

Data Visualization

- Matplotlib
- Seaborn

Matplotlib

- Matplotlib is one of the best library in python for visulization

Different types of plots:

- Lineplot
- Scatterplot
- Bar plot
- Box plot
- Histogram
- Pie plot
- how to read an image and display an image
- how to save the image

```
In [1]: 1 import matplotlib.pyplot as plt
```

In [2]: `print(dir(plt))`

```
['Annotation', 'Arrow', 'Artist', 'AutoLocator', 'Axes', 'Button', 'Circle', 'Figure', 'FigureCanvasBase', 'FixedFormatter', 'FixedLocator', 'FormatStrFormatter', 'Formatter', 'FuncFormatter', 'GridSpec', 'IndexLocator', 'Line2D', 'LinearLocator', 'Locator', 'LogFormatter', 'LogFormatterExponent', 'LogFormatterMathtext', 'LogLocator', 'MaxNLocator', 'MultipleLocator', 'Normalize', 'NullFormatter', 'NullLocator', 'Number', 'PolarAxes', 'Polygon', 'Rectangle', 'ScalarFormatter', 'Slider', 'Subplot', 'SubplotTool', 'Text', 'TickHelper', 'Widget', '_INSTALL_FIG_OBSERVER', '_IP_REGISTERED', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_auto_draw_if_interactive', '_autogen_docstring', '_backend_mod', '_get_running_interactive_framework', '_interactive_bk', '_log', '_pylab_helpers', '_setp', '_setup_pyplot_info_docstrings', '_show', '_string_to_bool', 'acorr', 'angle_spectrum', 'annotate', 'arrow', 'autoscale', 'autumn', 'axes', 'axhline', 'axhspan', 'axis', 'axvline', 'axvspan', 'bar', 'barbs', 'barh', 'bone', 'box', 'boxplot', 'broken_barh', 'cla', 'clabel', 'clf', 'clim', 'close', 'cm', 'cohere', 'colorbar', 'colormaps', 'connect', 'contour', 'contourf', 'cool', 'copper', 'csd', 'cycler', 'dedent', 'delaxes', 'deprecated', 'disconnect', 'docstring', 'draw', 'draw_all', 'draw_if_interactive', 'errorbar', 'eventplot', 'figaspect', 'figimage', 'figlegend', 'fignum_exists', 'figtext', 'figure', 'fill', 'fill_between', 'fill_betweenx', 'findobj', 'flag', 'gca', 'gcf', 'gci', 'get', 'get_backend', 'get_cmap', 'get_current_fig_manager', 'get_figlabels', 'get_fignums', 'get_plot_commands', 'get_scale_docs', 'get_scale_names', 'getp', 'ginput', 'gray', 'grid', 'hexbin', 'hist', 'hist2d', 'hlines', 'hot', 'hsv', 'importlib', 'imread', 'imsave', 'imshow', 'inferno', 'inspect', 'install_repl_displayhook', 'interactive', 'ioff', 'ion', 'isinteractive', 'jet', 'legend', 'locator_params', 'logging', 'loglog', 'magma', 'magnitude_spectrum', 'margins', 'matplotlib', 'matshow', 'minorticks_off', 'minorticks_on', 'mlab', 'new_figure_manager', 'nipy_spectral', 'np', 'pause', 'pcolor', 'pcolormesh', 'phase_spectrum', 'pie', 'pink', 'plasma', 'plot', 'plot_date', 'plotfile', 'plotting', 'polar', 'prism', 'psd', 'pylab_setup', 'quiver', 'quiverkey', 'rc', 'rcParams', 'rcParamsDefault', 'rcParamsOrig', 'rc_context', 'rcdefaults', 'rcsetup', 're', 'register_cmap', 'rgrids', 'savefig', 'sca', 'scatter', 'sci', 'semilogx', 'semilogy', 'set_cmap', 'setp', 'show', 'silent_list', 'specgram', 'spring', 'spy', 'stackplot', 'stem', 'step', 'streamplot', 'style', 'subplot', 'subplot2grid', 'subplot_tool', 'subplots', 'subplots_adjust', 'summer', 'suptitle', 'switch_backend', 'sys', 'table', 'text', 'thetagrids', 'tick_params', 'ticklabel_format', 'tight_layout', 'time', 'title', 'tricontour', 'tricontourf', 'tripcolor', 'tripplot', 'twinx', 'twiny', 'uninstall_repl_displayhook', 'violinplot', 'viridis', 'vlines', 'waitforbuttonpress', 'warn_deprecated', 'warnings', 'winter', 'xcorr', 'xkcd', 'xlabel', 'xlim', 'xscale', 'xticks', 'ylabel', 'ylim', 'yscale', 'yticks']
```

[Matplotlib official website \(https://matplotlib.org/\)](https://matplotlib.org/)

In [4]: 1 `help(plt.title)`

Help on function title in module matplotlib.pyplot:

```
title(label, fontdict=None, loc='center', pad=None, **kwargs)
    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'],
 'verticalalignment': 'baseline',
 'horizontalalignment': loc}
```

loc : {'center', 'left', 'right'}, str, optional
Which title to set, defaults to 'center'

pad : float
The offset of the title from the top of the axes, in points. Default is ``None`` to use rcParams['axes.titlepad'].

Returns

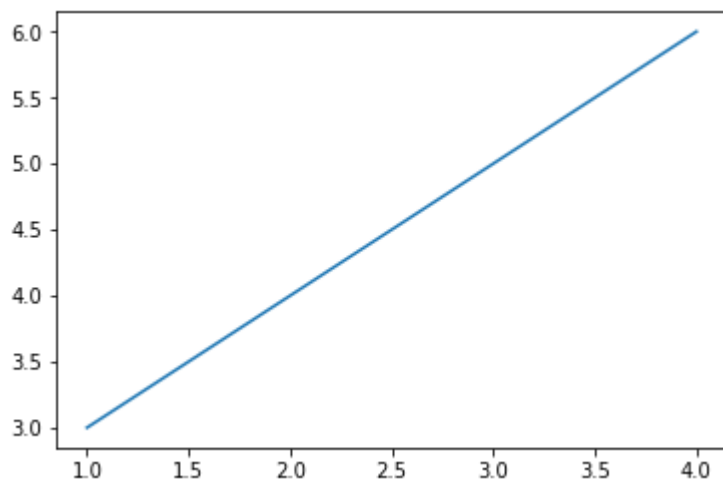
text : :class:`~matplotlib.text.Text`
The matplotlib text instance representing the title

Other Parameters

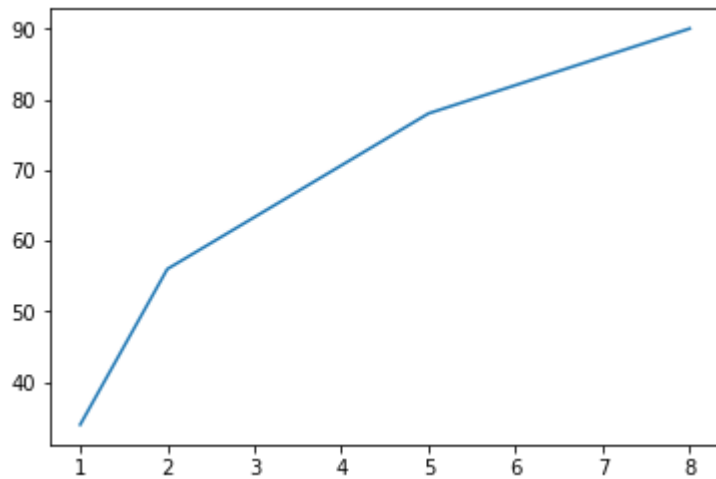
**kwargs : `~matplotlib.text.Text` properties
Other keyword arguments are text properties, see :class:`~matplotlib.text.Text` for a list of valid text properties.

Line plot

```
In [6]: 1 x = [1,2,3,4]
        2 y = [3,4,5,6]
        3 plt.plot(x,y)
        4 plt.show()
```



```
In [7]: 1 import numpy as np
        2 a = np.array([1,2,5,8])
        3 b = np.array([34,56,78,90])
        4 plt.plot(a,b)
        5 plt.show()
```



In [8]: 1 `help(plt.plot)`

Help on function plot in module matplotlib.pyplot:

`plot(*args, scalex=True, scaley=True, data=None, **kwargs)`
 Plot y versus x as lines and/or markers.

Call signatures::

```
plot([x], y, [fmt], data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
```

The coordinates of the points or line nodes are given by `*x*`, `*y*`.

The optional parameter `*fmt*` is a convenient way for defining basic formatting like color, marker and linestyle. It's a shortcut string notation described in the `*Notes*` section below.

```
>>> plot(x, y)          # plot x and y using default line style and col
or
>>> plot(x, y, 'bo')    # plot x and y using blue circle markers
>>> plot(y)             # plot y using x as index array 0..N-1
>>> plot(y, 'r+')       # ditto, but with red plusses
```

You can use ``.Line2D`` properties as keyword arguments for more control on the appearance. Line properties and `*fmt*` can be mixed. The following two calls yield identical results:

```
>>> plot(x, y, 'go--', linewidth=2, markersize=12)
>>> plot(x, y, color='green', marker='o', linestyle='dashed',
...       linewidth=2, markersize=12)
```

When conflicting with `*fmt*`, keyword arguments take precedence.

****Plotting labelled data****

There's a convenient way for plotting objects with labelled data (i.e.

data that can be accessed by index ```obj['y']```). Instead of giving the data in `*x*` and `*y*`, you can provide the object in the `*data*` parameter and just give the labels for `*x*` and `*y*::`

```
>>> plot('xlabel', 'ylabel', data=obj)
```

All indexable objects are supported. This could e.g. be a ``dict``, a ``pandas.DataFrame`` or a structured numpy array.

****Plotting multiple sets of data****

There are various ways to plot multiple sets of data.

- The most straight forward way is just to call ``plot`` multiple times.

Example:

```
>>> plot(x1, y1, 'bo')
>>> plot(x2, y2, 'go')
```

- Alternatively, if your data is already a 2d array, you can pass it directly to `*x*`, `*y*`. A separate data set will be drawn for every column.

Example: an array ```a``` where the first column represents the `*x*` values and the other columns are the `*y*` columns::

```
>>> plot(a[0], a[1:])
```

- The third way is to specify multiple sets of `*[x]*`, `*y*`, `*[fmt]*` groups::

```
>>> plot(x1, y1, 'g^', x2, y2, 'g-')
```

In this case, any additional keyword argument applies to all datasets. Also this syntax cannot be combined with the `*data*` parameter.

By default, each line is assigned a different style specified by a 'style cycle'. The `*fmt*` and line property parameters are only necessary if you want explicit deviations from these defaults. Alternatively, you can also change the style cycle using the 'axes.prop_cycle' rcParam.

Parameters

`x, y` : array-like or scalar

The horizontal / vertical coordinates of the data points. `*x*` values are optional. If not given, they default to ```[0, ..., N-1]```.

Commonly, these parameters are arrays of length `N`. However, scalars are supported as well (equivalent to an array with constant value).

The parameters can also be 2-dimensional. Then, the columns represent separate data sets.

`fmt` : str, optional

A format string, e.g. 'ro' for red circles. See the `*Notes*` section for a full description of the format strings.

Format strings are just an abbreviation for quickly setting basic line properties. All of these and more can also be controlled by keyword arguments.

`data` : indexable object, optional

An object with labelled data. If given, provide the label names to

o

plot in `*x*` and `*y*`.

.. note::

Technically there's a slight ambiguity in calls where the second label is a valid `*fmt*`. `plot('n', 'o', data=obj)``

could be ``plt(x, y)`` or ``plt(y, fmt)``. In such cases, the former interpretation is chosen, but a warning is issued. You may suppress the warning by adding an empty format string ``plot('n', 'o', '', data=obj)``.

Other Parameters

`scalex, scaley : bool, optional, default: True`

These parameters determined if the view limits are adapted to the data limits. The values are passed on to ``autoscale_view``.

`**kwargs : `.Line2D` properties, optional`

`*kwargs*` are used to specify properties like a line label (for auto legends), linewidth, antialiasing, marker face color.

Example::

```
>>> plot([1,2,3], [1,2,3], 'go-', label='line 1', linewidth=2)
>>> plot([1,2,3], [1,4,9], 'rs', label='line 2')
```

If you make multiple lines with one plot command, the kwargs apply to all those lines.

Here is a list of available ``Line2D`` properties:

```
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
animated: bool
antialiased: bool
clip_box: `.Bbox`
clip_on: bool
clip_path: [(`~matplotlib.path.Path`, `.Transform`) | `.Patch` | None]
color: color
contains: callable
dash_capstyle: {'butt', 'round', 'projecting'}
dash_joinstyle: {'miter', 'round', 'bevel'}
dashes: sequence of floats (on/off ink in points) or (None, None)
drawstyle: {'default', 'steps', 'steps-pre', 'steps-mid', 'steps-post'}
figure: `.Figure`
fillstyle: {'full', 'left', 'right', 'bottom', 'top', 'none'}
gid: str
in_layout: bool
label: object
linestyle: {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
linewidth: float
marker: unknown
markeredgecolor: color
markeredgewidth: float
markerfacecolor: color
markerfacecoloralt: color
markersize: float
markevery: unknown
path_effects: `.AbstractPathEffect`
picker: float or callable[[Artist, Event], Tuple[bool, dict]]
```

```

pickradius: float
rasterized: bool or None
sketch_params: (scale: float, length: float, randomness: float)
snap: bool or None
solid_capstyle: {'butt', 'round', 'projecting'}
solid_joinstyle: {'miter', 'round', 'bevel'}
transform: matplotlib.transforms.Transform
url: str
visible: bool
xdata: 1D array
ydata: 1D array
zorder: float

```

Returns

lines

A list of `.Line2D`` objects representing the plotted data.

See Also

`scatter` : XY scatter plot with markers of varying size and/or color (sometimes also called bubble chart).

Notes

****Format Strings****

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

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

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.

****Colors****

The following color abbreviations are supported:

| ===== | ===== |
|----------------------|---------|
| character | color |
| ===== | ===== |
| <code>``'b'``</code> | blue |
| <code>``'g'``</code> | green |
| <code>``'r'``</code> | red |
| <code>``'c'``</code> | cyan |
| <code>``'m'``</code> | magenta |
| <code>``'y'``</code> | yellow |
| <code>``'k'``</code> | black |
| <code>``'w'``</code> | white |
| ===== | ===== |

If the color is the only part of the format string, you can additionally use any ``matplotlib.colors`` spec, e.g. full names (```'green'```) or hex strings (```'#008000'```).

****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 |

Example format strings::

```
'b'      # blue markers with default shape
'ro'     # red circles
'g-'     # green solid line
'--'     # dashed line with default color
'k^:'    # black triangle_up markers connected by a dotted line
```

.. note::

In addition to the above described arguments, this function can take a

****data**** keyword argument. If such a ****data**** argument is given, the

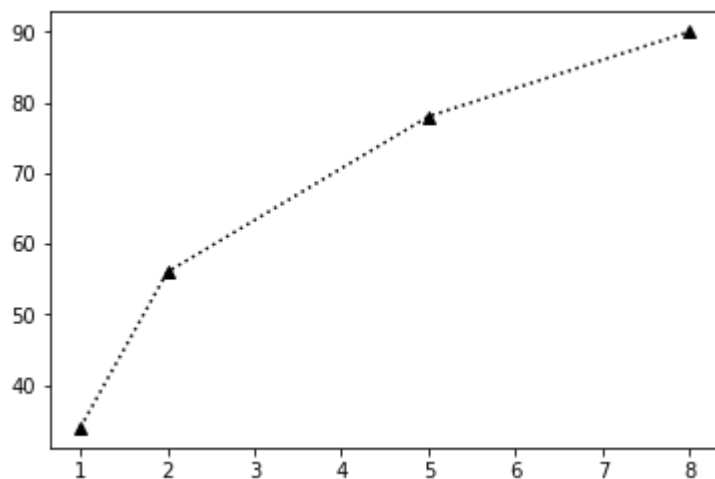
following arguments are replaced by ****data[<arg>]****:

* All arguments with the following names: 'x', 'y'.

Objects passed as ****data**** must support item access (``data[<arg>]``) and membership test (``<arg> in data``).

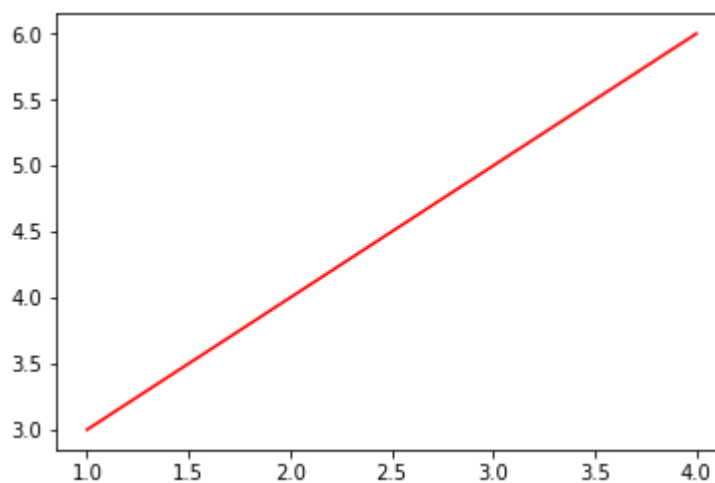
```
In [9]: 1 plt.plot(a,b,"k^:")
```

```
Out[9]: [matplotlib.lines.Line2D at 0x2818ca3eb38]
```

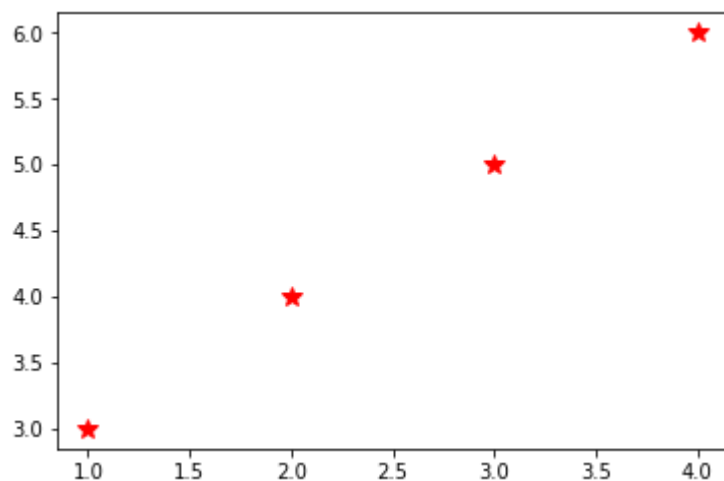


```
In [10]: 1 plt.plot(x,y,color="r")
```

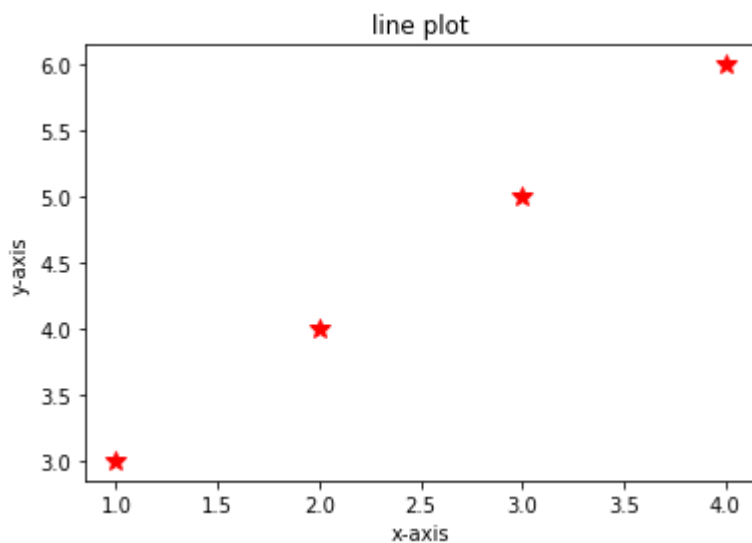
```
Out[10]: [matplotlib.lines.Line2D at 0x2818cac72e8]
```



```
In [15]: 1 plt.plot(x,y,"r*",markersize=10)
2 plt.show()
```



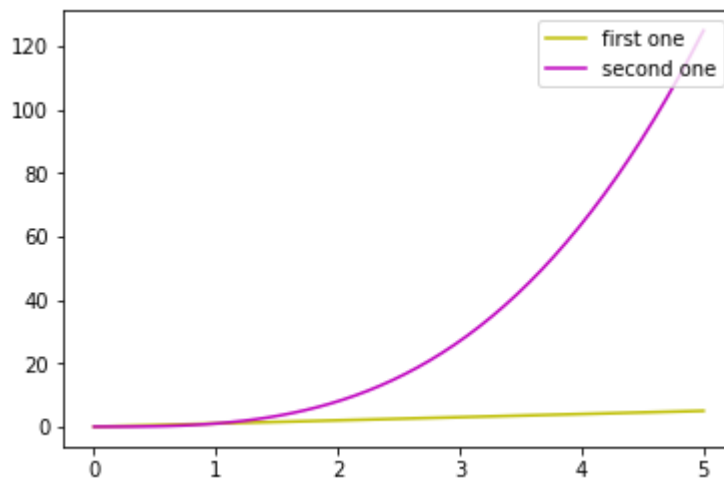
```
In [14]: 1 plt.plot(x,y,"r*",markersize=10)
2 plt.title("line plot")
3 plt.xlabel("x-axis")
4 plt.ylabel("y-axis")
5 plt.show()
```



```
In [16]: 1 x = np.linspace(0,5,50)
          2 x
```

```
Out[16]: array([0.          , 0.10204082, 0.20408163, 0.30612245, 0.40816327,
                0.51020408, 0.6122449 , 0.71428571, 0.81632653, 0.91836735,
                1.02040816, 1.12244898, 1.2244898 , 1.32653061, 1.42857143,
                1.53061224, 1.63265306, 1.73469388, 1.83673469, 1.93877551,
                2.04081633, 2.14285714, 2.24489796, 2.34693878, 2.44897959,
                2.55102041, 2.65306122, 2.75510204, 2.85714286, 2.95918367,
                3.06122449, 3.16326531, 3.26530612, 3.36734694, 3.46938776,
                3.57142857, 3.67346939, 3.7755102 , 3.87755102, 3.97959184,
                4.08163265, 4.18367347, 4.28571429, 4.3877551 , 4.48979592,
                4.59183673, 4.69387755, 4.79591837, 4.89795918, 5.          ])
```

```
In [23]: 1 plt.plot(x,x,"y",label = "first one")
          2 plt.plot(x,x**3,"m",label = "second one")
          3 #plt.legend(loc = 3)
          4 plt.legend(loc = 'upper right' )
          5 plt.show()
```



```
In [19]: 1 help(plt.legend)
```

Help on function legend in module matplotlib.pyplot:

```
legend(*args, **kwargs)
    Place a legend on the axes.
```

Call signatures::

```
    legend()
    legend(labels)
    legend(handles, labels)
```

The call signatures correspond to three different ways how to use this method.

****1. Automatic detection of elements to be shown in the legend****

The elements to be added to the legend are automatically determined, when you do not pass in any extra arguments.

In this case, the labels are taken from the artist. You can specify them either at artist creation or by calling the :meth:`~Artist.set_label` method on the artist::

```
    line, = ax.plot([1, 2, 3], label='Inline label')
    ax.legend()
```

or::

```
    line.set_label('Label via method')
    line, = ax.plot([1, 2, 3])
    ax.legend()
```

Specific lines can be excluded from the automatic legend element selection by defining a label starting with an underscore. This is default for all artists, so calling `Axes.legend` without any arguments and without setting the labels manually will result in no legend being drawn.

****2. Labeling existing plot elements****

To make a legend for lines which already exist on the axes (via plot for instance), simply call this function with an iterable of strings, one for each legend item. For example::

```
    ax.plot([1, 2, 3])
    ax.legend(['A simple line'])
```

Note: This way of using is discouraged, because the relation between plot elements and labels is only implicit by their order and can easily be mixed up.

****3. Explicitly defining the elements in the legend****

For full control of which artists have a legend entry, it is possible to pass an iterable of legend artists followed by an iterable of legend labels respectively::

```
legend((line1, line2, line3), ('label1', 'label2', 'label3'))
```

Parameters

handles : sequence of `.Artist``, optional

A list of Artists (lines, patches) to be added to the legend.

Use this together with `*labels*`, if you need full control on what is shown in the legend and the automatic mechanism described above is not sufficient.

The length of handles and labels should be the same in this case. If they are not, they are truncated to the smaller length.

labels : sequence of strings, optional

A list of labels to show next to the artists.

Use this together with `*handles*`, if you need full control on what is shown in the legend and the automatic mechanism described above is not sufficient.

Other Parameters

loc : int or string or pair of floats, default: `:rc:`legend.loc`` ('best' for axes, 'upper right' for figures)

The location of the legend. Possible codes are:

| ===== | ===== |
|-----------------|---------------|
| Location String | Location Code |
| ===== | ===== |
| 'best' | 0 |
| 'upper right' | 1 |
| 'upper left' | 2 |
| 'lower left' | 3 |
| 'lower right' | 4 |
| 'right' | 5 |
| 'center left' | 6 |
| 'center right' | 7 |
| 'lower center' | 8 |
| 'upper center' | 9 |
| 'center' | 10 |
| ===== | ===== |

Alternatively can be a 2-tuple giving ``x, y`` of the lower-left corner of the legend in axes coordinates (in which case ``bbox_to_anchor`` will be ignored).

The 'best' option can be quite slow for plots with large amounts of data. Your plotting speed may benefit from providing a specific location.

`bbox_to_anchor` : ``BboxBase``, 2-tuple, or 4-tuple of floats
 Box that is used to position the legend in conjunction with `*loc*`.
 Defaults to ``axes.bbox`` (if called as a method to ``Axes.legend``) or
``figure.bbox`` (if ``Figure.legend``). This argument allows arbitrary
 placement of the legend.

Bbox coordinates are interpreted in the coordinate system given by
``bbox_transform``, with the default transform
 Axes or Figure coordinates, depending on which ``legend`` is called.

If a 4-tuple or ``BboxBase`` is given, then it specifies the bbox
``(x, y, width, height)`` that the legend is placed in.
 To put the legend in the best location in the bottom right
 quadrant of the axes (or figure)::

```
loc='best', bbox_to_anchor=(0.5, 0., 0.5, 0.5)
```

A 2-tuple ``(x, y)`` places the corner of the legend specified by `*loc*` at
`x, y`. For example, to put the legend's upper right-hand corner in the
 center of the axes (or figure) the following keywords can be used::

```
loc='upper right', bbox_to_anchor=(0.5, 0.5)
```

`ncol` : integer
 The number of columns that the legend has. Default is 1.

`prop` : None or `:class:`matplotlib.font_manager.FontProperties`` or dict
 The font properties of the legend. If None (default), the current
`:data:`matplotlib.rcParams`` will be used.

`fontsize` : int or float or `{'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'}`
 Controls the font size of the legend. If the value is numeric the
 size will be the absolute font size in points. String values are
 relative to the current default font size. This argument is only
 used if ``prop`` is not specified.

`numpoints` : None or int
 The number of marker points in the legend when creating a legend
 entry for a ``Line2D`` (line).
 Default is ``None``, which will take the value from
`:rc:`legend.numpoints``.

`scatterpoints` : None or int
 The number of marker points in the legend when creating
 a legend entry for a ``PathCollection`` (scatter plot).
 Default is ``None``, which will take the value from
`:rc:`legend.scatterpoints``.

`scatteryoffsets` : iterable of floats
 The vertical offset (relative to the font size) for the markers

created for a scatter plot legend entry. 0.0 is at the base the legend text, and 1.0 is at the top. To draw all markers at the same height, set to ```[0.5]```. Default is ```[0.375, 0.5, 0.3125]```.

markerscale : None or int or float

The relative size of legend markers compared with the originally drawn ones.

Default is ```None```, which will take the value from `:rc:`legend.markerscale``.

markerfirst : bool

If `*True*`, legend marker is placed to the left of the legend label. If `*False*`, legend marker is placed to the right of the legend label.

Default is `*True*`.

frameon : None or bool

Control whether the legend should be drawn on a patch (frame).

Default is ```None```, which will take the value from `:rc:`legend.frameon``.

fancybox : None or bool

Control whether round edges should be enabled around the `:class:`~matplotlib.patches.FancyBboxPatch`` which makes up the legend's background.

Default is ```None```, which will take the value from `:rc:`legend.fancybox``.

shadow : None or bool

Control whether to draw a shadow behind the legend.

Default is ```None```, which will take the value from `:rc:`legend.shadow``.

framealpha : None or float

Control the alpha transparency of the legend's background.

Default is ```None```, which will take the value from `:rc:`legend.framealpha``. If shadow is activated and `*framealpha*` is ```None```, the default value is ignored.

facecolor : None or "inherit" or a color spec

Control the legend's background color.

Default is ```None```, which will take the value from `:rc:`legend.facecolor``. If ```"inherit"```, it will take `:rc:`axes.facecolor``.

edgecolor : None or "inherit" or a color spec

Control the legend's background patch edge color.

Default is ```None```, which will take the value from `:rc:`legend.edgecolor``. If ```"inherit"```, it will take `:rc:`axes.edgecolor``.

mode : {"expand", None}

If ``mode`` is set to ```"expand"``` the legend will be horizontally expanded to fill the axes area (or ``bbox_to_anchor`` if defines the legend's size).

`bbox_transform` : None or :class:`matplotlib.transforms.Transform`
 The transform for the bounding box (``bbox_to_anchor``). For a value of ``None`` (default) the Axes' :data:`~matplotlib.axes.Axes.transAxes` transform will be used.

`title` : str or None
 The legend's title. Default is no title (``None``).

`title_fontsize`: str or None
 The fontsize of the legend's title. Default is the default fontsize.

e.

`borderpad` : float or None
 The fractional whitespace inside the legend border. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.borderpad`.

`labelspacing` : float or None
 The vertical space between the legend entries. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.labelspacing`.

`handlelength` : float or None
 The length of the legend handles. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.handlelength`.

`handletextpad` : float or None
 The pad between the legend handle and text. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.handletextpad`.

`borderaxespad` : float or None
 The pad between the axes and legend border. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.borderaxespad`.

`columnspacing` : float or None
 The spacing between columns. Measured in font-size units. Default is ``None``, which will take the value from :rc:`legend.columnspacing`.

`handler_map` : dict or None
 The custom dictionary mapping instances or types to a legend handler. This ``handler_map`` updates the default handler map found at :func:`~matplotlib.legend.Legend.get_legend_handler_map`.

Returns

```
:class:`matplotlib.legend.Legend` instance
```

Notes

Not all kinds of artist are supported by the legend command. See `:doc:`/tutorials/intermediate/legend_guide`` for details.

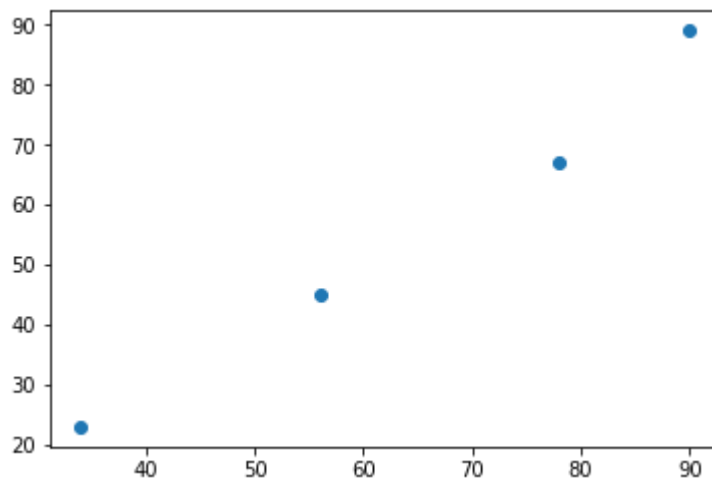
Examples

```
.. plot:: gallery/text_labels_and_annotations/legend.py
```

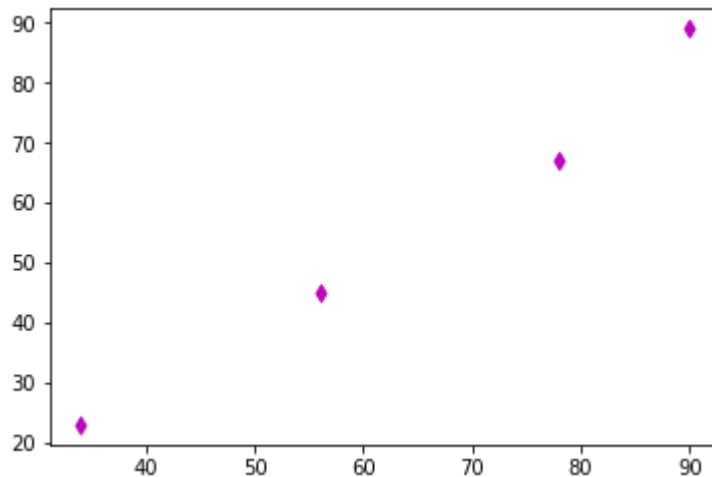
Scatter plot

- Display the data in points

```
In [26]: 1 x = np.array([34,56,78,90])  
2 y = np.array([23,45,67,89])  
3 plt.scatter(x,y)  
4 plt.show()
```



```
In [30]: 1 plt.scatter(x,y,color="m",marker="d")  
2 plt.show()
```



```
In [32]: 1 x = np.linspace(1,20,30)
```

```
In [33]: 1 x
```

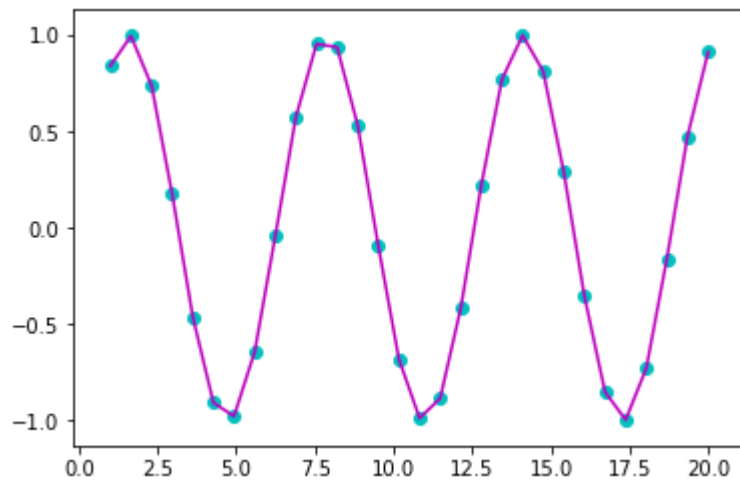
```
Out[33]: array([ 1.          ,  1.65517241,  2.31034483,  2.96551724,  3.62068966,  
                4.27586207,  4.93103448,  5.5862069 ,  6.24137931,  6.89655172,  
                7.55172414,  8.20689655,  8.86206897,  9.51724138, 10.17241379,  
                10.82758621, 11.48275862, 12.13793103, 12.79310345, 13.44827586,  
                14.10344828, 14.75862069, 15.4137931 , 16.06896552, 16.72413793,  
                17.37931034, 18.03448276, 18.68965517, 19.34482759, 20.          ])
```

```
In [34]: 1 y = np.sin(x)
```

```
In [35]: 1 y
```

```
Out[35]: array([ 0.84147098,  0.99644245,  0.73877292,  0.17516702, -0.46097803,  
                -0.90622553, -0.97619215, -0.64190371, -0.04179382,  0.57562349,  
                 0.95466692,  0.9383692 ,  0.53347946, -0.09233172, -0.67990702,  
                -0.98592315, -0.88365449, -0.41545193,  0.22479518,  0.77195141,  
                 0.99943158,  0.81303292,  0.28994576, -0.35321206, -0.85009974,  
                -0.99494903, -0.72777578, -0.15922022,  0.47527069,  0.91294525])
```

```
In [37]: 1 plt.plot(x,y,"m")  
2 plt.scatter(x,y,color = "c")  
3 plt.show()
```

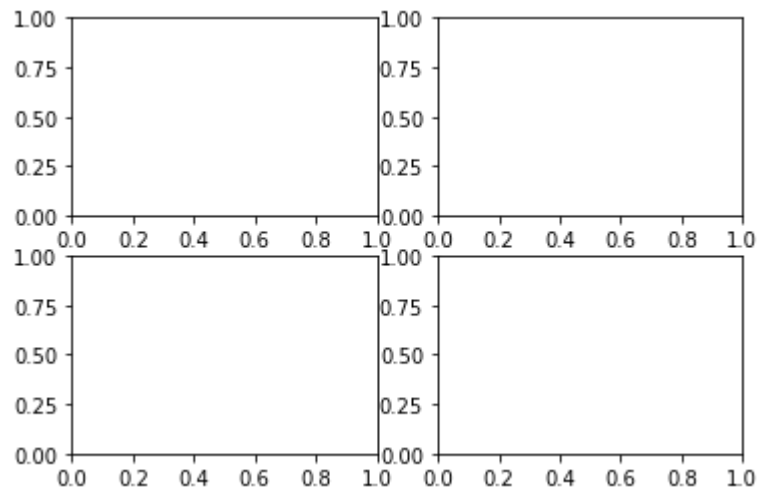


Subplots

- syntax : `plt.subplots(row,column,position)`

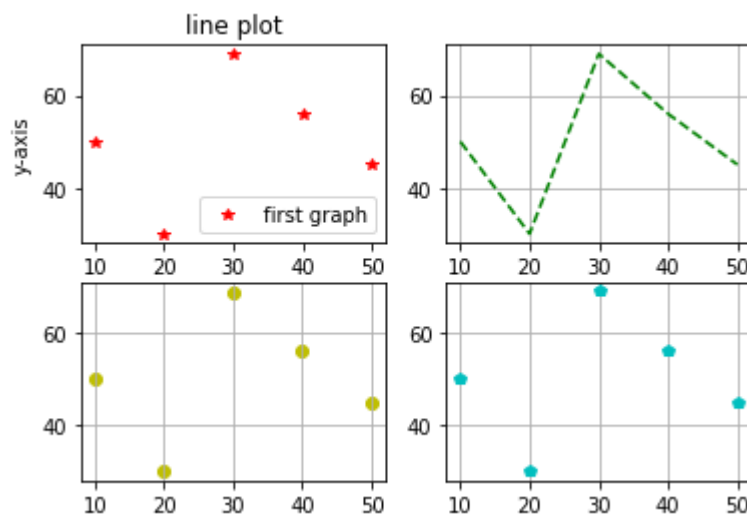
In [38]:

```
1 plt.subplots(2,2)
2 plt.show()
```



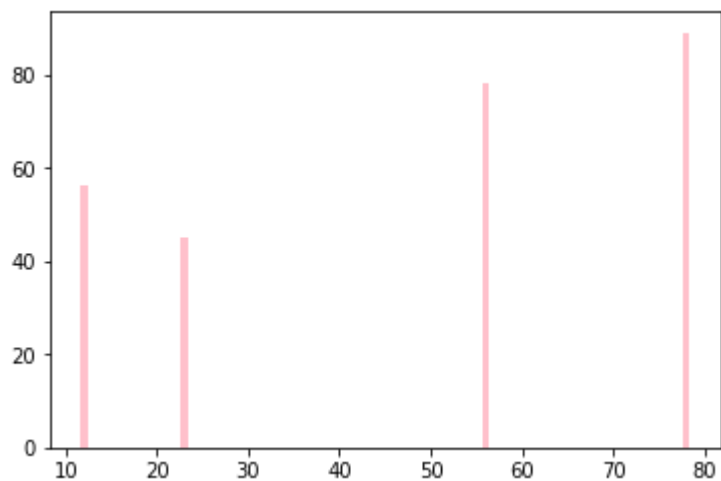
In [50]:

```
1 x = np.array([10,20,30,40,50])
2 y = np.array([50,30,69,56,45])
3 plt.subplot(2,2,1)# 2 rows , 2 columns and 1st position
4 plt.plot(x,y,"r*",label = "first graph")
5 plt.legend()
6 plt.title("line plot")
7 plt.xlabel("x-axis")
8 plt.ylabel("y-axis")
9 plt.subplot(2,2,2)
10 plt.plot(x,y,"g--")
11 plt.grid()
12 plt.subplot(2,2,3)
13 plt.scatter(x,y,color="y")
14 plt.grid()
15 plt.subplot(2,2,4)
16 plt.plot(x,y,"cp")
17 plt.grid()
```

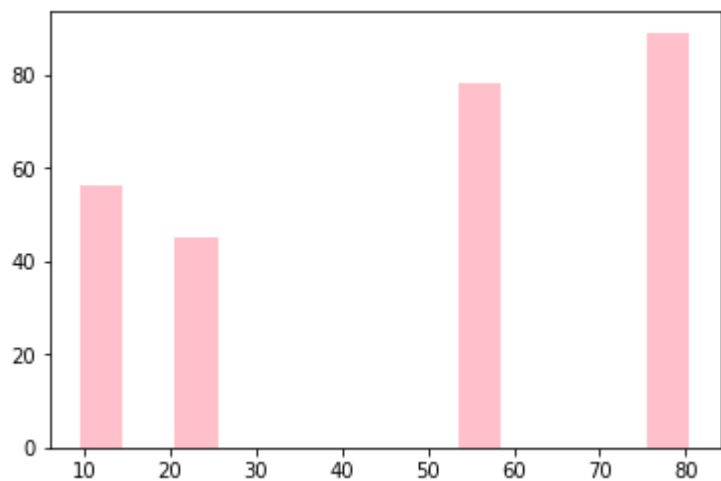


Bar graphs

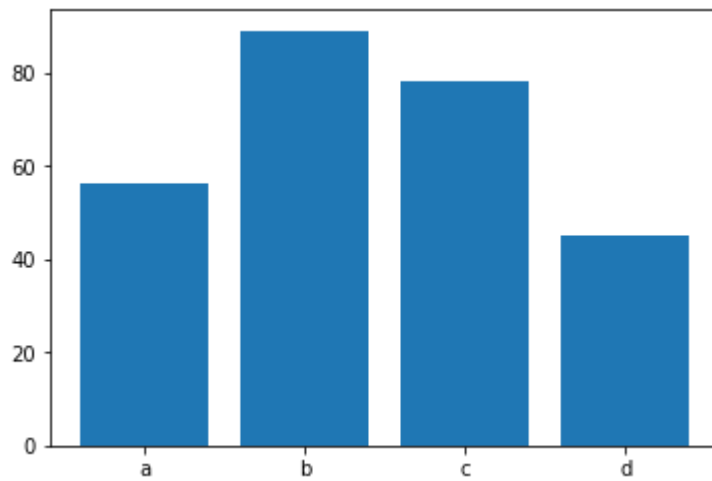
```
In [53]: 1 x = np.array([12,23,56,78])  
2 y = np.array([56,45,78,89])  
3 plt.bar(x,y,color = "pink")  
4 plt.show()
```



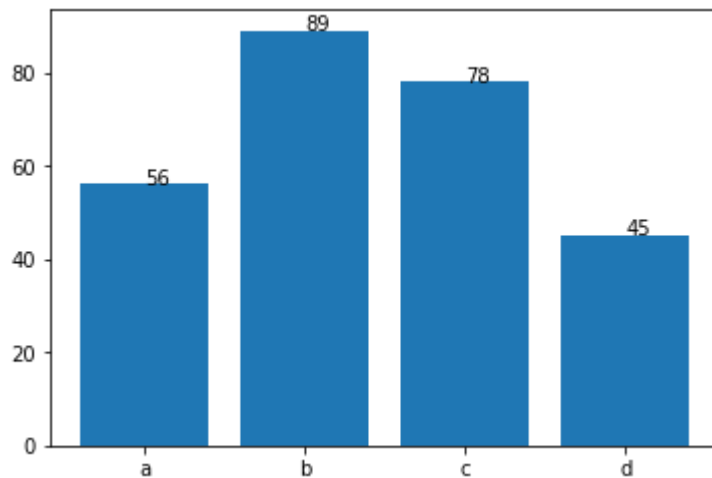
```
In [54]: 1 x = np.array([12,23,56,78])  
2 y = np.array([56,45,78,89])  
3 plt.bar(x,y,color = "pink",width = 5)  
4 plt.show()
```



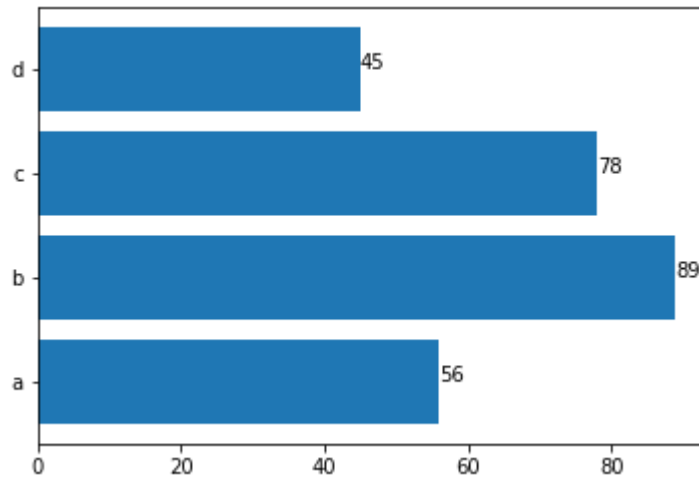
```
In [55]: 1 x = ["a", "b", "c", "d"]  
2 y = [56, 89, 78, 45]  
3 plt.bar(x, y)  
4 plt.show()
```



```
In [61]: 1 x = ["a", "b", "c", "d"]  
2 y = [56, 89, 78, 45]  
3 plt.bar(x, y)  
4 for a, b in zip(x, y):  
5     plt.text(a, b, str(b))  
6 plt.show()
```



```
In [65]: 1 x = ["a", "b", "c", "d"]  
2 y = [56, 89, 78, 45]  
3 plt.barh(x, y)  
4 for a, b in zip(y, x):  
5     plt.text(a, b, str(a))  
6 plt.show()
```



Hist plot


```
In [66]: 1 x = np.random.randn(1000)
        2 x
```

```
Out[66]: array([ 1.89737825e-01, -9.40183479e-01,  2.05466501e+00, -3.24997685e-01,
  8.04017649e-01, -7.08489537e-03,  9.34743149e-01,  3.18803888e-01,
 -1.61491471e+00, -9.97430990e-01,  7.52985379e-01, -1.51284962e-01,
 -1.28211619e+00, -7.44484003e-01,  6.75987211e-01, -1.62859369e+00,
  1.40845141e-01,  2.06877911e+00, -7.02125300e-02,  1.40042191e+00,
  1.14423683e-01, -3.67378399e-01, -5.20354458e-01, -8.63605240e-01,
  2.50565639e-01, -4.21883867e-01, -3.95000526e-01,  1.47436073e+00,
  4.29642565e-01, -7.27300909e-01,  6.95660484e-02, -3.78323695e-01,
 -6.94107741e-01,  7.97690698e-01, -1.41430630e+00,  6.94069064e-01,
  1.85709717e+00, -5.64195600e-01, -9.32490791e-02,  3.10390366e-01,
  7.22527336e-01,  7.19026872e-01, -5.83871874e-01,  1.17346159e+00,
 -6.91884792e-01, -1.33304145e+00, -1.10913007e+00,  1.03842947e-01,
  3.38740713e-01,  2.21582643e-01,  1.18113960e-01, -2.97260956e-01,
 -5.09952640e-01, -5.05333855e-01, -4.35709503e-01,  8.26249482e-02,
  1.14256105e+00,  1.73985878e+00,  1.27801818e-01,  7.67101014e-01,
 -5.33767747e-01, -1.26811659e+00, -1.56972635e+00, -6.69977972e-01,
 -5.66998699e-01,  3.47568970e-01,  6.74557062e-01,  5.90489702e-01,
  4.63987574e-01, -8.56914376e-01,  2.81034958e-01, -1.29062492e+00,
  5.19392877e-01,  1.92429847e-01,  1.13727891e+00,  2.44620772e+00,
  2.82973009e-01,  1.09746606e+00,  1.13890963e+00, -2.56174819e+00,
  5.84349706e-01, -2.60769957e+00,  1.28311684e+00, -1.37558493e-01,
 -7.39453693e-01,  1.12292810e+00, -4.97375373e-01, -1.24114989e+00,
  3.22414150e-01, -9.62772241e-01, -8.10856466e-01, -6.19463528e-01,
  1.01320972e+00,  1.49274108e-01,  2.43893302e-01,  7.54412615e-01,
 -1.03813716e-01, -7.09778258e-01,  1.14045938e+00, -4.41092028e-01,
  2.41789617e+00,  7.97580769e-01,  3.59172338e-01,  1.21701534e+00,
 -1.02736493e-01,  4.75984378e-01, -9.97367854e-01, -8.45928410e-01,
 -2.51932760e+00,  4.74328928e-01, -1.05759823e+00,  3.67026235e-01,
 -7.90195796e-01,  7.74621363e-01, -1.29486285e+00,  1.08163964e-01,
 -1.72870648e+00,  1.44983353e-01, -1.12377068e+00, -1.40552024e+00,
  1.02730566e+00, -1.68196666e+00, -1.56319991e-01,  1.22674175e+00,
  1.80330249e-01,  8.65627784e-01, -9.60911034e-03, -3.04562473e-01,
  4.18996177e-01, -2.62196984e-01,  2.10184229e+00,  4.82092440e-01,
  5.67610859e-01,  2.11963936e-01, -7.63690013e-01,  5.66184570e-01,
  6.39233912e-02, -1.05272312e-01,  3.70619441e-01, -6.32741672e-01,
  8.05795790e-01,  2.11810140e+00,  7.22731166e-01,  6.55861899e-01,
  4.90084001e-01,  1.44071819e-01, -1.88501117e+00,  1.44241921e+00,
  6.73830822e-01, -9.89154066e-01,  2.26721973e-01, -1.15939426e+00,
  4.93046315e-01, -1.68927117e+00,  4.06589185e-01,  9.85479971e-01,
 -1.63176602e-01, -1.09138652e+00, -3.13818234e-01,  1.62658555e+00,
  1.56776319e-01, -2.68443143e+00, -1.48614028e+00, -3.49553103e-01,
  1.28128446e-02,  1.27843232e+00, -1.59656788e-01,  5.98213510e-01,
 -6.53302750e-01,  6.63667282e-01, -1.97808337e-02, -9.52352070e-02,
  4.08612836e-01,  4.69262830e-01,  1.59949817e+00,  2.30014437e+00,
  4.37321274e-01,  4.99201099e-02,  4.78512753e-01, -9.20856364e-01,
 -3.47566981e-02,  1.30185039e+00, -9.39863473e-01, -6.68418282e-01,
 -1.61432710e+00,  9.84113876e-01,  1.11615670e+00,  1.40286437e-02,
 -8.40567030e-01, -3.68941694e-01, -8.53874128e-02, -3.68503842e-01,
 -1.60292967e+00, -1.24000752e+00,  4.04807001e-01, -1.52455781e+00,
 -1.78575072e+00, -9.77450976e-01,  5.22870729e-01,  3.85509259e-01,
  8.03649816e-01,  3.09834556e-01, -7.46851311e-01, -1.14583735e+00,
  1.73246821e+00,  4.22170516e-01,  2.15424239e-01, -9.90846892e-01,
  1.14526802e-01,  1.01289419e+00, -4.88306667e-01, -4.87681735e-01,
```

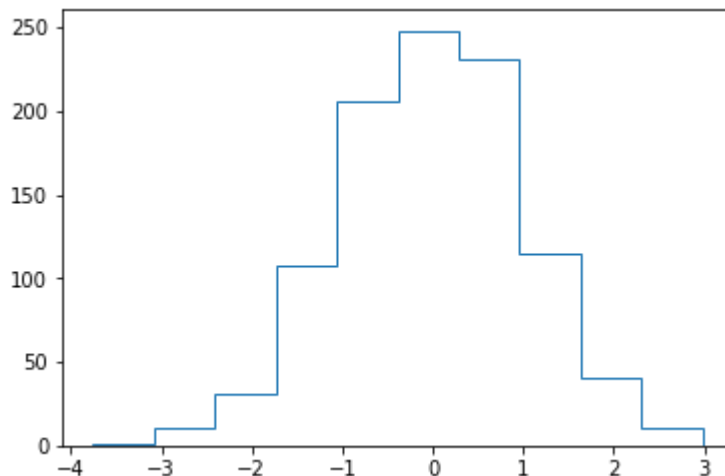
```
-9.94214797e-01, 5.19272096e-01, -2.11784253e+00, -3.99686984e-01,
1.23792911e+00, 6.15219379e-01, -1.15043412e+00, -8.27476819e-01,
1.54876982e+00, -8.15657218e-01, 4.78253102e-01, -8.25033579e-01,
9.33565020e-01, 1.26252702e+00, 1.38369772e+00, -1.14344706e+00,
1.21908211e+00, -1.34765951e+00, 4.35105048e-01, 7.60934283e-03,
-4.24760019e-01, -9.16094054e-01, 6.01445648e-01, -2.86852579e+00,
-1.10847427e+00, -4.27920011e-01, -2.77909019e-01, -1.05986765e+00,
-3.75611119e-01, 6.76423209e-02, 6.00549469e-01, -9.84263597e-01,
-1.61101263e+00, 1.35930915e+00, 8.09951141e-01, -9.16310312e-01,
-1.41041169e+00, -1.07428951e-01, 6.40619074e-01, 3.60063384e-01,
-4.73822899e-02, -5.34148697e-01, -1.50685102e-01, -1.38978733e+00,
4.54697634e-01, -7.09821424e-03, -1.72928936e-01, 5.57387873e-01,
1.32160434e+00, 5.87790385e-02, 1.91526315e+00, -1.70931626e+00,
-1.46844130e-01, 5.34040052e-02, -1.59069335e+00, 3.63829347e-01,
-1.55029165e+00, -1.08812205e+00, -9.53285808e-01, 2.88359922e-01,
2.51615905e-02, 5.95623039e-01, 8.64169148e-01, -7.47065205e-01,
-1.36954175e+00, 1.36617742e-01, -1.10390376e+00, -8.23025323e-01,
1.12194358e+00, -5.65538511e-01, 9.20526402e-01, 1.21301173e+00,
-5.74200372e-01, 1.61029682e-01, -3.75590920e+00, 8.83725826e-01,
-9.74703316e-02, -3.39958601e-01, 1.55250648e+00, 1.74363531e-01,
-1.42500755e+00, 2.44641175e+00, -2.69674558e-01, -1.35846999e+00,
-4.12273484e-01, -6.60823309e-02, 3.71152587e-01, -3.14543021e-01,
-3.43736091e-01, 1.52800267e-02, 1.25489078e+00, 1.04549249e+00,
1.50860572e+00, 1.27503095e+00, -1.62772347e-01, 5.85431978e-01,
-2.01142238e-01, -1.38817027e+00, 1.00344361e+00, -1.74669892e+00,
2.25499765e+00, 2.83262650e-01, 4.10243701e-01, 3.67151703e-01,
-5.68235249e-02, 5.27806924e-01, -1.74347244e-01, 6.67171447e-01,
-1.32810198e+00, 7.03103751e-01, -1.24212581e+00, -2.45997391e+00,
-6.50319822e-01, -7.04641872e-01, 9.58200096e-02, -3.66011593e-01,
2.44503694e+00, 5.22170917e-01, 1.38078552e-01, 7.63034328e-01,
2.89240440e-01, -1.86572911e+00, 4.10533519e-01, 3.84266031e-01,
-4.89652827e-01, 4.62078286e-01, -4.16887400e-01, -4.35466657e-01,
-1.20469544e-01, -9.88920515e-02, 7.86759769e-01, 1.75453072e+00,
8.96969120e-02, 1.49734160e+00, 1.62116003e-01, -1.62466435e+00,
-3.15523502e-01, 2.04149455e-01, -3.85796023e-01, -1.43086819e-01,
1.48098924e-01, 7.35596512e-01, 6.48849689e-01, 1.37584012e+00,
4.27155674e-01, -7.03306296e-01, -7.58612834e-01, 3.60131464e-01,
-1.00409287e+00, 9.32809334e-01, 4.45757023e-01, 9.18022560e-01,
-4.58289536e-01, -5.93197341e-01, -6.74480727e-01, 3.73424207e-01,
2.80688870e+00, -1.54906290e-01, -1.98594990e+00, 5.43590093e-01,
4.89340207e-01, 4.14116779e-01, 4.53899747e-01, -8.75736128e-01,
-1.06939510e+00, 4.81368550e-01, 7.75006253e-01, -1.01942569e+00,
-3.16208552e-01, -6.61025193e-01, -6.78053478e-02, 2.78665721e+00,
8.47788416e-02, -8.81176885e-01, 6.01571615e-01, 6.32872614e-01,
1.03509976e+00, -4.09765160e-01, 1.08026316e+00, -9.14002728e-01,
-2.05767986e-01, -1.21377788e+00, 7.85689690e-01, -4.22045243e-01,
2.76196143e-01, 3.10090026e-01, -1.50573458e-01, -6.63729664e-01,
-4.29090504e-01, 5.23120688e-01, -8.81175964e-01, 2.27828914e-01,
-1.45416970e+00, 4.92309633e-01, -4.70945790e-01, 4.76749970e-02,
4.83931825e-01, 1.87539381e+00, 1.33472783e-01, 1.75028350e-01,
-1.92261709e-01, -9.14324759e-01, -2.42049499e-01, 1.15522846e-01,
-1.78445364e-01, -5.16831479e-01, -7.26528580e-01, 2.88228794e-01,
-1.49529464e+00, -4.38630232e-01, 7.16286833e-01, 9.12811176e-01,
-3.70549025e-01, 4.67887471e-01, -1.74264327e+00, 5.62713554e-01,
8.39932570e-02, -1.96071872e+00, -7.22038004e-01, 2.20705913e+00,
1.74469083e+00, -1.42110835e+00, -1.54762417e-01, 5.66156930e-01,
1.76231682e+00, -1.44322622e+00, -2.70267453e-01, -1.01926495e+00,
```

```
-1.58497510e+00, -4.79901915e-01, -1.38447656e+00, 9.39839356e-01,
1.36209739e+00, -1.13939719e+00, -6.94097733e-01, 5.04383050e-01,
1.30817822e+00, -4.53762588e-01, -5.88837759e-02, 4.69235972e-01,
2.23101470e+00, -4.78784330e-01, 1.28420116e+00, -7.35573473e-01,
-8.59462189e-01, 1.63272436e-01, -1.02955564e+00, 9.51044169e-01,
6.16300255e-01, -4.36191738e-01, 2.68838314e-01, -6.25234764e-01,
-8.76641756e-01, -7.06224356e-01, -6.04123001e-01, -9.43048773e-04,
-5.84228704e-03, -1.37478559e+00, 6.71229764e-01, -3.92506909e-01,
1.12659539e+00, 1.45383891e+00, 9.24094658e-01, 1.13088495e+00,
2.44081983e-01, -5.40528504e-01, 9.09642064e-01, -3.74346482e-02,
-6.50198584e-01, -5.20932084e-01, -1.31943573e+00, 3.88613042e-01,
9.06928122e-02, 1.32132008e-01, 7.24312532e-01, 1.10021596e+00,
-1.87496355e-01, -1.29759803e+00, 1.02222522e+00, 8.01850167e-01,
-7.24384414e-01, -7.70450083e-02, 2.09954778e+00, -9.27309886e-02,
-2.86719500e-02, 9.05682213e-01, -4.62050633e-01, 3.38666869e-01,
1.82268095e-01, -4.45240316e-02, 1.53066014e+00, 7.81346785e-02,
-2.26953970e-01, -2.24502504e-02, 2.08053084e+00, -4.26588496e-01,
-1.33719362e-01, 3.40376855e-01, 1.26119515e+00, 5.24260712e-01,
-7.84423776e-01, -6.18582179e-01, -5.28634088e-01, -1.68789075e+00,
1.96043446e+00, -1.35578848e-01, 1.15886831e+00, -1.39711347e+00,
5.78035345e-02, -9.92855098e-01, -8.28112362e-01, -2.17998634e-01,
1.00724255e-01, -3.22611446e-01, 1.14543419e-01, 1.60075386e+00,
-2.71518584e-01, 1.11258849e+00, -1.50428196e-01, -4.78648255e-02,
2.07714659e-01, 9.79680024e-01, 1.12117207e+00, -1.63926328e-01,
2.98196380e+00, -1.26653146e+00, 1.13653981e+00, 6.47240040e-01,
-1.38051916e+00, 1.17204534e+00, -1.46554024e+00, 1.24593995e+00,
-1.67084060e+00, -7.07705516e-01, -6.96851873e-01, 6.69252645e-01,
1.88168373e+00, -2.70282670e-01, 1.67218324e+00, -1.35810918e-01,
-1.01514916e+00, -9.23259129e-01, 5.79023948e-01, 7.10614236e-01,
-1.46242677e+00, -1.14214912e+00, 8.61547899e-01, -1.17092334e+00,
-6.47110317e-01, -4.54329063e-01, -4.02577073e-01, 8.04464924e-01,
1.18422626e+00, 8.34767436e-01, -1.24306785e+00, -2.22871548e+00,
-2.68059779e-01, -5.54827106e-01, 1.19362915e+00, 7.22876945e-01,
-1.25909499e+00, -1.16946977e+00, 1.66275186e+00, 3.50994818e-01,
-4.30313878e-01, 3.05178741e-02, 2.39854132e+00, 6.63388773e-02,
3.05148486e-01, 2.40850644e-01, -1.04068948e+00, 1.69818199e+00,
-7.41443373e-01, -5.12998584e-01, 5.56448324e-01, 3.68214945e-01,
1.06716702e+00, 1.21140208e+00, -1.15167377e+00, 8.98126670e-01,
1.03361317e+00, 2.01175990e-01, 2.11163312e-01, -9.82591486e-01,
8.57985773e-01, -4.35235732e-01, 3.33908418e-01, 2.44087339e-01,
-4.24966257e-02, 1.30501805e+00, 1.30854110e+00, 7.73080361e-02,
1.31227577e+00, -2.10625511e-01, -1.82915325e+00, -4.48357303e-01,
3.53735262e-01, 6.60149258e-01, -5.82265445e-01, -3.99077039e-01,
-5.87479392e-01, 4.42192306e-01, 9.44008697e-01, -2.50803118e-02,
1.78952132e-01, -2.06248081e+00, 2.10140222e-01, 1.30822331e+00,
-3.58942504e-01, 1.37895738e+00, 6.38206083e-01, 9.46059141e-01,
-9.51943962e-01, -6.09963094e-01, -2.64890602e+00, -2.10115241e+00,
-1.14321277e+00, -1.02793230e-01, 1.60036168e+00, -1.07594432e+00,
-1.07754629e+00, -3.39383782e-01, 1.14329268e+00, 1.71044242e+00,
-5.41066126e-01, 1.30579636e+00, 1.44897546e+00, 4.91602923e-02,
8.58360031e-01, -4.57675158e-01, -1.08910595e-01, 1.37552591e-01,
2.84109143e-01, -2.68160965e-01, -2.51874217e-01, -1.50449453e+00,
4.84556874e-01, -4.01971739e-01, -9.04255549e-01, -1.46809426e+00,
-1.12181719e+00, -2.03365005e-01, -2.11063055e-02, 2.14358552e+00,
-8.34958700e-01, 1.13567184e+00, -1.21069045e+00, 2.87675229e-01,
1.77685191e+00, 1.74778615e-01, 1.44960904e+00, -6.22220563e-02,
1.59557992e-02, 1.23572579e+00, -7.18769421e-01, 5.01619450e-01,
```

```
1.92922064e-01, -3.07693010e-01, -2.19347835e-01, 1.12585258e+00,
7.43173956e-01, -7.75094258e-01, -8.49555340e-02, 2.63383635e-01,
1.03765954e+00, 4.33721782e-01, 3.79807437e-01, 6.43766462e-02,
-1.98995719e+00, -6.79073396e-01, 1.52236550e+00, -3.38824923e-01,
1.56563842e+00, -1.65007205e+00, 9.62151190e-02, 1.31605644e+00,
1.27885350e+00, 5.81965129e-01, 5.56107080e-01, 2.26992999e+00,
1.62315861e+00, -4.06247424e-01, -9.02333914e-01, -1.39370962e+00,
-6.85963526e-01, -8.87266384e-01, -1.94519944e+00, -1.58629739e+00,
1.32252166e+00, -2.90530047e-01, 1.13437296e+00, -9.29131787e-01,
-3.68474933e-01, 7.81706200e-01, 1.07966320e+00, 5.87355044e-01,
8.93393352e-02, 4.09823240e-01, 2.80223500e-01, -1.07743854e-01,
6.49543822e-01, -4.47733308e-01, -1.68393037e-01, -8.96161162e-01,
-9.08876927e-01, 1.11096324e+00, 4.50626025e-01, 1.80445754e+00,
-8.04205276e-01, -4.20699825e-01, 8.16437416e-01, 9.48611610e-01,
-1.73689723e-01, -2.45325406e+00, 1.42001389e+00, -2.37971349e-01,
-1.56640139e+00, 6.15644148e-01, 2.25261893e+00, -7.06820100e-01,
5.97343240e-01, 1.22850560e+00, -5.20487474e-01, -4.07965888e-01,
-1.42570078e+00, -2.37926811e+00, 6.86679044e-01, -2.39915026e-01,
-3.72302494e-03, -8.41749092e-01, 1.20463384e+00, 7.98921507e-02,
1.56707016e+00, 9.11849399e-01, -1.57441232e-01, 1.41820726e+00,
3.15814029e-01, 1.54401255e-02, -4.17641530e-01, -8.57953702e-01,
5.05083543e-01, 1.70009064e+00, -1.39748123e+00, 3.23539863e-03,
9.84187969e-01, 2.48168306e+00, -5.48222485e-01, 8.78623636e-01,
-2.86857594e-01, -3.04266707e-01, 5.90629622e-01, -1.24247718e+00,
-8.59988337e-01, 7.12656955e-01, 1.21572451e-01, 1.10016912e+00,
-5.32377994e-01, -5.59541225e-01, 1.38564937e+00, -3.61756326e-01,
-9.91198107e-01, -5.57474228e-01, 6.63286613e-01, 2.50997170e+00,
5.43052138e-01, 2.05672372e-01, -2.37178718e-01, 7.92753284e-01,
5.10041219e-01, -9.94780760e-01, -1.34154446e+00, -6.69582023e-01,
-1.61358795e+00, -5.57674465e-01, 5.36954510e-01, -2.24186720e+00,
6.16784887e-02, 3.30601755e-01, -3.56449952e-01, -1.62404209e+00,
-1.73641828e-01, -1.41193697e+00, 2.32303016e-01, 9.57407066e-01,
-7.22772315e-01, 1.11328147e+00, 5.06161422e-01, -1.24842590e-01,
7.06690401e-01, -5.23486618e-01, 6.33564712e-01, 4.45599350e-01,
-7.52846631e-01, -7.88832103e-01, 5.72220781e-01, 9.07058549e-01,
4.86330631e-01, -3.38796401e-01, -2.04108558e-01, 5.36196815e-01,
9.30361371e-01, 2.05605194e+00, -9.91248812e-01, -1.83481849e+00,
-5.26057458e-01, 5.08893447e-01, -2.04692774e+00, -1.90713266e+00,
-1.96472544e+00, -1.44287930e+00, 3.10650949e-03, -5.13752538e-01,
1.19485721e+00, -2.52448416e-01, 7.62947296e-01, -2.05667114e+00,
-5.04646300e-01, 6.52677458e-01, 9.01196332e-02, -1.06369030e+00,
1.28668986e+00, -6.37249224e-01, -5.73944968e-01, 5.22907768e-02,
-1.89756501e+00, -5.10036191e-02, -1.29952570e+00, -6.14892059e-01,
-4.26178878e-01, -9.59826304e-01, -5.47664242e-01, -1.60285720e+00,
1.88652715e+00, 6.84736995e-01, 4.71289453e-01, 9.78570048e-01,
6.01712595e-01, 8.18866004e-01, -5.51823871e-01, 1.13847275e+00,
7.23719080e-01, -2.22322646e+00, -5.01442593e-01, -9.97867661e-01,
4.48715576e-02, 9.32914407e-01, -1.99239990e-01, 5.74569453e-02,
1.25684548e-01, -3.53952277e-01, 7.55584671e-01, 1.17058297e+00,
-1.09217467e+00, 1.69889555e+00, 4.38924775e-01, -4.11579856e-01,
-3.31726982e-01, -8.26696925e-01, 1.02626902e+00, 2.33398998e-01,
-9.35639605e-01, -1.38354125e+00, 1.14459990e+00, 6.08711850e-01,
-1.19884044e+00, -1.04743264e-01, 2.97615768e-01, -1.85335598e+00,
-1.39168117e+00, 7.59564685e-01, -1.63093155e+00, -7.35152260e-01,
-1.59226012e+00, -1.40618751e-01, -1.12653042e+00, -7.24750347e-01,
-1.00045706e+00, 3.54354148e-01, 5.79183003e-01, -1.05693266e+00,
-9.60123452e-01, 4.61108140e-01, 2.09692792e+00, -8.94810758e-01,
```

```
-6.27096971e-01, 1.94917831e-01, 1.98302750e+00, -2.91429942e+00,
5.34460927e-01, 5.31169554e-02, 1.99049634e+00, -1.79894774e-01,
2.29123102e+00, -4.43802236e-01, -2.79402960e-01, -1.41677573e+00,
1.16602706e+00, -2.87670657e-01, 2.61900931e-02, 4.89743426e-01,
-1.33612778e+00, 7.37262528e-01, 4.47523176e-02, 6.78413944e-01,
1.55346604e+00, -9.81130147e-01, 4.30249589e-02, -2.38836044e+00,
6.25204941e-01, -9.73460949e-02, -1.73865231e+00, 2.03330756e-01,
-2.38403112e+00, 1.27249558e+00, 1.59009271e-01, -8.33022644e-01,
2.56120274e-01, -6.72352623e-02, 5.38316913e-01, 6.07080818e-02,
9.41938366e-01, 3.22948868e-01, -4.11778578e-01, 1.35722512e+00,
-1.34926911e+00, -3.86662305e-01, -1.15514073e+00, 5.81740971e-01,
7.24880460e-01, -1.11494041e+00, 2.03180277e+00, -3.09398267e-01,
2.95173313e-01, -3.26188218e-02, 1.21414819e+00, 2.36138269e-01,
1.03485360e-01, 8.16460824e-01, -5.45326853e-01, -5.07519053e-01,
1.07864361e+00, -2.52448486e+00, 1.02691569e-01, -2.62004232e-01,
2.14338518e+00, 4.24583100e-01, -1.45689709e+00, -1.64818013e-01,
-9.61929394e-01, -1.59270407e+00, 1.69630008e+00, 1.55325624e+00,
6.84505543e-01, 7.38884689e-02, -1.11841795e+00, 5.68270508e-01,
1.05336516e+00, 8.28903228e-01, -6.72188709e-02, 5.61543642e-01,
1.56503103e+00, 6.34136896e-01, 9.22832552e-01, -7.43029111e-01,
7.36656635e-01, -1.84588306e+00, -6.66802910e-01, 4.89131997e-01,
-1.09623095e+00, 6.60367019e-01, 2.69254460e-01, 5.90077140e-01,
6.38867226e-01, -1.85402164e+00, -2.43862085e+00, -1.81756405e+00,
-4.20998937e-01, 1.35084818e+00, 7.18591461e-01, 2.79170261e-01,
-1.63037685e+00, 1.41088104e+00, -7.98960724e-01, -3.22465808e-01,
-7.02174394e-01, 3.21829046e-01, -2.09186277e+00, -9.36667504e-01])
```

In [67]: `plt.hist(x, histtype="step")`
`plt.show()`



In [68]: 1 `help(plt.hist)`

Help on function hist in module matplotlib.pyplot:

```
hist(x, bins=None, range=None, density=None, weights=None, cumulative=False,
     bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=
     None, log=False, color=None, label=None, stacked=False, normed=None, da
     ta=None, **kwargs)
```

Plot a histogram.

Compute and draw the histogram of `*x*`. The return value is a tuple `(*n*, *bins*, *patches*)` or `([*n0*, *n1*, ...], *bins*, [*patches0*, *patches1*,...])` if the input contains multiple data.

Multiple data can be provided via `*x*` as a list of datasets of potentially different length `([*x0*, *x1*, ...])`, or as a 2-D ndarray in which each column is a dataset. Note that the ndarray form is transposed relative to the list form.

Masked arrays are not supported at present.

Parameters

`x` : (n,) array or sequence of (n,) arrays

Input values, this takes either a single array or a sequence of arrays which are not required to be of the same length.

`bins` : int or sequence or str, optional

If an integer is given, ```bins + 1``` bin edges are calculated and returned, consistent with ``numpy.histogram``.

If ``bins`` is a sequence, gives bin edges, including left edge of first bin and right edge of last bin. In this case, ``bins`` is returned unmodified.

All but the last (righthand-most) bin is half-open. In other words, if ``bins`` is::

```
[1, 2, 3, 4]
```

then the first bin is ```[1, 2)``` (including 1, but excluding 2) and the second ```[2, 3)```. The last bin, however, is ```[3, 4]```, which `*includes* 4`.

Unequally spaced bins are supported if `*bins*` is a sequence.

With Numpy 1.11 or newer, you can alternatively provide a string describing a binning strategy, such as `'auto'`, `'sturges'`, `'fd'`, `'doane'`, `'scott'`, `'rice'`, `'sturges'` or `'sqrt'`, see ``numpy.histogram``.

The default is taken from `:rc:`hist.bins``.

`range` : tuple or None, optional

The lower and upper range of the bins. Lower and upper outliers

are ignored. If not provided, `*range*` is `((x.min(), x.max()))`. Range has no effect if `*bins*` is a sequence.

If `*bins*` is a sequence or `*range*` is specified, autoscaling is based on the specified bin range instead of the range of `x`.

Default is `None`

`density : bool, optional`

If `True`, the first element of the return tuple will be the counts normalized to form a probability density, i.e., the area (or integral) under the histogram will sum to 1. This is achieved by dividing the count by the number of observations times the bin width and not dividing by the total number of observations. If `*stacked*` is also `True`, the sum of the histograms is normalized to 1.

Default is `None` for both `*normed*` and `*density*`. If either is set, then that value will be used. If neither are set, then the args will be treated as `False`.

If both `*density*` and `*normed*` are set an error is raised.

`weights : (n,) array_like or None, optional`

An array of weights, of the same shape as `*x*`. Each value in `*x*` only contributes its associated weight towards the bin count (instead of 1). If `*normed*` or `*density*` is `True`, the weights are normalized, so that the integral of the density over the range remains 1.

Default is `None`

`cumulative : bool, optional`

If `True`, then a histogram is computed where each bin gives the counts in that bin plus all bins for smaller values. The last bin gives the total number of datapoints. If `*normed*` or `*density*` is also `True` then the histogram is normalized such that the last bin equals 1. If `*cumulative*` evaluates to less than 0 (e.g., -1), the direction of accumulation is reversed. In this case, if `*normed*` and/or `*density*` is also `True`, then the histogram is normalized such that the first bin equals 1.

Default is `False`

`bottom : array_like, scalar, or None`

Location of the bottom baseline of each bin. If a scalar, the base line for each bin is shifted by the same amount. If an array, each bin is shifted independently and the length of bottom must match the number of bins. If `None`, defaults to 0.

Default is `None`

`histtype : {'bar', 'barstacked', 'step', 'stepfilled'}, optional`

The type of histogram to draw.

- 'bar' is a traditional bar-type histogram. If multiple data

are given the bars are arranged side by side.

- 'barstacked' is a bar-type histogram where multiple data are stacked on top of each other.
- 'step' generates a lineplot that is by default unfilled.
- 'stepfilled' generates a lineplot that is by default filled.

Default is 'bar'

`align` : {'left', 'mid', 'right'}, optional
Controls how the histogram is plotted.

- 'left': bars are centered on the left bin edges.
- 'mid': bars are centered between the bin edges.
- 'right': bars are centered on the right bin edges.

Default is 'mid'

`orientation` : {'horizontal', 'vertical'}, optional
If 'horizontal', `~matplotlib.pyplot.barh` will be used for bar-type histograms and the `*bottom*` kwarg will be the left edges.

`rwidth` : scalar or None, optional
The relative width of the bars as a fraction of the bin width. If ```None```, automatically compute the width.

Ignored if `*histtype*` is 'step' or 'stepfilled'.

Default is ```None```

`log` : bool, optional
If ```True```, the histogram axis will be set to a log scale. If `*log*` is ```True``` and `*x*` is a 1D array, empty bins will be filtered out and only the non-empty ```(n, bins, patches)``` will be returned.

Default is ```False```

`color` : color or array_like of colors or None, optional
Color spec or sequence of color specs, one per dataset. Default (```None```) uses the standard line color sequence.

Default is ```None```

`label` : str or None, optional
String, or sequence of strings to match multiple datasets. Bar charts yield multiple patches per dataset, but only the first gets the label, so that the legend command will work as expected.

default is ```None```

stacked : bool, optional
 If ``True``, multiple data are stacked on top of each other If
 ``False`` multiple data are arranged side by side if histtype is
 'bar' or on top of each other if histtype is 'step'

 Default is ``False``

normed : bool, optional
 Deprecated; use the density keyword argument instead.

Returns

n : array or list of arrays
 The values of the histogram bins. See **normed** or **density**
 and **weights** for a description of the possible semantics.
 If input **x** is an array, then this is an array of length
nbins. If input is a sequence of arrays
 ``[data1, data2,..]``, then this is a list of arrays with
 the values of the histograms for each of the arrays in the
 same order.

bins : array
 The edges of the bins. Length nbins + 1 (nbins left edges and right
 edge of last bin). Always a single array even when multiple data
 sets are passed in.

patches : list or list of lists
 Silent list of individual patches used to create the histogram
 or list of such list if multiple input datasets.

Other Parameters

****kwargs** : `~matplotlib.patches.Patch`` properties

See also

hist2d : 2D histograms

Notes

.. [Notes section required for data comment. See #10189.]

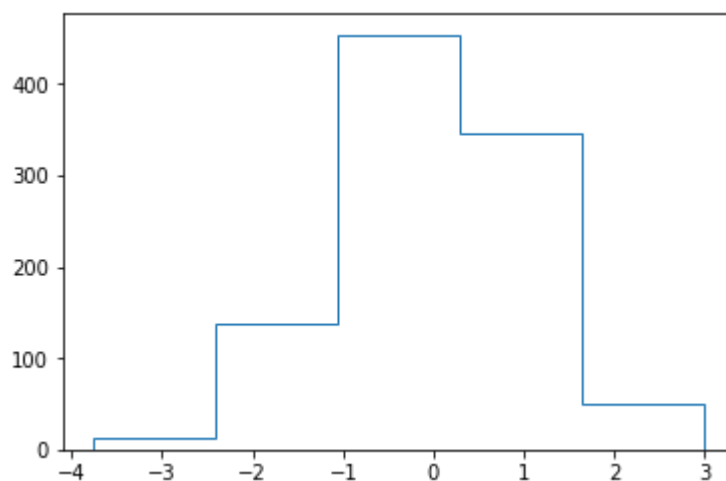
.. note::

In addition to the above described arguments, this function can tak
 e a
 e
****data**** keyword argument. If such a ****data**** argument is given, th
 following arguments are replaced by ****data[<arg>]****:

 * All arguments with the following names: 'weights', 'x'.

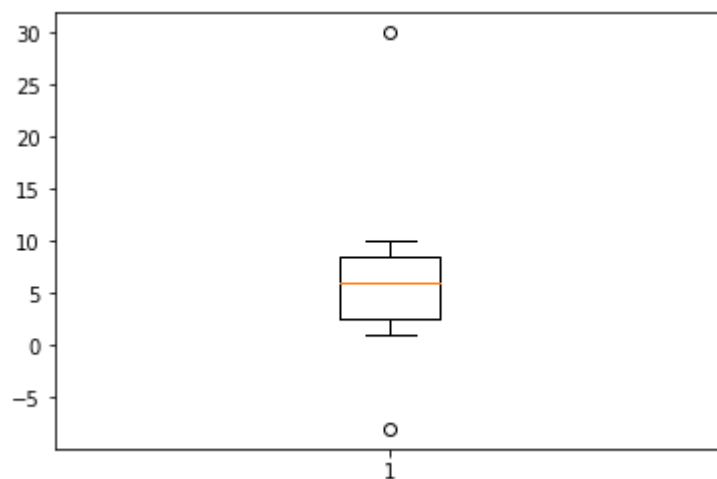
 Objects passed as ****data**** must support item access (`data[<arg>]`
) and
 membership test (`<arg> in data`).

```
In [71]: 1 plt.hist(x,histtype="step",bins=5)
        2 plt.show()
```



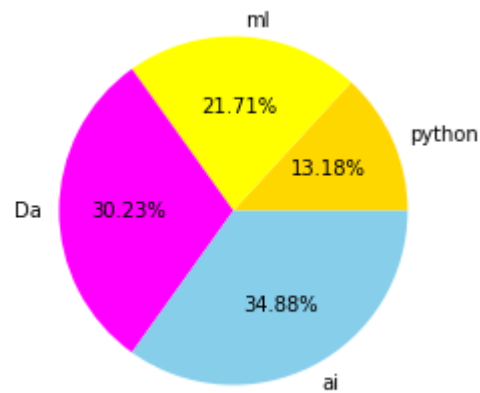
Boxplot /Whisker plot

```
In [74]: 1 a = [1,2,3,4,6,7,8,9,10,30,-8]
        2 plt.boxplot(a)
        3 plt.show()
```

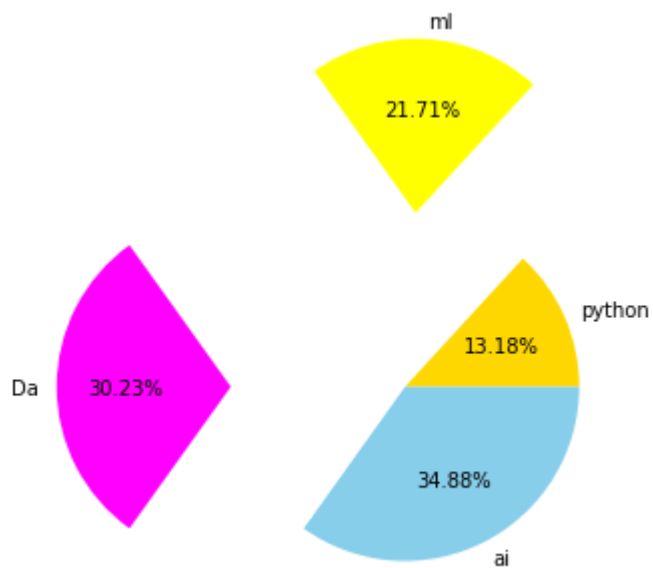


Pieplot

```
In [81]: 1 l = ["python", "ml", "Da", "ai"]
2 s = [34, 56, 78, 90]
3 c = ["gold", "yellow", "magenta", "skyblue"]
4 plt.pie(s, labels=l, autopct="%1.2f%%", colors=c)
5 plt.show()
```

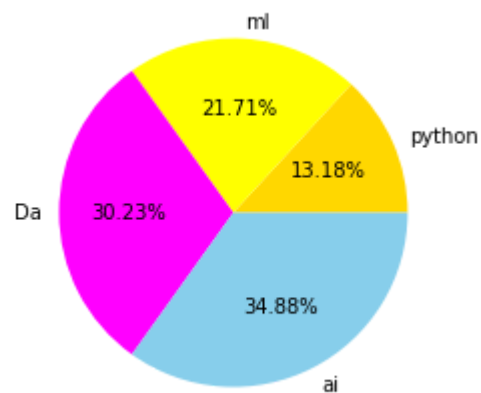


```
In [85]: 1 l = ["python", "ml", "Da", "ai"]
2 s = [34, 56, 78, 90]
3 c = ["gold", "yellow", "magenta", "skyblue"]
4 explode = (0, 1, 1, 0)
5 plt.pie(s, labels=l, autopct="%1.2f%%", colors=c, explode=explode)
6 plt.show()
```



Save plot

```
In [91]: ▶ 1 l = ["python", "ml", "Da", "ai"]  
2 s = [34, 56, 78, 90]  
3 c = ["gold", "yellow", "magenta", "skyblue"]  
4 plt.pie(s, labels=l, autopct="%1.2f%", colors=c)  
5 plt.savefig("pieplot.jpg")  
6 plt.show()
```



Reading an image

```
In [90]: 1 a = plt.imread("flower.jpeg") #c://foldername//filename
        2 a # pixel values
```

```
Out[90]: array([[[ 54,  61,  53],
                  [ 54,  61,  53],
                  [ 54,  61,  53],
                  ...,
                  [138, 144, 144],
                  [138, 144, 144],
                  [138, 144, 144]],

                [[ 54,  61,  53],
                  [ 54,  61,  53],
                  [ 54,  61,  53],
                  ...,
                  [139, 145, 145],
                  [139, 145, 145],
                  [139, 145, 145]],

                [[ 54,  61,  53],
                  [ 54,  61,  53],
                  [ 54,  61,  53],
                  ...,
                  [139, 145, 145],
                  [139, 145, 145],
                  [139, 145, 145]],

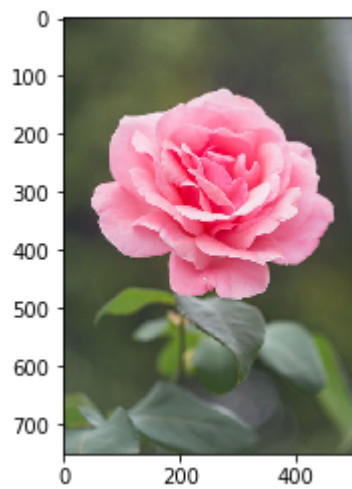
                ...,

                [[120, 142, 130],
                  [120, 142, 130],
                  [120, 142, 130],
                  ...,
                  [ 82,  92,  65],
                  [ 82,  92,  65],
                  [ 82,  92,  65]],

                [[117, 139, 127],
                  [117, 139, 127],
                  [118, 140, 128],
                  ...,
                  [ 82,  92,  65],
                  [ 82,  92,  65],
                  [ 82,  92,  65]],

                [[114, 136, 124],
                  [115, 137, 125],
                  [118, 140, 128],
                  ...,
                  [ 82,  92,  65],
                  [ 82,  92,  65],
                  [ 82,  92,  65]]], dtype=uint8)
```

```
In [97]: ▶ 1 plt.imshow(a)
          2 #plt.figure(figsize=(100,15))
          3 plt.show()
```



<Figure size 7200x1080 with 0 Axes>

```
In [ ]: ▶ 1
```