
```






```
ax.text(x=0, y=0,
        va='top', ha='left',
        s="Hello",
        fontsize=30)
```

4. Text Alignment(ha, va Alignment)

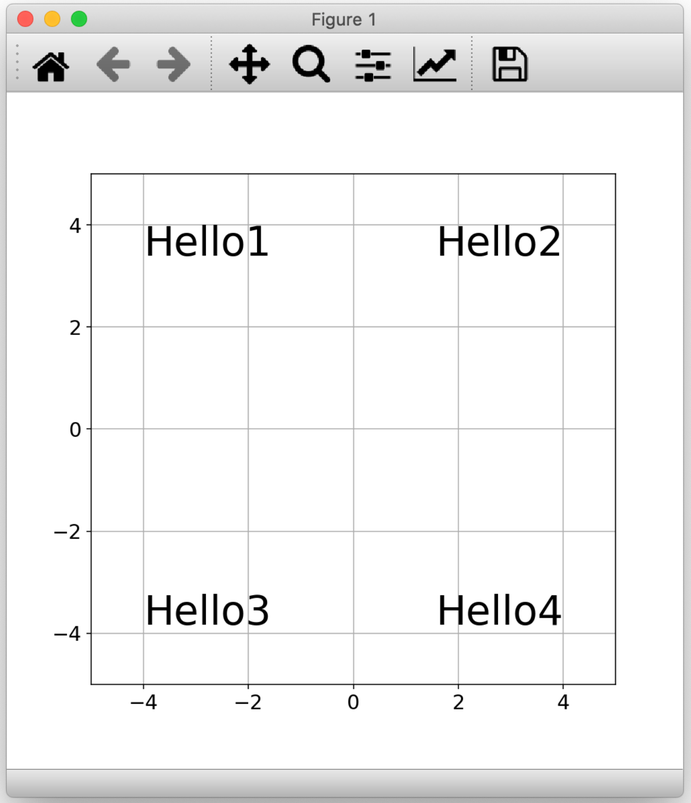
```
import matplotlib.pyplot as plt

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

ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])

ax.grid()
ax.tick_params(axis='both',
               labelsize=15)
```

```
ax.text(x=-4, y=4,
        va='top', ha='left',
        s="Hello1",
        fontsize=30)
```

```
ax.text(x=-4, y=-4,
        va='bottom', ha='left',
        s="Hello3",
        fontsize=30)
```

```
ax.text(x=4, y=4,
        va='top', ha='right',
        s="Hello2",
        fontsize=30)
```

```
ax.text(x=4, y=-4,
        va='bottom', ha='right',
        s="Hello4",
        fontsize=30)
```



5. Title Alignment

matplotlib.pyplot.suptitle

`matplotlib.pyplot.suptitle(t, **kwargs)` [source]

Add a centered title to the figure.

Parameters:

t : str

The title text.

x : float, default 0.5

The x location of the text in figure coordinates.

y : float, default 0.98

The y location of the text in figure coordinates.

horizontalalignment, ha : {'center', 'left', right'}, default: 'center'

The horizontal alignment of the text relative to (*x*, *y*).

verticalalignment, va : {'top', 'center', 'bottom', 'baseline'}, default: 'top'

The vertical alignment of the text relative to (*x*, *y*).

fontsize, size : default: `rcParams["figure.titlesize"]` (default: `'large'`)

The font size of the text. See `Text.set_size` for possible values.

fontweight, weight : default: `rcParams["figure.titleweight"]` (default: `'normal'`)

The font weight of the text. See `Text.set_weight` for possible values.

Other Parameters:

fontproperties : None or dict, optional

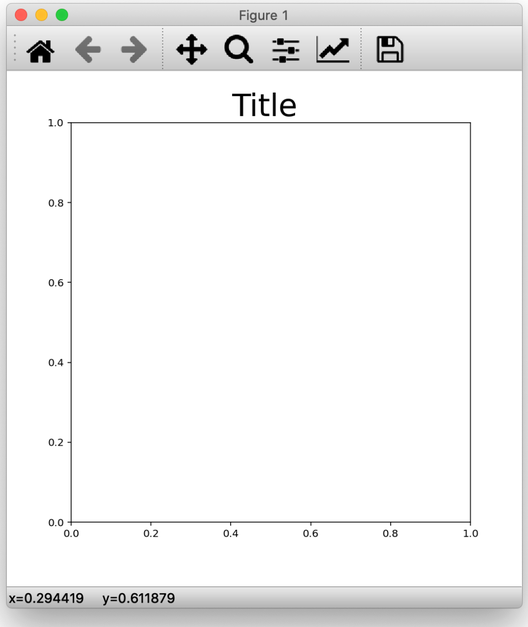
A dict of font properties. If *fontproperties* is given the default values for font size and weight are taken from the `FontProperties` defaults. `rcParams["figure.titlesize"]` (default: `'large'`) and `rcParams["figure.titleweight"]` (default: `'normal'`) are ignored in this case.

****kwargs**

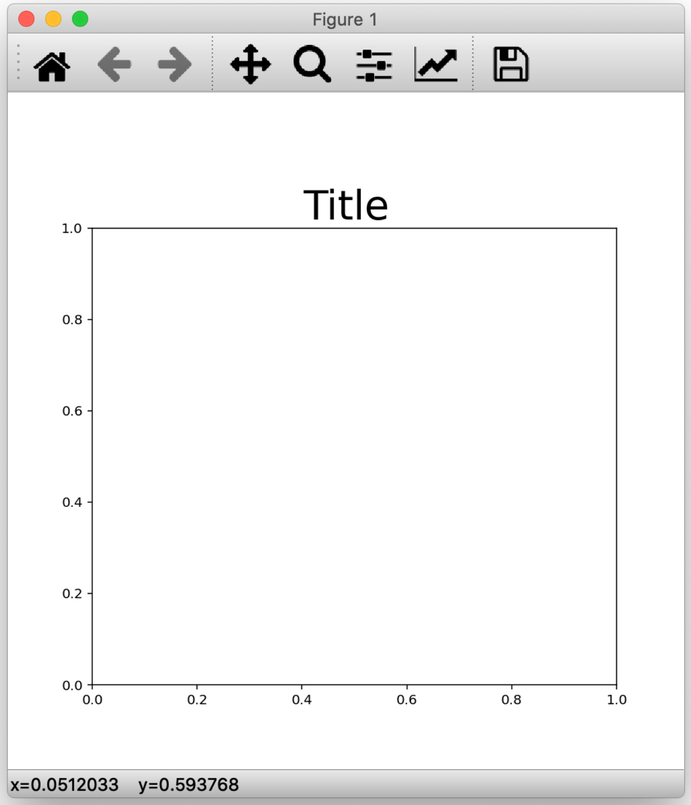
Additional kwargs are `matplotlib.text.Text` properties.

5. Title Alignment(title_bottom Variable)

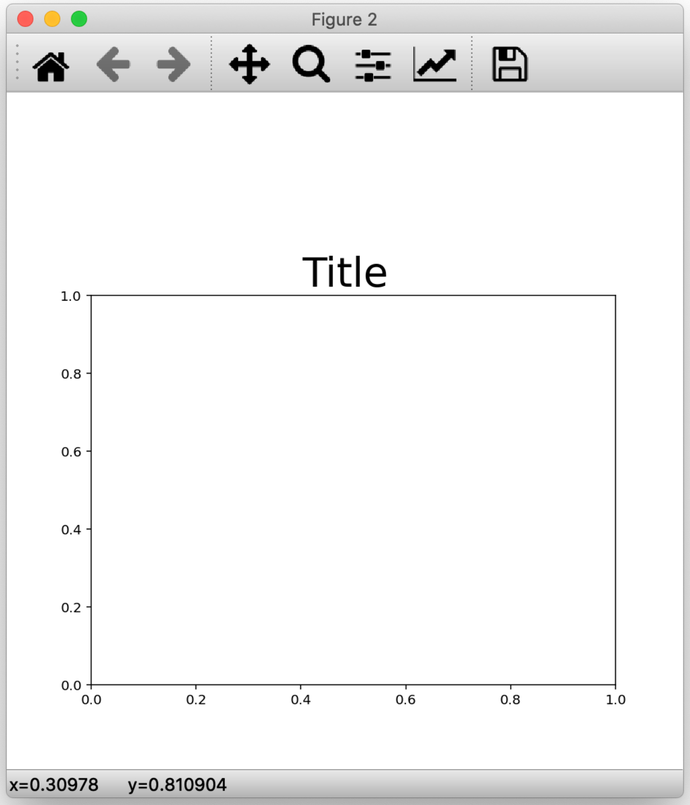
```
import matplotlib.pyplot as plt

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

title_bottom = 0.9

fig.suptitle("Title",
             fontsize=30,
             y=title_bottom,
             va='bottom')

fig.subplots_adjust(top=title_bottom)
```



title_bottom = 0.8



title_bottom = 0.7



5. Title Alignment(with ax.set_title)

```
import matplotlib.pyplot as plt

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

title_bottom = 0.9
fig.suptitle("Title",
             fontsize=30,
             y=title_bottom,
             va='bottom')

ax.set_title("Ax Title",
             fontsize=20)

fig.subplots_adjust(top=title_bottom)
fig.subplots_adjust(top=title_bottom-0.1)
```

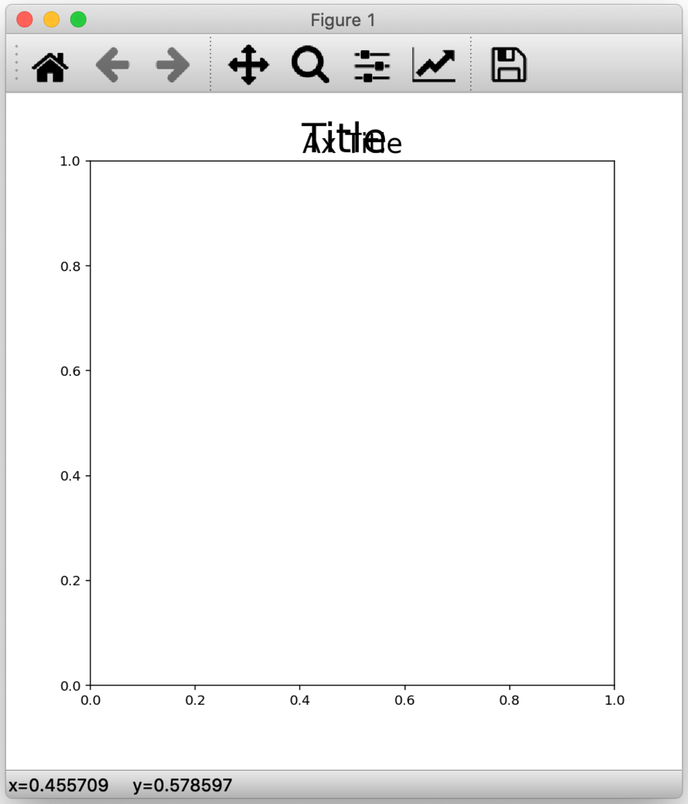


fig.subplots_adjust(top=title_bottom)

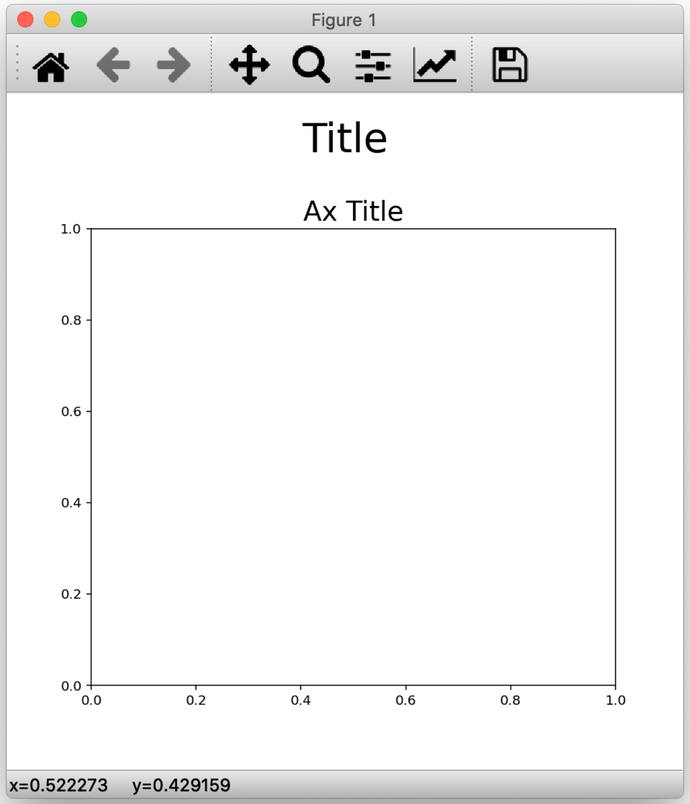
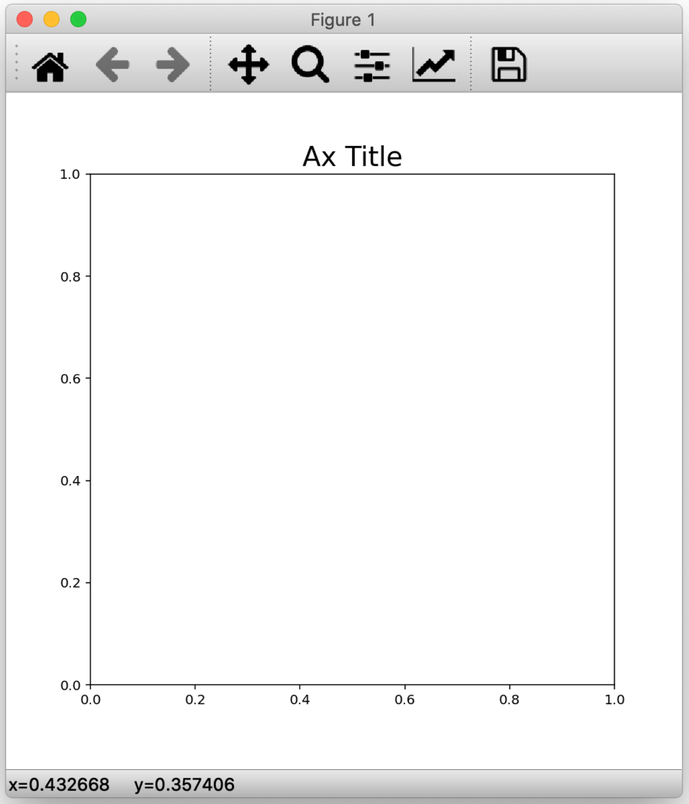


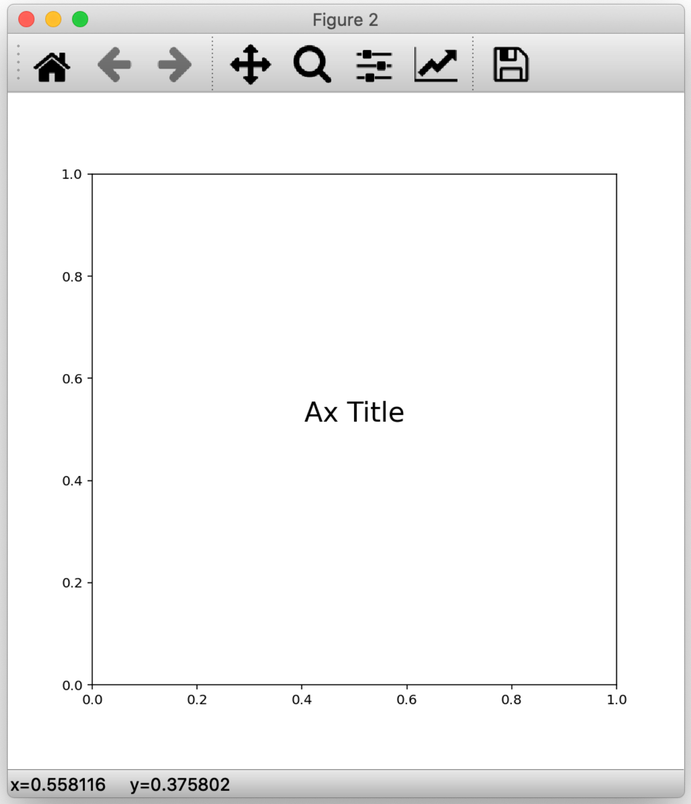
fig.subplots_adjust(top=title_bottom-0.1)

5. Title Alignment(y Argument)

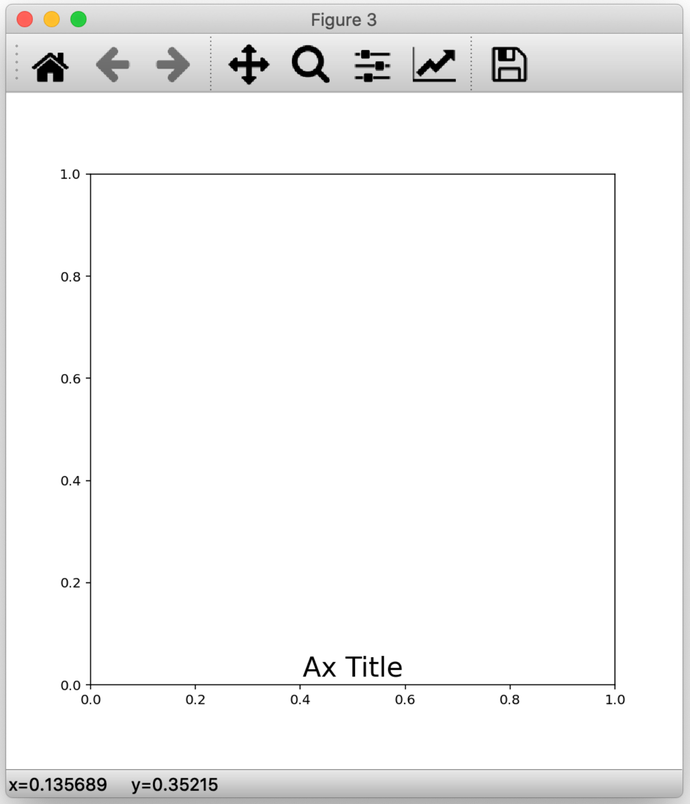
```
ax.set_title("Ax Title",
             fontsize=20,
             y=1)
```




```
ax.set_title("Ax Title",
             fontsize=20,
             y=0.5)
```

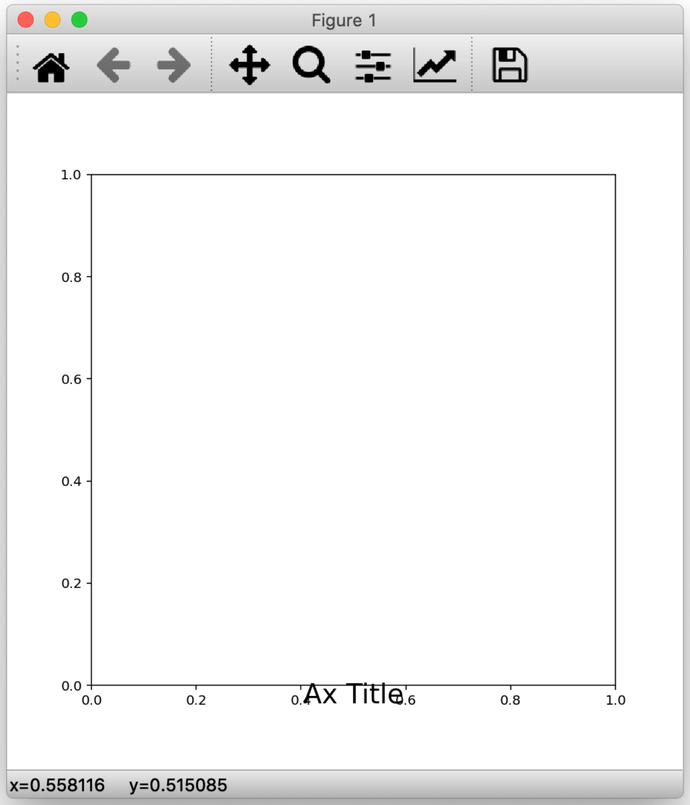



```
ax.set_title("Ax Title",
             fontsize=20,
             y=0)
```

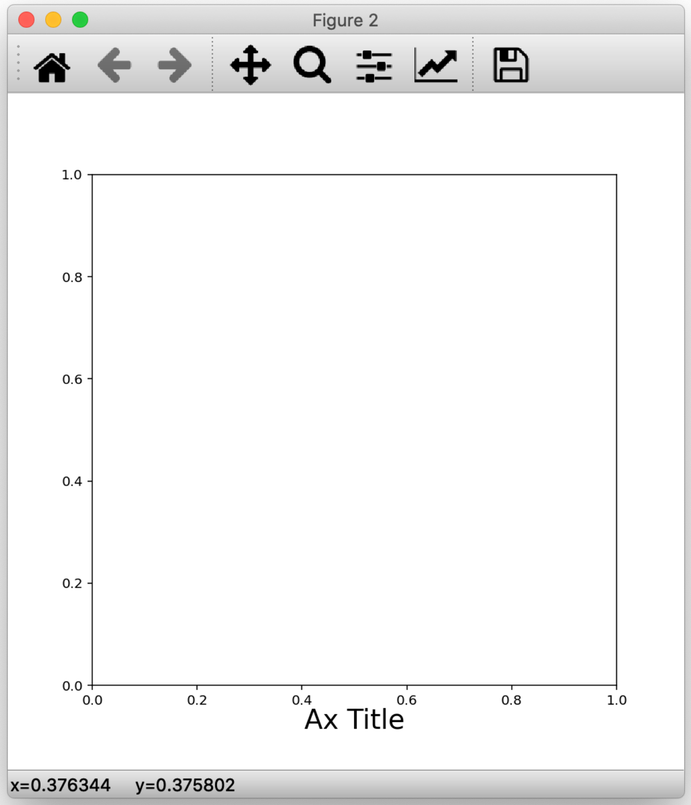


5. Title Alignment(y Argument)

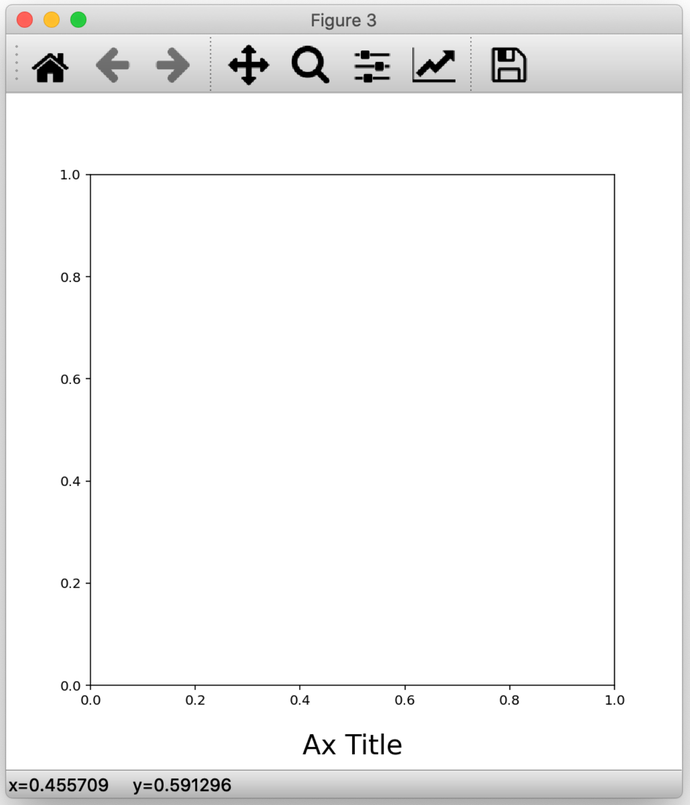
```
ax.set_title("Ax Title",
             fontsize=20,
             y=-0.05)
```




```
ax.set_title("Ax Title",
             fontsize=20,
             y=-0.1)
```

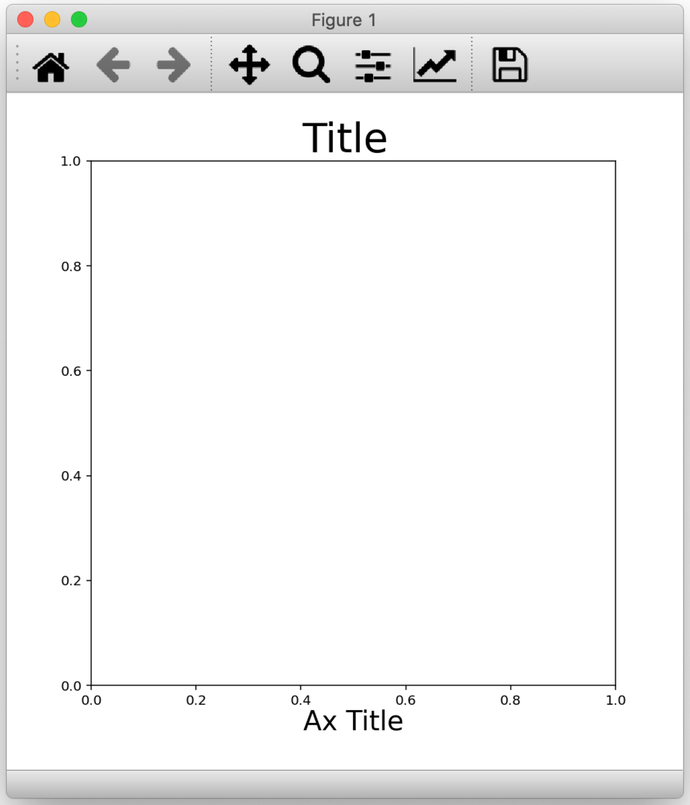



```
ax.set_title("Ax Title",
             fontsize=20,
             y=-0.15)
```

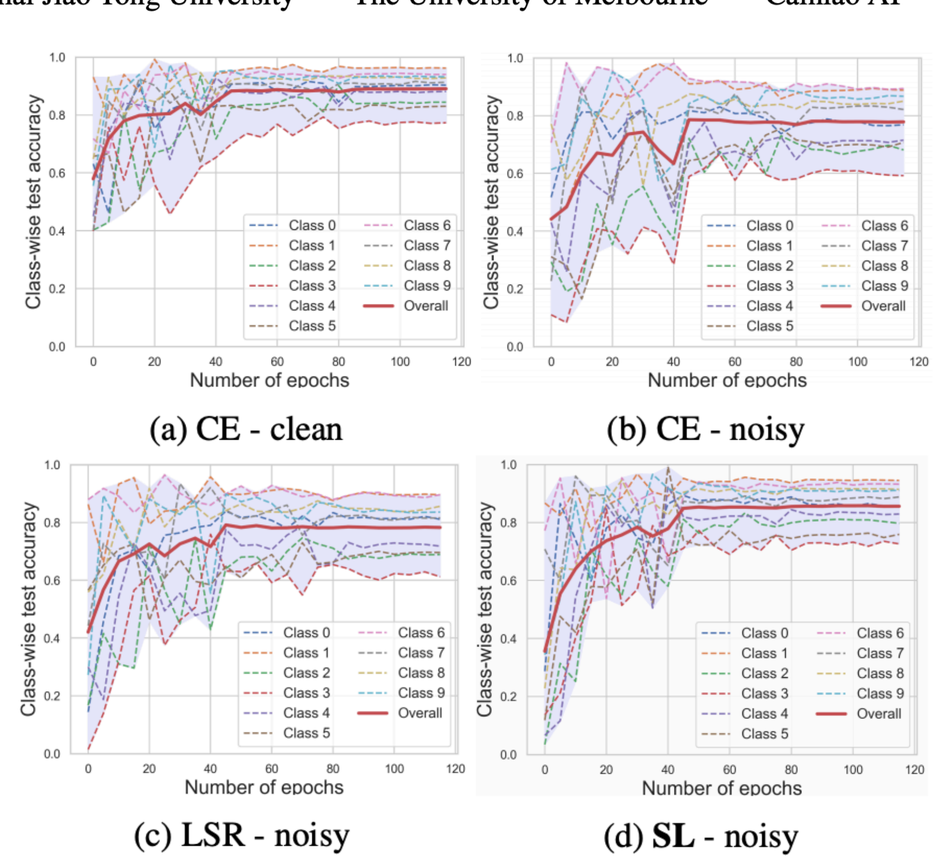


5. Title Alignment(Title Under Ax)

```
import matplotlib.pyplot as plt

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

title_bottom = 0.9

fig.suptitle("Title",
             fontsize=30,
             y=title_bottom,
             va='bottom')


ax.set_title("Ax Title",
              fontsize=20,
              y=-0.1)

fig.subplots_adjust(top=title_bottom)
```



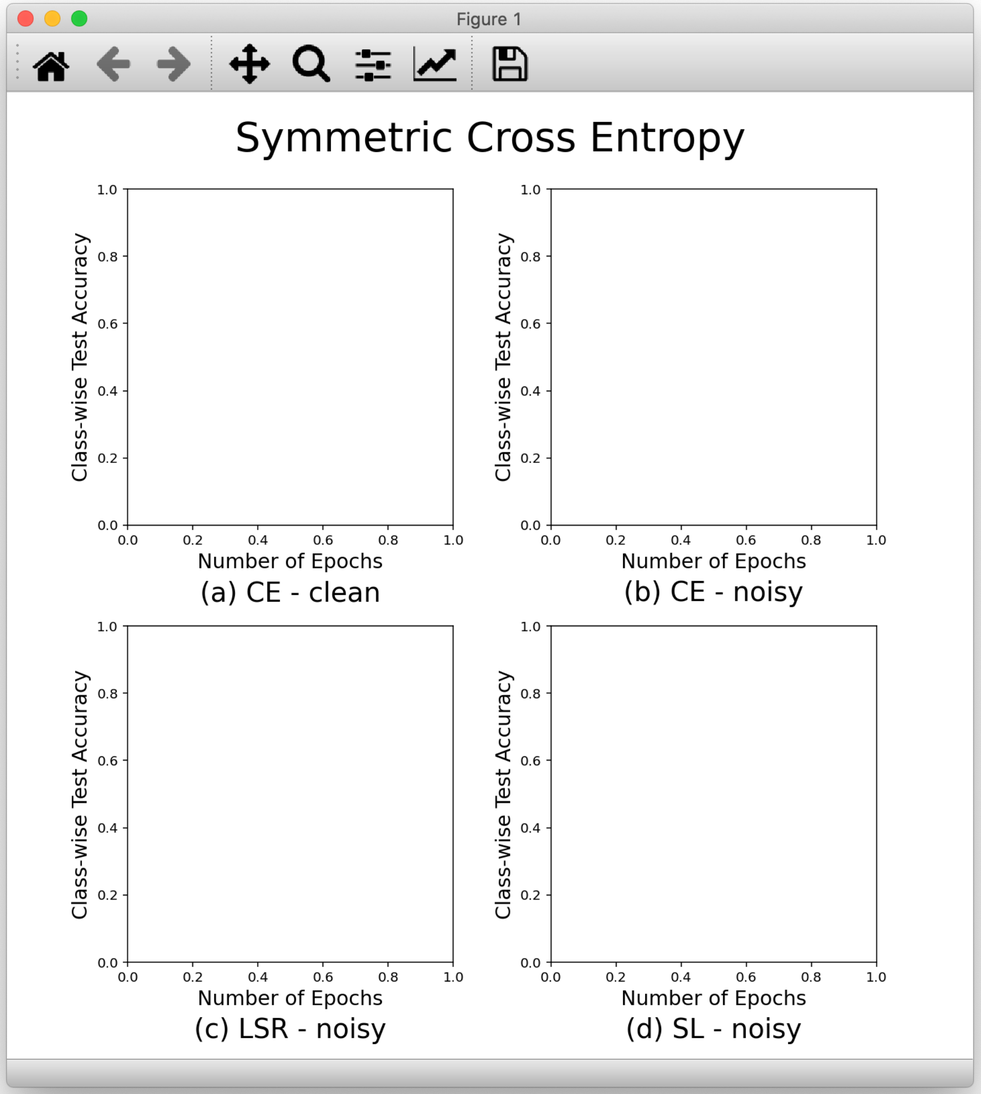
5. Title Alignment(Exercise)




```
ax_title_list = ['(a) CE - clean', '(b) CE - noisy',
                 '(c) LSR - noisy', '(d) SL - noisy']
title_bottom = 0.9

figsize = (10, 10)
fig, axes = plt.subplots(2, 2,
                         figsize=figsize)

fig.suptitle("Symmetric Cross Entropy",
             fontsize=30,
             y=title_bottom+0.03,
             va='bottom')

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(ax_title_list[ax_idx],
                 fontsize=20,
                 y=-0.25)
    ax.set_xlabel("Number of Epochs",
                  fontsize=15)
    ax.set_ylabel("Class-wise Test Accuracy",
                  fontsize=15)
fig.subplots_adjust(top=title_bottom, bottom=0.1,
                    hspace=0.3, wspace=0.3)
```

5. Title Alignment(Exercise)

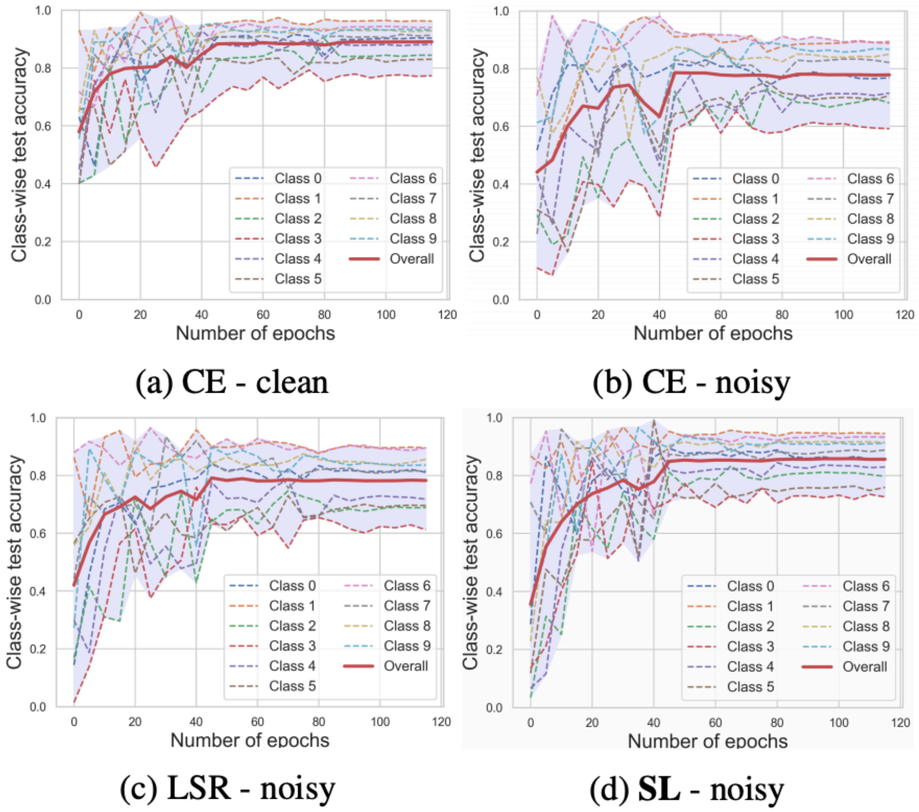
```
ax_title_list = ['(a) CE - clean', '(b) CE - noisy',
                 '(c) LSR - noisy', '(d) SL - noisy']
title_bottom = 0.9

figsize = (10, 10)
fig, axes = plt.subplots(2, 2,
                         figsize=figsize)

fig.suptitle("Symmetric Cross Entropy",
             fontsize=30,
             y=title_bottom+0.03,
             va='bottom')

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(ax_title_list[ax_idx],
                 fontsize=20,
                 y=-0.25)
    ax.set_xlabel("Number of Epochs",
                  fontsize=15)
    ax.set_ylabel("Class-wise Test Accuracy",
                  fontsize=15)
fig.subplots_adjust(top=title_bottom, bottom=0.1,
                    hspace=0.3, wspace=0.3)
```



5. Title Alignment(Exercise)

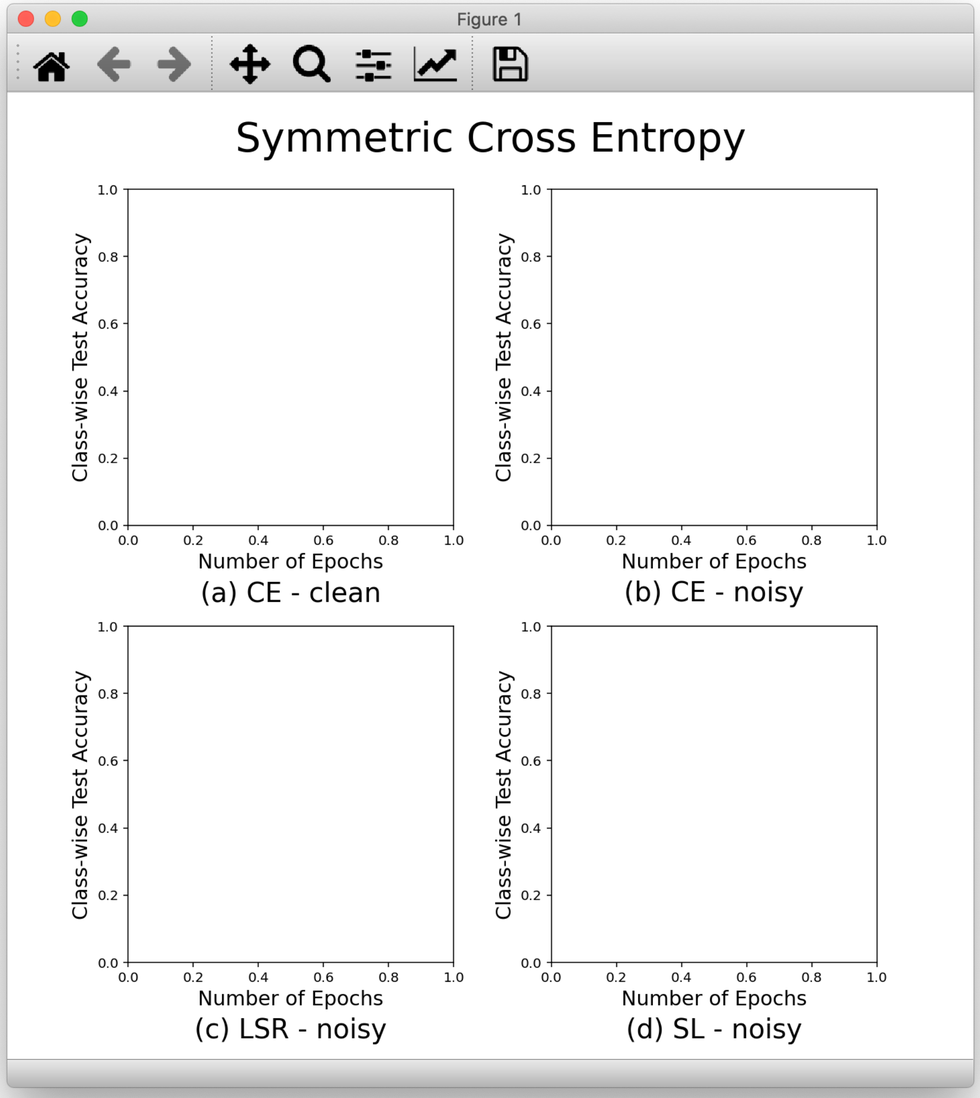
Symmetric Cross Entropy for Robust Learning with Noisy Labels

Yisen Wang[1*†] Xingjun Ma[2*†] Zaiyi Chen[3] Yuan Luo[1] Jinfeng Yi[4] James Bailey[2]
[1]Shanghai Jiao Tong University [2]The University of Melbourne [3]Cainiao AI [4]JD AI



(a) CE - clean (b) CE - noisy

(c) LSR - noisy (d) **SL** - noisy



6. Text Properties(Document)

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color
contains	unknown
figure	Figure
fontfamily or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
fontproperties or font or font_properties	font_manager.FontProperties or str or pathlib.Path
fontsize or size	float or {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}
fontstretch or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
fontstyle or style	{'normal', 'italic', 'oblique'}
fontvariant or variant	{'normal', 'small-caps'}
fontweight or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}
gid	str
horizontalalignment or ha	{'center', 'right', 'left'}
in_layout	bool
label	object
linespacing	float (multiple of font size)
multialignment or ma	{'left', 'right', 'center'}
path_effects	AbstractPathEffect
picker	None or bool or callable
position	(float, float)
rasterized	bool or None
rotation	float or {'vertical', 'horizontal'}
rotation_mode	{None, 'default', 'anchor'}
sketch_params	(scale: float, length: float, randomness: float)
snap	bool or None
text	object
transform	Transform
url	str
usetex	bool or None
verticalalignment or va	{'center', 'top', 'bottom', 'baseline', 'center_baseline'}
visible	bool
wrap	bool
x	float
y	float
zorder	float

6. Text Properties(fontsize)

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color
contains	unknown
figure	Figure
fontfamily or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
fontproperties or font or font_properties	font_manager.FontProperties or str or pathlib.Path
fontstretch or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
fontstyle or style	{'normal', 'italic', 'oblique'}
fontvariant or variant	{'normal', 'small-caps'}
fontweight or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}
gid	str
horizontalalignment or ha	{'center', 'right', 'left'}
in_layout	bool
label	object
linespacing	float (multiple of font size)
multialignment or ma	{'left', 'right', 'center'}
path_effects	AbstractPathEffect
picker	None or bool or callable
position	(float, float)
rasterized	bool or None
rotation	float or {'vertical', 'horizontal'}
rotation_mode	{None, 'default', 'anchor'}
sketch_params	(scale: float, length: float, randomness: float)
snap	bool or None
text	object
transform	Transform
url	str
usetex	bool or None
verticalalignment or va	{'center', 'top', 'bottom', 'baseline', 'center_baseline'}
visible	bool
wrap	bool
x	float
y	float
zorder	float

6. Text Properties(fontsize)

`fontfamily` or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
`fontproperties` or font or font_properties	`font_manager.FontProperties` or `str` or `pathlib.Path`
`fontsize` or size	float or {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}
`fontstretch` or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
`fontstyle` or style	{'normal', 'italic', 'oblique'}
`fontvariant` or variant	{'normal', 'small-caps'}
`fontweight` or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}

`set_fontsize(`*`self`*`, `*`fontsize`*`)` [source]

Set the font size.

Parameters:

fontsize : float or {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}

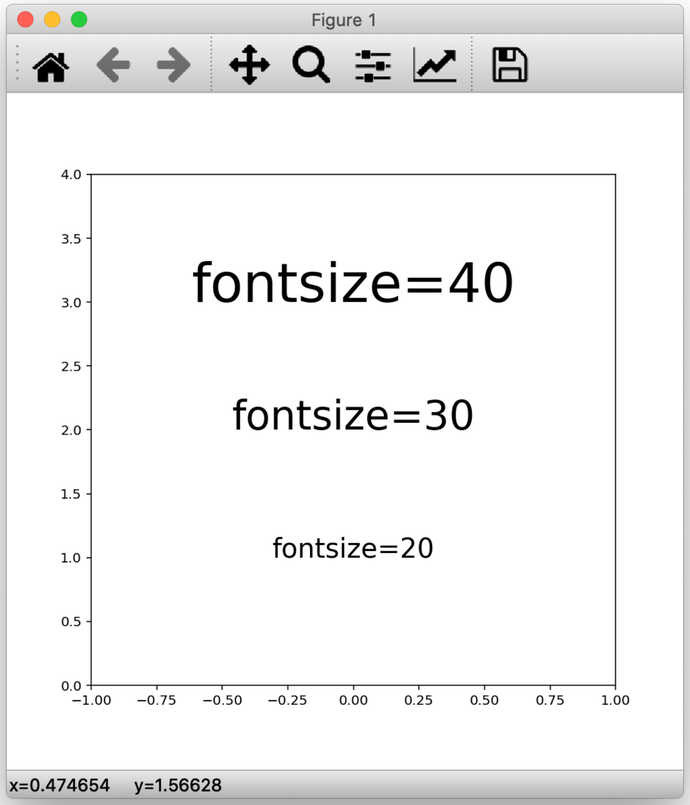
If float, the fontsize in points. The string values denote sizes relative to the default font size.

https://matplotlib.org/api/text_api.html#matplotlib.text.Text

https://matplotlib.org/api/text_api.html#matplotlib.text.Text.set_fontsize

6. Text Properties(fontsize)

```
import matplotlib.pyplot as plt

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

fontsize_list = [20, 30, 40]

ax.set_xlim([-1, 1])
ax.set_ylim([0, len(fontsize_list)+1])

for font_idx, fontsize in enumerate(fontsize_list):
    ax.text(0, font_idx+1, ha='center',
            s="fontsize="+str(fontsize),
            fontsize=fontsize)
```



6. Text Properties(fontfamily)

`fontfamily` or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
`fontproperties` or font or font_properties	`font_manager.FontProperties` or `str` or `pathlib.Path`
`fontsize` or size	float or {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}
`fontstretch` or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
`fontstyle` or style	{'normal', 'italic', 'oblique'}
`fontvariant` or variant	{'normal', 'small-caps'}
`fontweight` or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}

`set_fontfamily(`*`self`*`, `*`fontname`*`)` [source]

Set the font family. May be either a single string, or a list of strings in decreasing priority. Each string may be either a real font name or a generic font class name. If the latter, the specific font names will be looked up in the corresponding rcParams.

If a `Text` instance is constructed with `fontfamily=None`, then the font is set to `rcParams["font.family"]` (default: `['sans-serif']`), and the same is done when `set_fontfamily()` is called on an existing `Text` instance.

Parameters:

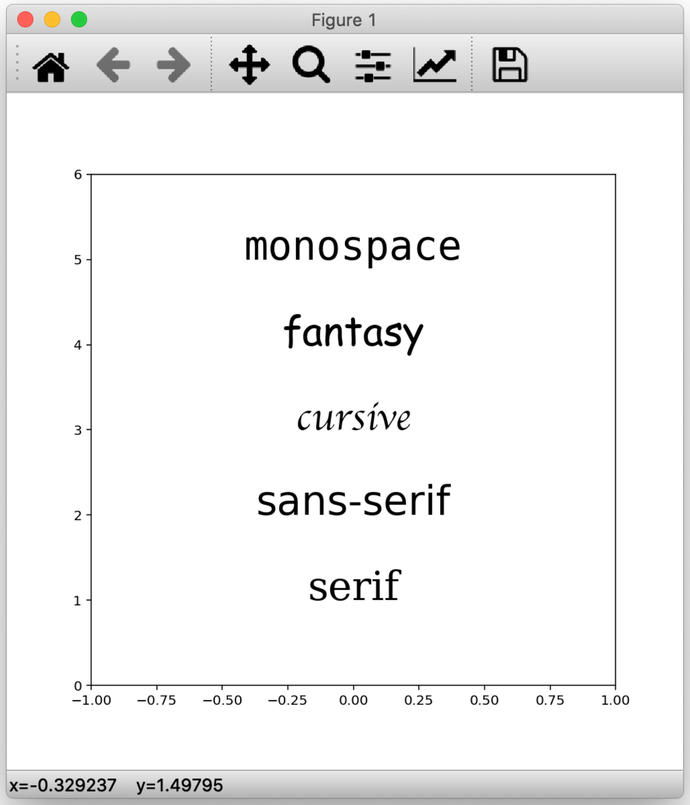
fontname : {FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}

https://matplotlib.org/api/text_api.html#matplotlib.text.Text

https://matplotlib.org/api/text_api.html#matplotlib.text.Text.set_fontfamily

6. Text Properties(fontfamily)

```
fontfamily_list = ['serif', 'sans-serif',
                   'cursive', 'fantasy',
                   'monospace']

ax.set_xlim([-1, 1])
ax.set_ylim([0, len(fontfamily_list)+1])

for font_idx, fontfamily in enumerate(fontfamily_list):
    ax.text(0, font_idx+1, ha="center",
            s=fontfamily,
            fontsize=30,
            fontfamily=fontfamily)
```



6. Text Properties(fontweight)

`fontfamily` or family	{FONTNAME, 'serif', 'sans-serif', 'cursive', 'fantasy', 'monospace'}
`fontproperties` or font or font_properties	`font_manager.FontProperties` or `str` or `pathlib.Path`
`fontsize` or size	float or {'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}
`fontstretch` or stretch	{a numeric value in range 0-1000, 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'normal', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'}
`fontstyle` or style	{'normal', 'italic', 'oblique'}
`fontvariant` or variant	{'normal', 'small-caps'}
`fontweight` or weight	{a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}

`set_fontweight(self, weight)` [source]

Set the font weight.

Parameters:

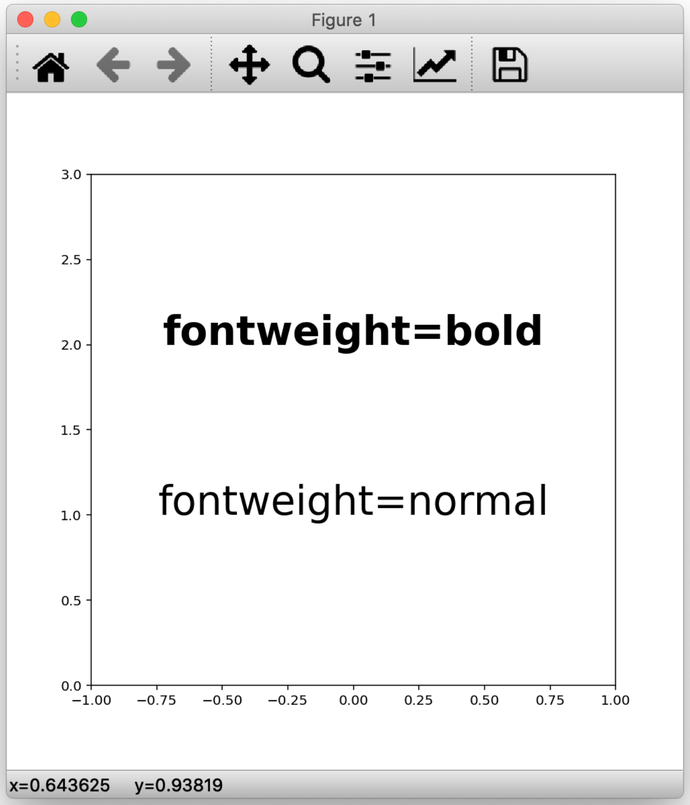
weight : {a numeric value in range 0-1000, 'ultralight', 'light', 'normal', 'regular', 'book', 'medium', 'roman', 'semibold', 'demibold', 'demi', 'bold', 'heavy', 'extra bold', 'black'}

https://matplotlib.org/api/text_api.html#matplotlib.text.Text

https://matplotlib.org/api/text_api.html#matplotlib.text.Text.set_fontweight

6. Text Properties(fontweight)

```
fontweight_list = ['normal', 'bold']

ax.set_xlim([-1, 1])
ax.set_ylim([0, len(fontweight_list)+1])

for font_idx, fontweight in enumerate(fontweight_list):
    ax.text(0, font_idx+1, ha="center",
            s='fontweight='+fontweight,
            fontsize=30,
            fontweight=fontweight)
```



6. Text Properties(alpha)

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color

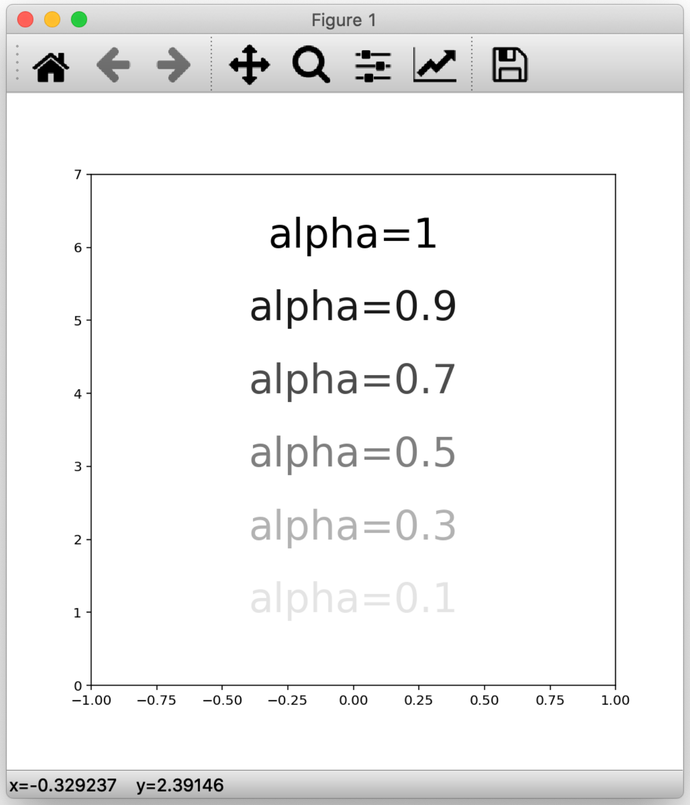
Artist.set_alpha(*self*, *alpha*)

Set the alpha value used for blending - not supported on all backends.

Parameters: **alpha** : float or None

6. Text Properties(alpha)

```
alpha_list = [0.1, 0.3, 0.5, 0.7, 0.9, 1]

ax.set_xlim([-1, 1])
ax.set_ylim([0, len(alpha_list)+1])

for a_idx, alpha in enumerate(alpha_list):
    ax.text(0, a_idx+1, ha="center",
            s='alpha='+str(alpha),
            fontsize=30,
            alpha=alpha)
```

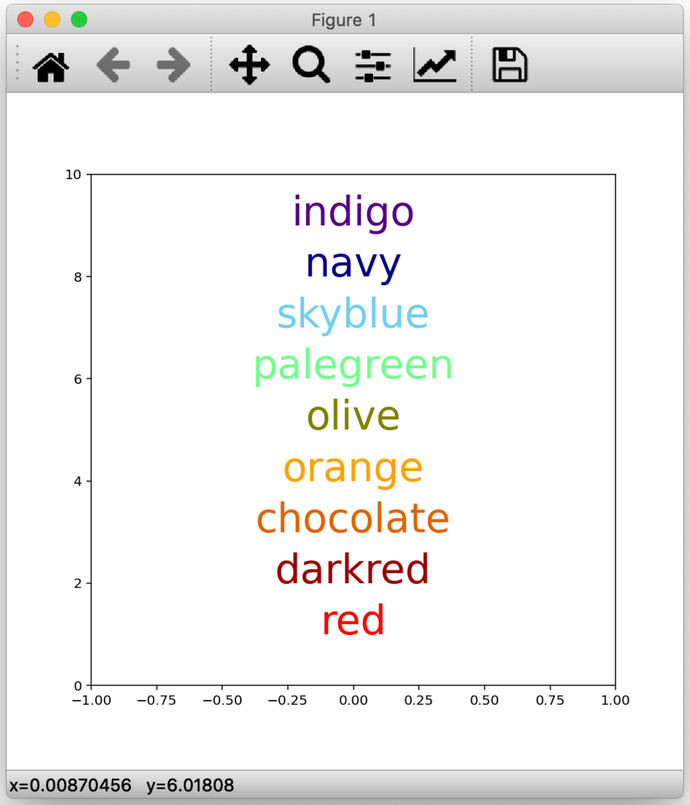


6. Text Properties(color)

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color

6. Text Properties(color)

```
color_list = ['red', 'darkred', 'chocolate',
              'orange', 'olive', 'palegreen',
              'skyblue', 'navy', 'indigo']

ax.set_xlim([-1, 1])
ax.set_ylim([0, len(color_list)+1])

for c_idx, color in enumerate(color_list):
    ax.text(0, c_idx+1, ha="center",
            s=color,
            fontsize=30,
            color=color)
```



6. Text Properties(bbox)

Property	Description
agg_filter	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
alpha	float or None
animated	bool
backgroundcolor	color
bbox	dict with properties for patches.FancyBboxPatch
clip_box	Bbox
clip_on	bool
clip_path	Patch or (Path, Transform) or None
color or c	color

set_bbox(*self*, *rectprops*)

Draw a bounding box around self.

Parameters:

rectprops : dict with properties for patches.FancyBboxPatch

The default boxstyle is 'square'. The mutation scale of the patches.FancyBboxPatch is set to the fontsize.

6. Text Properties(bbox)

boxstyle : str or `matplotlib.patches.BoxStyle`

The style of the fancy box. This can either be a `BoxStyle` instance or a string of the style name and optionally comma seprarated attributes (e.g. "Round, pad=0.2"). This string is passed to `BoxStyle` to construct a `BoxStyle` object. See there for a full documentation.

The following box styles are available:

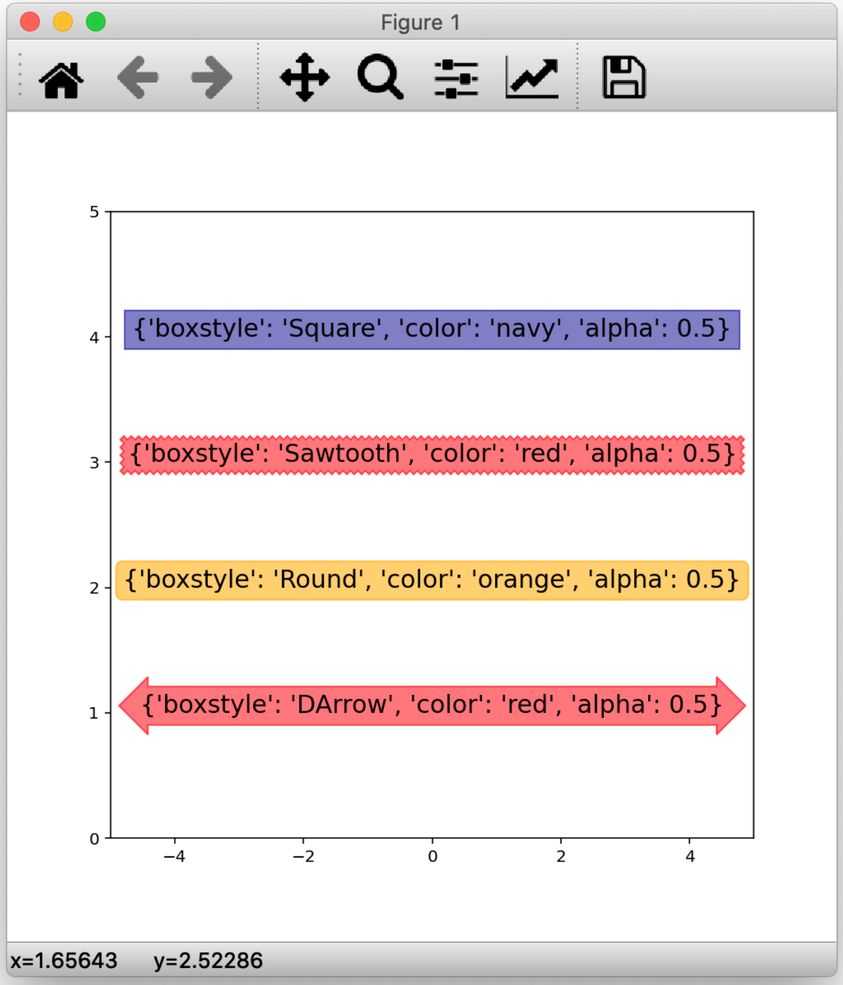
Class	Name	Attrs
Circle	`circle`	pad=0.3
DArrow	`darrow`	pad=0.3
LArrow	`larrow`	pad=0.3
RArrow	`rarrow`	pad=0.3
Round	`round`	pad=0.3, rounding_size=None
Round4	`round4`	pad=0.3, rounding_size=None
Roundtooth	`roundtooth`	pad=0.3, tooth_size=None
Sawtooth	`sawtooth`	pad=0.3, tooth_size=None
Square	`square`	pad=0.3

****kwargs** : `Patch` properties

Property	Description
`agg_filter`	a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array
`alpha`	float or None
`animated`	bool
`antialiased` or aa	unknown
`capstyle`	{'butt', 'round', 'projecting'}
`clip_box`	`Bbox`
`clip_on`	bool
`clip_path`	Patch or (Path, Transform) or None
`color`	color
`contains`	unknown
`edgecolor` or ec	color or None or 'auto'
`facecolor` or fc	color or None

6. Text Properties(bbox)

```
bbox_list = [{'boxstyle': 'DArrow', 'color': 'red', 'alpha': 0.5},
             {'boxstyle': 'Round', 'color': 'orange', 'alpha': 0.5},
             {'boxstyle': 'Sawtooth', 'color': 'red', 'alpha': 0.5},
             {'boxstyle': 'Square', 'color': 'navy', 'alpha': 0.5}]


ax.set_xlim([-5, 5])
ax.set_ylim([0, len(bbox_list)+1])

for b_idx, bbox in enumerate(bbox_list):
    ax.text(0, b_idx+1, ha="center",
            s=bbox,
            fontsize=15,
            bbox=bbox)
```

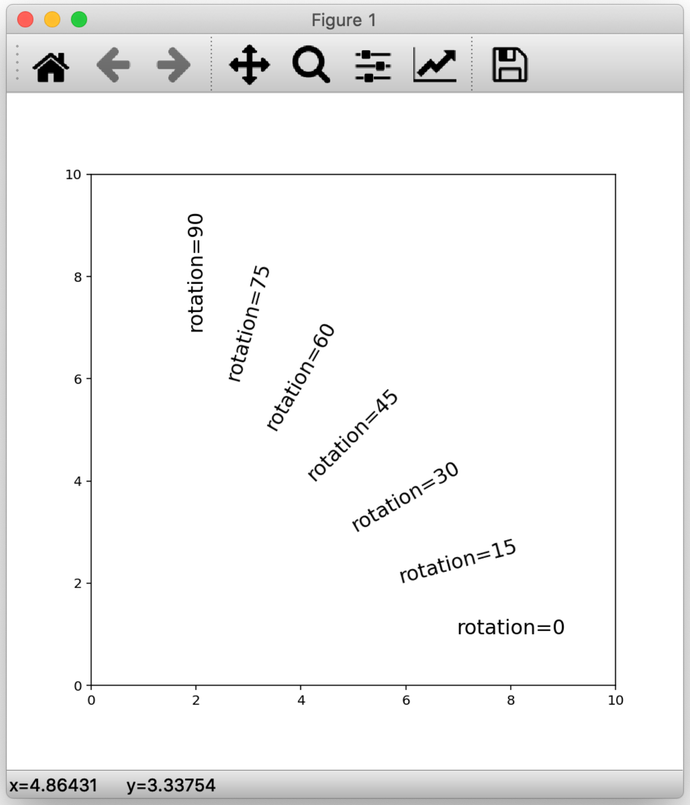


6. Text Properties(rotation)

rasterized	bool or None
rotation	float or {'vertical', 'horizontal'}
rotation_mode	{None, 'default', 'anchor'}
sketch_params	(scale: float, length: float, randomness: float)
snap	bool or None
text	object
transform	Transform
url	str
usetex	bool or None
verticalalignment or va	{'center', 'top', 'bottom', 'baseline', 'center_baseline'}
visible	bool
wrap	bool
x	float
y	float
zorder	float

6. Text Properties(rotation)

```
rotation_list = [0, 15, 30, 45, 60, 75, 90]

ax.set_xlim([0, len(rotation_list)+3])
ax.set_ylim([0, len(rotation_list)+3])

for r_idx, rotation in enumerate(rotation_list):
    ax.text(len(rotation_list)+1 - r_idx, r_idx+1, ha="center",
            s='rotation='+str(rotation),
            fontsize=15,
            rotation=rotation)
```



7. Font Dictionary

matplotlib.axes.Axes.set_title

`Axes.set_title(self, label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)` [source]

Set a title for the axes.

Set one of the three available axes titles. The available titles are positioned above the axes in the center, flush with the left edge, and flush with the right edge.

matplotlib.axes.Axes.set_xlabel

`Axes.set_xlabel(self, xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the x-axis.

matplotlib.axes.Axes.set_ylabel

`Axes.set_ylabel(self, ylabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)` [source]

Set the label for the y-axis.

matplotlib.axes.Axes.text

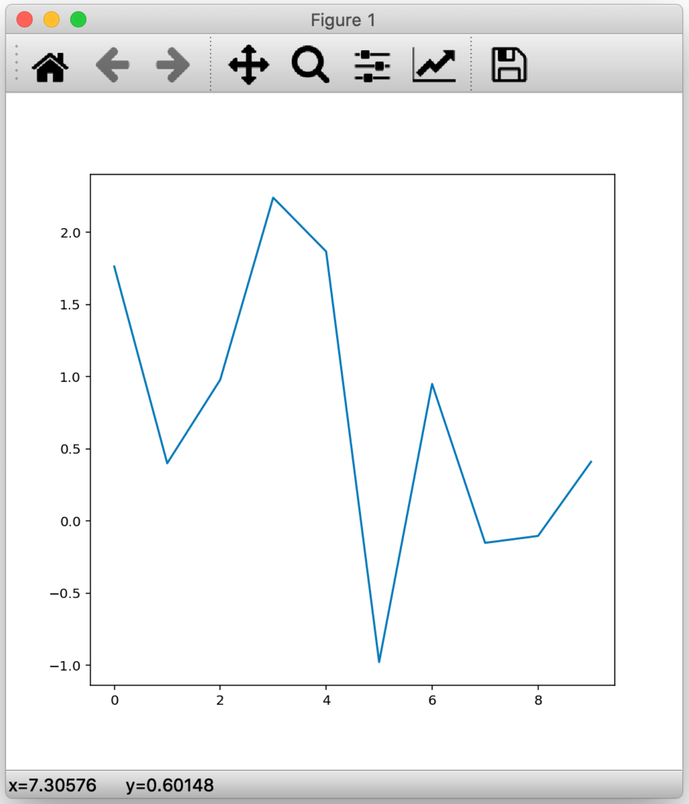
`Axes.text(self, x, y, s, fontdict=None, **kwargs)` [source]

Add text to the axes.

Add the text *s* to the axes at location *x*, *y* in data coordinates.

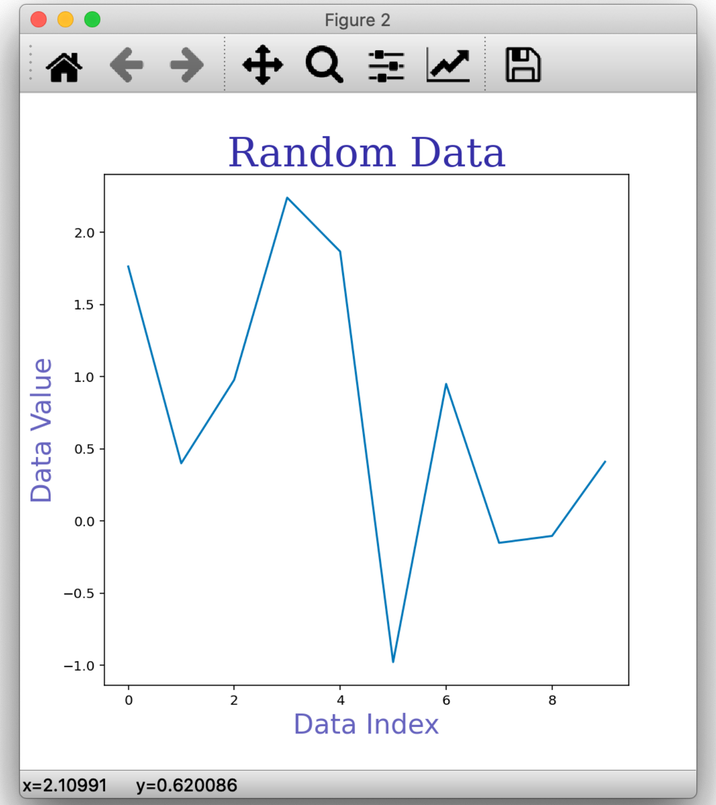
7. Font Dictionary

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

np.random.seed(0)
fig, ax = plt.subplots(figsize=(7, 7))

data = np.random.normal(0, 1, (10,))
M_idx, M_val = np.argmax(data), np.round(np.max(data),2)
m_idx, m_val = np.argmin(data), np.round(np.min(data), 2)

M_string = "Max: ({},{})".format(str(M_idx), str(M_val))
m_string = "min: ({},{})".format(str(m_idx), str(m_val))

ax.plot(data)
```



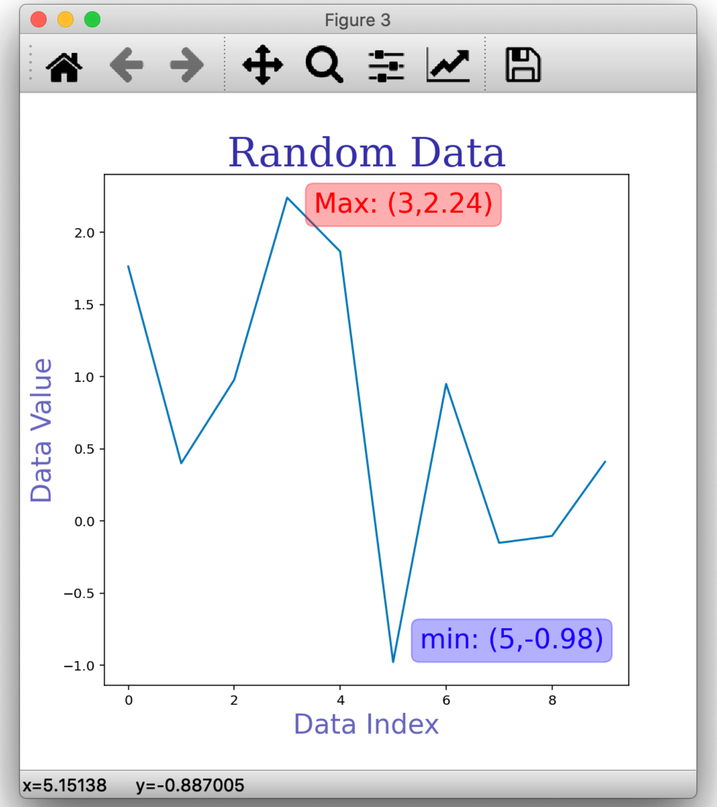
7. Font Dictionary

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

np.random.seed(0)

fig, ax = plt.subplots(figsize=(7, 7))
data = np.random.normal(0, 1, (10,))

M_idx, M_val = np.argmax(data), np.round(np.max(data),2)
m_idx, m_val = np.argmin(data), np.round(np.min(data), 2)

M_string = "Max: ({},{})".format(str(M_idx), str(M_val))
m_string = "min: ({},{})".format(str(m_idx), str(m_val))

ax.plot(data)

ax.set_title("Random Data",
             fontsize=30,
             fontfamily='serif',
             color='darkblue',
             alpha=0.8)
ax.set_xlabel("Data Index",
             fontsize=20,
             color='darkblue',
             alpha=0.6)
ax.set_ylabel("Data Value",
             fontsize=20,
             color='darkblue',
             alpha=0.6)
```



7. Font Dictionary

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

np.random.seed(0)

fig, ax = plt.subplots(figsize=(7, 7))
data = np.random.normal(0, 1, (10,))

M_idx, M_val = np.argmax(data), np.round(np.max(data),2)
m_idx, m_val = np.argmin(data), np.round(np.min(data), 2)

M_string = "Max: ({},{})".format(str(M_idx), str(M_val))
m_string = "min: ({},{})".format(str(m_idx), str(m_val))

ax.plot(data)

ax.set_title("Random Data",
             fontsize=30,
             fontfamily='serif',
             color='darkblue',
             alpha=0.8)

ax.set_xlabel("Data Index",
             fontsize=20,
             color='darkblue',
             alpha=0.6)

ax.set_ylabel("Data Value",
             fontsize=20,
             color='darkblue',
             alpha=0.6)

ax.text(x=M_idx+0.5, y=M_val-0.1,
        s=M_string,
        fontsize=20,
        color='r',
        bbox={'boxstyle': 'Round',
              'color': 'r', 'alpha': 0.3})

ax.text(x=m_idx+0.5, y=m_val+0.1,
        s=m_string,
        fontsize=20,
        color='b',
        bbox={'boxstyle': 'Round',
              'color': 'b', 'alpha': 0.3})
```



7. Font Dictionary

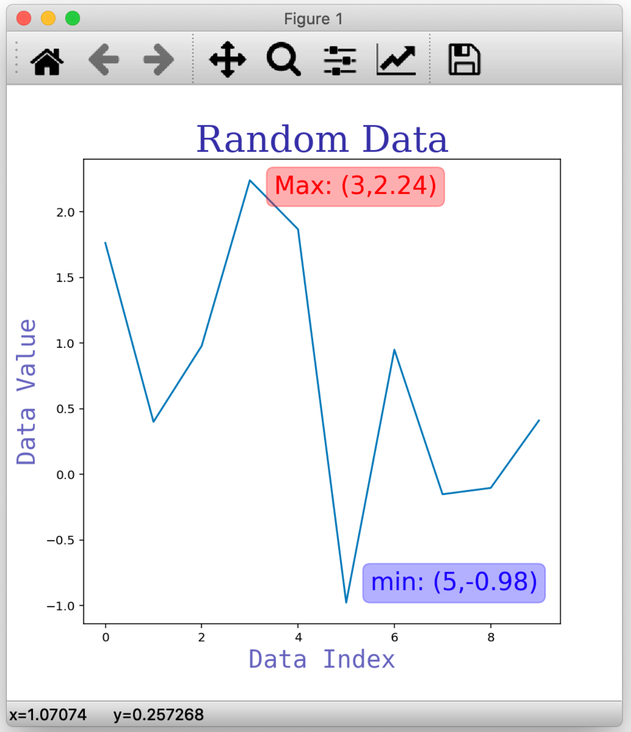
```
ax.set_title("Random Data",
             fontsize=30,
             fontfamily='serif',
             color='darkblue',
             alpha=0.8)
ax.set_xlabel("Data Index",
             fontsize=20,
             color='darkblue',
             alpha=0.6)
ax.set_ylabel("Data Value",
             fontsize=20,
             color='darkblue',
             alpha=0.6)


ax.text(x=M_idx+0.5, y=M_val-0.1,
        s=M_string,
        fontsize=20,
        color='r',
        bbox={'boxstyle': ‘Round’,
              'color': 'r', 'alpha': 0.3})
ax.text(x=m_idx+0.5, y=m_val+0.1,
        s=m_string,
        fontsize=20,
        color='b',
        bbox={'boxstyle': ‘Round',
              ‘color': 'b', 'alpha': 0.3})
```

```
title_font_dict = {'fontsize': 30, 'fontfamily': 'serif',
                   'color': 'darkblue', 'alpha': 0.8}
xylabel_font_dict = {'fontsize': 20, 'fontfamily': 'monospace',
                     'color': 'darkblue', 'alpha': 0.6}
M_font_dict = {'fontsize': 20, 'color': 'r',
               'bbox': {'boxstyle': ‘Round’,
                        'color': 'r', 'alpha': 0.3}}
m_font_dict = {'fontsize': 20, 'color': 'b',
               'bbox': {'boxstyle': ‘Round',
                        ‘color': 'b', 'alpha': 0.3}}

ax.set_title("Random Data",
             fontdict=title_font_dict)
ax.set_xlabel("Data Index",
              fontdict=xylabel_font_dict)
ax.set_ylabel("Data Value",
              fontdict=xylabel_font_dict)

ax.text(x=M_idx+0.5, y=M_val-0.1,
        s=M_string,
        fontdict=M_font_dict)

ax.text(x=m_idx+0.5, y=m_val+0.1,
        s=m_string,
        fontdict=m_font_dict)
```

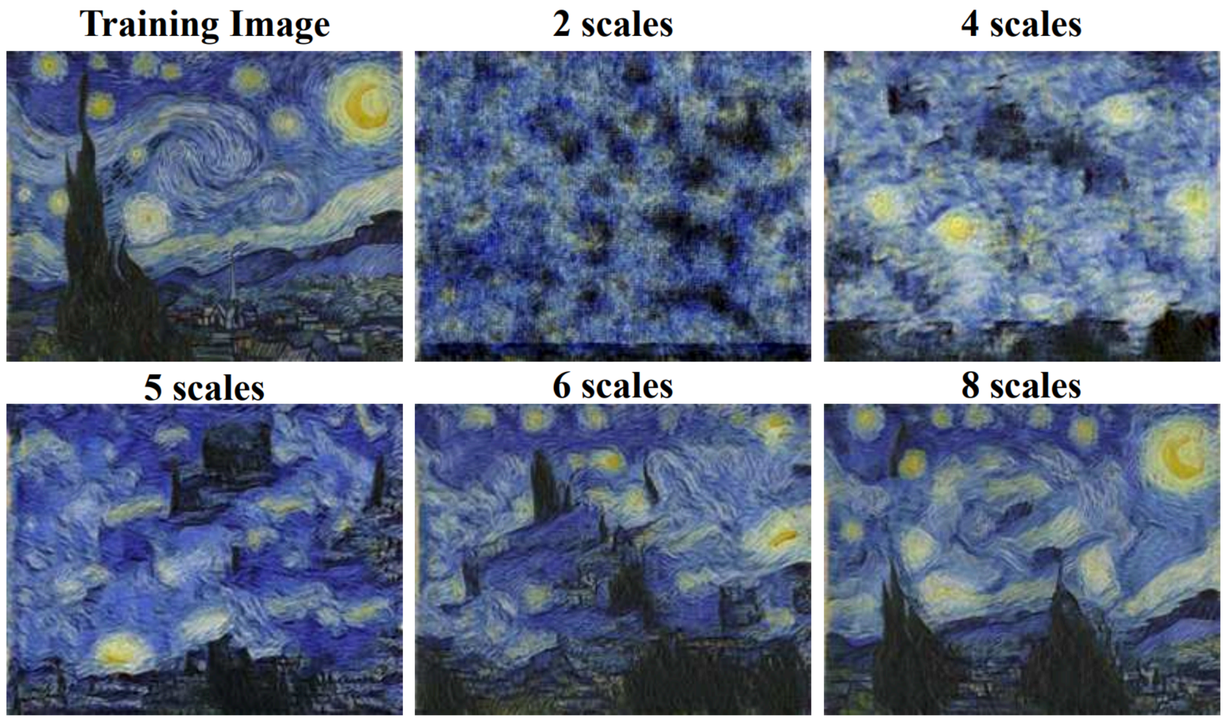
7. Font Dictionary

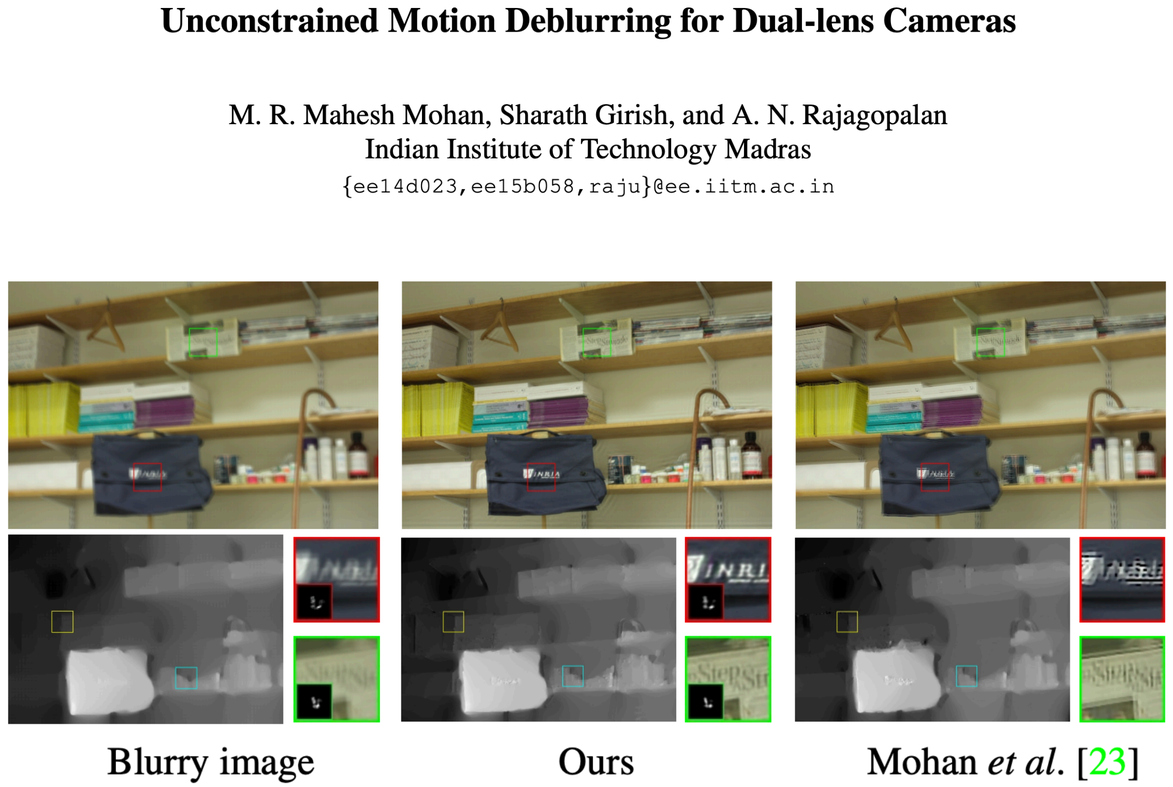
```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)

fig, ax = plt.subplots(figsize=(7, 7))
data = np.random.normal(0, 1, (10,))

M_idx, M_val = np.argmax(data), np.round(np.max(data),2)
m_idx, m_val = np.argmin(data), np.round(np.min(data), 2)

M_string = "Max: ({},{})".format(str(M_idx), str(M_val))
m_string = "min: ({},{})".format(str(m_idx), str(m_val))

ax.plot(data)

title_font_dict = {'fontsize': 30, 'fontfamily': 'serif',
                   'color': 'darkblue', 'alpha': 0.8}
xylabel_font_dict = {'fontsize': 20, 'fontfamily': 'monospace',
                     'color': 'darkblue', 'alpha': 0.6}
M_font_dict = {'fontsize': 20, 'color': 'r',
               'bbox': {'boxstyle': 'Round',
                        'color': 'r', 'alpha': 0.3}}
m_font_dict = {'fontsize': 20, 'color': 'b',
               'bbox': {'boxstyle': 'Round',
                        'color': 'b', 'alpha': 0.3}}
```

```
ax.set_title("Random Data",
             fontdict=title_font_dict)

ax.set_xlabel("Data Index",
              fontdict=xylabel_font_dict)

ax.set_ylabel("Data Value",
              fontdict=xylabel_font_dict)


ax.text(x=M_idx+0.5, y=M_val-0.1,
        s=M_string,
        fontdict=M_font_dict)


ax.text(x=m_idx+0.5, y=m_val+0.1,
        s=m_string,
        fontdict=m_font_dict)
```



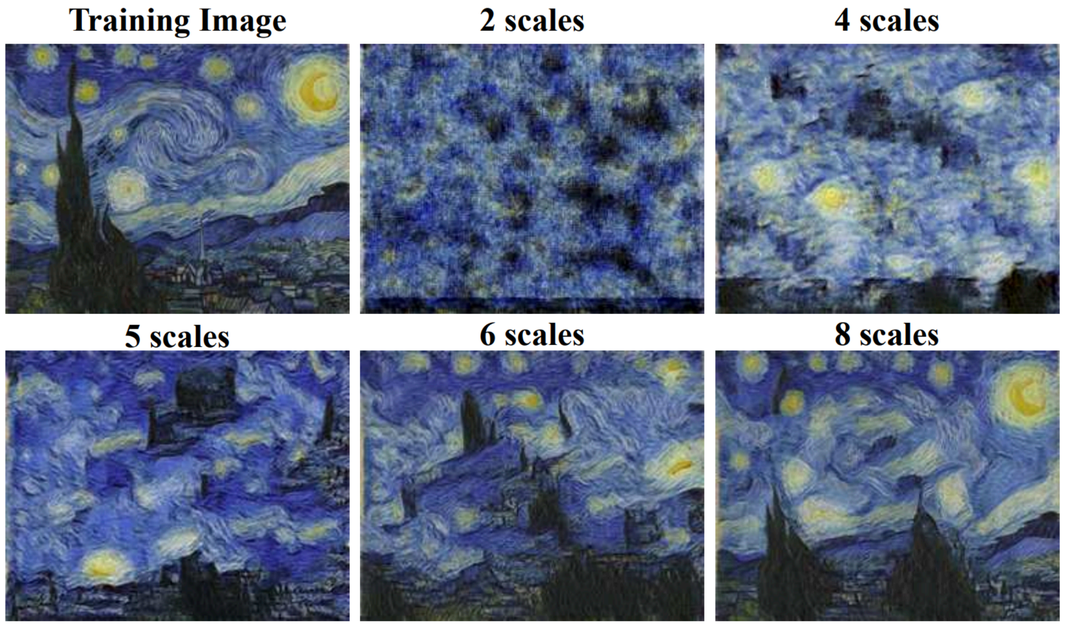
8. Title Exercise

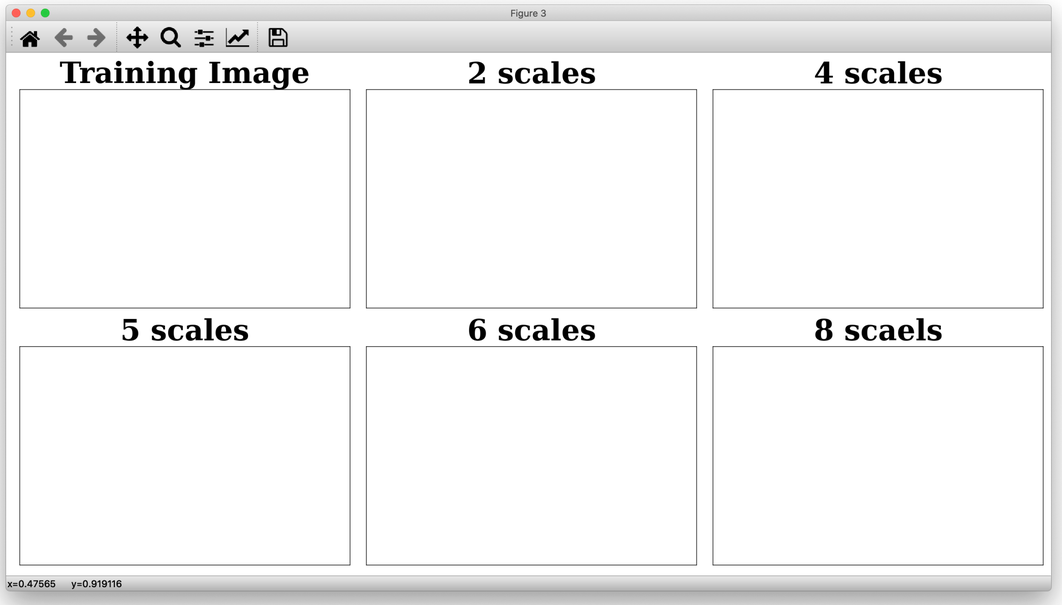




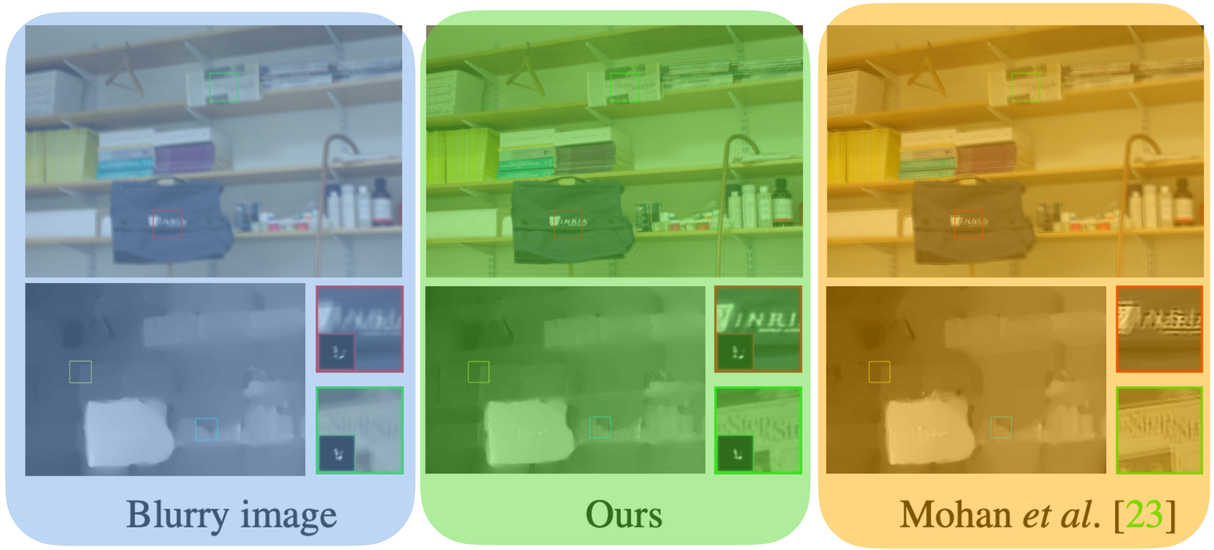
8. Title Exercise1

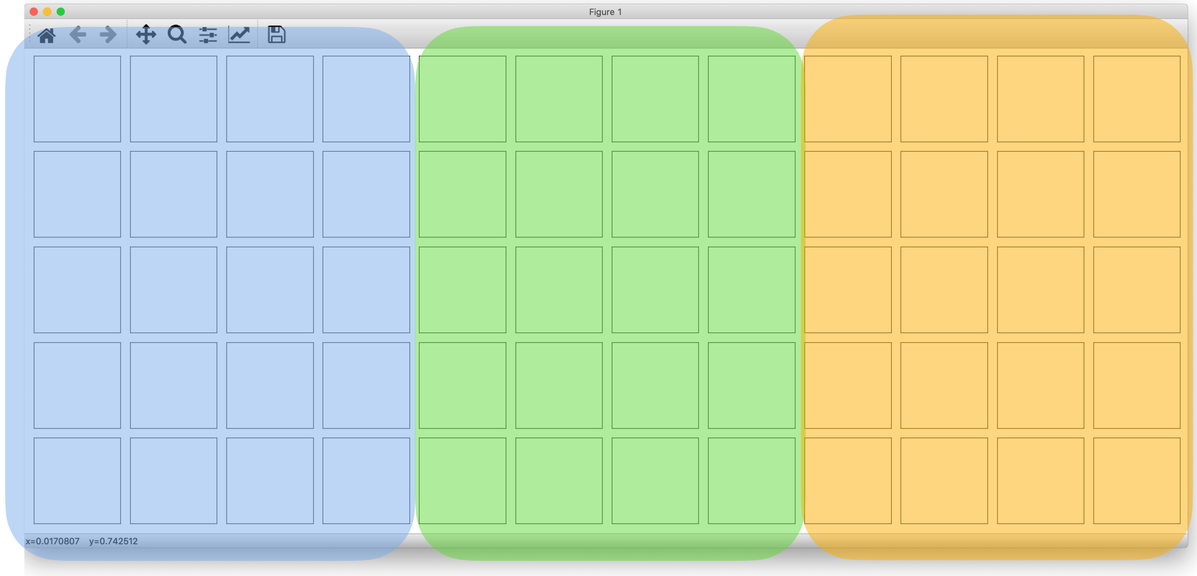
```
import matplotlib.pyplot as plt

title_list = ['Training Image', '2 scales',
              '4 scales', '5 scales',
              '6 scales', '8 scaels']

title_font_dict = {'fontsize': 20,
                   'fontweight': 'bold'}

fig, axes = plt.subplots(2, 3,
                         figsize=(20, 10))

for ax_idx, ax in enumerate(axes.flat):
    ax.set_title(title_list[ax_idx],
                 fontdict=title_font_dict)

    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

fig.tight_layout()
```



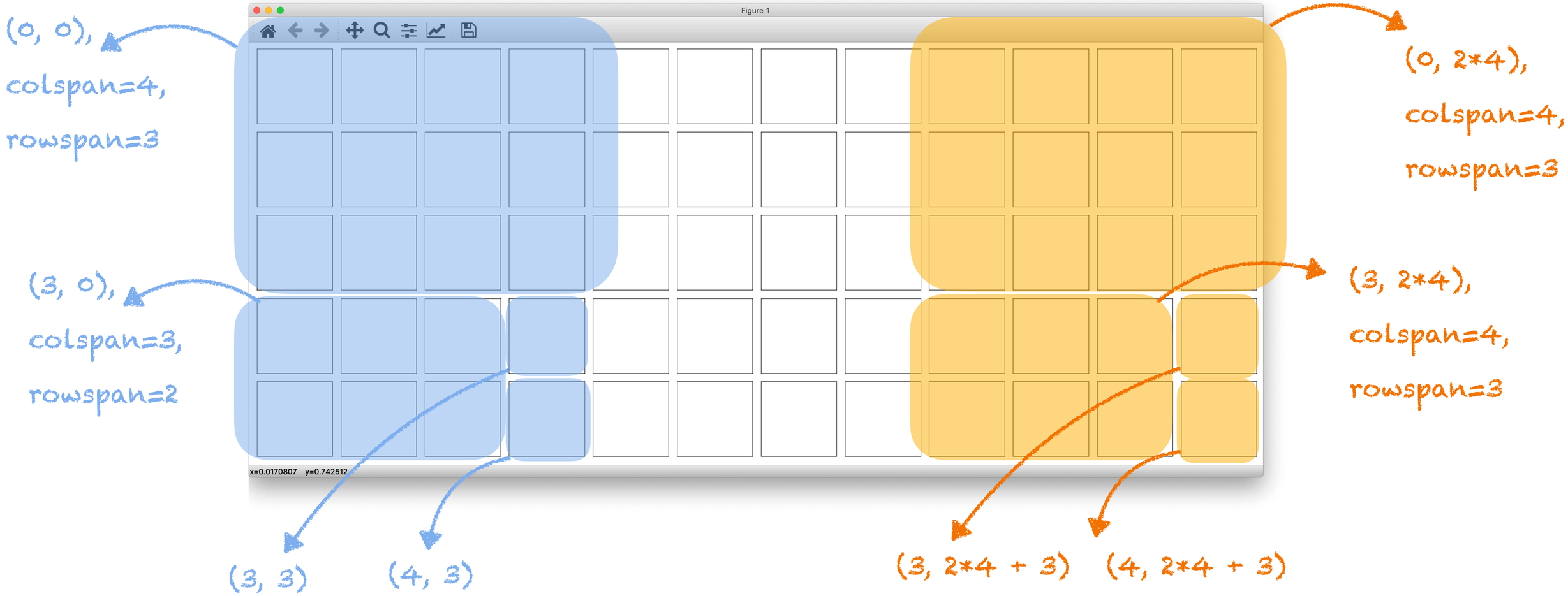


8. Title Exercise2

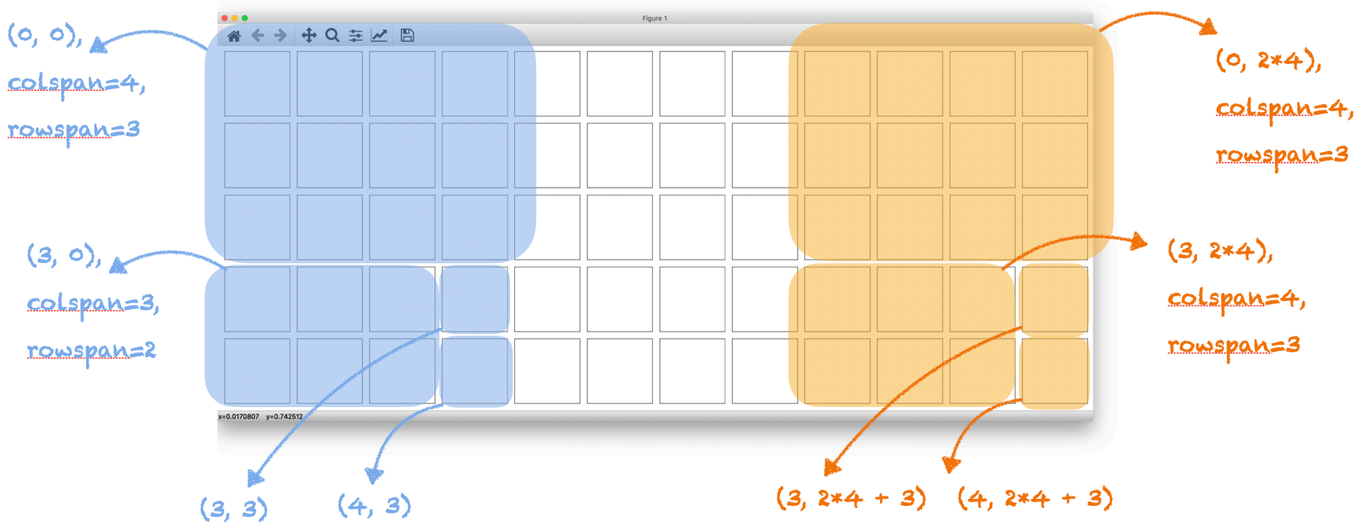




8. Title Exercise2

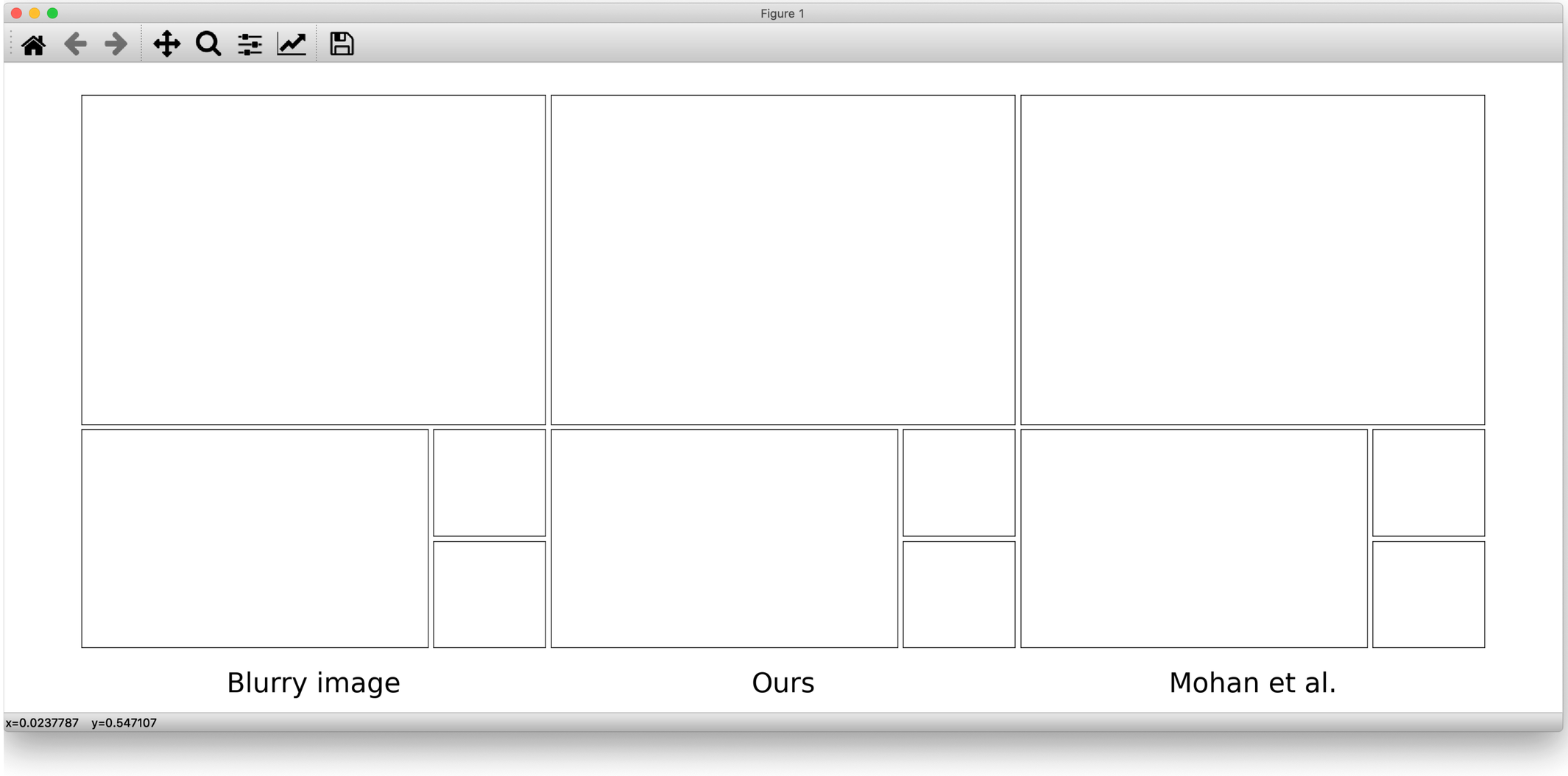


8. Title Exercise2




```
import matplotlib.pyplot as plt

title_list = ['Blurry image', 'Ours', 'Mohan et al.']
fig = plt.figure(figsize=(24, 10))

axes = np.empty((0, 4))
n_g, n_col, n_row = 3, 4, 5


for g_idx in range(n_g):
    axes_row = np.empty((0,))

    axes_row = np.append(axes_row,
                         plt.subplot2grid((5, 12), (0, g_idx*n_col),
                                          colspan=4, rowspan=3,
                                          fig=fig))
    axes_row = np.append(axes_row,
                         plt.subplot2grid((5, 12), (3, g_idx*n_col),
                                          colspan=3, rowspan=2,
                                          fig=fig))
    axes_row = np.append(axes_row,
                         plt.subplot2grid((5, 12), (3, 3 + g_idx*n_col),
                                          fig=fig))
    axes_row = np.append(axes_row,
                         plt.subplot2grid((5, 12), (4, 3 + g_idx*n_col),
                                          fig=fig))

    for ax in axes_row:
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)
    axes = np.vstack((axes, axes_row))
    axes_row[0].set_title(title_list[g_idx],
                          fontsize=30,
                          y=-0.83)

fig.subplots_adjust(bottom=0.1, top=0.95,
                    left=0.05, right=0.95,
                    hspace=0.05, wspace=0.05)
```

8. Title Exercise2



Python for Data Visualization

-Chapter.1 Matplotlib Anatomy -

1-03. Titles, Labels and Font Dict

1. Title APIs
2. fig.suptitle and ax.set_title
3. ax.set_xlabel and ax.set_ylabel
4. Text Alignment
5. Title Alignment
6. Text Properties
7. Font Dictionary
8. Title Exercise