Name → Vinayak Rajeshwar kottawar
Roll No → 28 [S2] [APP] SYCSE

Date :

Lab Work # 11

Aim :→ Study of Matplotlib Library Functions in python.

Objective → The objective of this lab work is:

1] students will be able to list functions of MatplotLib.
2] students will be able to configure MatplotLib.
3] students will be able to apply functions of MatplotLib.

Outcomes :→ After completing the Lab Work # 11 students will be able to:

1] Configure Matplotlib in their system.
2] Understand use of Matplotlibfunctions.
3] Apply Matplotlib Functions.

Pre-Requisites →

1] Basic syntax of python.

Theory :→

Installing an official release.

Matplotlib & its dependencies are available as wheel packages for macOS, windows & Linux distributions.

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

If this command results in Matplotlib being compiled from source and there is trouble with compilation, you can add --prefer-binary to select the newest version of Matplotlib for which there is a precompiled wheel for your OS & python.

Note :-

The following backends work out of the box :

Agg, PS, PDF, SVG.

Python is typically shipped with tk bindings which are used by TkAgg.

For support of other GUI framework, LaTeX rendering, saving animation & a larger selection of file formats, you need to install additional dependencies.

Although not required we suggest also installing IPython for interactive use. To easily install a complete scientific python stack, see scientific python Distribution below.

Test data

The wheels (*.whl) on the PyPI download page do not contain test data or example code.

If you want to try the many demos that come in the Matplotlib source distribution, download the *.tar.gz file and look in the examples subdirectory.

To run the test suite :-

→ extract the lib/matplotlib/tests or lib/mpl_toolkits/tests directories from the source distribution.
→ Install test dependencies : pytest, MikTex, Ghostscript, ffmpeg, avconv, ImageMagick & Inkscape.
→ run python -m pytest.

Third-party distributions of Matplotlib.
Scientific python Distribution.

Anaconda & Activestate are excellent choices that "just work" out of the box for Windows, macOS & common Linux platforms. winpython is an option for Windows users. All of these distributions include Matplotlib & lots of other useful (data) science tools.

Linux : using your package manager.
If you are on Linux, you might prefer to use your package manager. Matplotlib is packaged for almost every major Linux distribution.
→ Debian / Ubuntu : sudo apt-get install python3-matplotlib
→ Fedora : sudo dnf install python3-matplotlib.
→ Red Hat : sudo yum install python3-matplotlib.
→ Arch : Sudo pacman -S python-matplotlib.

Page No. ( )

Date :

Installing from source.

If you are interested in contributing to Matplotlib development, running the latest source code, or just like to build everything yourself, it is not difficult to build Matplotlib from source. Grab the latest tar.gz release file from the PyPI files page, or if you want to develop Matplotlib or just need the latest bugfixed version, grab the latest git version, and see Install from source.

Matplotlib can be installed from the source directory with a simple

Python -m pip install.

We provide a setup.cfg file which you can use to customize that build process. For example, which default backend to use, whether some of the optional libraries that Matplotlib ships with are installed, & so on. This file will be particularly useful to those packaging Matplotlib.

Welcome To Colaboratory - Cola | SyntaxError: invalid syntax site:sta | APP_SY_2021-22 | Basic Usage —

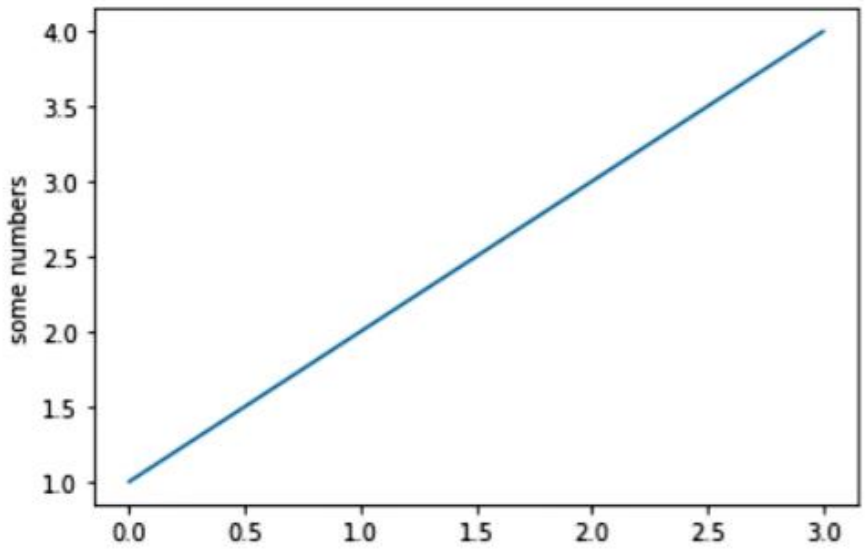
colab.research.google.com/?utm_source=scs-index#scrollTo=NqbxDEy5vP0v&uniqifier=4

Welcome To Colaboratory

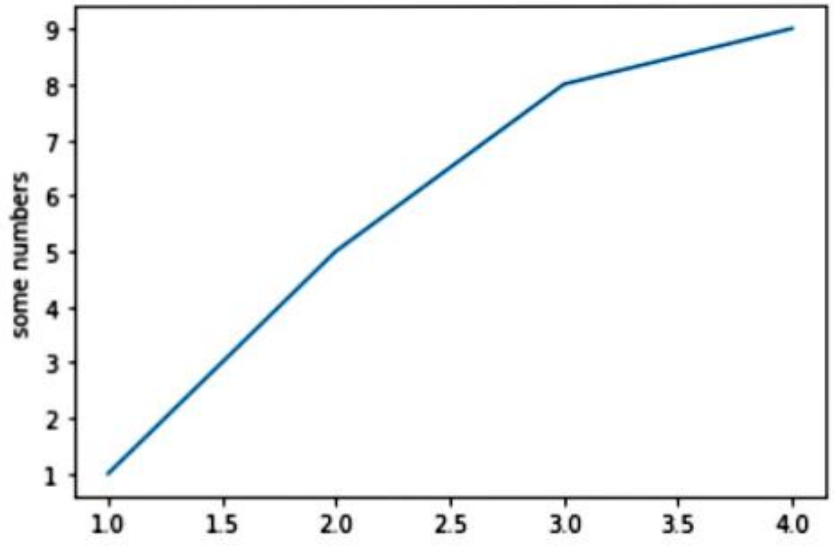
File Edit View Insert Runtime Tools Help Cannot save changes

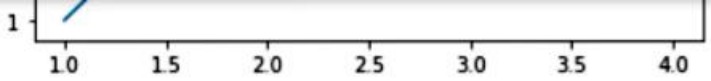
+ Code + Text Copy to Drive

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



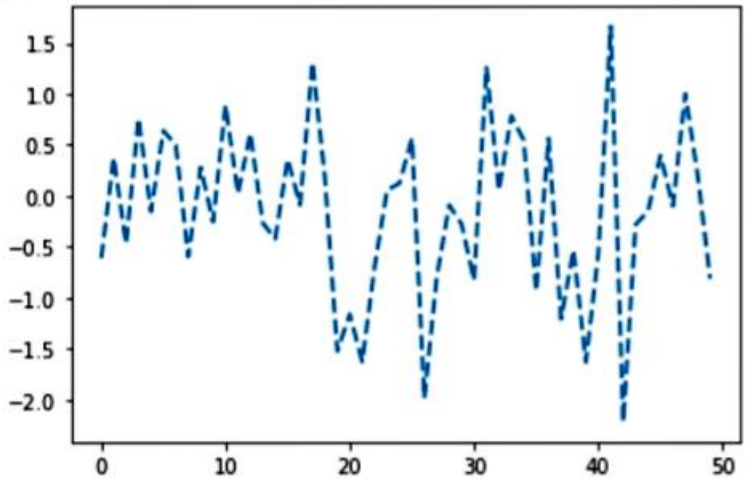
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4],[1,5,8,9])
plt.ylabel('some numbers')
plt.show()
```





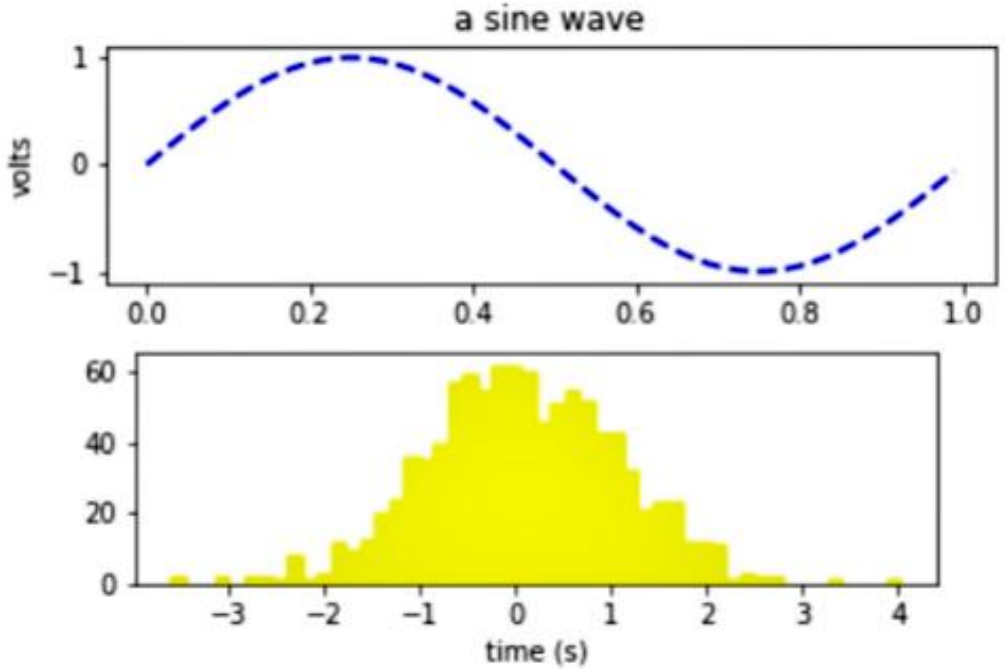

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from cycler import cycler
mpl.rcParams['lines.linewidth'] = 2
mpl.rcParams['lines.linestyle'] = '--'
data = np.random.randn(50)
plt.plot(data)
```

```
[<matplotlib.lines.Line2D at 0x7f34a1e64b50>]
```



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

fig = plt.figure()
fig.subplots_adjust(top=0.8)
ax1 = fig.add_subplot(211)
ax1.set_ylabel('volts')
ax1.set_title('a sine wave')

t = np.arange(0.0, 1.0, 0.01)
s = np.sin(2*np.pi*t)
line, = ax1.plot(t, s, color='blue', lw=2)

# Fixing random state for reproducibility
np.random.seed(19680801)

ax2 = fig.add_axes([0.15, 0.1, 0.7, 0.3])
n, bins, patches = ax2.hist(np.random.randn(1000), 50,
                            facecolor='yellow', edgecolor='yellow')
ax2.set_xlabel('time (s)')

plt.show()
```

```
ax2.set_xlabel('time (s)')

plt.show()
```



Welcome To Colaboratory - Cola | APP_SY_2021-22 | Text properties and layout — Ma

colab.research.google.com/?utm_source=scs-index#scrollTo=aTFWPxib36Lo&uniqifier=4

Welcome To Colaboratory

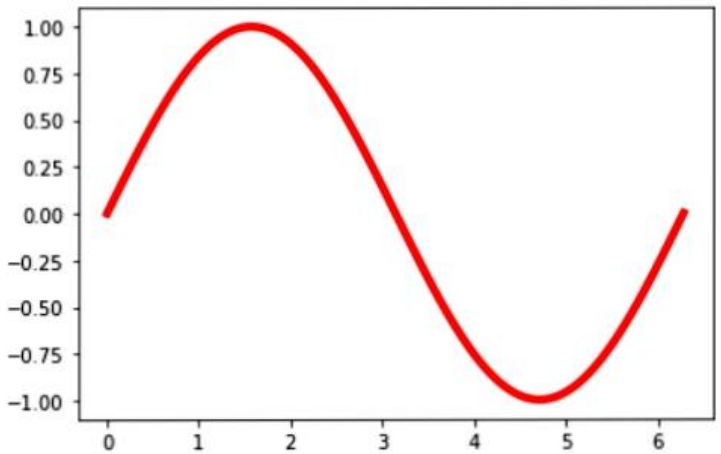
File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

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

x = np.linspace(0, 2 * np.pi, 100)

fig, ax = plt.subplots()

# animated=True tells matplotlib to only draw the artist when we
# explicitly request it
(ln,) = ax.plot(x, np.sin(x), animated=True)

# make sure the window is raised, but the script keeps going
plt.show(block=False)

# stop to admire our empty window axes and ensure it is rendered at
# least once.
#
# We need to fully draw the figure at its final size on the screen
# before we continue on so that :
#  a) we have the correctly sized and drawn background to grab
#  b) we have a cached renderer so that ``ax.draw_artist`` works
# so we spin the event loop to let the backend process any pending operations
plt.pause(0.1)

# get copy of entire figure (everything inside fig.bbox) sans animated artist
bg = fig.canvas.copy_from_bbox(fig.bbox)
# draw the animated artist, this uses a cached renderer
ax.draw_artist(ln)
# show the result to the screen, this pushes the updated RGBA buffer from the
# renderer to the GUI framework so you can see it
fig.canvas.blit(fig.bbox)
```

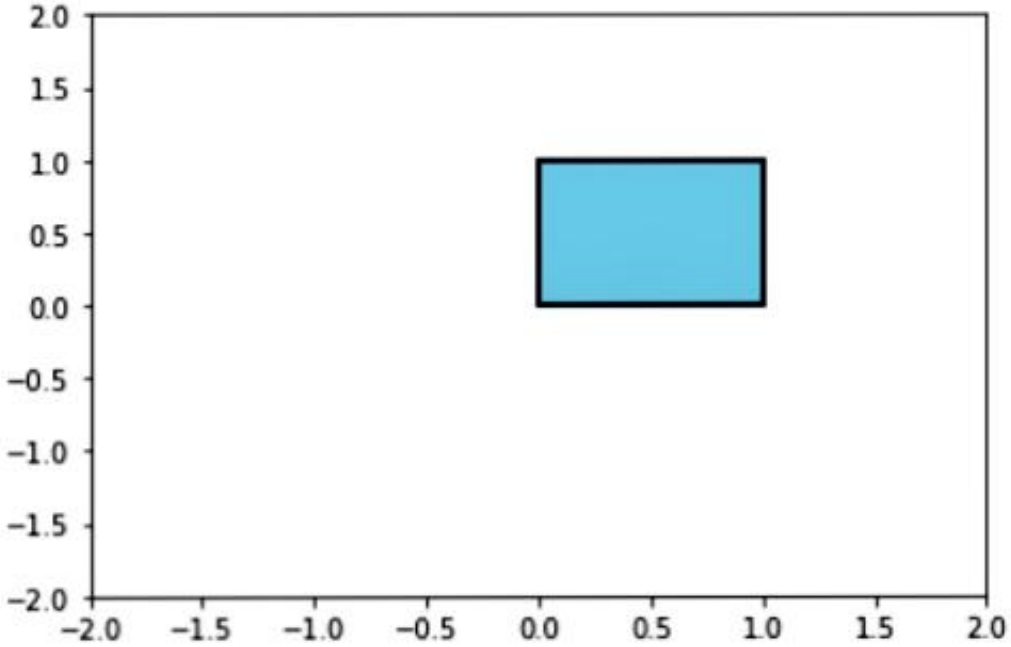
0s completed at 11:16 AM

[23]
```
# show the result to the screen, this pushes the updated RGBA buffer from the
# renderer to the GUI framework so you can see it
fig.canvas.blit(fig.bbox)

for j in range(100):
    # reset the background back in the canvas state, screen unchanged
    fig.canvas.restore_region(bg)
    # update the artist, neither the canvas state nor the screen have changed
    ln.set_ydata(np.sin(x + (j / 100) * np.pi))
    # re-render the artist, updating the canvas state, but not the screen
    ax.draw_artist(ln)
    # copy the image to the GUI state, but screen might not be changed yet
    fig.canvas.blit(fig.bbox)
    # flush any pending GUI events, re-painting the screen if needed
    fig.canvas.flush_events()
    # you can put a pause in if you want to slow things down
    # plt.pause(.1)
```

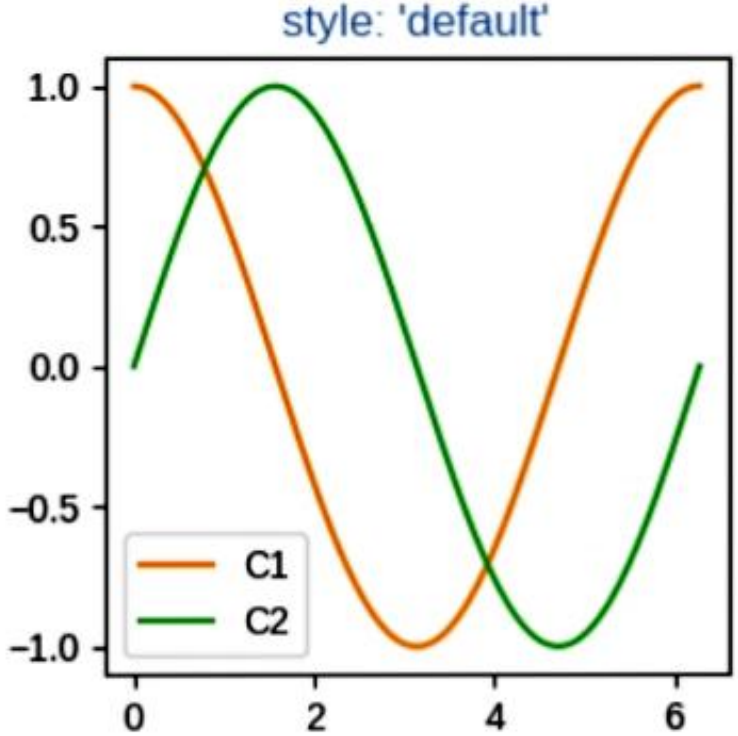


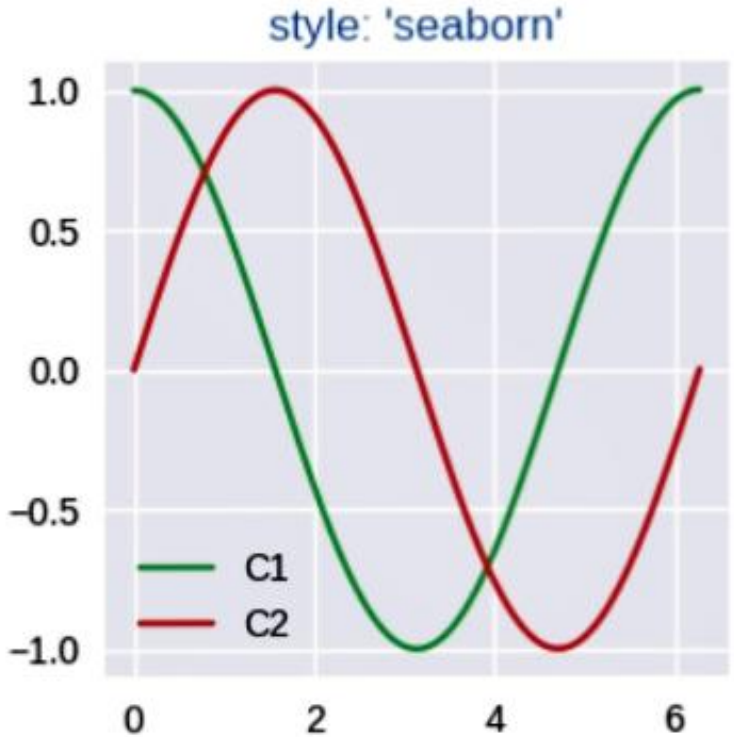
0 1 2 3 4 5 6

[26]
```
import matplotlib.pyplot as plt
from matplotlib.path import Path
import matplotlib.patches as patches

verts = [
   (0., 0.),  # left, bottom
   (0., 1.),  # left, top
   (1., 1.),  # right, top
   (1., 0.),  # right, bottom
   (0., 0.),  # ignored
]

codes = [
    Path.MOVETO,
    Path.LINETO,
    Path.LINETO,
    Path.LINETO,
    Path.CLOSEPOLY,
]

path = Path(verts, codes)

fig, ax = plt.subplots()
patch = patches.PathPatch(path, facecolor='skyblue', lw=2)
ax.add_patch(patch)
ax.set_xlim(-2, 2)
ax.set_ylim(-2, 2)
plt.show()
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

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

th = np.linspace(0, 2*np.pi, 128)


def demo(sty):
    mpl.style.use(sty)
    fig, ax = plt.subplots(figsize=(3, 3))

    ax.set_title('style: {!r}'.format(sty), color='C0')

    ax.plot(th, np.cos(th), 'C1', label='C1')
    ax.plot(th, np.sin(th), 'C2', label='C2')
    ax.legend()


demo('default')
demo('seaborn')
```


style: 'default'
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
C1
C2


style: 'seaborn'
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
C1
C2

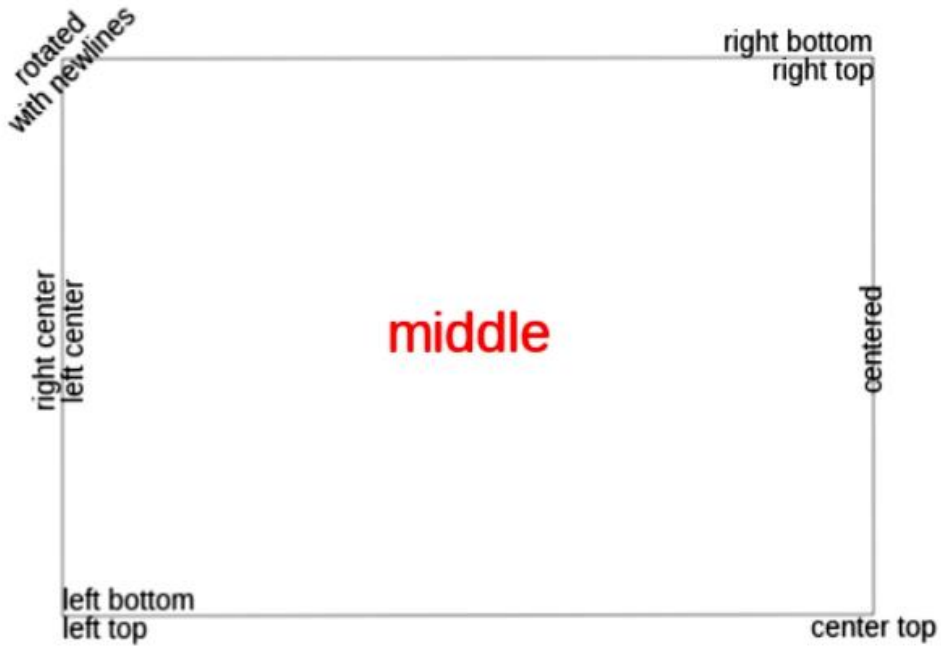
0s com

```python
import matplotlib.pyplot as plt
import matplotlib.patches as patches

# build a rectangle in axes coords
left, width = .25, .5
bottom, height = .25, .5
right = left + width
top = bottom + height

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])

# axes coordinates: (0, 0) is bottom left and (1, 1) is upper right
p = patches.Rectangle(
    (left, bottom), width, height,
    fill=False, transform=ax.transAxes, clip_on=False
    )

ax.add_patch(p)

ax.text(left, bottom, 'left top',
        horizontalalignment='left',
        verticalalignment='top',
        transform=ax.transAxes)

ax.text(left, bottom, 'left bottom',
        horizontalalignment='left',
        verticalalignment='bottom',
        transform=ax.transAxes)

ax.text(right, top, 'right bottom',
```

✓ 0s completed at 11:16 AM

```
        transform=ax.transAxes)

ax.text(right, top, 'right bottom',
        horizontalalignment='right',
        verticalalignment='bottom',
        transform=ax.transAxes)

ax.text(right, top, 'right top',
        horizontalalignment='right',
        verticalalignment='top',
        transform=ax.transAxes)

ax.text(right, bottom, 'center top',
        horizontalalignment='center',
        verticalalignment='top',
        transform=ax.transAxes)

ax.text(left, 0.5*(bottom+top), 'right center',
        horizontalalignment='right',
        verticalalignment='center',
        rotation='vertical',
        transform=ax.transAxes)

ax.text(left, 0.5*(bottom+top), 'left center',
        horizontalalignment='left',
        verticalalignment='center',
        rotation='vertical',
        transform=ax.transAxes)

ax.text(0.5*(left+right), 0.5*(bottom+top), 'middle',
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=20, color='red',
```

```
        horizontalalignment='left',
        verticalalignment='center',
        rotation='vertical',
        transform=ax.transAxes)

ax.text(0.5*(left+right), 0.5*(bottom+top), 'middle',
        horizontalalignment='center',
        verticalalignment='center',
        fontsize=20, color='red',
        transform=ax.transAxes)

ax.text(right, 0.5*(bottom+top), 'centered',
        horizontalalignment='center',
        verticalalignment='center',
        rotation='vertical',
        transform=ax.transAxes)

ax.text(left, top, 'rotated\nwith newlines',
        horizontalalignment='center',
        verticalalignment='center',
        rotation=45,
        transform=ax.transAxes)

ax.set_axis_off()
plt.show()
```


rotated
with newlines
right bottom
right top
right center
left center
middle
centered
left bottom
left top
center top