In [1]:

```
import seaborn as sn
```

In [3]:

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

In [4]:

```
iris
```

Out[4]:

	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

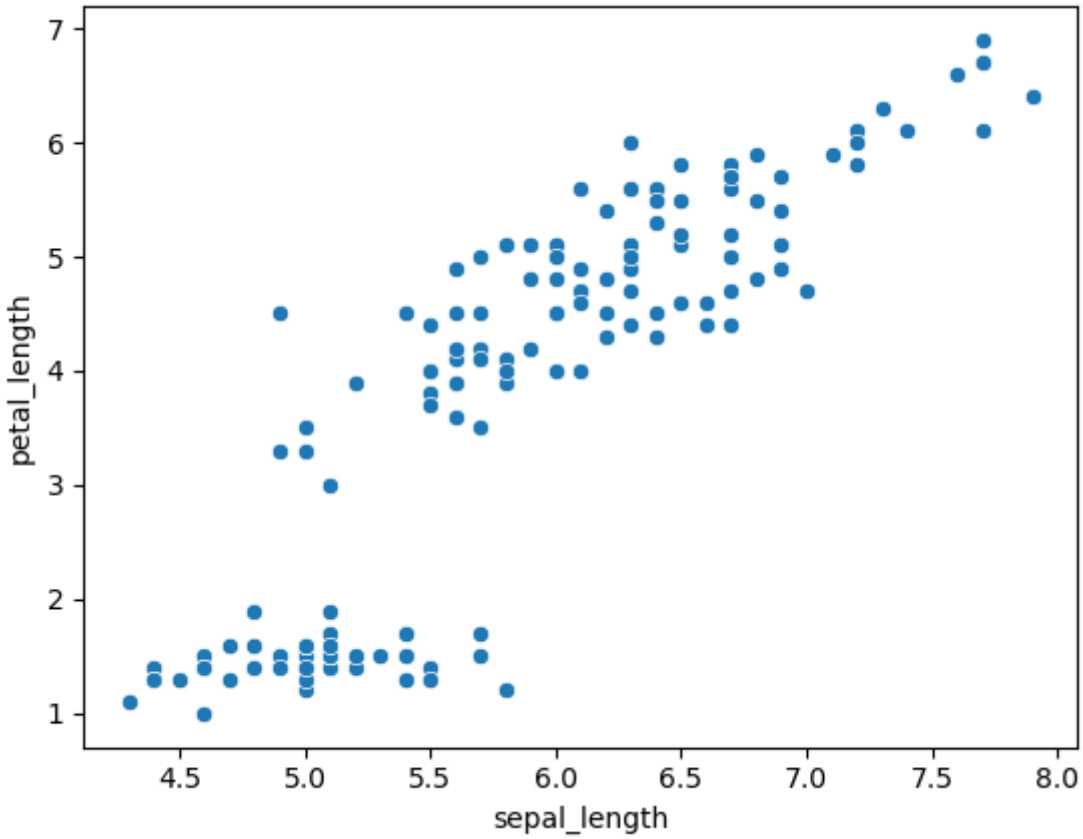
150 rows × 5 columns

In [7]:

```
sn.scatterplot(x= iris.sepal_length,y = iris.petal_length)
```

Out[7]:

```
<Axes: xlabel='sepal_length', ylabel='petal_length'>
```

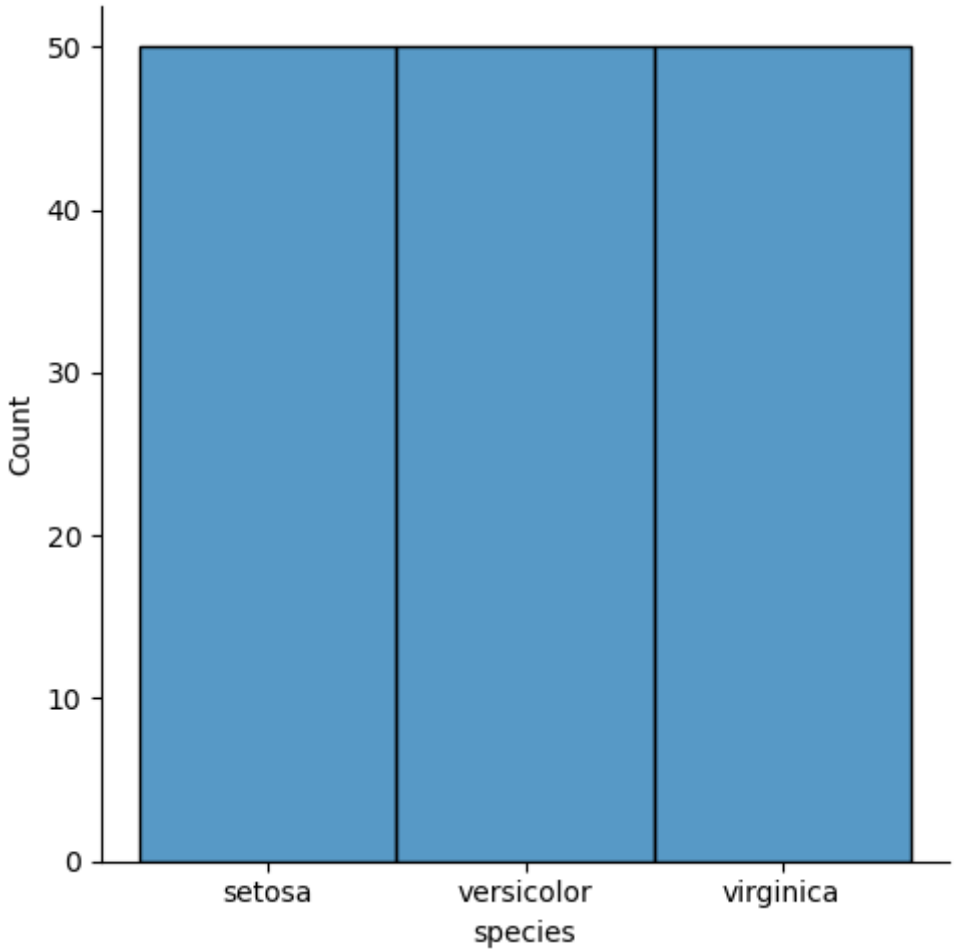


In [9]:

```
sn.displot(iris["species"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1f93d5d4460>
```

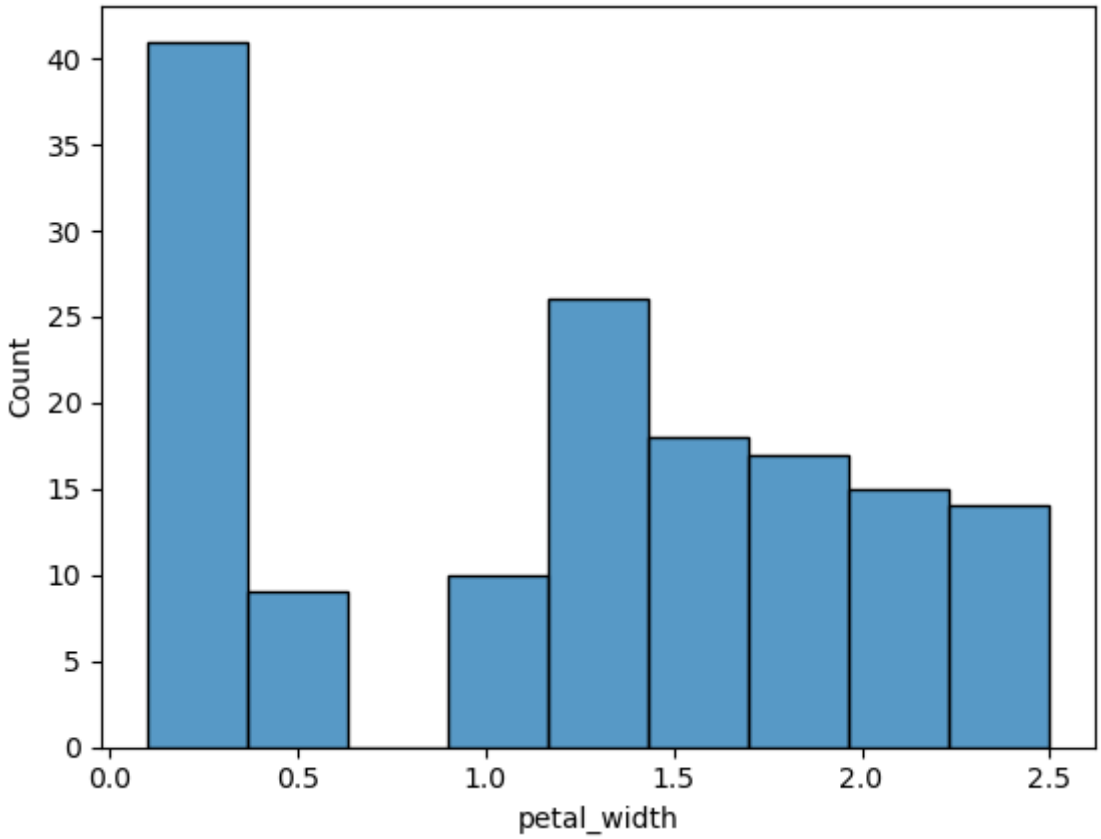


In [11]:

```
sn.histplot(iris.petal_width)
```

Out[11]:

```
<Axes: xlabel='petal_width', ylabel='Count'>
```



In [12]:

```
tips = sn.load_dataset("tips")
```

In [13]:

```
tips
```

Out[13]:

	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 [17]:

```
tips["smoker"].value_count()
```

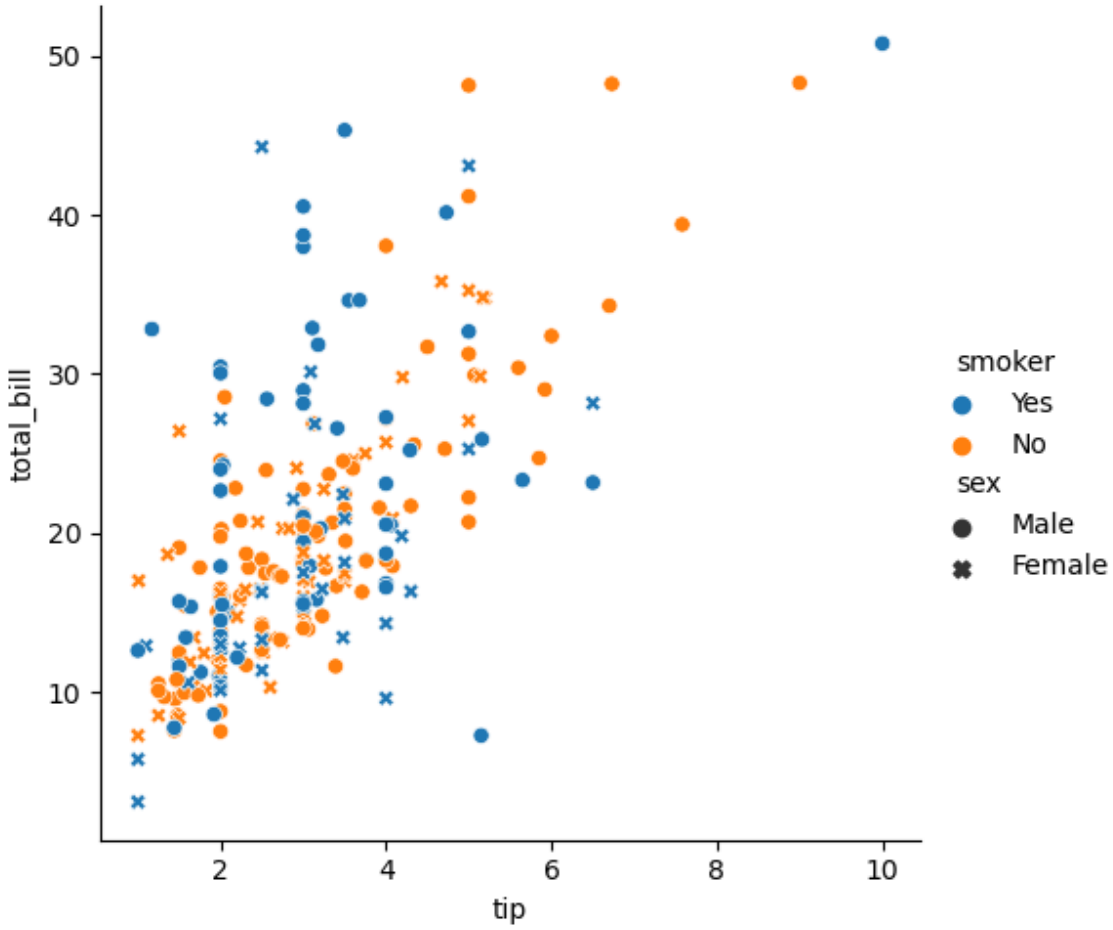
```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[17], line 1
----> 1 tips["smoker"].value_count()

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5902, in NDFram
e.__getattr__(self, name)
   5895 if (
   5896     name not in self._internal_names_set
   5897     and name not in self._metadata
   5898     and name not in self._accessors
   5899     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5900 ):
   5901     return self[name]
-> 5902 return object.__getattribute__(self, name)

AttributeError: 'Series' object has no attribute 'value_count'
```

In [22]:

```
sn.relplot(x =tips.tip,y = tips.total_bill,data= tips,style="sex",hue= "smoker" )
```

Out[22]:

```
<seaborn.axisgrid.FacetGrid at 0x1f944bfcca0>
```

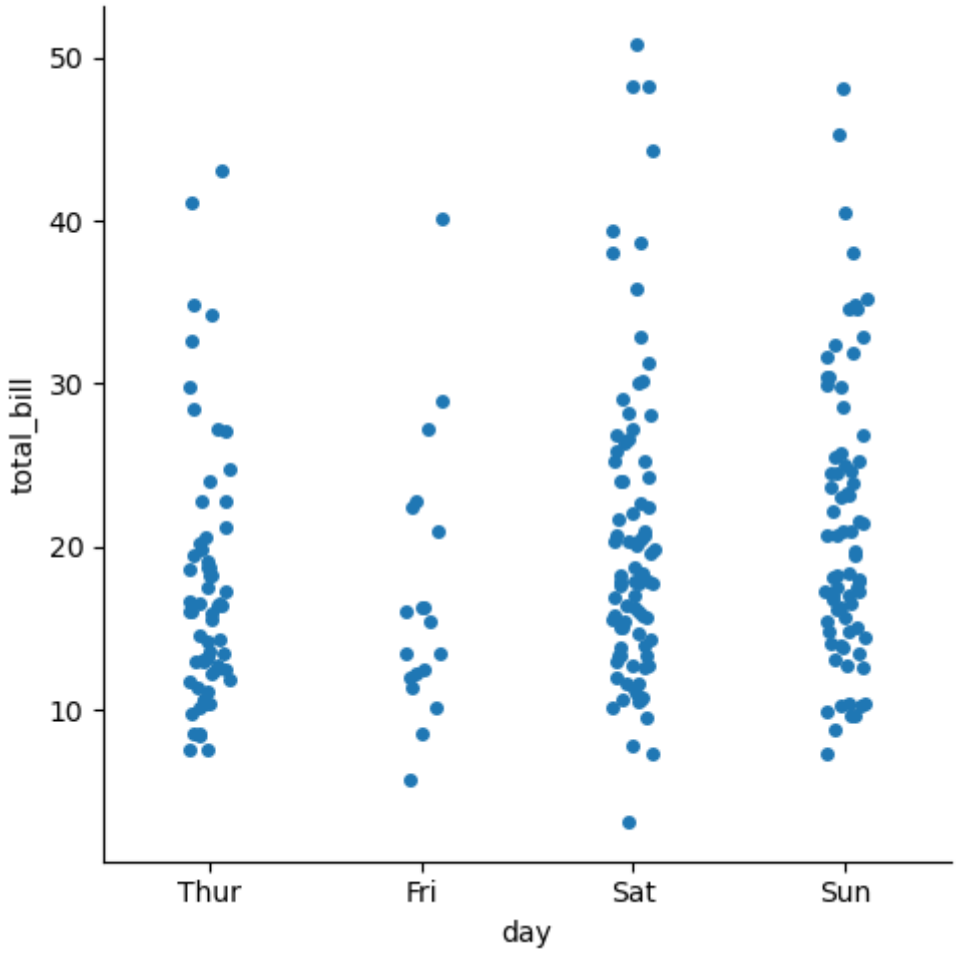


In [25]:

```
sn.catplot(x = tips.day,y = tips.total_bill,data= tips )
```

Out[25]:

```
<seaborn.axisgrid.FacetGrid at 0x1f944e575b0>
```

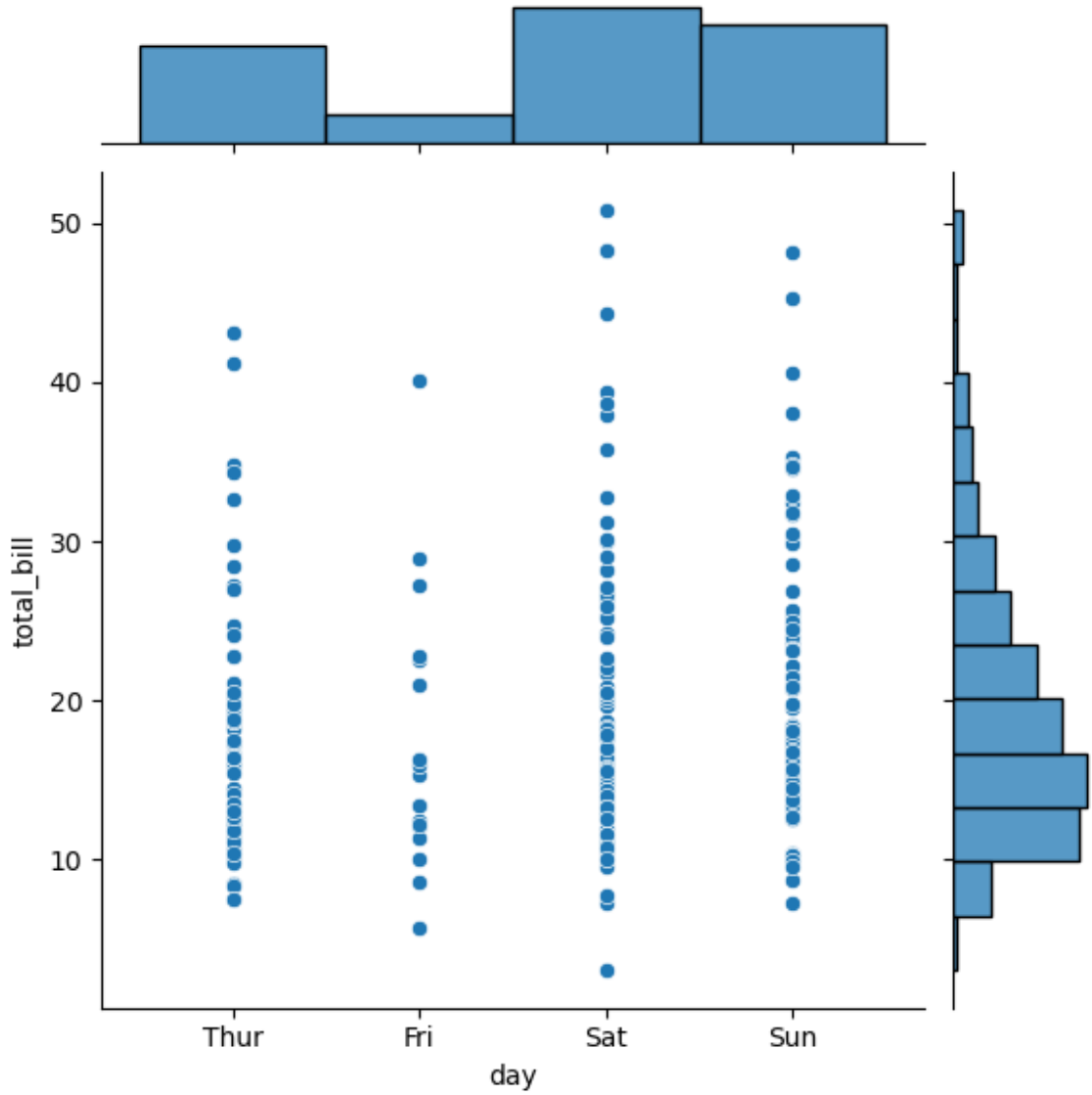


In [26]:

```
sn.jointplot(x = tips.day,y = tips.total_bill)
```

Out[26]:

```
<seaborn.axisgrid.JointGrid at 0x1f944e3fca0>
```

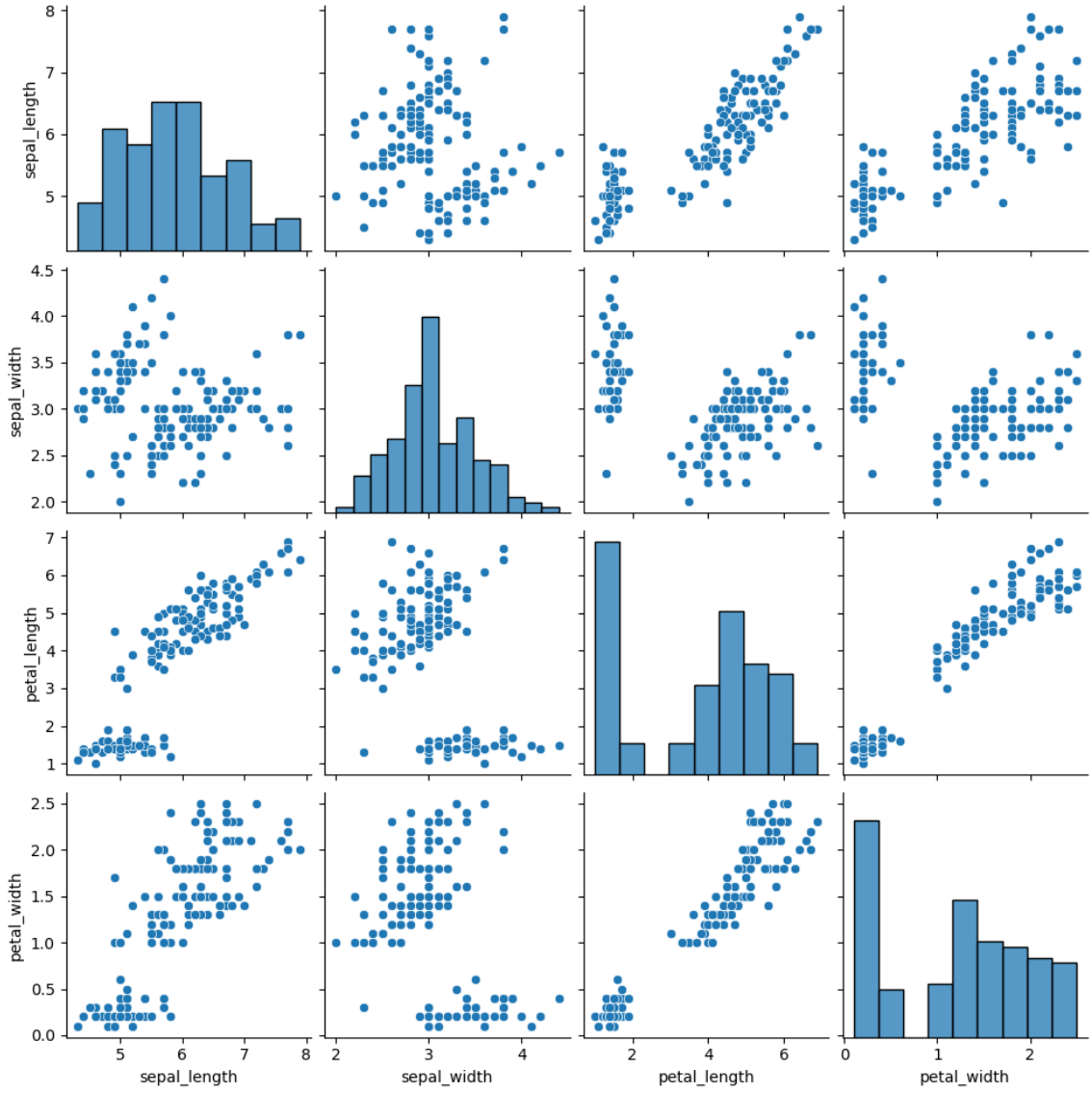


In [27]:

```
sn.pairplot(iris)
```

Out[27]:

```
<seaborn.axisgrid.PairGrid at 0x1f94522fac0>
```

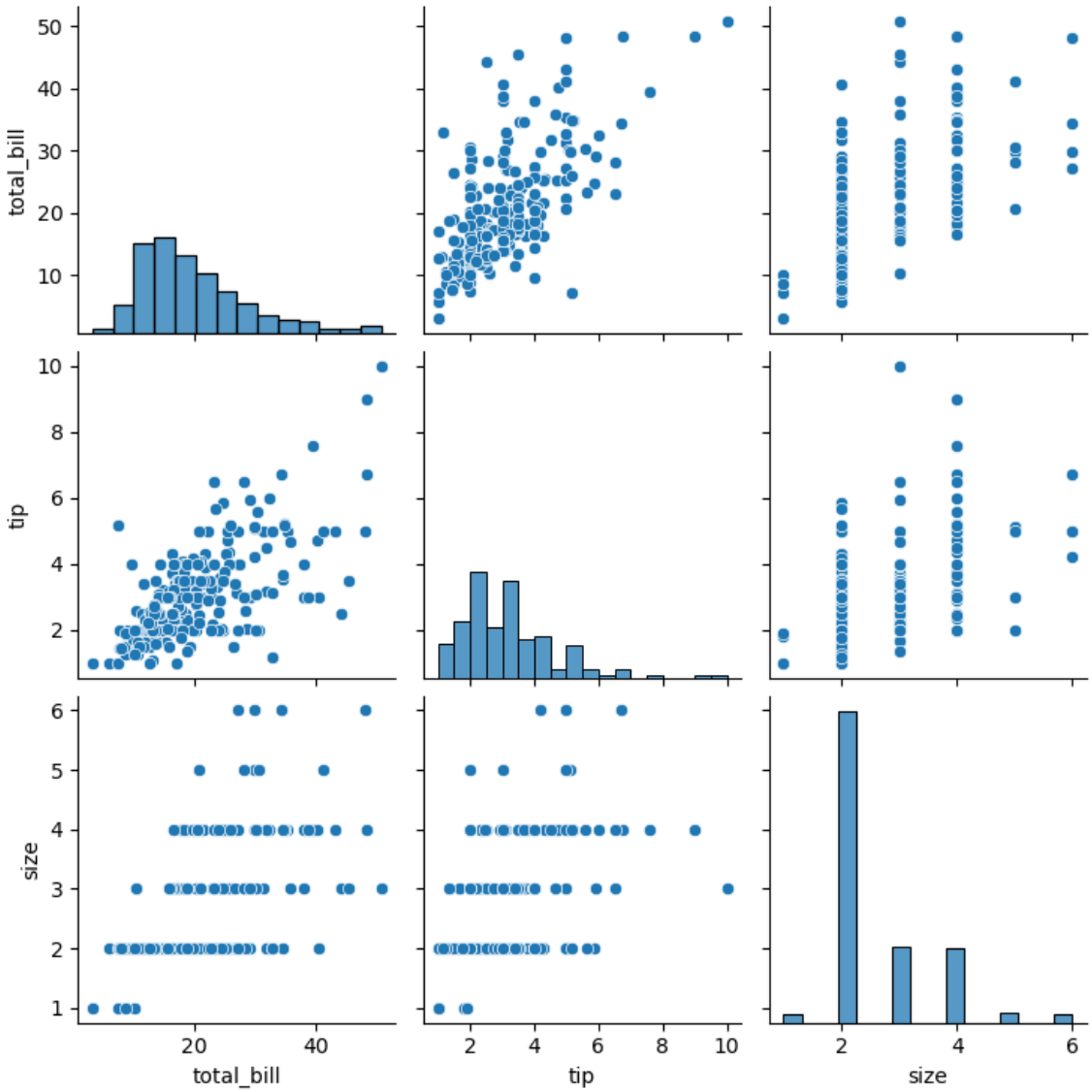


In [28]:

```
sn.pairplot(tips)
```

Out[28]:

```
<seaborn.axisgrid.PairGrid at 0x1f941a21090>
```

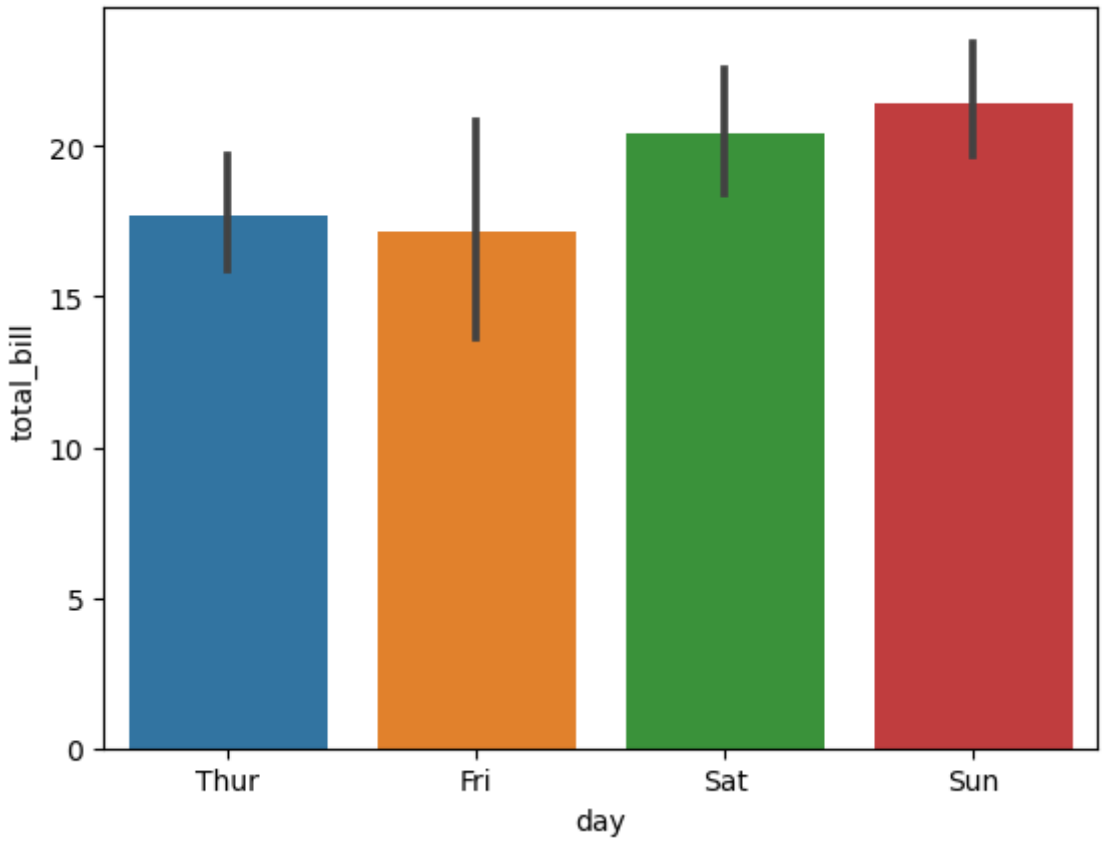


In [31]:

```
sn.barplot(x= tips.day,y = tips.total_bill)
```

Out[31]:

```
<Axes: xlabel='day', ylabel='total_bill'>
```

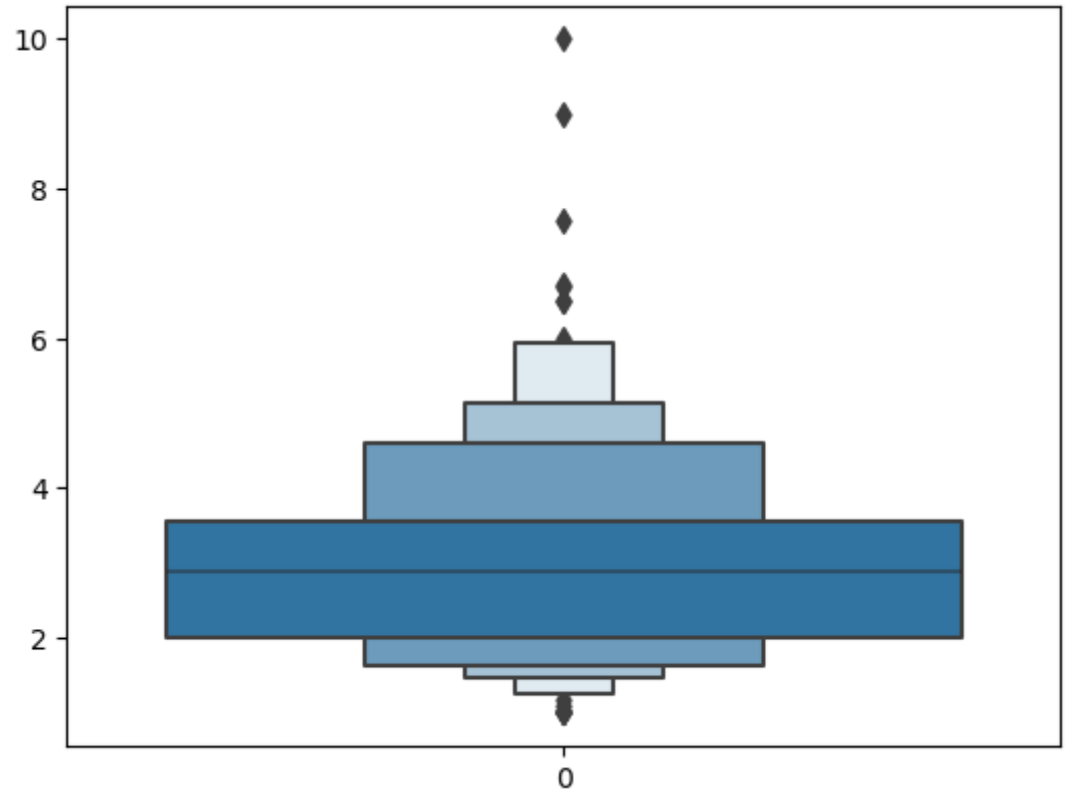


In [34]:

```
sn.boxenplot(tips.tip)
```

Out[34]:

<Axes: >

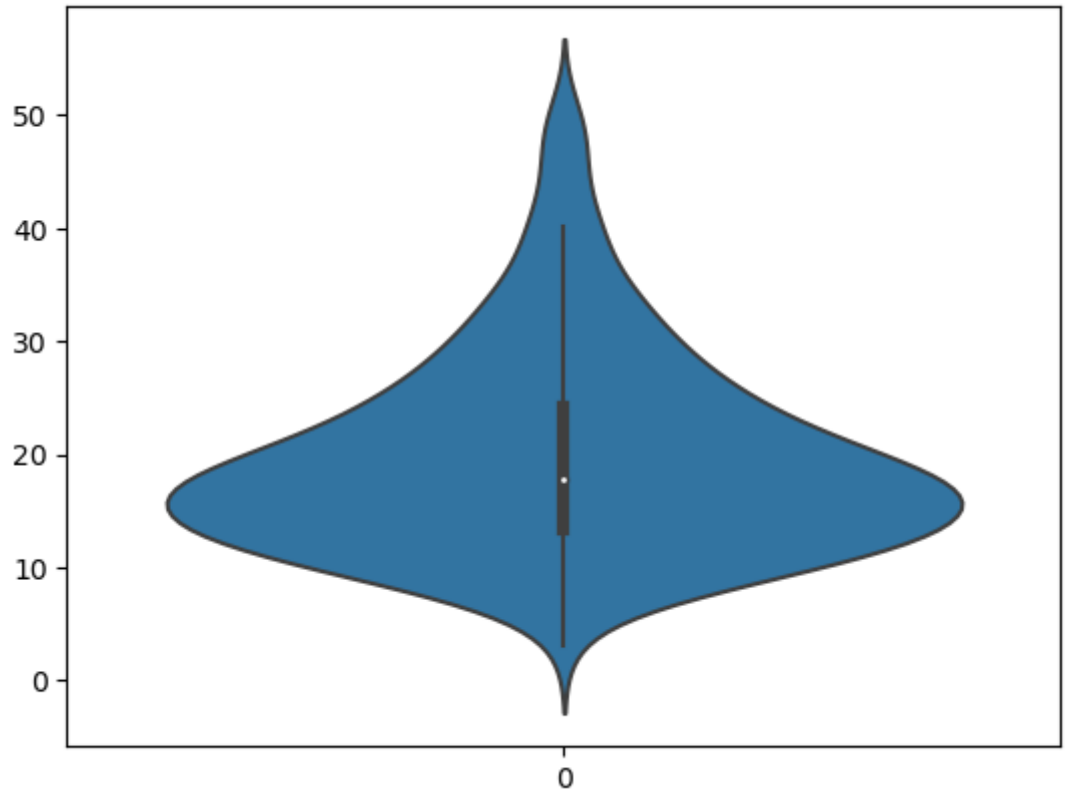


In [36]:

```
sn.violinplot(tips.total_bill)
```

Out[36]:

<Axes: >

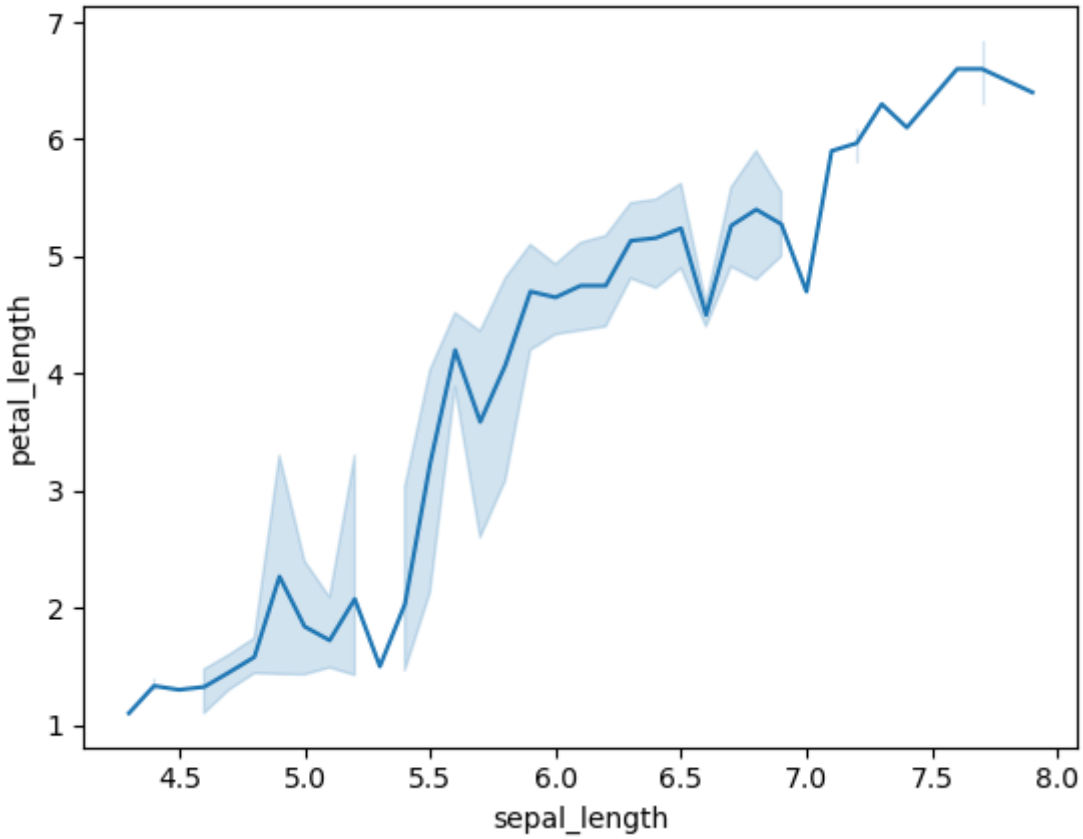


In [37]:

```
sn.lineplot(x= iris.sepal_length,y = iris.petal_length)
```

Out[37]:

```
<Axes: xlabel='sepal_length', ylabel='petal_length'>
```

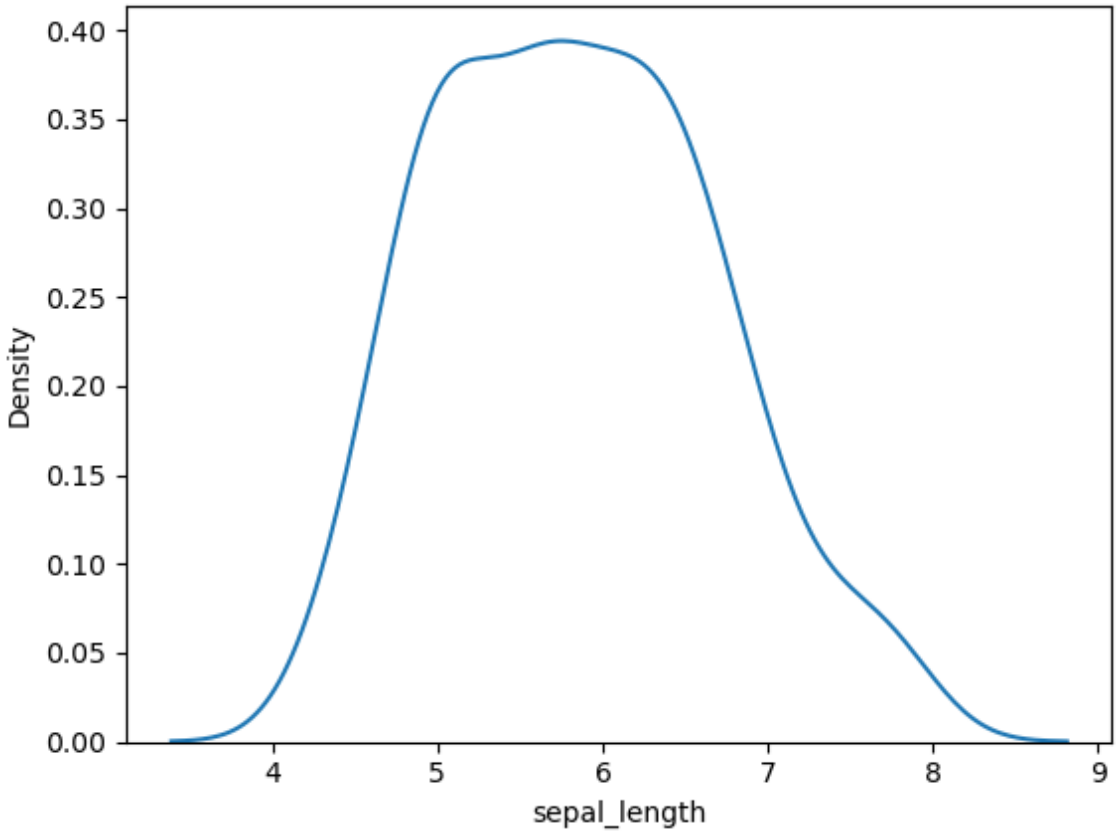


In [38]:

```
sn.kdeplot(iris.sepal_length)
```

Out[38]:

```
<Axes: xlabel='sepal_length', ylabel='Density'>
```



#heatmap Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

seaborn.heatmap() Syntax: seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, annot_kws=None, linewidths=0, linecolor='white', cbar=True, **kwargs)

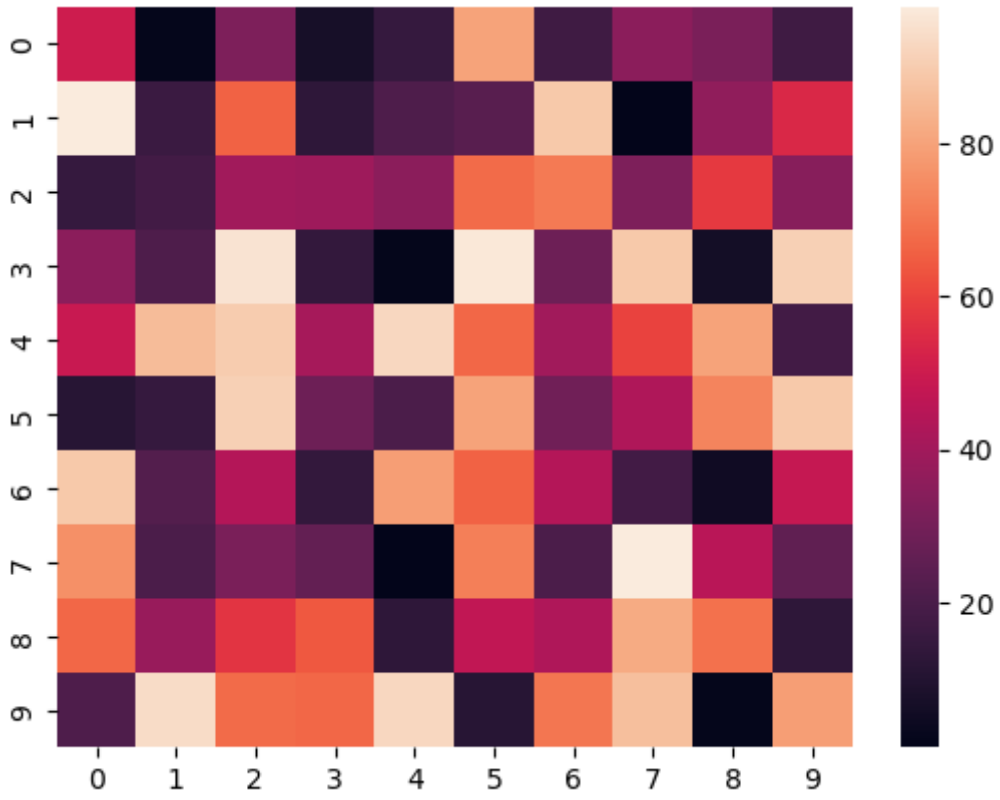
Important Parameters:

data: 2D dataset that can be coerced into an ndarray. vmin, vmax: Values to anchor the colormap, otherwise they are inferred from the data and other keyword arguments. cmap: The mapping from data values to color space. center: The value at which to center the colormap when plotting divergent data. annot: If True, write the data value in each cell. fmt: String formatting code to use when adding annotations. linewidths: Width of the lines that will divide each cell. linecolor: Color of the lines that will divide each cell. cbar: Whether to draw a colorbar.

In [47]:

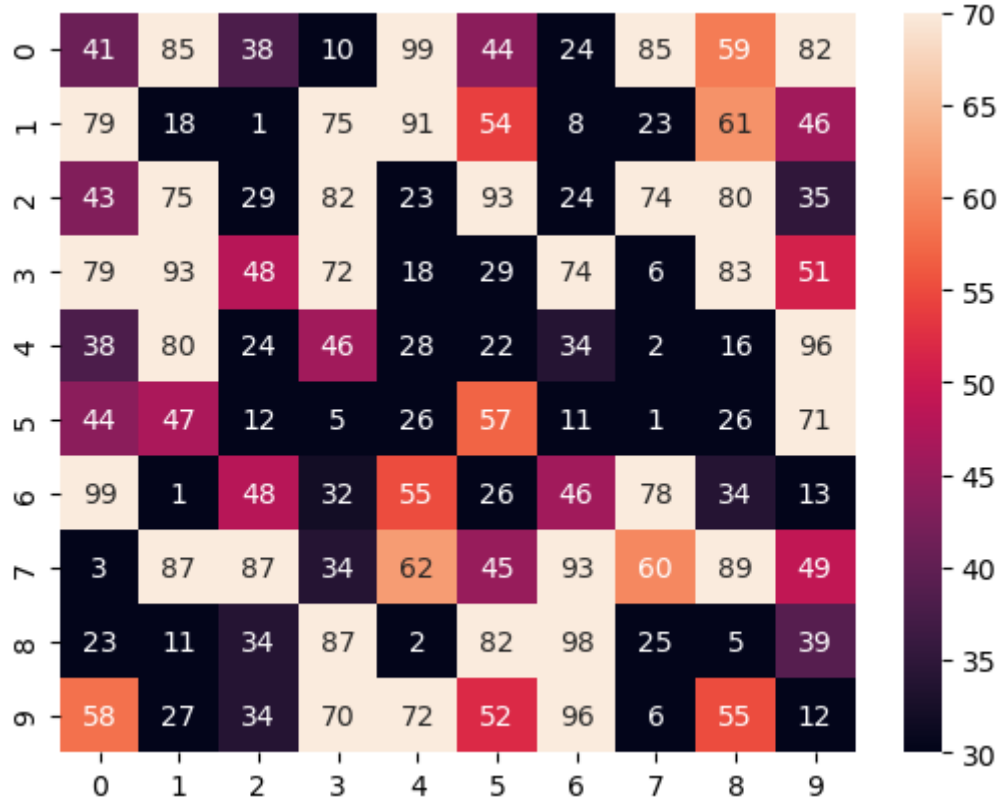
```
# importing the modules
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt

# generating 2-D 10x10 matrix of random numbers
# from 1 to 100
data = np.random.randint(low = 1,
                         high = 100,
                         size = (10, 10))
print("The data to be plotted:\n")
print(data)

# plotting the heatmap
hm = sn.heatmap(data = data)
```

```
The data to be plotted:

[[50  2 32  7 15 80 17 35 31 17]
 [98 16 66 13 21 23 89  1 36 54]
 [15 18 40 39 35 68 71 32 58 34]
 [35 21 96 14  2 97 28 89  6 91]
 [49 86 90 41 93 67 40 60 80 18]
 [11 15 91 28 20 80 29 43 73 89]
 [89 22 44 14 79 66 44 18  5 48]
 [76 20 31 26  1 72 20 98 45 25]
 [67 38 57 64 13 47 43 82 69 13]
 [21 94 68 67 93 11 70 87  2 79]]
```



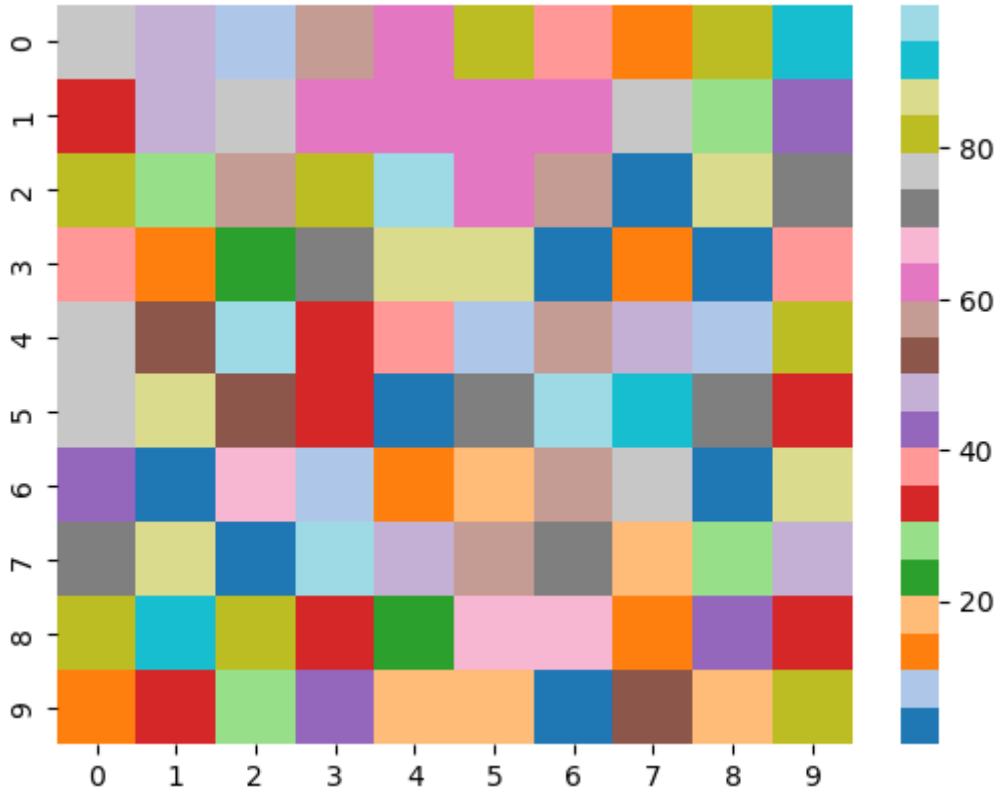
In [51]:

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

# generating 2-D 10x10 matrix of random numbers
# from 1 to 100
data = np.random.randint(low=1,
                         high=100,
                         size=(10, 10))

# setting the parameter values
vmin = 30
vmax = 70
#If we set the vmin value to 30 and the vmax value to 70, then only the cells with value

# plotting the heatmap
hm = sn.heatmap(data=data,
                vmin=vmin,
                vmax=vmax,
                annot= True
               )

# displaying the plotted heatmap
plt.show()
#If we set the vmin value to 30 and the vmax value to 70, then only the cells with value
```



In [53]:

```
# importing the modules
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt

# generating 2-D 10x10 matrix of random numbers
# from 1 to 100
data = np.random.randint(low=1,
                         high=100,
                         size=(10, 10))

# setting the parameter values
cmap = "tab20"

# plotting the heatmap
hm = sn.heatmap(data=data,
                cmap=cmap)

# displaying the plotted heatmap
plt.show()
```



In [ ]:

```

```