- Seaborn is a visualization library in python for statistical plotting
- Designed to work with Data frame objects in pandas
- It contains default attractive style
- It provides high level interface for drawing attractive and informative statistical graphs
- Seaborn official website : https://seaborn.pydata.org/ (https://seaborn.pydata.org/)

In [1]:

```
import seaborn as sns
```

In [2]:

```
sns.__version__
```

Out[2]:

```
'0.11.2'
```

In [3]:

```
print(dir(sns))
```

```
['FacetGrid', 'JointGrid', 'PairGrid', '__builtins__', '__cached__', '__do
c__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__
spec__', '__version__', '_core', '_decorators', '_docstrings', '_orig_rc_p
arams', '_statistics', 'algorithms', 'axes_style', 'axisgrid', 'barplot',
'blend_palette', 'boxenplot', 'boxplot', 'categorical', 'catplot', 'choose
_colorbrewer_palette', 'choose_cubehelix_palette', 'choose_dark_palette',
'choose_diverging_palette', 'choose_light_palette', 'clustermap', 'cm', 'c
olor_palette', 'colors', 'countplot', 'crayon_palette', 'crayons', 'cubehe
lix_palette', 'dark_palette', 'desaturate', 'despine', 'displot', 'distplo
t', 'distributions', 'diverging_palette', 'dogplot', 'ecdfplot', 'externa
l', 'factorplot', 'get_data_home', 'get_dataset_names', 'heatmap', 'histpl
ot', 'hls_palette', 'husl_palette', 'jointplot', 'kdeplot', 'light_palett
e', 'lineplot', 'lmplot', 'load_dataset', 'matrix', 'miscplot', 'move_lege
nd', 'mpl', 'mpl_palette', 'pairplot', 'palettes', 'palplot', 'plotting_co
ntext', 'pointplot', 'rcmod', 'regplot', 'regression', 'relational', 'relp
lot', 'reset_defaults', 'reset_orig', 'residplot', 'rugplot', 'saturate',
'scatterplot', 'set', 'set_color_codes', 'set_context', 'set_hls_values',
'set_palette', 'set_style', 'set_theme', 'stripplot', 'swarmplot', 'util
s', 'violinplot', 'widgets', 'xkcd_palette', 'xkcd_rgb']
```

Types of plots

- Color palette
- Plotting with categorical data
- Jointplot
- Pairplot
- Heat Maps

color_palette

- It is an interface to generate few colors in seaborn

- sns.color_palette()
- Seaborn has 6 variations of its default color palette: deep,muted,pastel,bright,dark and colorblind

In [5]:

```
sns.color_palette()
```

Out[5]:

In [18]:

```
help(sns.color_palette)
```

```
Help on function color_palette in module seaborn.palettes:

color_palette(palette=None, n_colors=None, desat=None, as_cmap=False)
    Return a list of colors or continuous colormap defining a palette.

    Possible ``palette`` values include:
        - Name of a seaborn palette (deep, muted, bright, pastel, dark, co
lorblind)
        - Name of matplotlib colormap
        - 'husl' or 'hls'
        - 'ch:<cubehelix arguments>'
        - 'light:<color>', 'dark:<color>', 'blend:<color>,<color>',
        - A sequence of colors in any format matplotlib accepts

    Calling this function with ``palette=None`` will return the current
    matplotlib color cycle.

    This function can also be used in a ``with`` statement to temporarily
    set the color cycle for a plot or set of plots.

    See the :ref:`tutorial <palette_tutorial>` for more information.

    Parameters
    ----------
    palette : None, string, or sequence, optional
        Name of palette or None to return current palette. If a sequence,
input
        colors are used but possibly cycled and desaturated.
    n_colors : int, optional
        Number of colors in the palette. If ``None``, the default will dep
end
        on how ``palette`` is specified. Named palettes default to 6 color
s,
        but grabbing the current palette or passing in a list of colors wi
ll
        not change the number of colors unless this is specified. Asking f
or
        more colors than exist in the palette will cause it to cycle. Igno
red
        when ``as_cmap`` is True.
    desat : float, optional
        Proportion to desaturate each color by.
    as_cmap : bool
        If True, return a :class:`matplotlib.colors.Colormap`.

    Returns
    -------
    list of RGB tuples or :class:`matplotlib.colors.Colormap`

    See Also
    --------
    set_palette : Set the default color cycle for all plots.
    set_color_codes : Reassign color codes like ``"b"``, ``"g"``, etc. to
                      colors from one of the seaborn palettes.

    Examples
    --------

    .. include:: ../docstrings/color_palette.rst
```

palplot()

- creates a plot for the colors of the palette

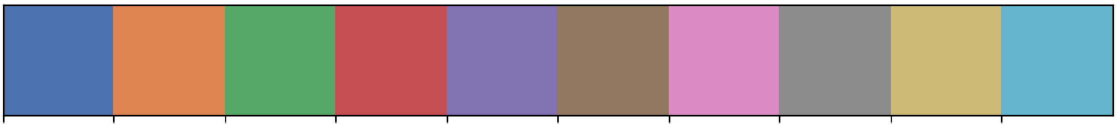
In [11]:

```
sns.color_palette('deep')
```

Out[11]:

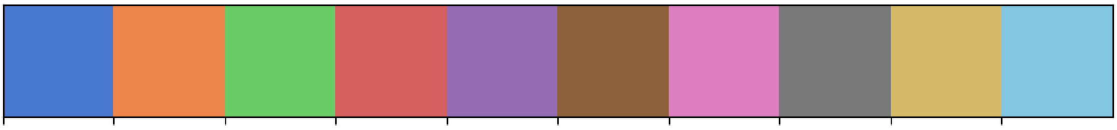
In [12]:

```
sns.palplot(sns.color_palette('deep'))
```



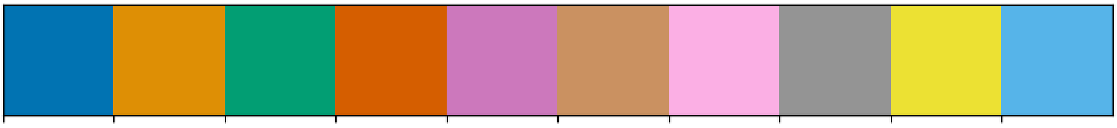
In [13]:

```
sns.palplot(sns.color_palette('muted'))
```



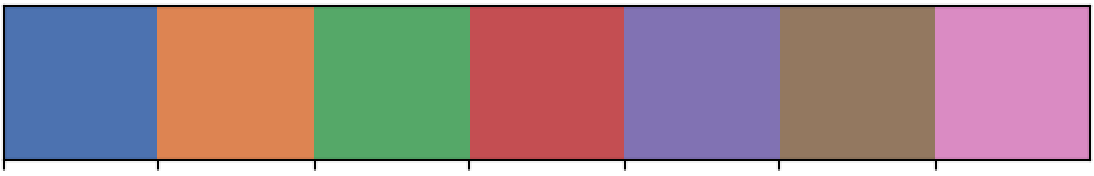
In [14]:

```
sns.palplot(sns.color_palette('colorblind'))
```



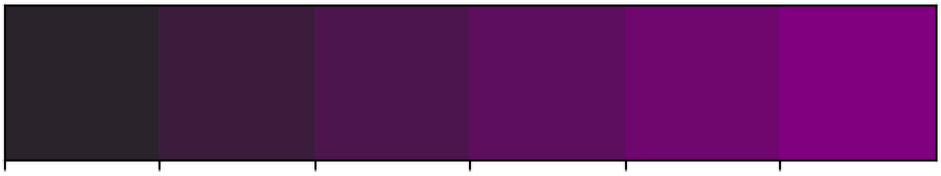
In [16]:

```
sns.palplot(sns.color_palette('deep',n_colors=7))   # no of colours u want to display in
```



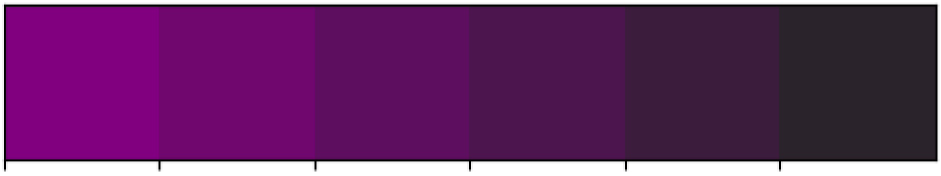
In [19]:

```
sns.palplot(sns.dark_palette("purple"))
```



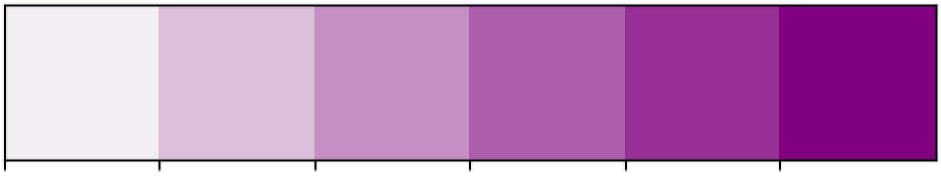
In [21]:

```
sns.palplot(sns.dark_palette("purple",reverse=True))
```



In [20]:

```
sns.palplot(sns.light_palette("purple"))
```



In [22]:

```
sns.get_dataset_names()
```

Out[22]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [23]:

```
iris=sns.load_dataset("iris")
```

In [24]:

```
iris
```

Out[24]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [25]:

```
iris.head()
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [26]:

```
iris.tail()
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [27]:

```
iris.shape
```

Out[27]:

```
(150, 5)
```

In [28]:

```
iris.isnull().sum()
```

Out[28]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [30]:

```
iris["species"].value_counts()
```

Out[30]:

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

Categorical plot

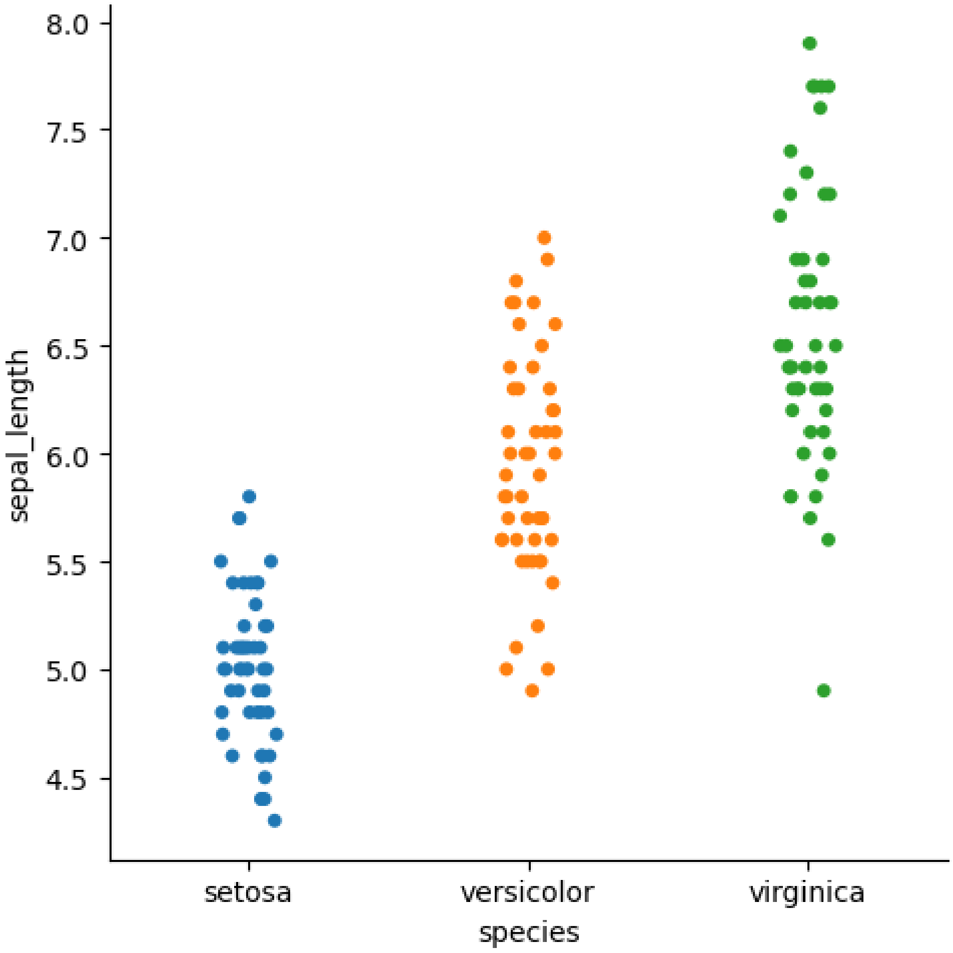
- Categorical plots shows the relationship between a numerical variable and one or more categorical variable
- catplot() is used to plot categorical plots.
- By Default it returns scatter plot
- sns.catplot()

In [31]:

```
help(sns.catplot)
```

```
Help on function catplot in module seaborn.categorical:

catplot(*, x=None, y=None, hue=None, data=None, row=None, col=None, col
_wrap=None, estimator=<function mean at 0x0000025E4E776C10>, ci=95, n_b
oot=1000, units=None, seed=None, order=None, hue_order=None, row_order=
None, col_order=None, kind='strip', height=5, aspect=1, orient=None, co
lor=None, palette=None, legend=True, legend_out=True, sharex=True, shar
ey=True, margin_titles=False, facet_kws=None, **kwargs)
    Figure-level interface for drawing categorical plots onto a FacetGr
id.

    This function provides access to several axes-level functions that
    show the relationship between a numerical and one or more categoric
al
    variables using one of several visual representations. The ``kind``
    parameter selects the underlying axes-level function to use:

    Categorical scatterplots:

    - :func:`stripplot` (with ``kind="strip"``; the default)
```

```
In [32]:
sns.catplot(x="species",y="sepal_length",data=iris)
Out[32]:
<seaborn.axisgrid.FacetGrid at 0x25e5431d2e0>
```



Categorical distribution plots:

- boxplot (with kind="box")
- violinplot (with kind="violin")
- boxenplot (with kind="boxen")

Categorical distribution data

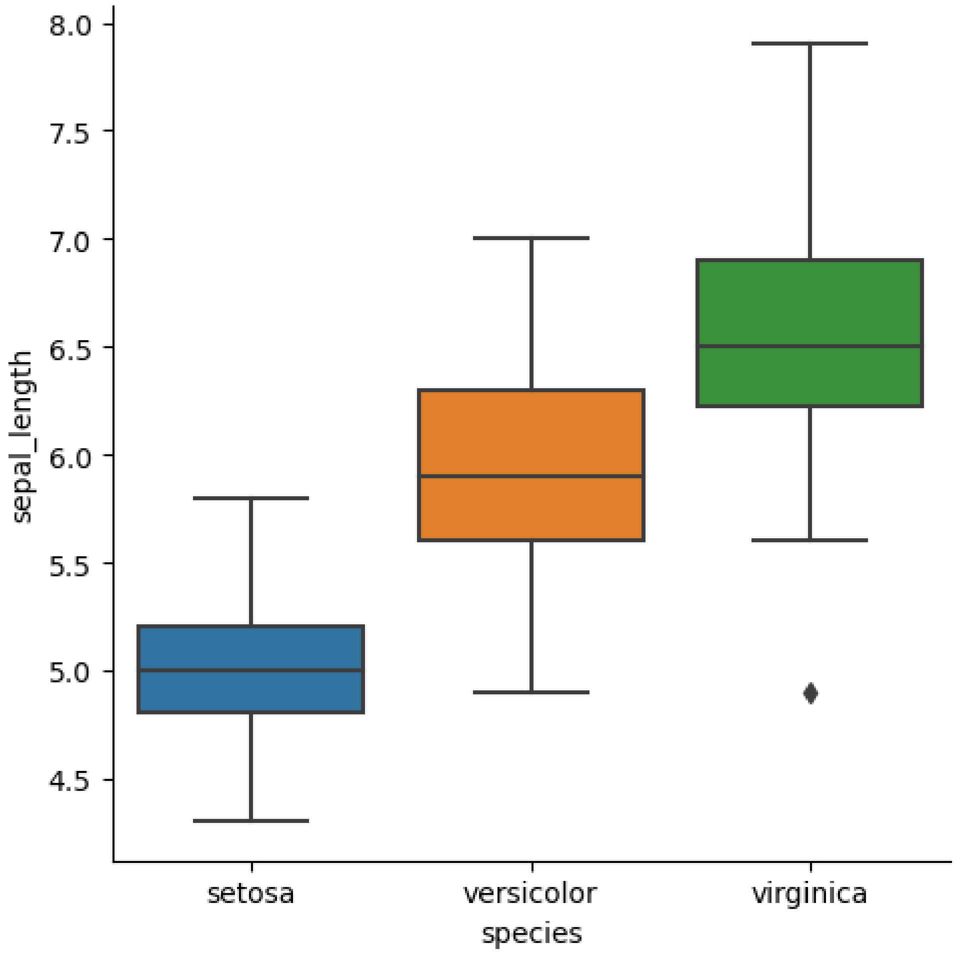
- Box plot

In [33]:

```
sns.catplot(x="species",y="sepal_length",data=iris,kind='box')
```

Out[33]:

```
<seaborn.axisgrid.FacetGrid at 0x25e5411bfd0>
```

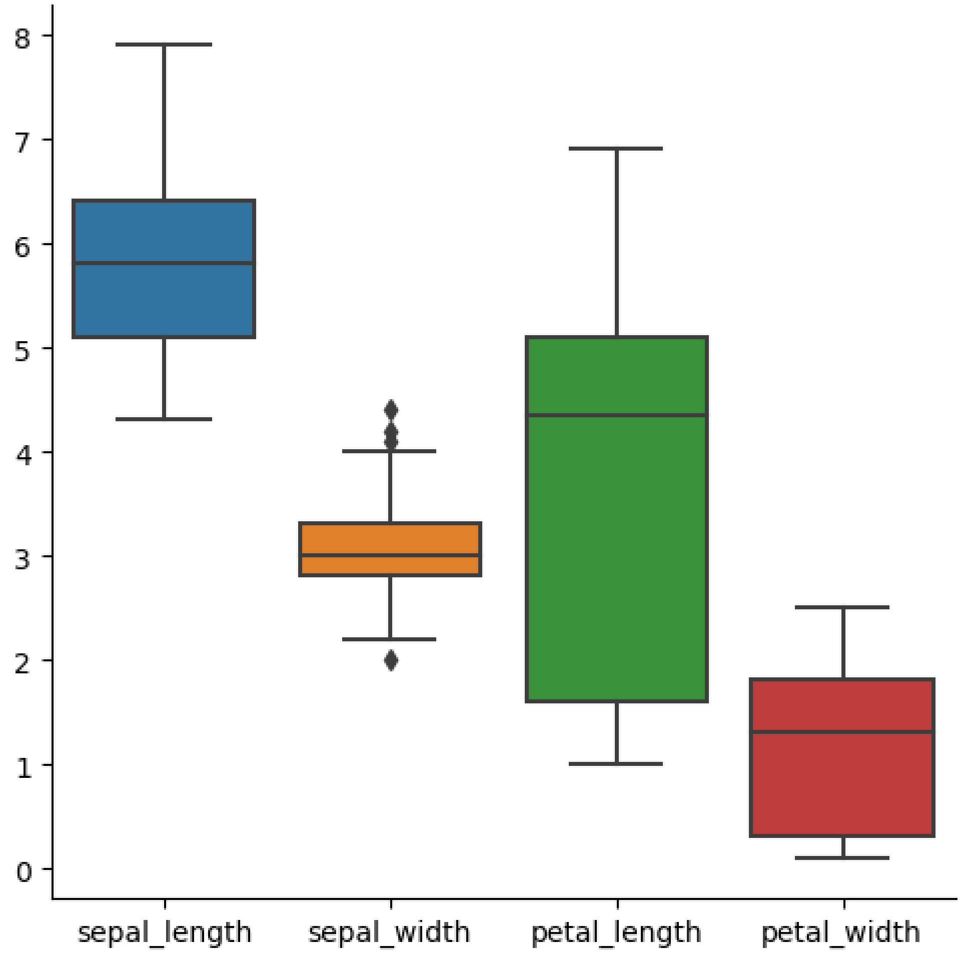


In [35]:

```
sns.catplot(data=iris,kind='box')   # plotting for numerical data
```

Out[35]:

```
<seaborn.axisgrid.FacetGrid at 0x25e555a50d0>
```

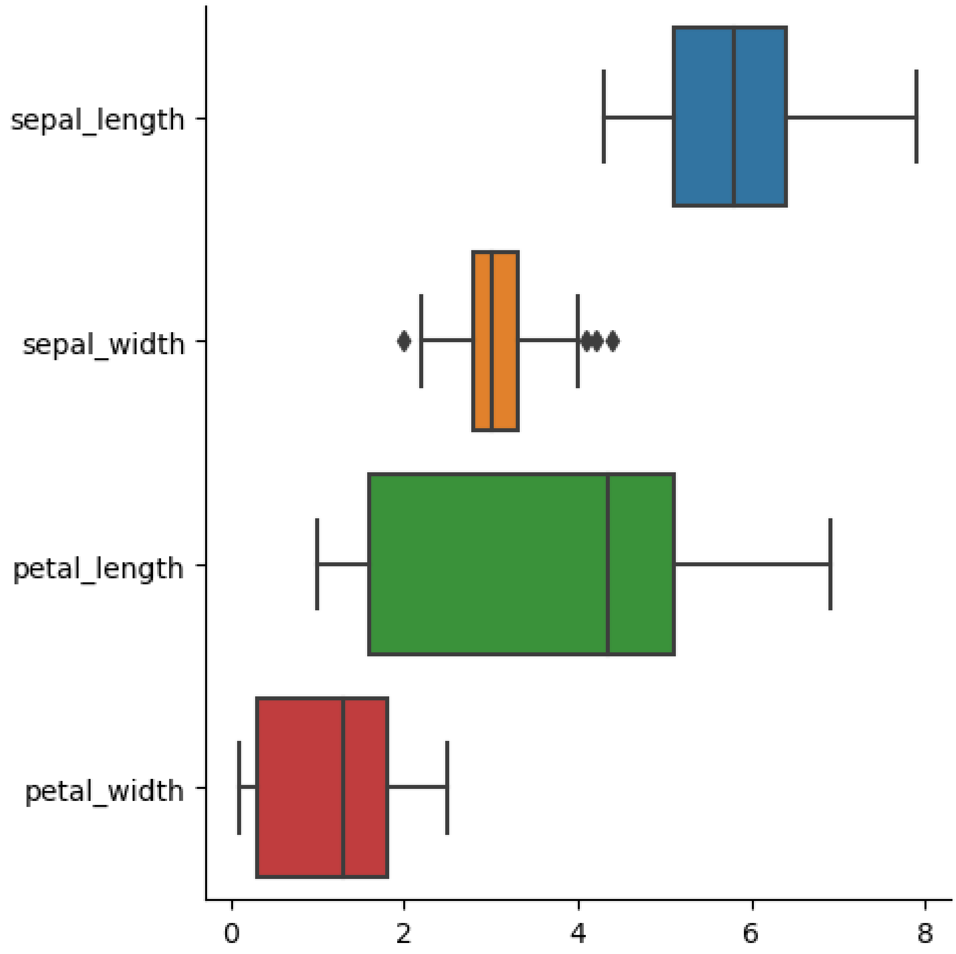


In [36]:

```
sns.catplot(data=iris,kind='box',orient='h')
```

Out[36]:

```
<seaborn.axisgrid.FacetGrid at 0x25e555d53a0>
```



Violin plot

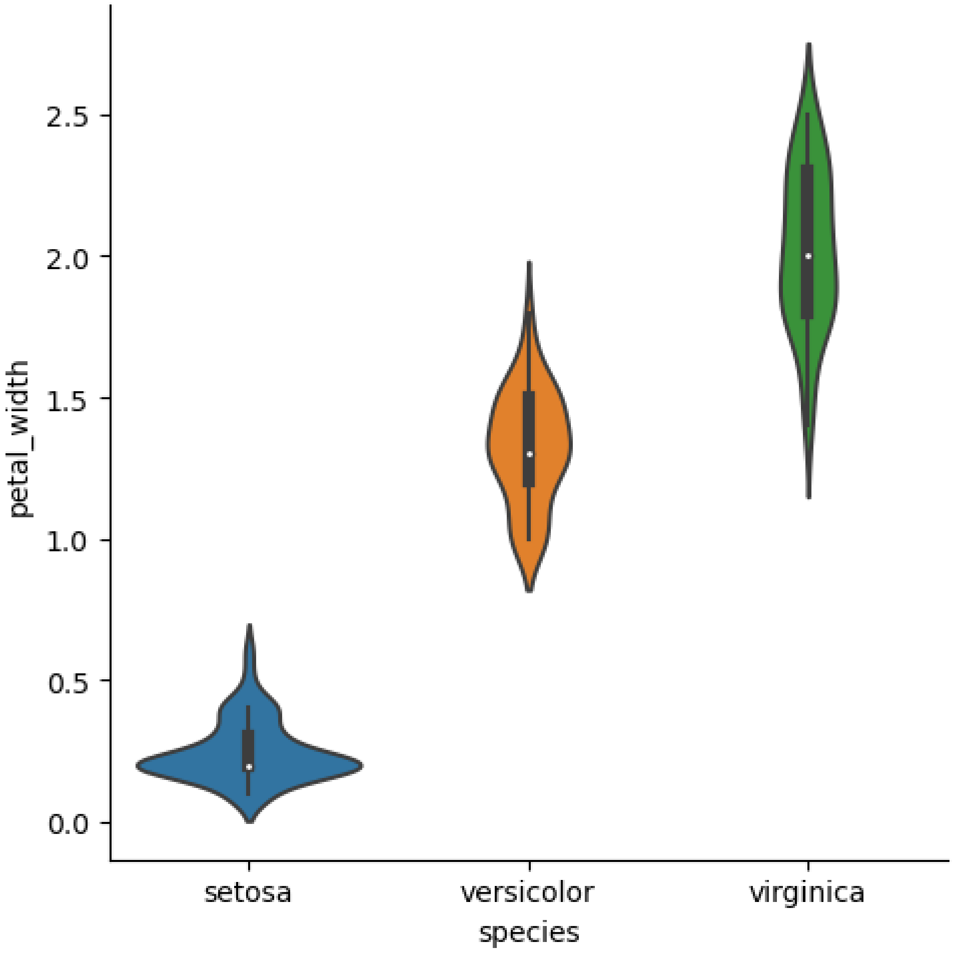
- similar to boxplot

In [37]:

```
sns.catplot(x="species",y="petal_width",data=iris,kind='violin')
```

Out[37]:

```
<seaborn.axisgrid.FacetGrid at 0x25e55574e50>
```



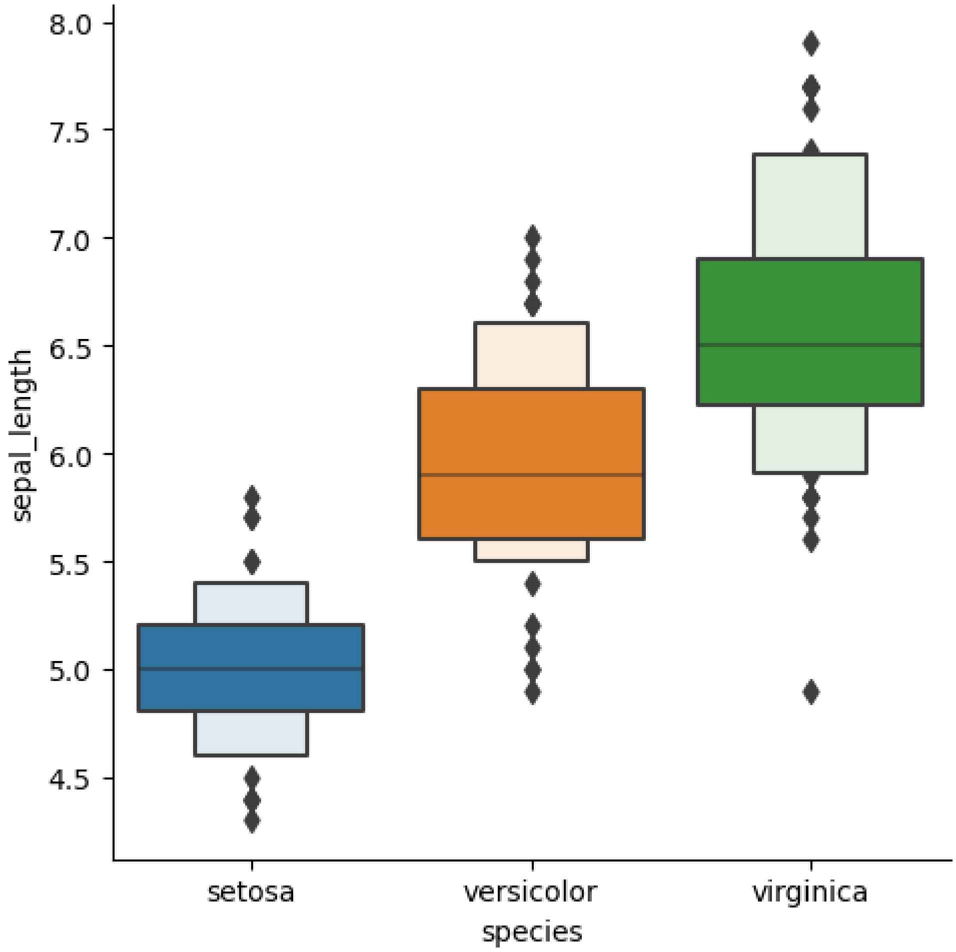
Boxen plot

In [38]:

```
sns.catplot(x="species",y="sepal_length",data=iris,kind='boxen')
```

Out[38]:

```
<seaborn.axisgrid.FacetGrid at 0x25e4dfb18b0>
```



In [39]:

```
titanic=sns.load_dataset("titanic")
```

In [40]:

```
titanic.head()
```

Out[40]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru

In [41]:

```
titanic.isnull().sum()
```

Out[41]:

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [42]:

```
titanic.isnull().sum().sum()
```

Out[42]:

```
869
```

In [44]:

```
titanic["class"].value_counts()
```

Out[44]:

```
Third     491
First     216
Second    184
Name: class, dtype: int64
```

Categorical estimate plots:

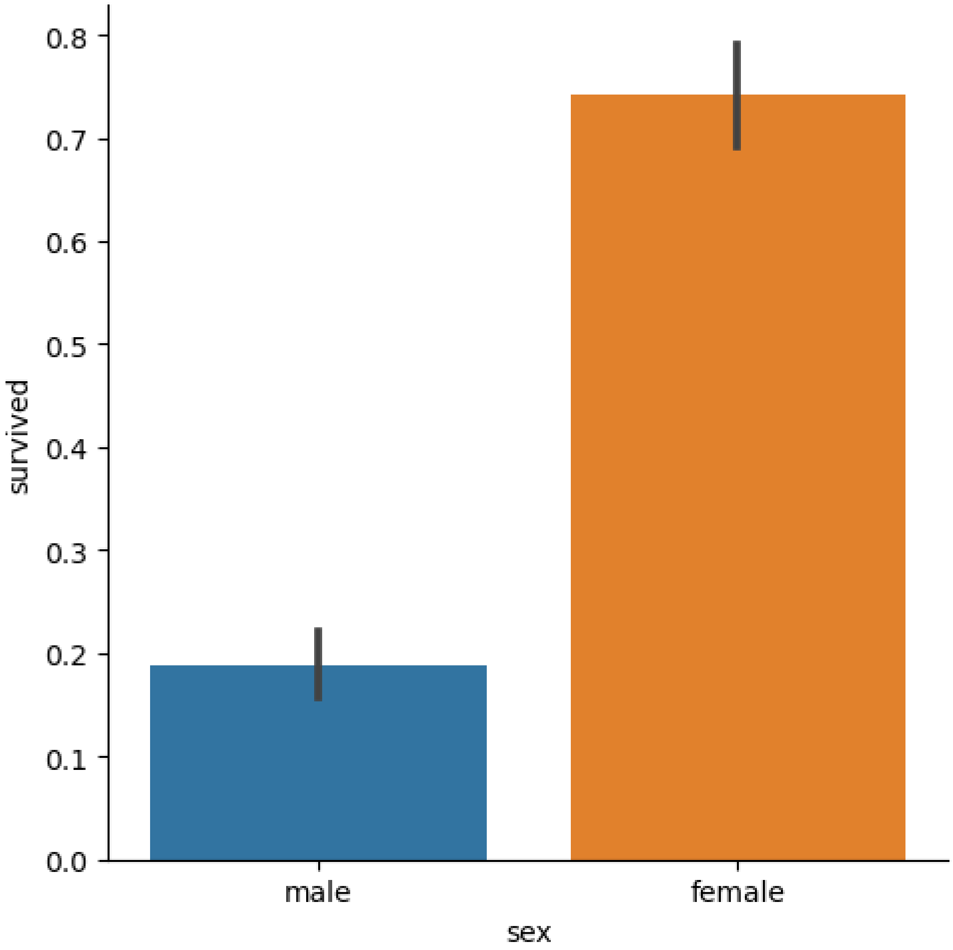
- pointplot (with kind="point")
- barplot (with kind="bar")
- countplot (with kind="count")

In [46]:

```
sns.catplot(x="sex",y="survived",data=titanic,kind='bar')
```

Out[46]:

```
<seaborn.axisgrid.FacetGrid at 0x25e4dee60d0>
```

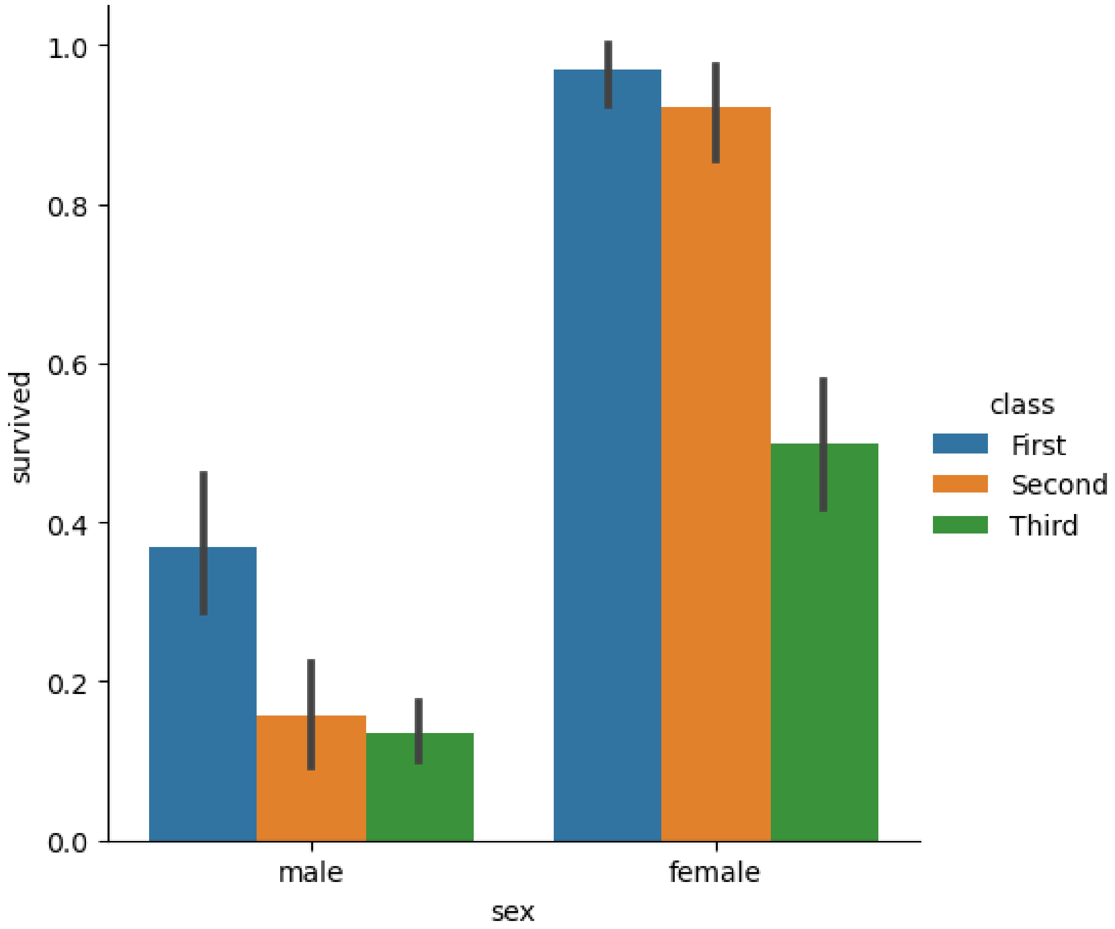


In [47]:

```
sns.catplot(x="sex",y="survived",data=titanic,hue="class",kind='bar')
```

Out[47]:

```
<seaborn.axisgrid.FacetGrid at 0x25e53949cd0>
```



joint plot

- Combination of two plots. By default it have scatterplot, histogram
- sns.jointplot()
- kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }

In [48]:

```
help(sns.jointplot)
```

```
Help on function jointplot in module seaborn.axisgrid:

jointplot(*, x=None, y=None, data=None, kind='scatter', color=None, height
=6, ratio=5, space=0.2, dropna=False, xlim=None, ylim=None, marginal_ticks
=False, joint_kws=None, marginal_kws=None, hue=None, palette=None, hue_ord
er=None, hue_norm=None, **kwargs)
    Draw a plot of two variables with bivariate and univariate graphs.

    This function provides a convenient interface to the :class:`JointGrid
`
    class, with several canned plot kinds. This is intended to be a fairly
    lightweight wrapper; if you need more flexibility, you should use
    :class:`JointGrid` directly.

    Parameters
    ----------
    x, y : vectors or keys in ``data``
        Variables that specify positions on the x and y axes.
    data : :class:`pandas.DataFrame`, :class:`numpy.ndarray`, mapping, or
sequence
        Input data structure. Either a long-form collection of vectors tha
t can be
        assigned to named variables or a wide-form dataset that will be in
ternally
        reshaped.
    kind : { "scatter" | "kde" | "hist" | "hex" | "reg" | "resid" }
        Kind of plot to draw. See the examples for references to the under
lying functions.
    color : :mod:`matplotlib color <matplotlib.colors>`
        Single color specification for when hue mapping is not used. Other
wise, the
        plot will try to hook into the matplotlib property cycle.
    height : numeric
        Size of the figure (it will be square).
    ratio : numeric
        Ratio of joint axes height to marginal axes height.
    space : numeric
        Space between the joint and marginal axes
    dropna : bool
        If True, remove observations that are missing from ``x`` and ``y`
`.
    {x, y}lim : pairs of numbers
        Axis limits to set before plotting.
    marginal_ticks : bool
        If False, suppress ticks on the count/density axis of the marginal
plots.
    {joint, marginal}_kws : dicts
        Additional keyword arguments for the plot components.
    hue : vector or key in ``data``
        Semantic variable that is mapped to determine the color of plot el
ements.
        Semantic variable that is mapped to determine the color of plot el
ements.
    palette : string, list, dict, or :class:`matplotlib.colors.Colormap`
        Method for choosing the colors to use when mapping the ``hue`` sem
antic.
        String values are passed to :func:`color_palette`. List or dict va
lues
        imply categorical mapping, while a colormap object implies numeric
mapping.
    hue_order : vector of strings
```

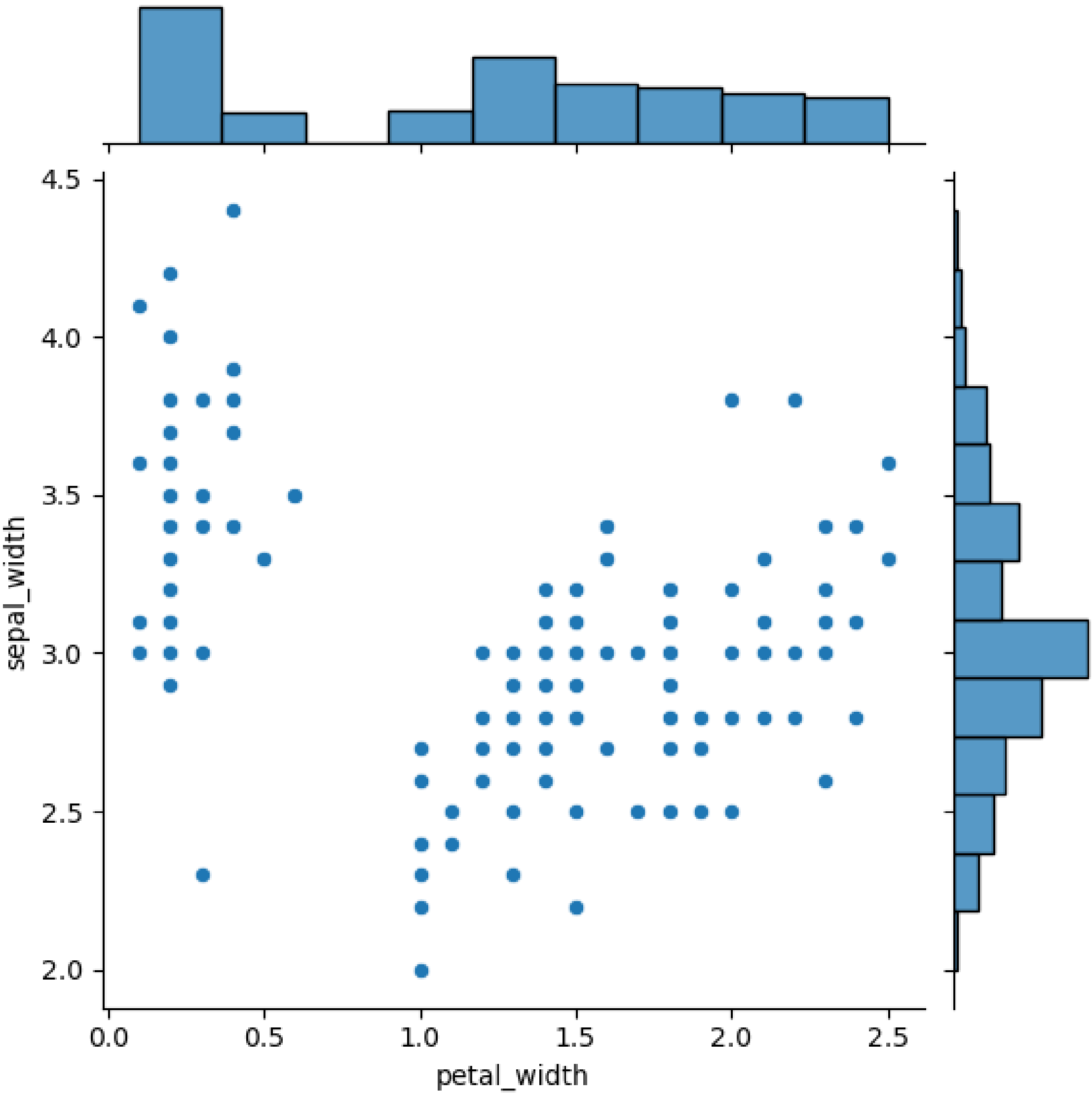
```
        Specify the order of processing and plotting for categorical level
s of the
        ``hue`` semantic.
    hue_norm : tuple or :class:`matplotlib.colors.Normalize`
        Either a pair of values that set the normalization range in data u
nits
        or an object that will map from data units into a [0, 1] interval.
Usage
        implies numeric mapping.
    kwargs
        Additional keyword arguments are passed to the function used to
        draw the plot on the joint Axes, superseding items in the
        ``joint_kws`` dictionary.

    Returns
    -------
    :class:`JointGrid`
        An object managing multiple subplots that correspond to joint and
marginal axes
        for plotting a bivariate relationship or distribution.

    See Also
    --------
    JointGrid : Set up a figure with joint and marginal views on bivariate
data.
    PairGrid : Set up a figure with joint and marginal views on multiple v
ariables.
    jointplot : Draw multiple bivariate plots with univariate marginal dis
tributions.

    Examples
    --------

    .. include:: ../docstrings/jointplot.rst
```

In [49]:

```
sns.jointplot(x="petal_width",y="sepal_width",data=iris)
```

Out[49]:

```
<seaborn.axisgrid.JointGrid at 0x25e54296310>
```

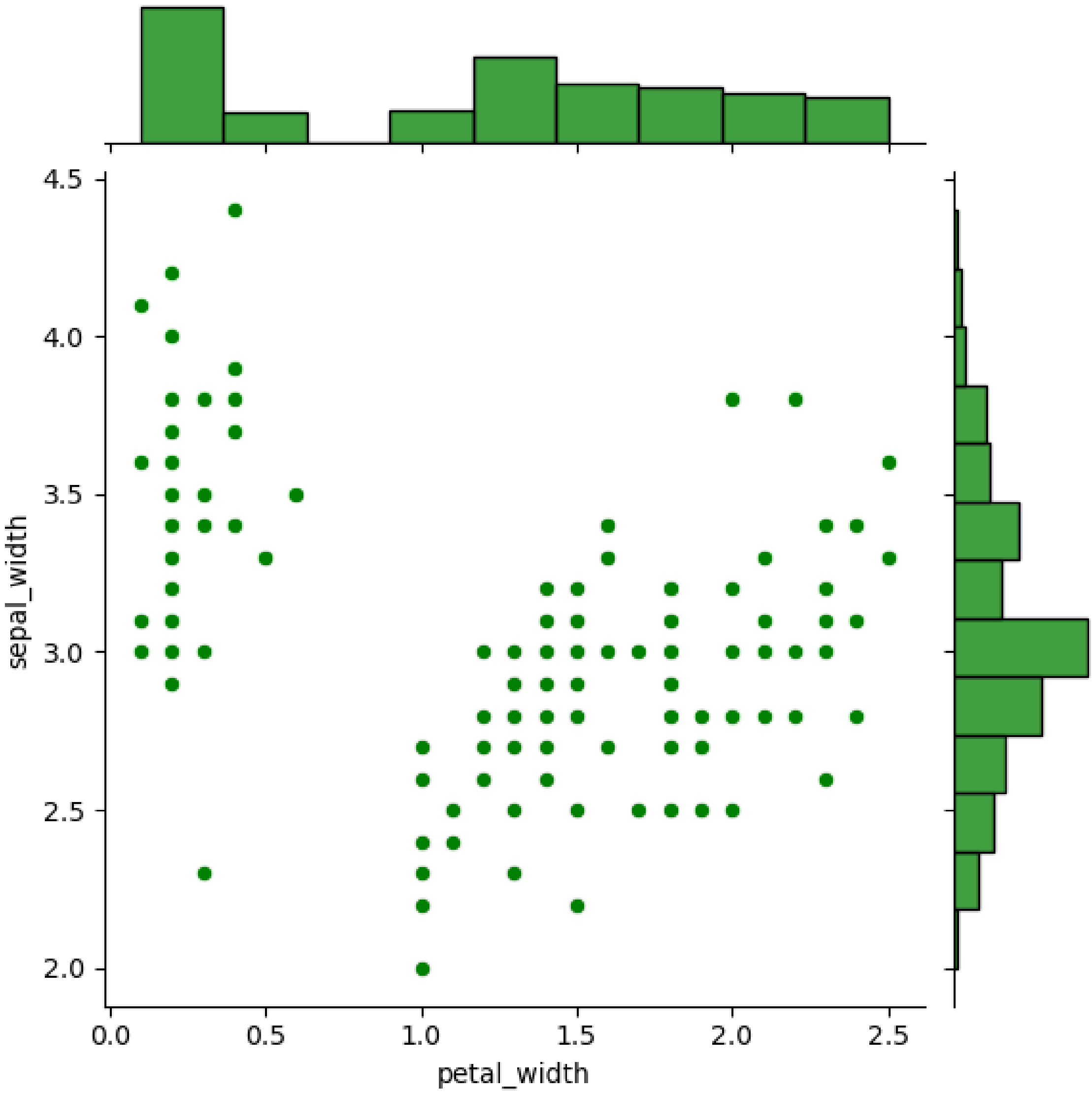


In [51]:

```
sns.jointplot(x="petal_width",y="sepal_width",data=iris,color="g")
```

Out[51]:

```
<seaborn.axisgrid.JointGrid at 0x25e56c9beb0>
```

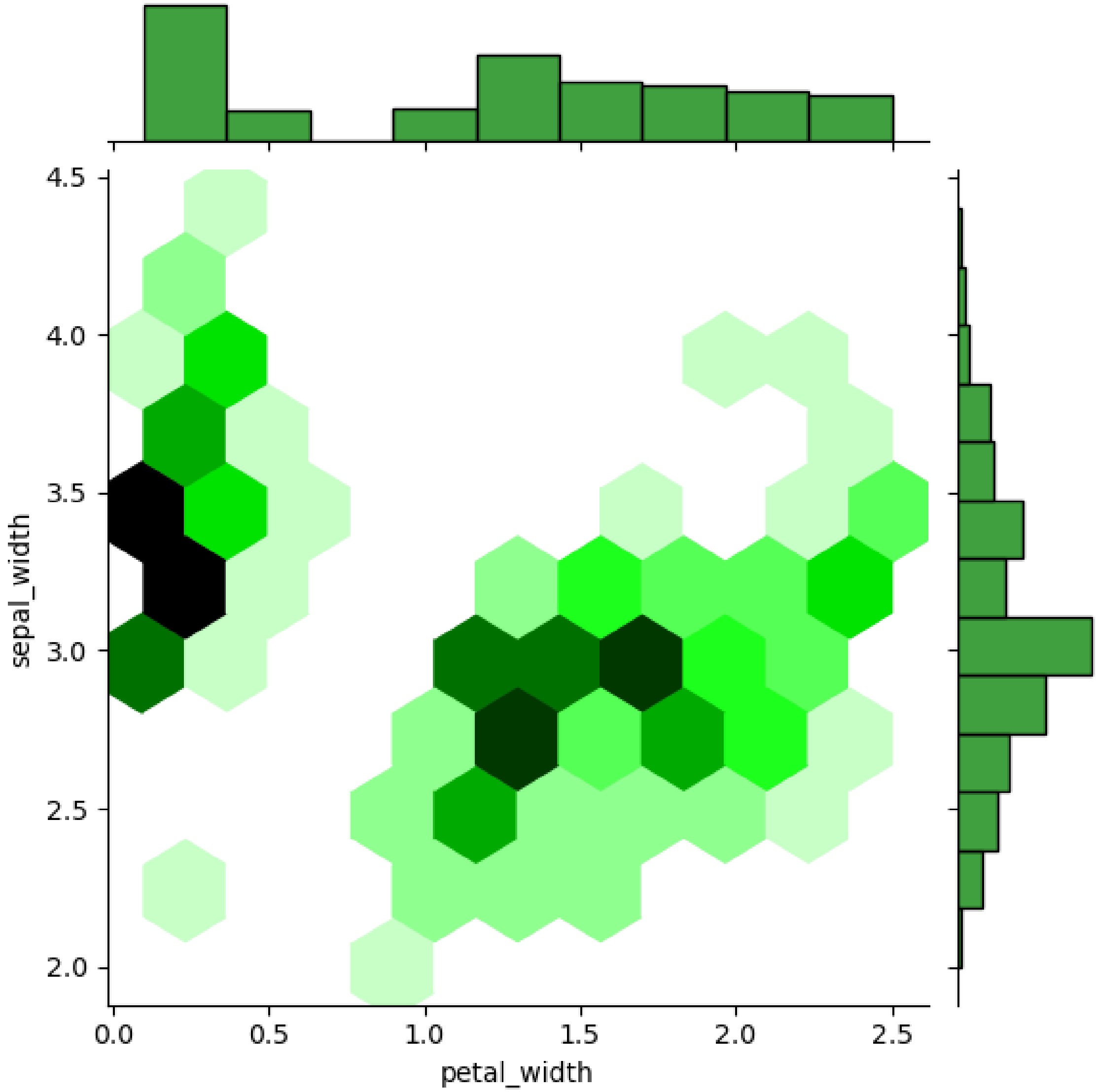


In [52]:

```
sns.jointplot(x="petal_width",y="sepal_width",kind="hex",data=iris,color="g")
```

Out[52]:

```
<seaborn.axisgrid.JointGrid at 0x25e56d6a640>
```



pairplot

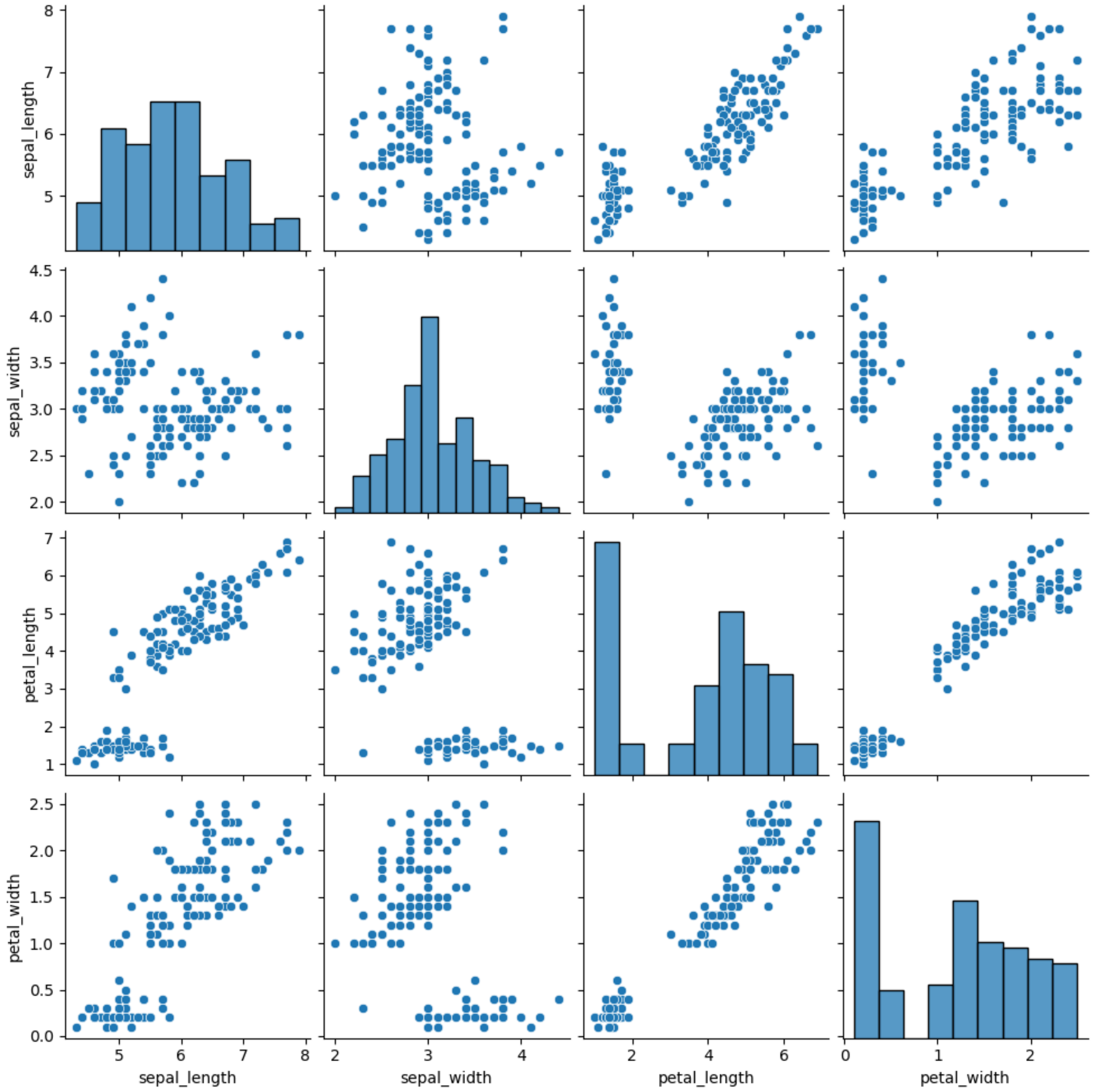
- It displays multiple plots at a time in single graph
- pairwise relationship
- By default it returns scatter plot
- sns.pairplot()

In [53]:

```
sns.pairplot(iris)
```

Out[53]:

```
<seaborn.axisgrid.PairGrid at 0x25e56767100>
```

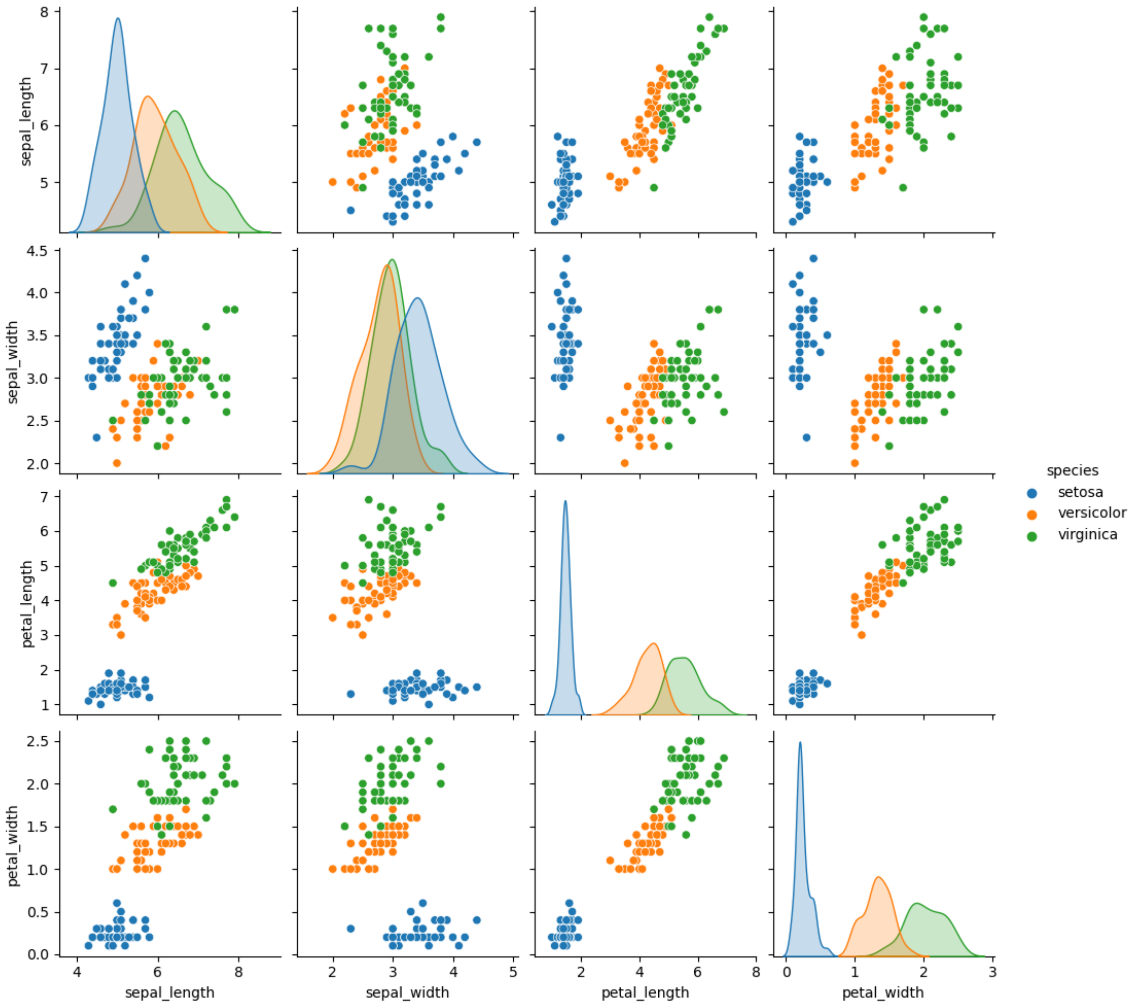


In [54]:

```
sns.pairplot(iris,hue="species")
```

Out[54]:

```
<seaborn.axisgrid.PairGrid at 0x25e543262b0>
```



In [56]:

```
help(sns.pairplot)
```

```
Help on function pairplot in module seaborn.axisgrid:

pairplot(data, *, hue=None, hue_order=None, palette=None, vars=None, x_var
s=None, y_vars=None, kind='scatter', diag_kind='auto', markers=None, heigh
t=2.5, aspect=1, corner=False, dropna=False, plot_kws=None, diag_kws=None,
grid_kws=None, size=None)
    Plot pairwise relationships in a dataset.

    By default, this function will create a grid of Axes such that each nu
meric
    variable in ``data`` will by shared across the y-axes across a single
row and
    the x-axes across a single column. The diagonal plots are treated
    differently: a univariate distribution plot is drawn to show the margi
nal
    distribution of the data in each column.

    It is also possible to show a subset of variables or plot different
    variables on the rows and columns.

    This is a high-level interface for :class:`PairGrid` that is intended
to
    make it easy to draw a few common styles. You should use :class:`PairG
rid`
    directly if you need more flexibility.

    Parameters
    ----------
    data : `pandas.DataFrame`
        Tidy (long-form) dataframe where each column is a variable and
        each row is an observation.
    hue : name of variable in ``data``
        Variable in ``data`` to map plot aspects to different colors.
    hue_order : list of strings
        Order for the levels of the hue variable in the palette
    palette : dict or seaborn color palette
        Set of colors for mapping the ``hue`` variable. If a dict, keys
        should be values  in the ``hue`` variable.
    vars : list of variable names
        Variables within ``data`` to use, otherwise use every column with
        a numeric datatype.
    {x, y}_vars : lists of variable names
        Variables within ``data`` to use separately for the rows and
        columns of the figure; i.e. to make a non-square plot.
    kind : {'scatter', 'kde', 'hist', 'reg'}
        Kind of plot to make.
    diag_kind : {'auto', 'hist', 'kde', None}
        Kind of plot for the diagonal subplots. If 'auto', choose based on
        whether or not ``hue`` is used.
    markers : single matplotlib marker code or list
        Either the marker to use for all scatterplot points or a list of m
arkers
        with a length the same as the number of levels in the hue variable
so that
        differently colored points will also have different scatterplot
        markers.
    height : scalar
        Height (in inches) of each facet.
    aspect : scalar
        Aspect * height gives the width (in inches) of each facet.
    corner : bool
```

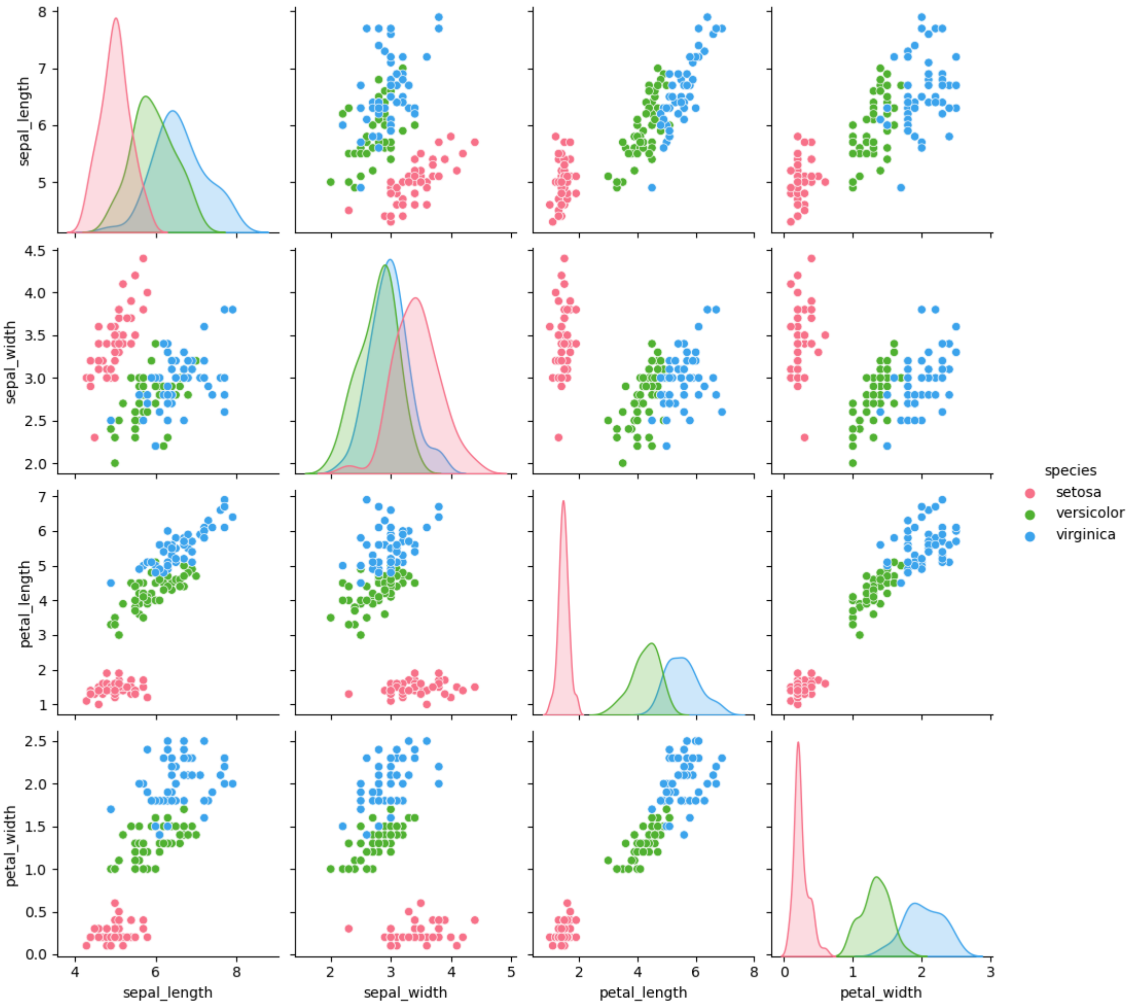
```
        If True, don't add axes to the upper (off-diagonal) triangle of th
e
        grid, making this a "corner" plot.
    dropna : boolean
        Drop missing values from the data before plotting.
    {plot, diag, grid}_kws : dicts
        Dictionaries of keyword arguments. ``plot_kws`` are passed to the
        bivariate plotting function, ``diag_kws`` are passed to the univar
iate
        plotting function, and ``grid_kws`` are passed to the :class:`Pair
Grid`
        constructor.

    Returns
    -------
    grid : :class:`PairGrid`
        Returns the underlying :class:`PairGrid` instance for further twea
king.

    See Also
    --------
    PairGrid : Subplot grid for more flexible plotting of pairwise relatio
nships.
    JointGrid : Grid for plotting joint and marginal distributions of two
variables.

    Examples
    --------

    .. include:: ../docstrings/pairplot.rst
```

In [57]:

```
sns.pairplot(iris,hue="species",palette="husl")
```

Out[57]:

```
<seaborn.axisgrid.PairGrid at 0x25e575f9fa0>
```

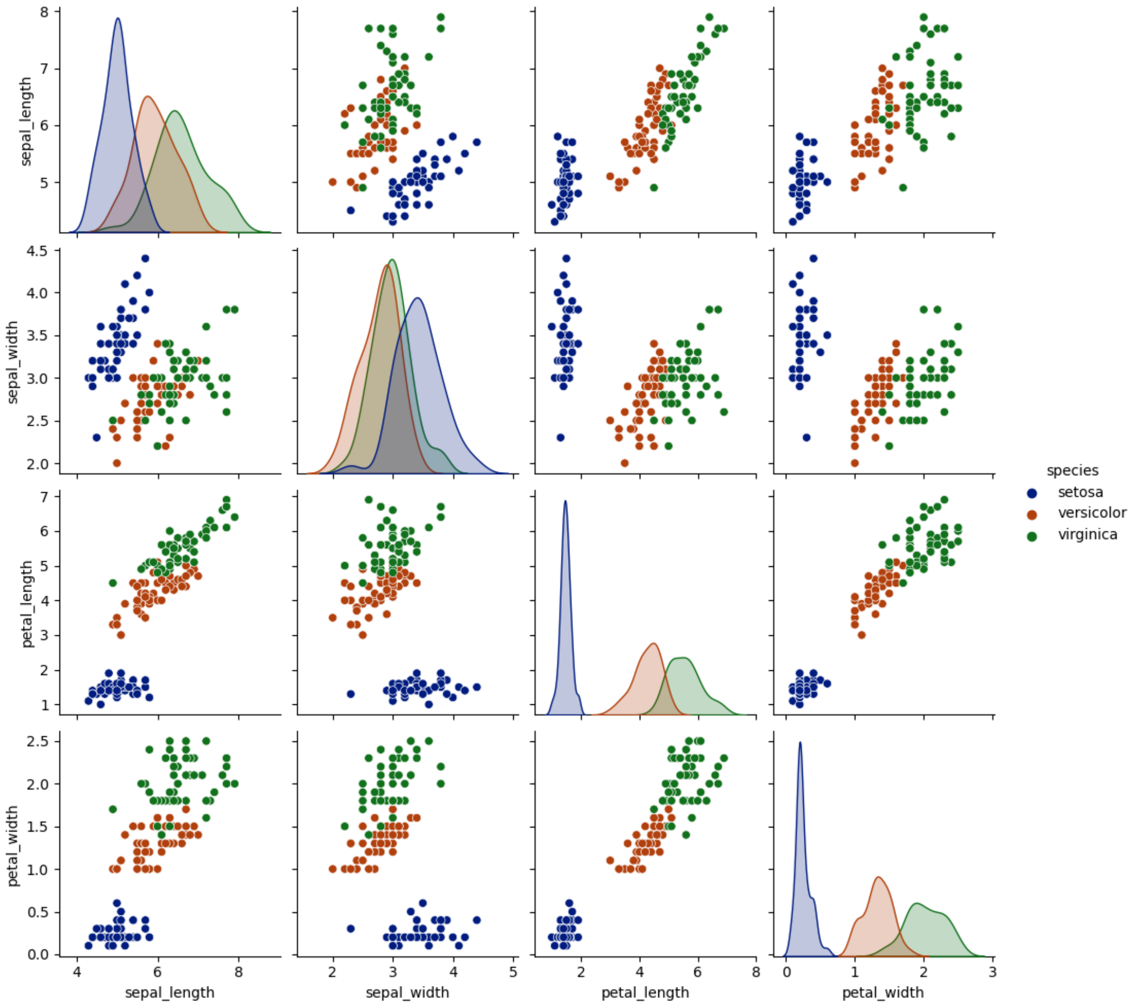


In [58]:

```
sns.pairplot(iris,hue="species",palette="dark")
```

Out[58]:

```
<seaborn.axisgrid.PairGrid at 0x25e59042b20>
```

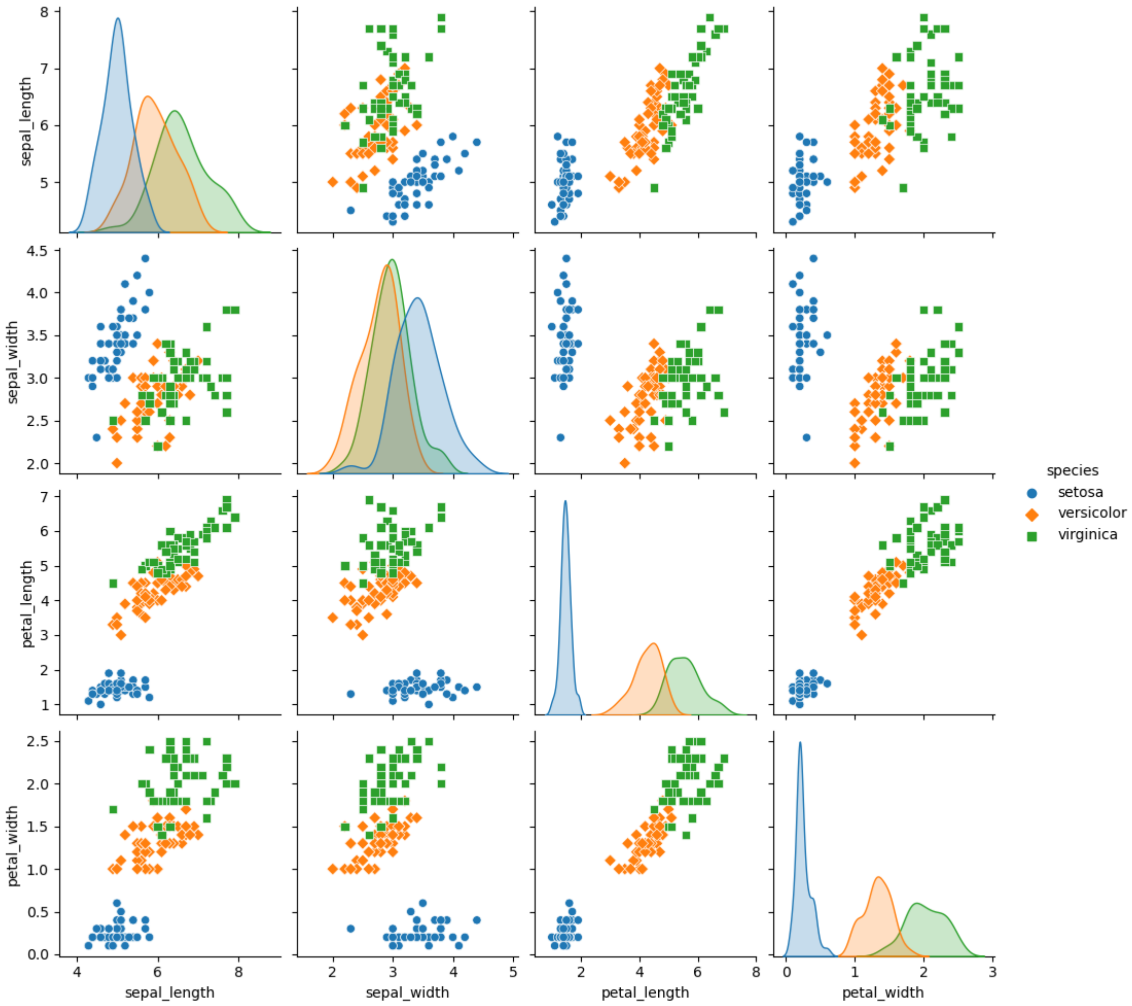


In [59]:

```
sns.pairplot(iris,hue="species",markers=['o','D','s'])
```

Out[59]:

```
<seaborn.axisgrid.PairGrid at 0x25e56d5cd60>
```

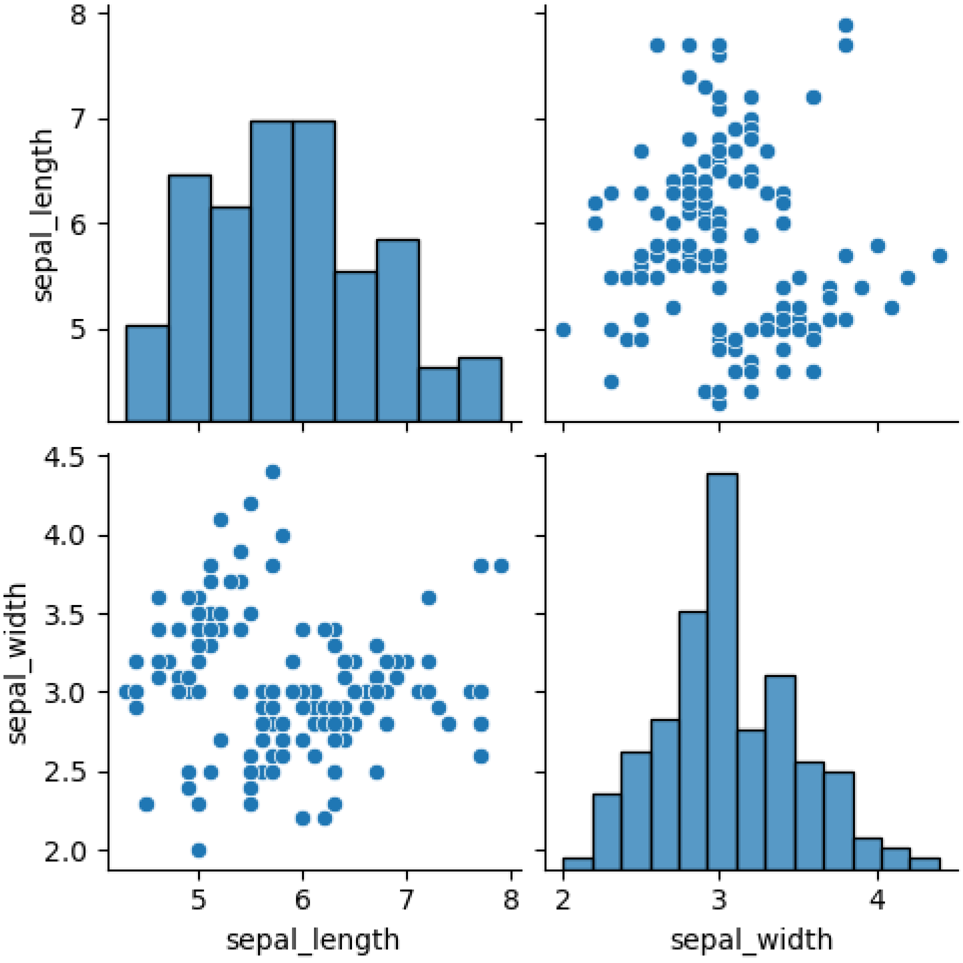


In [60]:

```
sns.pairplot(iris,vars=["sepal_length","sepal_width"])
```

Out[60]:

```
<seaborn.axisgrid.PairGrid at 0x25e5955c790>
```



Heatmaps

- it is used to correlation between two columns in data
- sns.heatmap()

In [62]:

```
help(sns.heatmap)
```

```
Help on function heatmap in module seaborn.matrix:

heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, robust=F
alse, annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='w
hite', cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabel
s='auto', yticklabels='auto', mask=None, ax=None, **kwargs)
    Plot rectangular data as a color-encoded matrix.

    This is an Axes-level function and will draw the heatmap into the
    currently-active Axes if none is provided to the ``ax`` argument.
Part of
    this Axes space will be taken and used to plot a colormap, unless `
`cbar``
    is False or a separate Axes is provided to ``cbar_ax``.

    Parameters
    ----------
    data : rectangular dataset
        2D dataset that can be coerced into an ndarray. If a Pandas Dat
```

In [64]:

```
t=sns.load_dataset("tips")
```

In [65]:

```
t
```

Out[65]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [66]:

```
t.head()
```

Out[66]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [67]:

```
t.corr()
```

Out[67]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [69]:

```
help(t.corr)
```

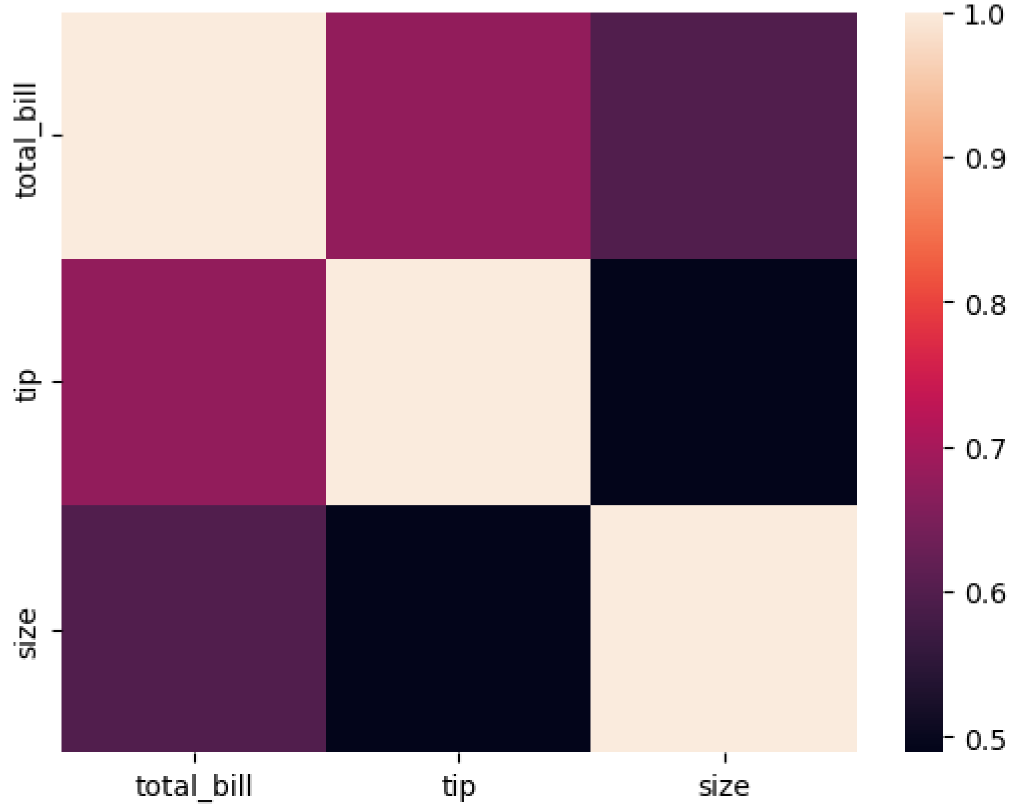
```
Help on method corr in module pandas.core.frame:

corr(method: 'str | Callable[[np.ndarray, np.ndarray], float]' = 'pearso
n', min_periods: 'int' = 1) -> 'DataFrame' method of pandas.core.frame.Dat
aFrame instance
    Compute pairwise correlation of columns, excluding NA/null values.

    Parameters
    ----------
    method : {'pearson', 'kendall', 'spearman'} or callable
        Method of correlation:

        * pearson : standard correlation coefficient
        * kendall : Kendall Tau correlation coefficient
        * spearman : Spearman rank correlation
        * callable: callable with input two 1d ndarrays
            and returning a float. Note that the returned matrix from corr
            will have 1 along the diagonals and will be symmetric
            regardless of the callable's behavior.
    min_periods : int, optional
        Minimum number of observations required per pair of columns
        to have a valid result. Currently only available for Pearson
        and Spearman correlation.

    Returns
    -------
    DataFrame
        Correlation matrix.

    See Also
    --------
    DataFrame.corrwith : Compute pairwise correlation with another
        DataFrame or Series.
    Series.corr : Compute the correlation between two Series.

    Examples
    --------
    >>> def histogram_intersection(a, b):
    ...     v = np.minimum(a, b).sum().round(decimals=1)
    ...     return v
    >>> df = pd.DataFrame([(.2, .3), (.0, .6), (.6, .0), (.2, .1)],
    ...                   columns=['dogs', 'cats'])
    >>> df.corr(method=histogram_intersection)
          dogs  cats
    dogs   1.0   0.3
    cats   0.3   1.0
```

In [68]:

```
sns.heatmap(t.corr())
```

Out[68]:

```
<AxesSubplot:>
```

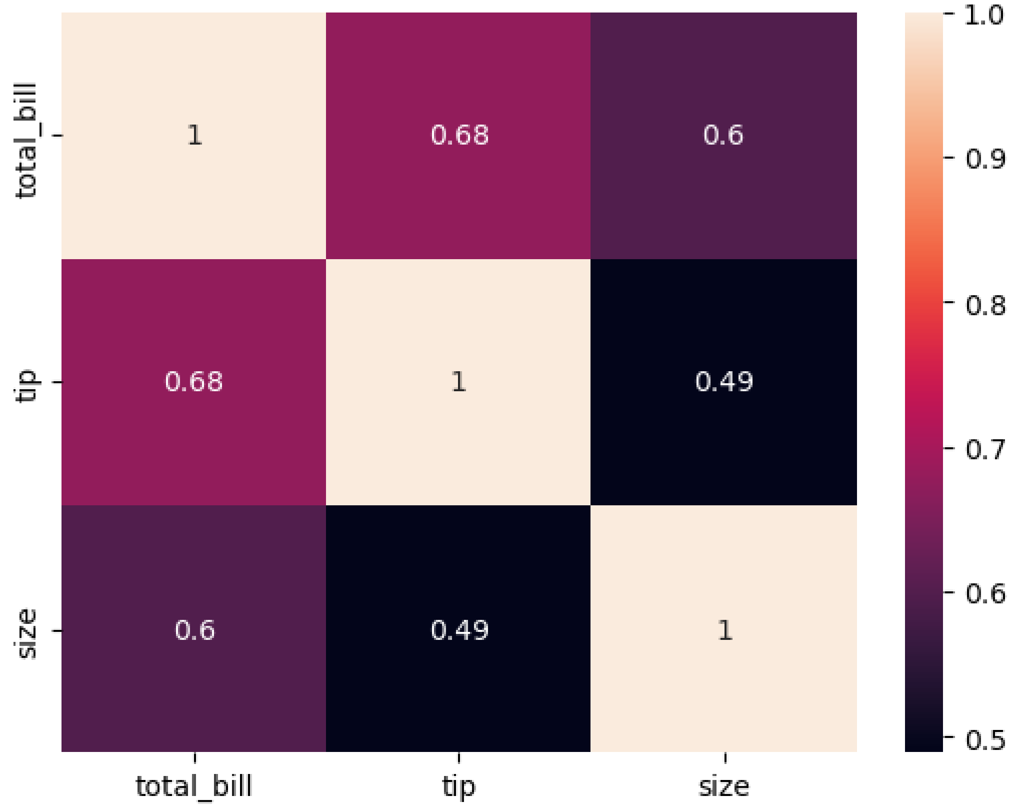


In [70]:

```
sns.heatmap(t.corr(),annot=True)
```

Out[70]:

```
<AxesSubplot:>
```

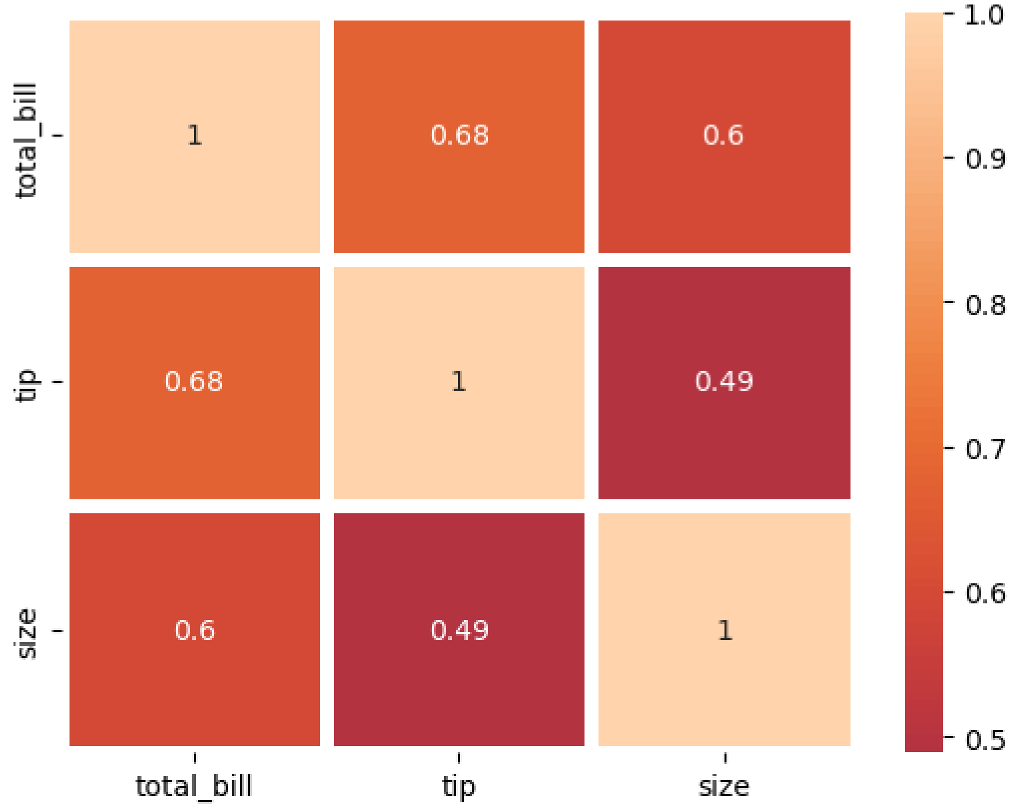


In [71]:

```
sns.heatmap(t.corr(),annot=True,center=0,linewidths=5,linecolor='white')
```

Out[71]:

```
<AxesSubplot:>
```

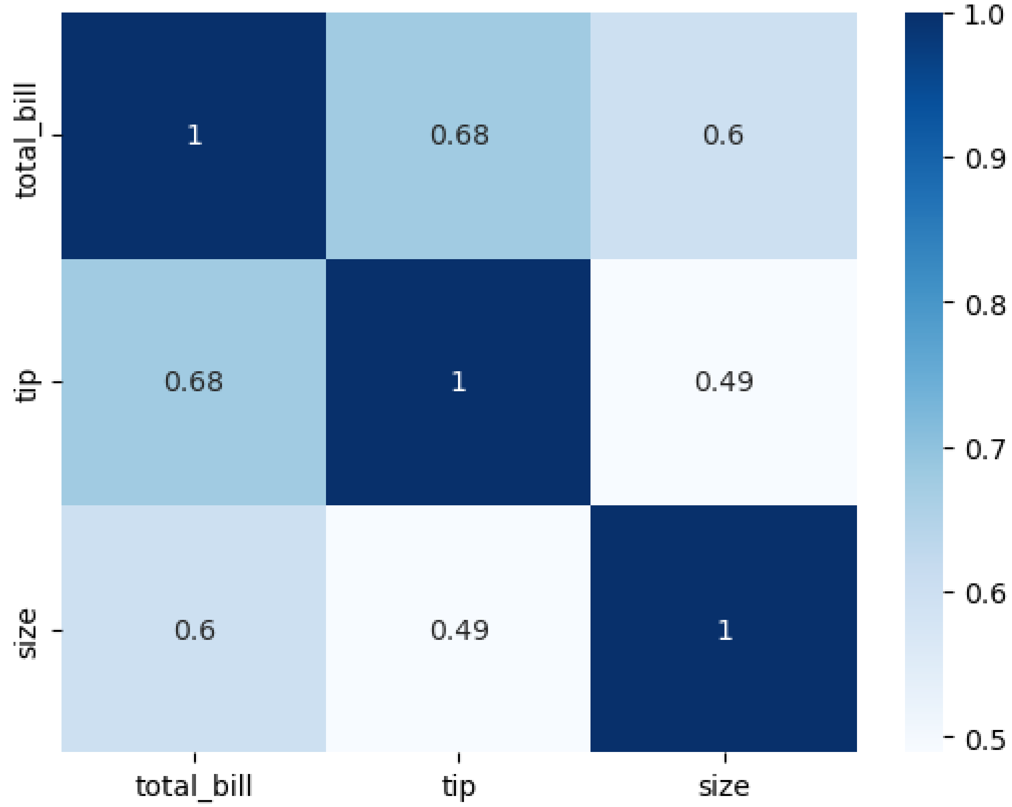


In [72]:

```
sns.heatmap(t.corr(),annot=True,cmap="Blues")
```

Out[72]:

```
<AxesSubplot:>
```

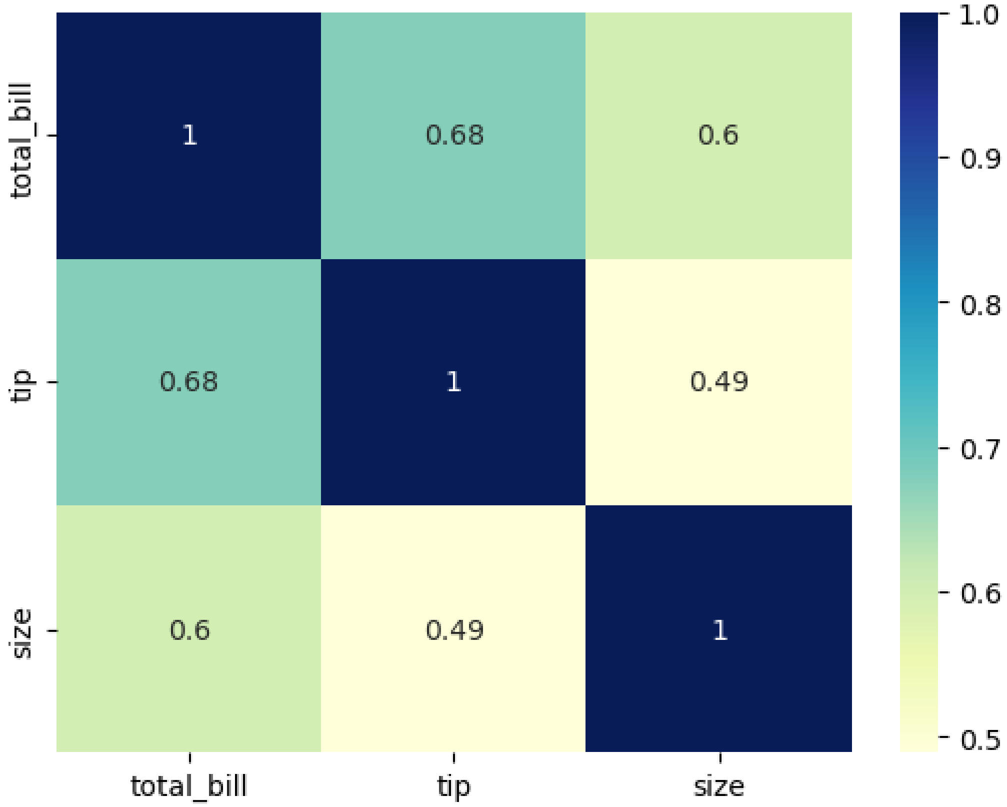


In [73]:

```
sns.heatmap(t.corr(),annot=True,cmap="YlGnBu")
```

Out[73]:

```
<AxesSubplot:>
```



In [ ]: