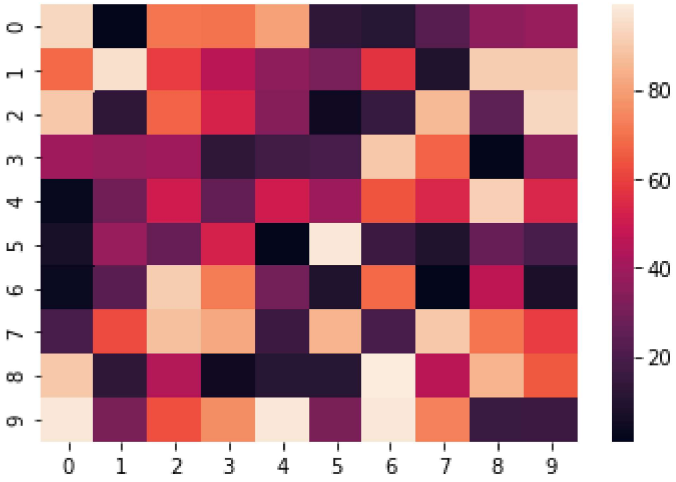
Heatmap: ¶

In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data)
```

```
[[94  1 71 70 80 13 11 23 36 38]
 [68 96 59 46 36 31 57  9 91 91]
 [90 13 67 53 34  5 15 86 25 94]
 [40 38 40 13 18 19 90 67  1 35]
 [ 3 29 51 26 51 39 64 54 92 54]
 [ 7 38 27 52  1 98 17  9 27 20]
 [ 4 24 91 72 30  9 68  1 47  8]
 [19 62 88 82 17 85 20 90 71 59]
 [90 13 44  5 11 11 99 46 85 65]
 [98 32 63 76 98 32 98 73 16 17]]
```



In [2]:

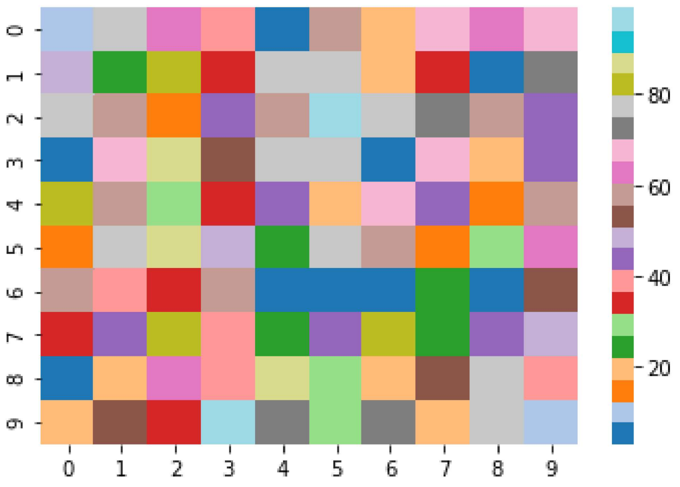
```
help(sns.heatmap)
```

```
        Axes in which to draw the colorbar, otherwise take space from the
        main Axes.
    square : bool, optional
        If True, set the Axes aspect to "equal" so each cell will be
        square-shaped.
    xticklabels, yticklabels : "auto", bool, list-like, or int, optional
        If True, plot the column names of the dataframe. If False, don't pl
ot
        the column names. If list-like, plot these alternate labels as the
        xticklabels. If an integer, use the column names but plot only ever
y
        n label. If "auto", try to densely plot non-overlapping labels.
    mask : bool array or DataFrame, optional
        If passed, data will not be shown in cells where ``mask`` is True.
        Cells with missing values are automatically masked.
    ax : matplotlib Axes, optional
        Axes in which to draw the plot, otherwise use the currently-active
        Axes.
    kwargs : other keyword arguments
        All other keyword arguments are passed to
```

In [5]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data,cmap="tab20")
```

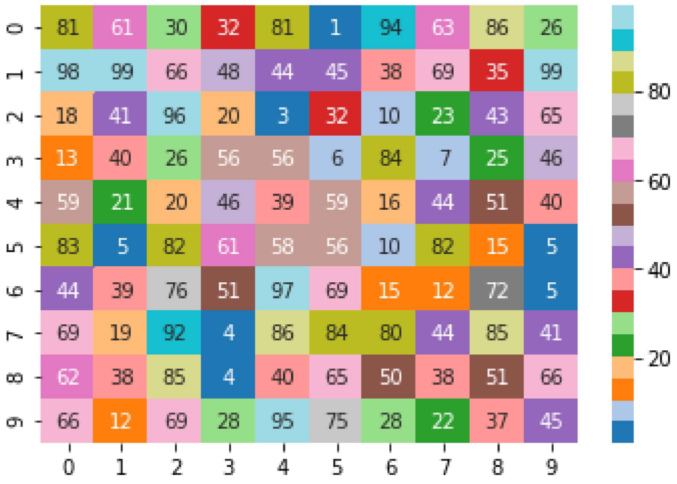
```
[[ 9 76 61 40  3 60 19 69 64 70]
 [48 23 82 36 79 76 21 35  5 71]
 [77 58 16 46 56 99 79 71 58 45]
 [ 6 69 88 53 77 76  7 67 19 46]
 [80 58 29 35 43 22 68 46 17 60]
 [13 78 87 49 26 77 57 13 27 65]
 [56 38 32 60  7  3  5 23  7 51]
 [32 44 84 39 23 42 84 25 43 47]
 [ 6 19 61 39 87 29 22 52 75 40]
 [21 52 33 97 72 28 72 22 75 10]]
```



In [6]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data,cmap="tab20",annot=True)
```

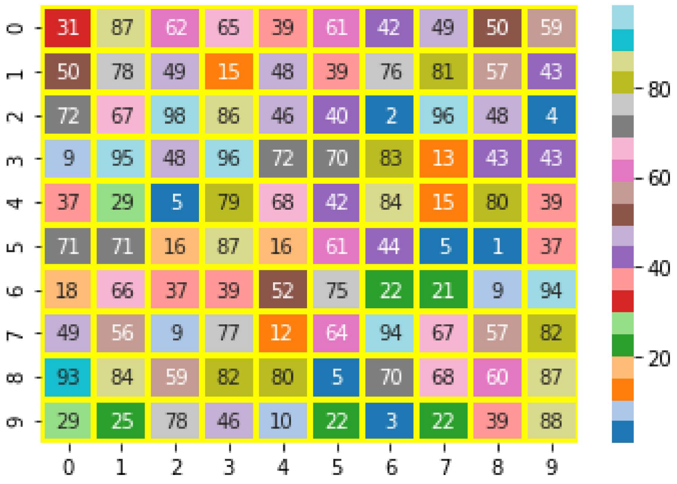
```
[[81 61 30 32 81  1 94 63 86 26]
 [98 99 66 48 44 45 38 69 35 99]
 [18 41 96 20  3 32 10 23 43 65]
 [13 40 26 56 56  6 84  7 25 46]
 [59 21 20 46 39 59 16 44 51 40]
 [83  5 82 61 58 56 10 82 15  5]
 [44 39 76 51 97 69 15 12 72  5]
 [69 19 92  4 86 84 80 44 85 41]
 [62 38 85  4 40 65 50 38 51 66]
 [66 12 69 28 95 75 28 22 37 45]]
```



In [7]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data,cmap="tab20",annot=True,linewidths=2,linecolor=
```

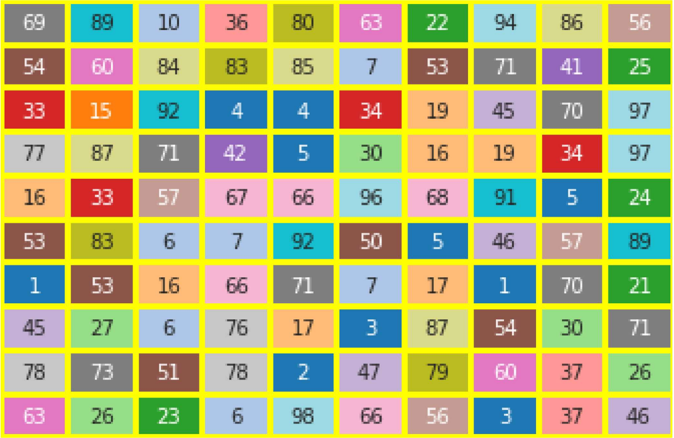
```
[[31 87 62 65 39 61 42 49 50 59]
 [50 78 49 15 48 39 76 81 57 43]
 [72 67 98 86 46 40  2 96 48  4]
 [ 9 95 48 96 72 70 83 13 43 43]
 [37 29  5 79 68 42 84 15 80 39]
 [71 71 16 87 16 61 44  5  1 37]
 [18 66 37 39 52 75 22 21  9 94]
 [49 56  9 77 12 64 94 67 57 82]
 [93 84 59 82 80  5 70 68 60 87]
 [29 25 78 46 10 22  3 22 39 88]]
```



In [9]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data,cmap="tab20",annot=True,linewidths=2,linecolor=
```

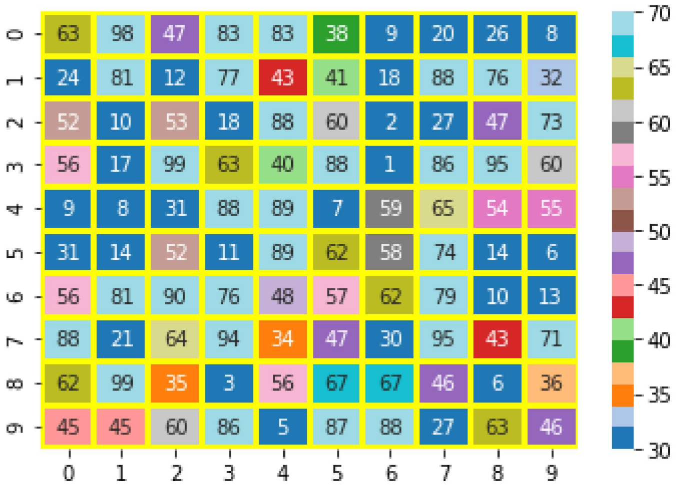
```
[[69 89 10 36 80 63 22 94 86 56]
 [54 60 84 83 85  7 53 71 41 25]
 [33 15 92  4  4 34 19 45 70 97]
 [77 87 71 42  5 30 16 19 34 97]
 [16 33 57 67 66 96 68 91  5 24]
 [53 83  6  7 92 50  5 46 57 89]
 [ 1 53 16 66 71  7 17  1 70 21]
 [45 27  6 76 17  3 87 54 30 71]
 [78 73 51 78  2 47 79 60 37 26]
 [63 26 23  6 98 66 56  3 37 46]]
```



In [10]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
hm = sns.heatmap(data=data,cmap="tab20",annot=True,linewidths=2,linecolor=
```

```
[[63 98 47 83 83 38  9 20 26  8]
 [24 81 12 77 43 41 18 88 76 32]
 [52 10 53 18 88 60  2 27 47 73]
 [56 17 99 63 40 88  1 86 95 60]
 [ 9  8 31 88 89  7 59 65 54 55]
 [31 14 52 11 89 62 58 74 14  6]
 [56 81 90 76 48 57 62 79 10 13]
 [88 21 64 94 34 47 30 95 43 71]
 [62 99 35  3 56 67 67 46  6 36]
 [45 45 60 86  5 87 88 27 63 46]]
```



In [13]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
df = pd.read_csv("auto-mpg.csv")
df
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

In [14]:

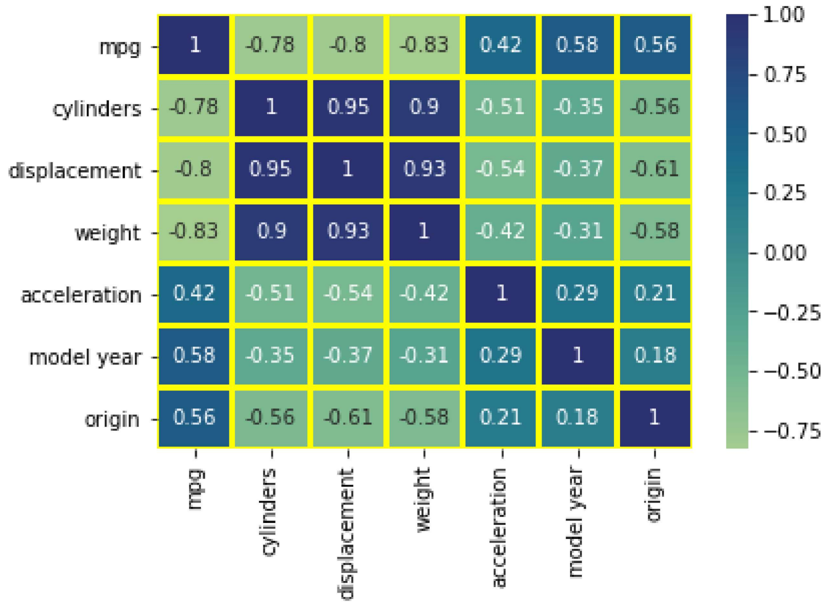
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
df = pd.read_csv("auto-mpg.csv")
df.corr()
```

Out[14]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

In [17]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
df = pd.read_csv("auto-mpg.csv")
hm = sns.heatmap(data=df.corr(),cmap="crest",annot=True,linewidths=2,linec
```



Networkx: only 3 marks all graph is pending

- directory graph
- undirectory graph

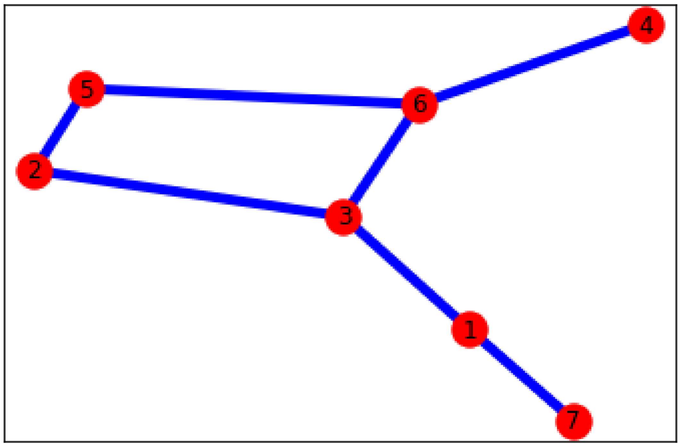
In [18]:

```
pip install networkx
```

```
Requirement already satisfied: networkx in c:\programdata\anaconda3\lib\site-
packages (2.5)
Requirement already satisfied: decorator>=4.3.0 in c:\programdata\anaconda3\l
ib\site-packages (from networkx) (4.4.2)
Note: you may need to restart the kernel to use updated packages.
```

In [20]:

```
import networkx as nx
import matplotlib.pyplot as plt
g = nx.Graph()
g.add_node(1)
g.add_nodes_from([2,3])
g.add_nodes_from(range(4,7))
g.add_edge(1,3)
g.add_edge(2,5)
g.add_edge(1,7)
g.add_edge(3,3)
g.add_edges_from([(2,3),(3,6),(4,6),(5,6)])
nx.draw_networkx(g,node_size=300,node_color="red",edge_color="blue",width=
plt.figsize = [150,150]
plt.show()
```



In [21]:

```
help(nx.draw_networkx)
```

```
    For directed graphs, arrows  are drawn at the head end.  Arrows can be
    turned off with keyword arrows=False.

    Examples
    --------
    >>> G = nx.dodecahedral_graph()
    >>> nx.draw(G)
    >>> nx.draw(G, pos=nx.spring_layout(G))  # use spring layout

    >>> import matplotlib.pyplot as plt
    >>> limits = plt.axis("off")  # turn of axis

    Also see the NetworkX drawing examples at
    https://networkx.github.io/documentation/latest/auto_examples/index.htm
l (https://networkx.github.io/documentation/latest/auto_examples/index.htm
l)

    See Also
    --------
```

In [22]:

```
help(nx.graph)
```

```
 |      **Nodes:**
 |
 |      Add one node at a time:
 |
 |      >>> G.add_node(1)
 |
 |      Add the nodes from any container (a list, dict, set or
 |      even the lines from a file or the nodes from another graph).
 |
 |      >>> G.add_nodes_from([2, 3])
 |      >>> G.add_nodes_from(range(100, 110))
 |      >>> H = nx.path_graph(10)
 |      >>> G.add_nodes_from(H)
 |
 |      In addition to strings and integers any hashable Python object
 |      (except None) can represent a node, e.g. a customized node object,
 |      or even another Graph.
 |
 |      >>> G.add_node(H)
 |
```

In [ ]:

```

```