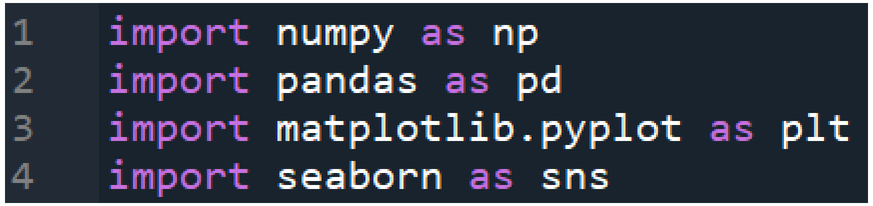
Lecture 14: Plotting II

Doing more with our visuals

Contents

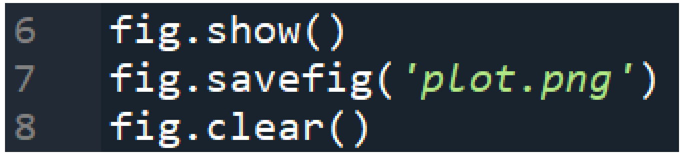
- Showing, saving, and clearing
- Dual-y axis plots
- Multiple subplots
- Other plot types
  - Scatter plots
  - Bar plots
  - Box plots
- Seaborn

First, import statements



```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

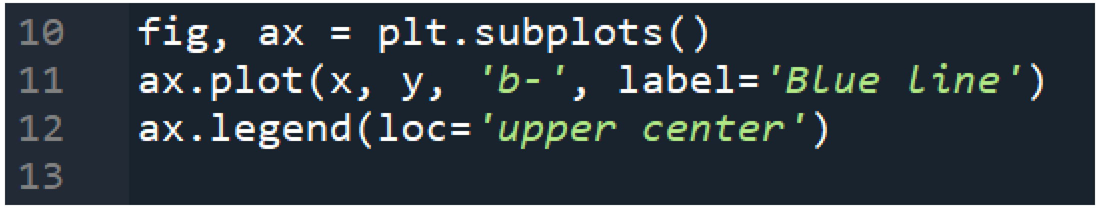
Finishing a plot



```
6    fig.show()
7    fig.savefig('plot.png')
8    fig.clear()
```

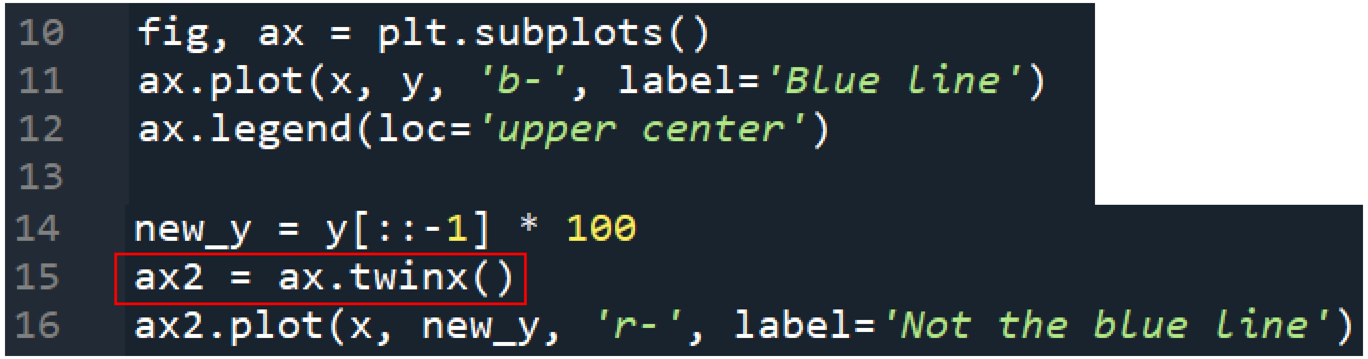
Can also use `plt.show()` to duplicate the code on lines 6 and 8.

Dual-y axis plots



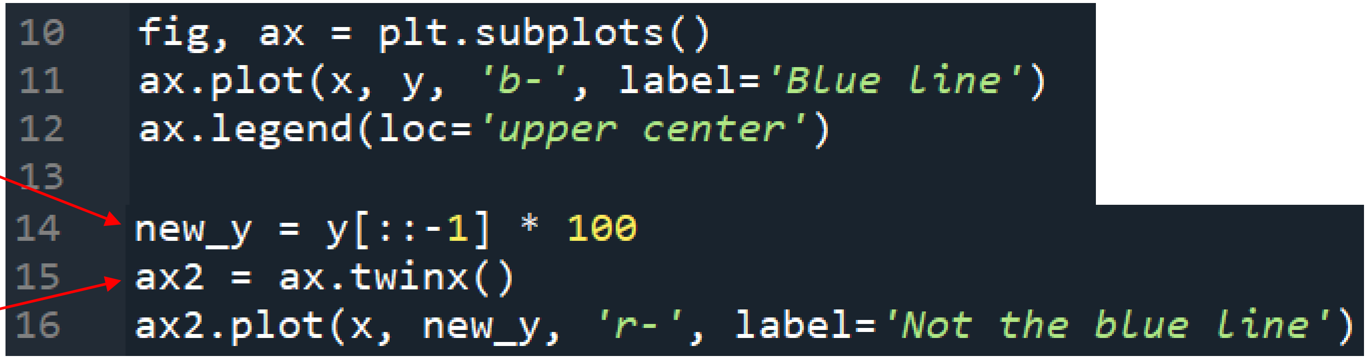
```python
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.legend(loc='upper center')

```

Dual-y axis plots



```python
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.legend(loc='upper center')

new_y = y[::-1] * 100
ax2 = ax.twinx()
ax2.plot(x, new_y, 'r-', label='Not the blue line')
```

Dual-y axis plots



```
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.legend(loc='upper center')

new_y = y[::-1] * 100
ax2 = ax.twinx()
ax2.plot(x, new_y, 'r-', label='Not the blue line')
```

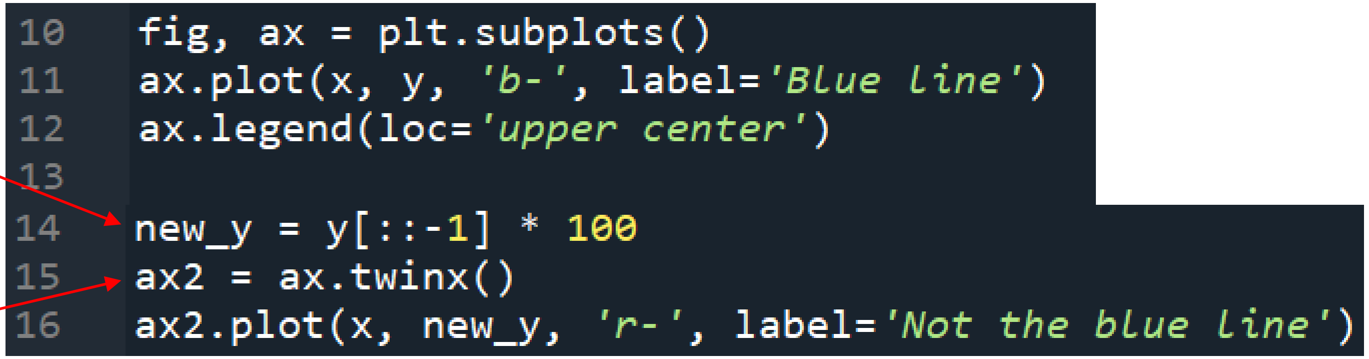
New y values, reversed and scaled up by 100x

A second axis object that mirrors the original axis, but does not contain the same lines

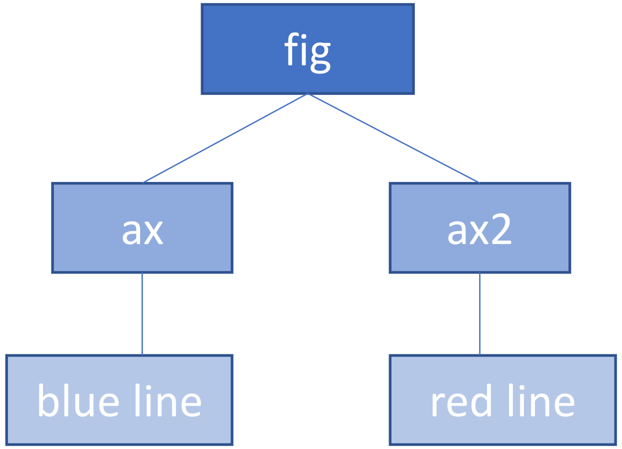
Dual-y axis plots

New y values, reversed and scaled up by 100x

A second axis object that mirrors the original axis, but does not contain the same lines



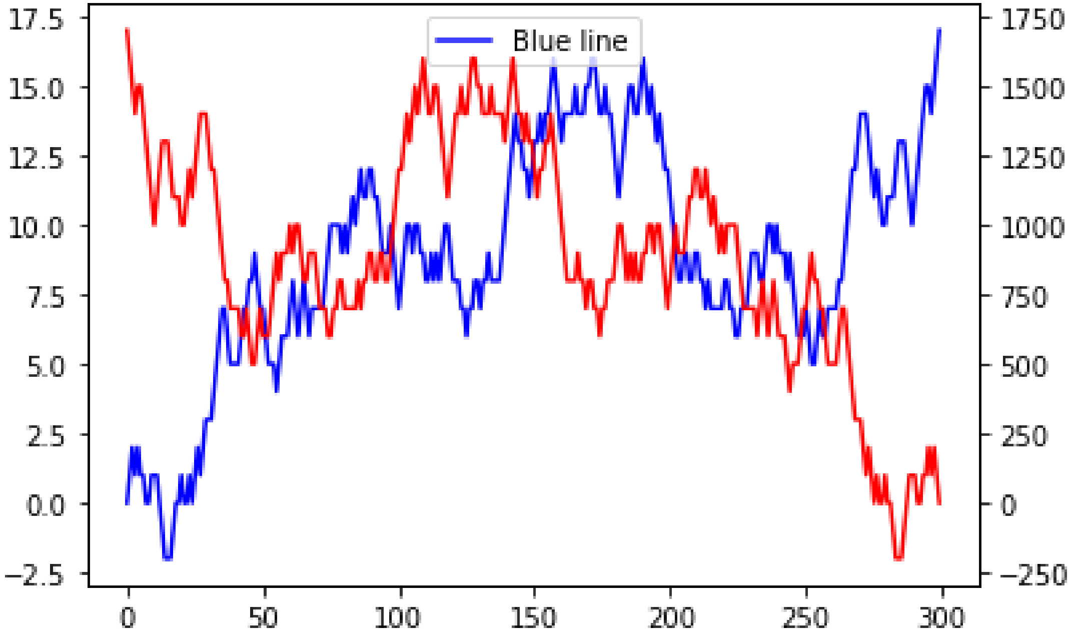
```
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.legend(loc='upper center')

new_y = y[::-1] * 100
ax2 = ax.twinx()
ax2.plot(x, new_y, 'r-', label='Not the blue line')
```



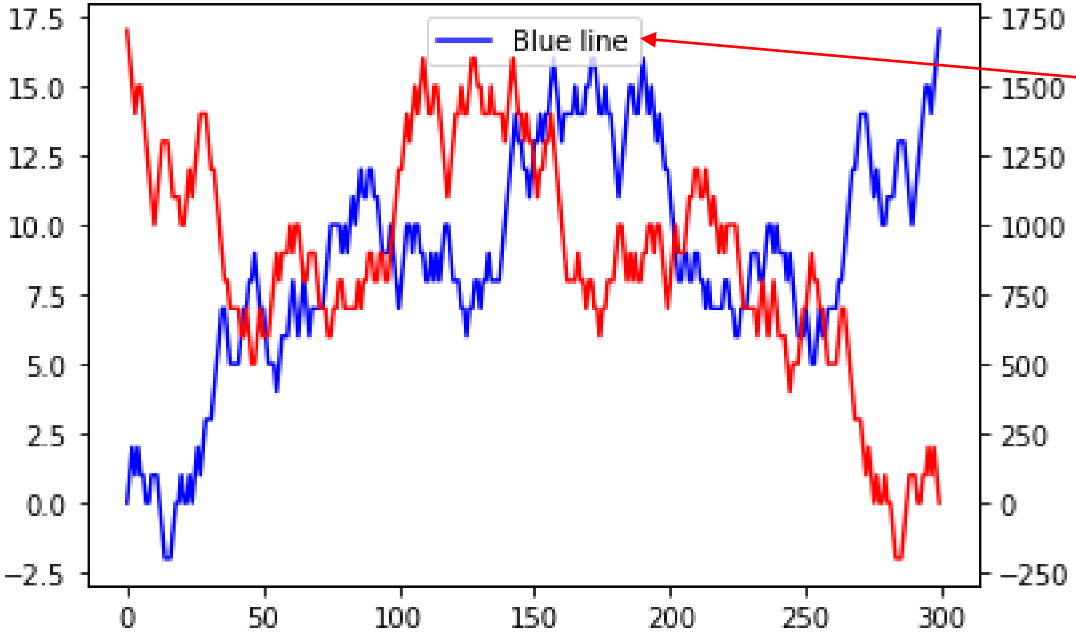
Both axis (ax and ax2) are in the same figure (fig)!

ax contains the first line (Blue line)
ax2 contains the second line (Red line)

Dual-y axis plots

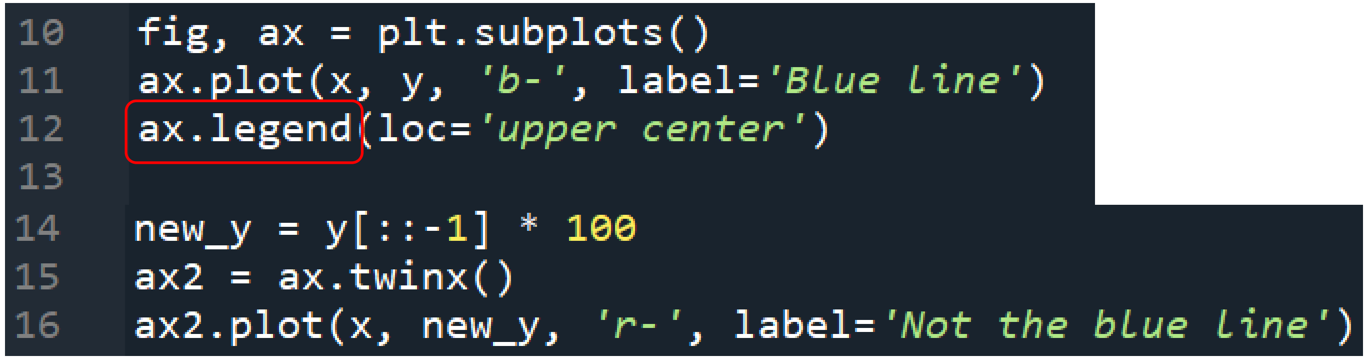


Dual-y axis plots

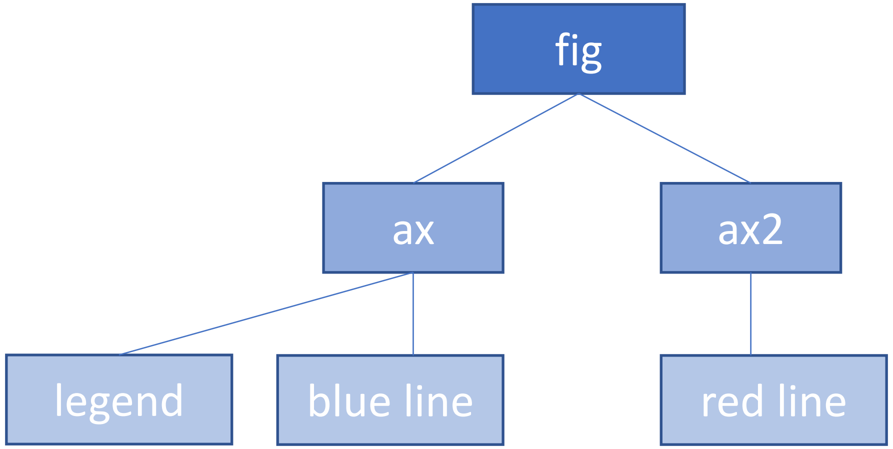


Only the blue line is in the legend?

Dual-y axis plots



```
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.legend(loc='upper center')

new_y = y[::-1] * 100
ax2 = ax.twinx()
ax2.plot(x, new_y, 'r-', label='Not the blue line')
```



fig

ax

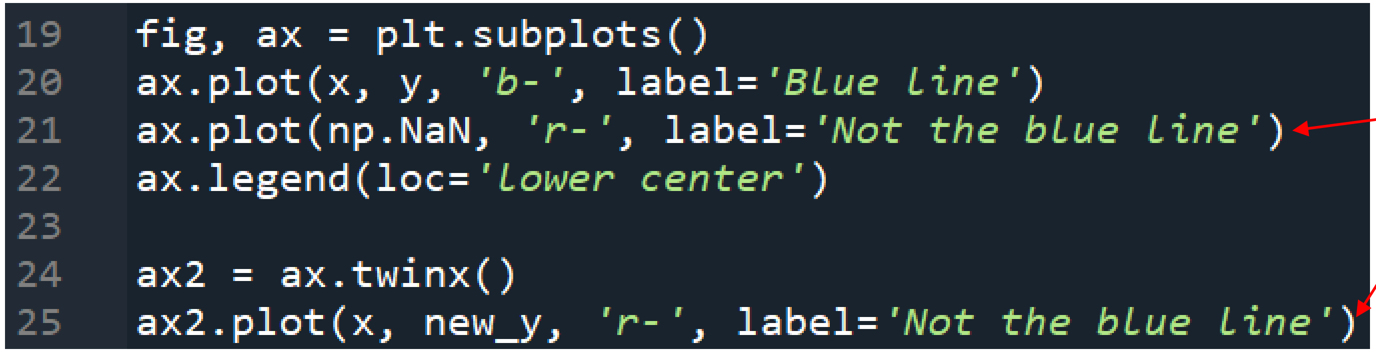
ax2

legend

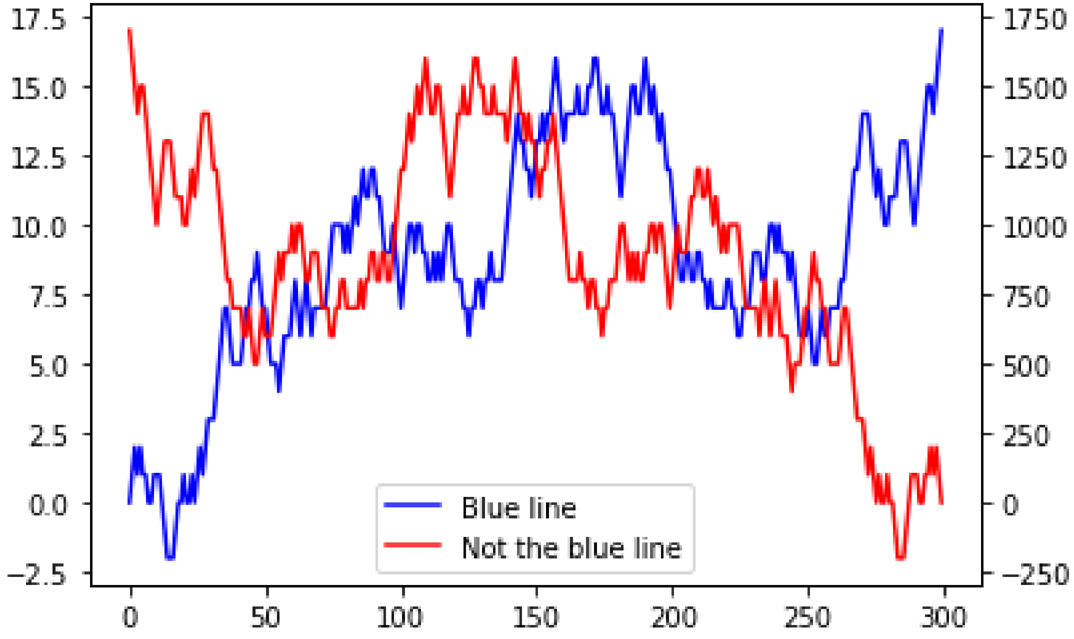
blue line

red line

Dual-y axis plots

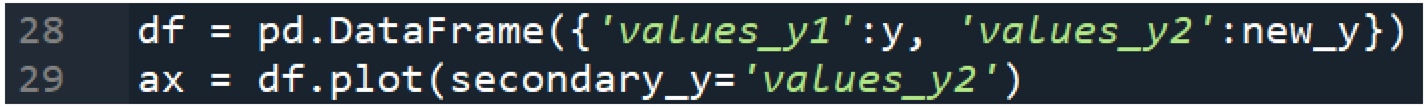


```
fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Blue line')
ax.plot(np.NaN, 'r-', label='Not the blue line')
ax.legend(loc='lower center')

ax2 = ax.twinx()
ax2.plot(x, new_y, 'r-', label='Not the blue line')
```

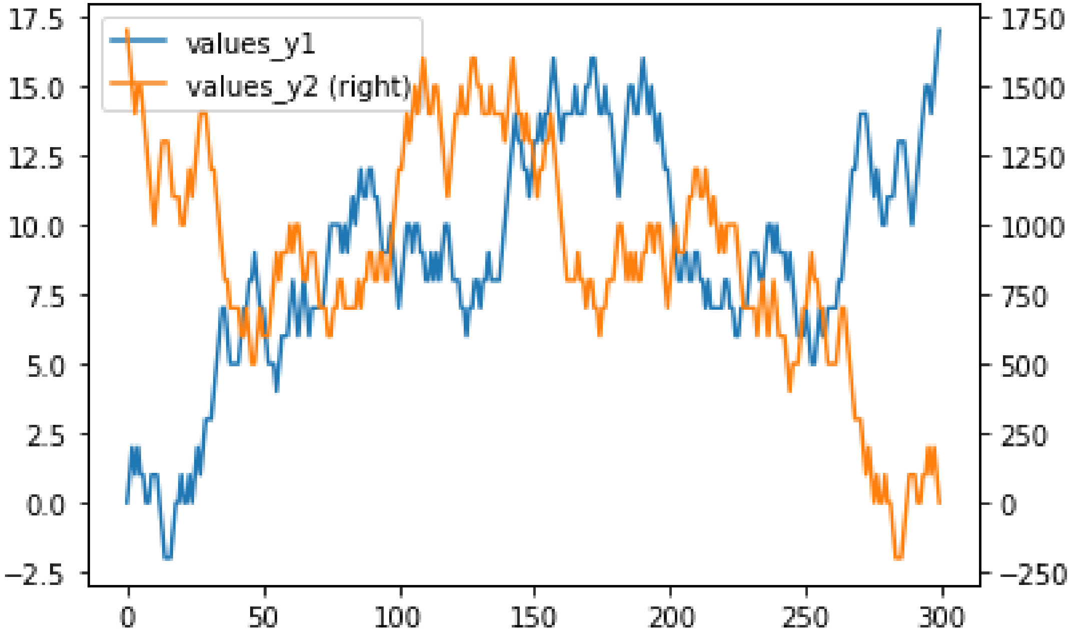


Just add a duplicate of the ax2 line, but with NaN as the values to plot (no line)

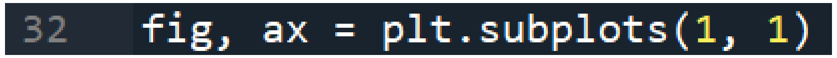
Or just use Pandas



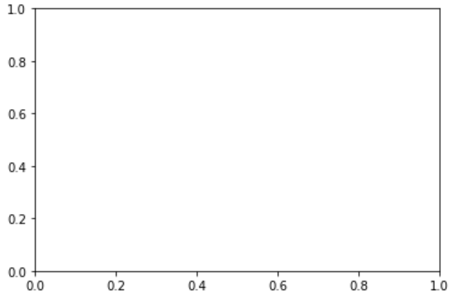
```
df = pd.DataFrame({'values_y1':y, 'values_y2':new_y})
ax = df.plot(secondary_y='values_y2')
```

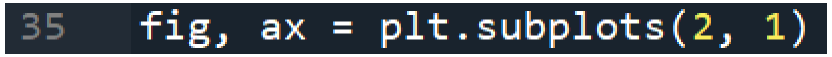


Multiple subplots

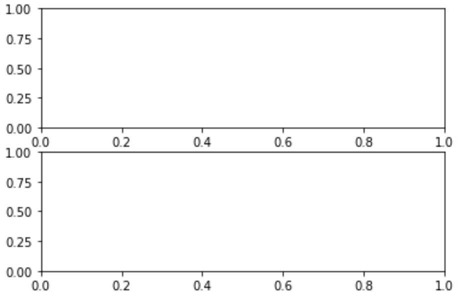


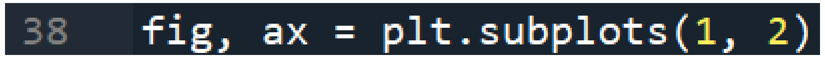
```python
fig, ax = plt.subplots(1, 1)
```



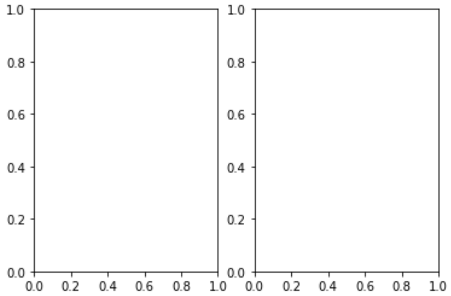


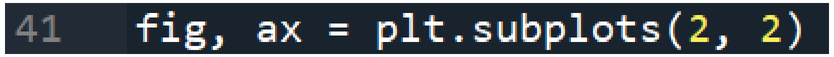
```python
fig, ax = plt.subplots(2, 1)
```



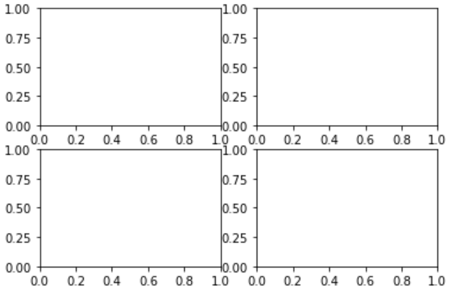


```python
fig, ax = plt.subplots(1, 2)
```

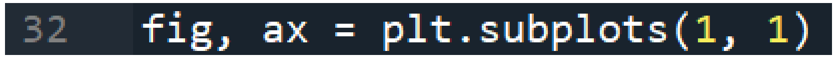




```python
fig, ax = plt.subplots(2, 2)
```

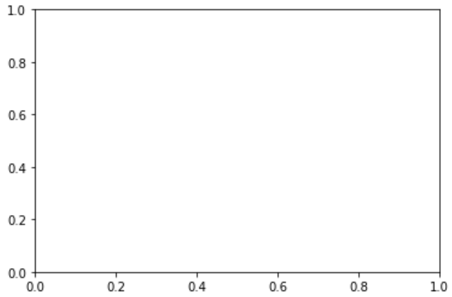


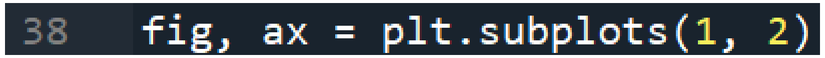
Multiple subplots



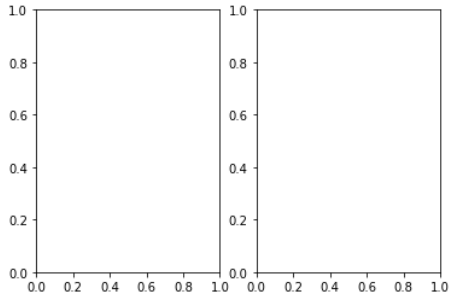
```python
fig, ax = plt.subplots(1, 1)
```

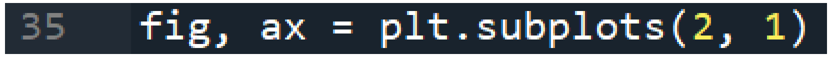
ax = MatPlotLib axis object



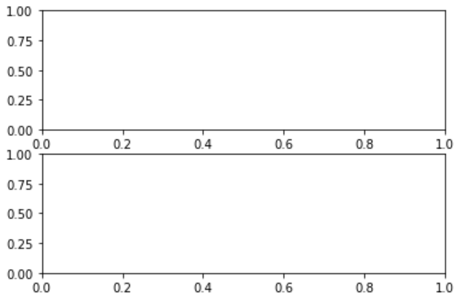


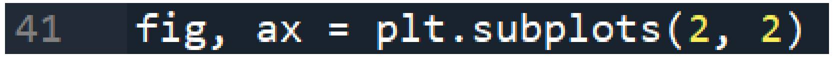
```python
fig, ax = plt.subplots(1, 2)
```



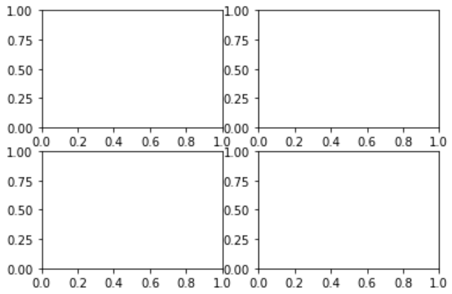


```python
fig, ax = plt.subplots(2, 1)
```

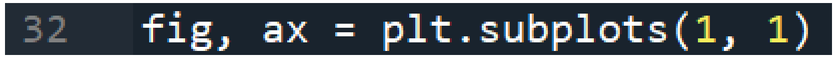




```python
fig, ax = plt.subplots(2, 2)
```

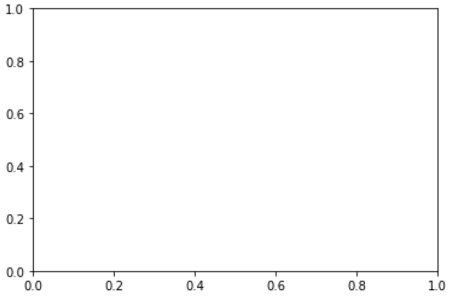


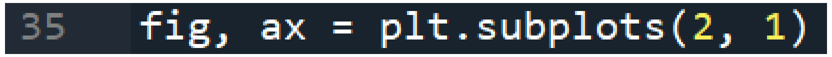
Multiple subplots



```
fig, ax = plt.subplots(1, 1)
```

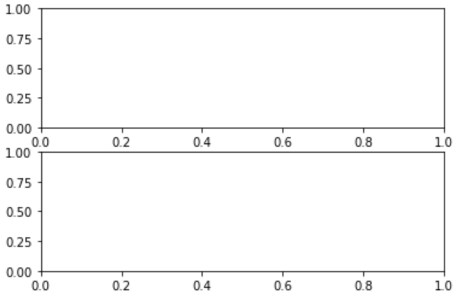
ax = MatPlotLib axis object

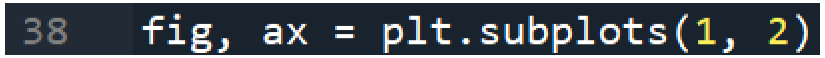




```
fig, ax = plt.subplots(2, 1)
```

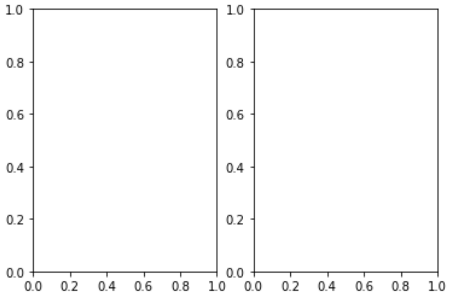
ax = List of two axis objects

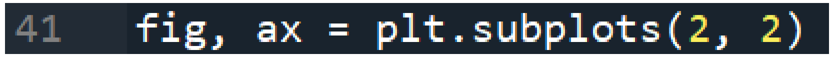




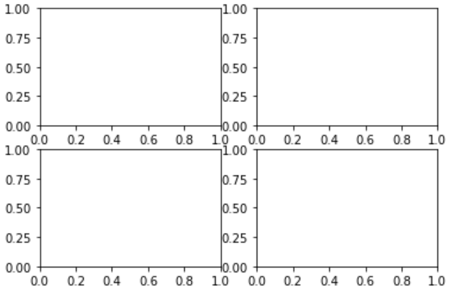
```
fig, ax = plt.subplots(1, 2)
```

ax = List of two axis objects

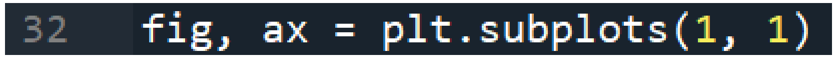




```
fig, ax = plt.subplots(2, 2)
```

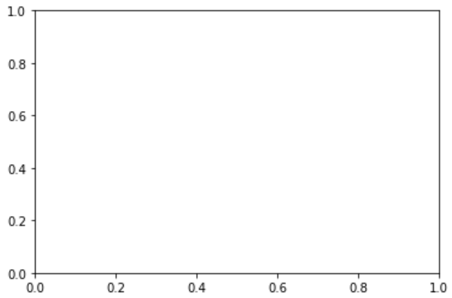


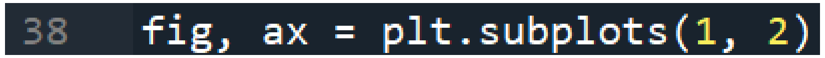
Multiple subplots



```
fig, ax = plt.subplots(1, 1)
```

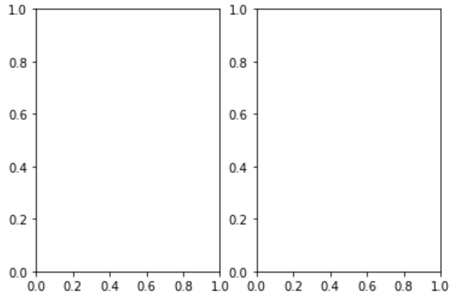
ax = MatPlotLib axis object

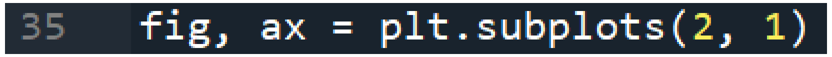




```
fig, ax = plt.subplots(1, 2)
```

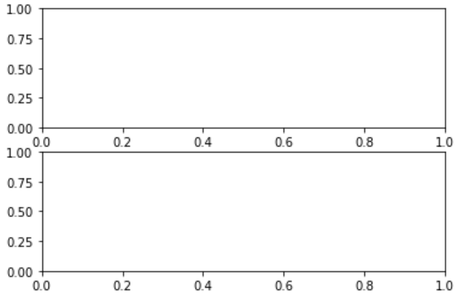
ax = List of two axis objects

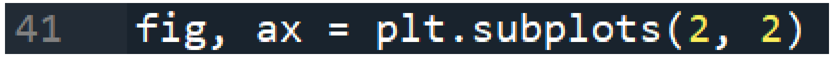




```
fig, ax = plt.subplots(2, 1)
```

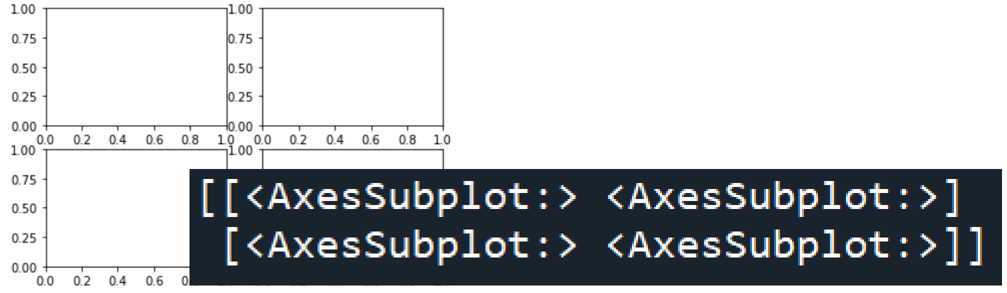
ax = List of two axis objects





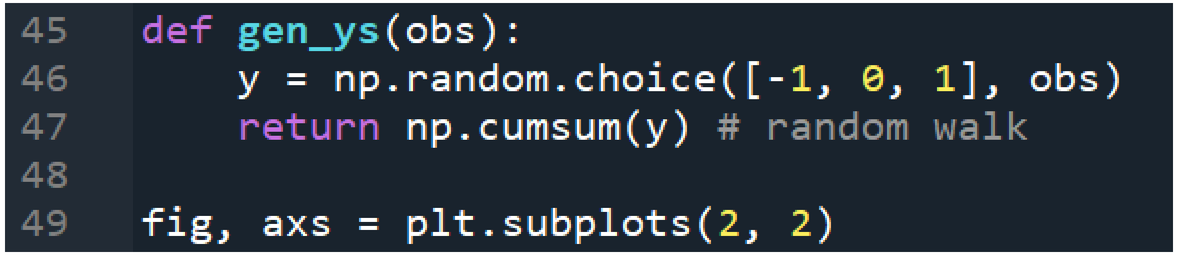
```
fig, ax = plt.subplots(2, 2)
```

ax = A list of two lists of two axis objects each



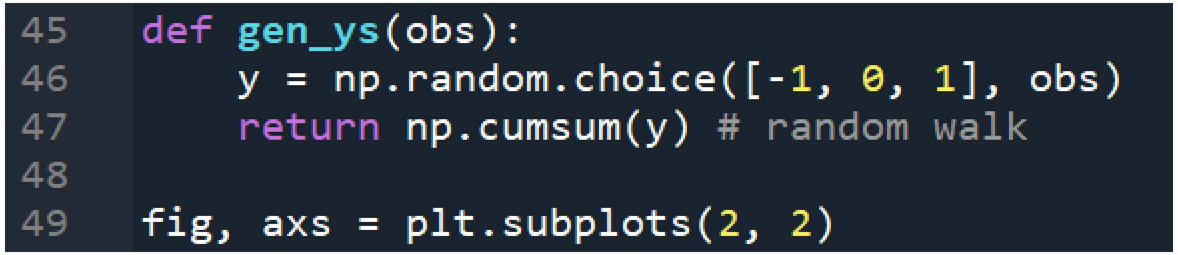
```
[[<AxesSubplot:> <AxesSubplot:>]
 [<AxesSubplot:> <AxesSubplot:>]]
```

Multiple subplots

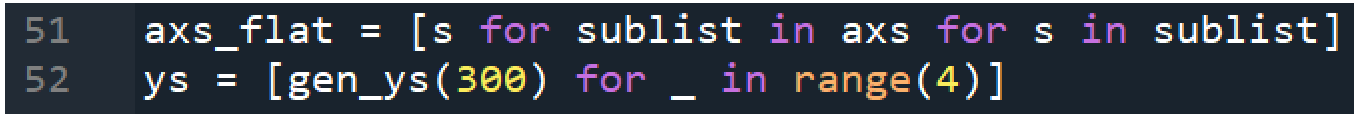


```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```

Multiple subplots

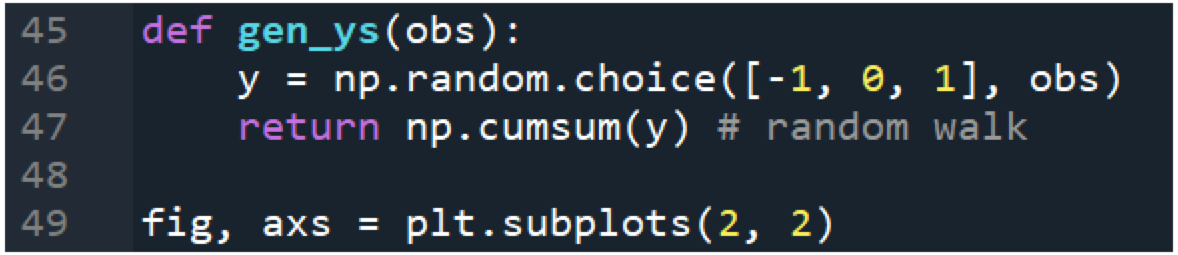


```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```

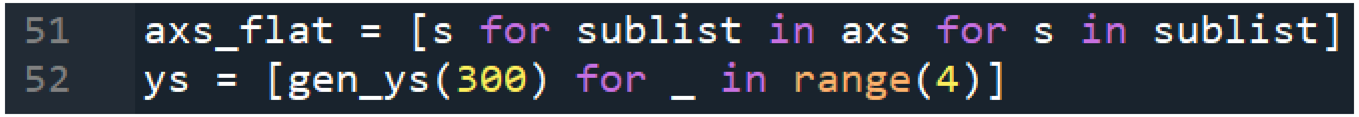


```python
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```

Multiple subplots



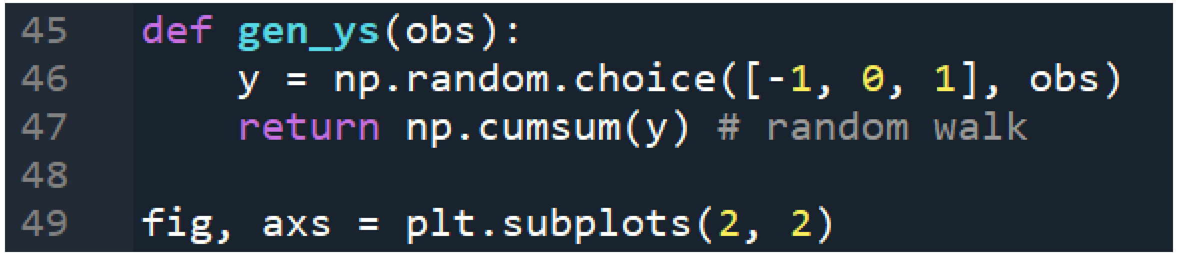
```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```



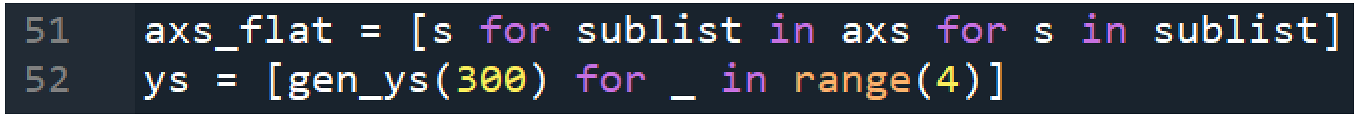
```python
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```

```
axs ->       [[<ax>, <ax>], [<ax>, <ax>]]

axs_flat -> [<ax>, <ax>, <ax>, <ax>]

ys ->        [[300 rw], [300 rw], [300 rw], [300 rw]]
```

Multiple subplots



```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```

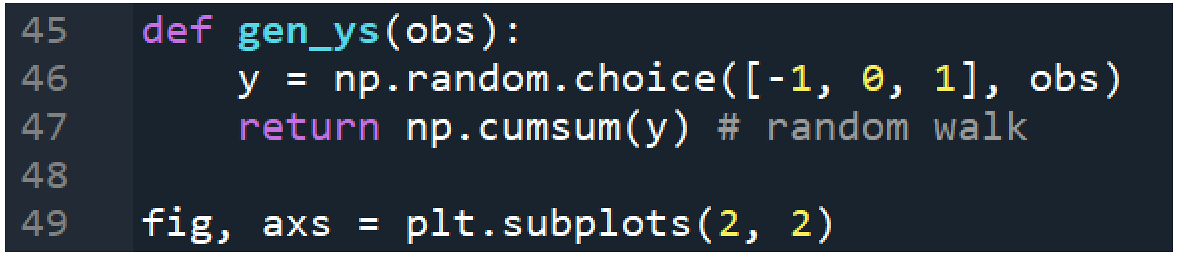


```python
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```

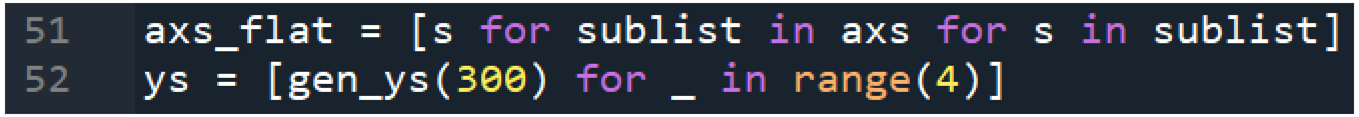
Three easy options:

```
axs[0][0]          ax1, ax2 = axs[0]
axs[0][1]          ax3, ax4 = axs[1]
axs[1][0]
axs[1][1]          ax1, ax2, ax3, ax4 = axs_flat
```

Multiple subplots



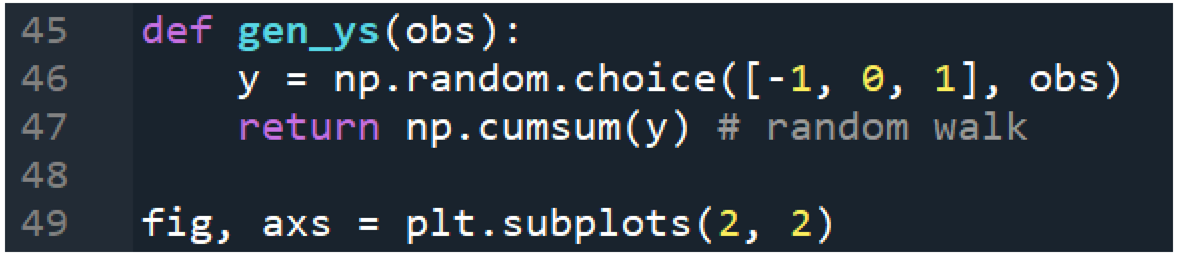
```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```



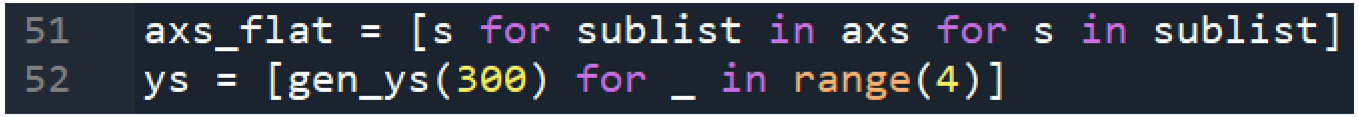
```python
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```

```
     len(axs) == 2    axs ->      [[<ax>, <ax>], [<ax>, <ax>]]

len(axs_flat) == 4    axs_flat -> [<ax>, <ax>, <ax>, <ax>]

      len(ys) == 4    ys ->       [[300 rw], [300 rw], [300 rw], [300 rw]]
```

Multiple subplots



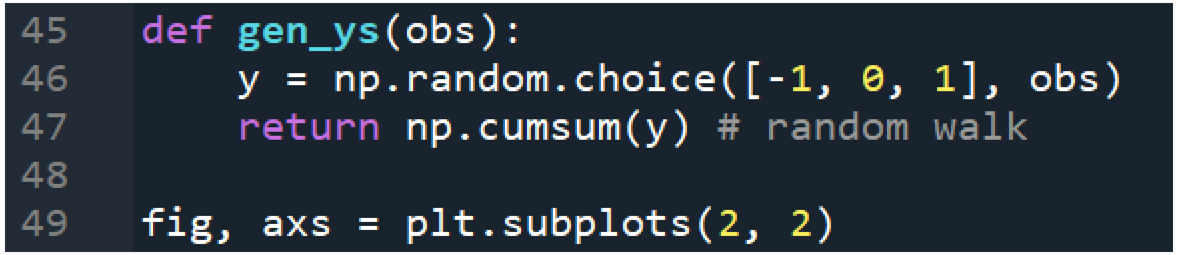
```
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```



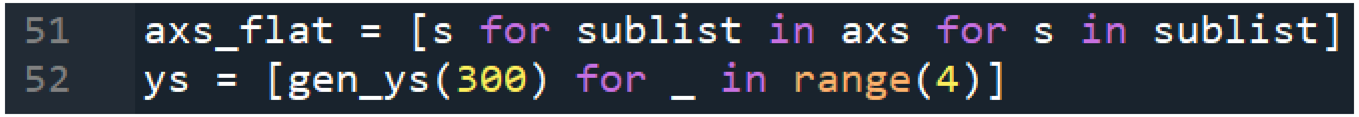
```
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```

```
    len(axs) == 2    axs ->      [[<ax>, <ax>], [<ax>, <ax>]]

len(axs_flat) == 4   axs_flat -> [<ax>, <ax>, <ax>, <ax>]

     len(ys) == 4    ys ->       [[300 rw], [300 rw], [300 rw], [300 rw]]


zip(axs_flat, ys)  -> (<ax>, [300 rw]), (<ax>, [300 rw]), (<ax>, [300 rw]), (<ax>, [300 rw])
```

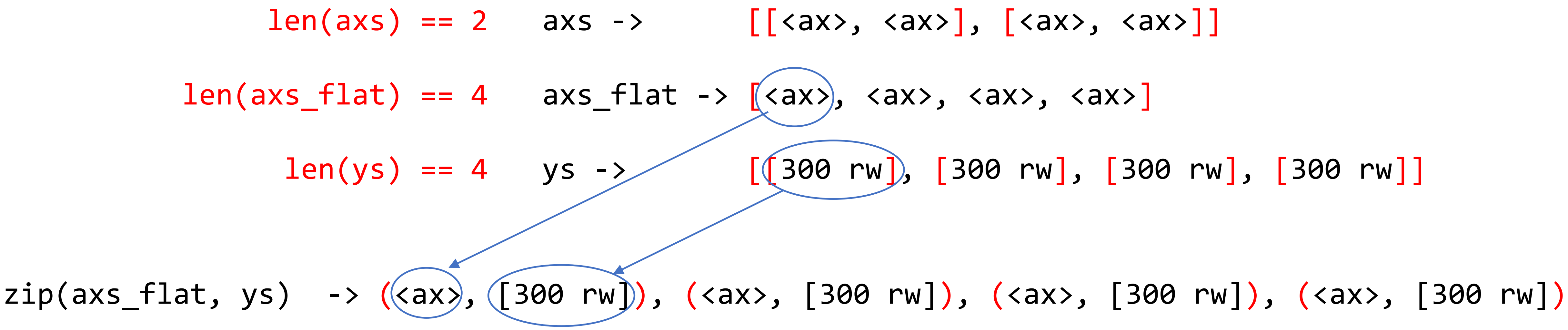
Multiple subplots



```
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```



```
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```



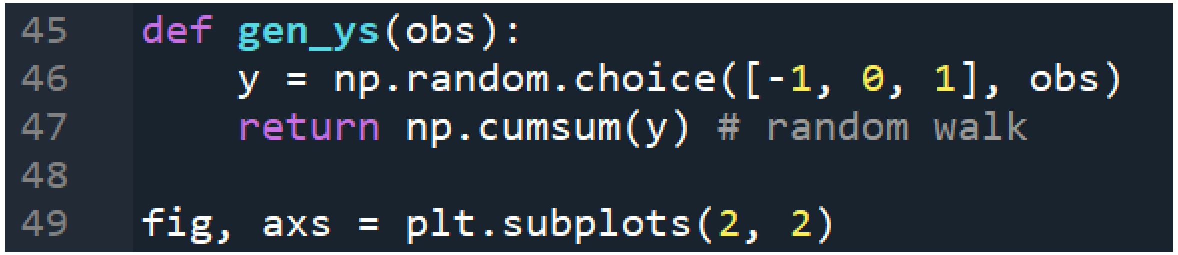
len(axs) == 2   axs ->   [[<ax>, <ax>], [<ax>, <ax>]]

len(axs_flat) == 4   axs_flat -> [<ax>, <ax>, <ax>, <ax>]

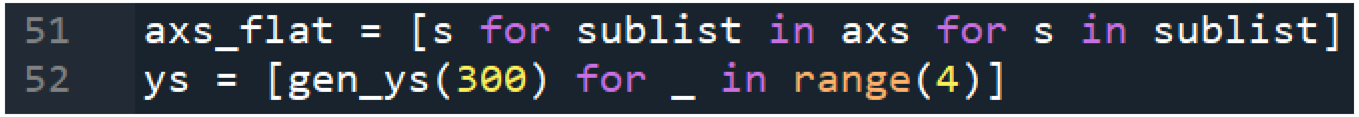
len(ys) == 4   ys ->   [[300 rw], [300 rw], [300 rw], [300 rw]]

zip(axs_flat, ys) -> (<ax>, [300 rw]), (<ax>, [300 rw]), (<ax>, [300 rw]), (<ax>, [300 rw])

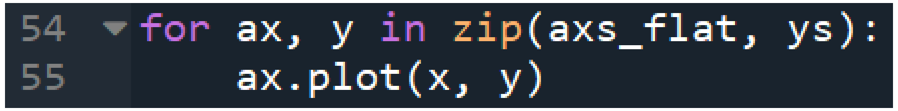
Multiple subplots



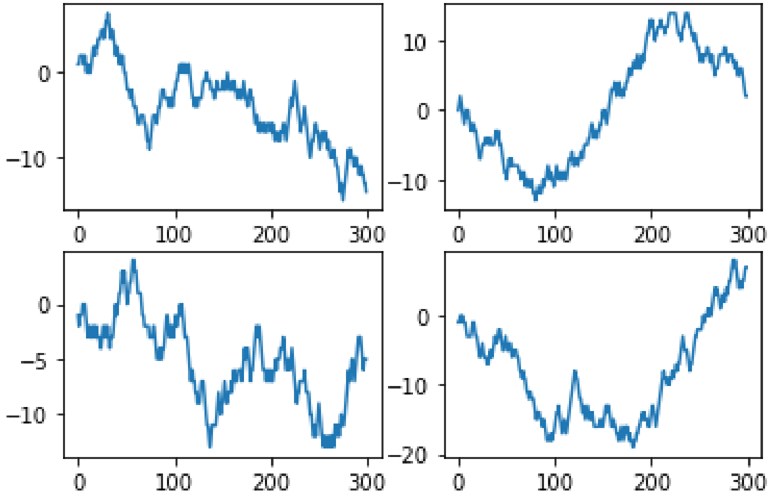
```python
def gen_ys(obs):
    y = np.random.choice([-1, 0, 1], obs)
    return np.cumsum(y) # random walk

fig, axs = plt.subplots(2, 2)
```



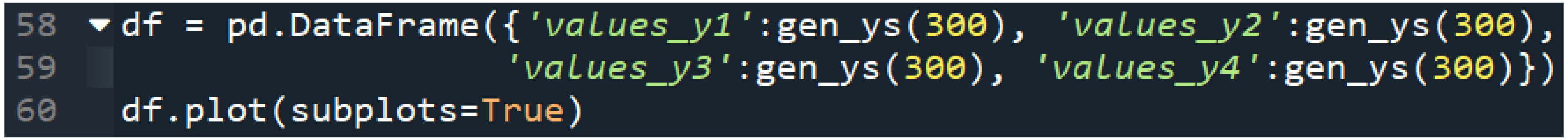
```python
axs_flat = [s for sublist in axs for s in sublist]
ys = [gen_ys(300) for _ in range(4)]
```



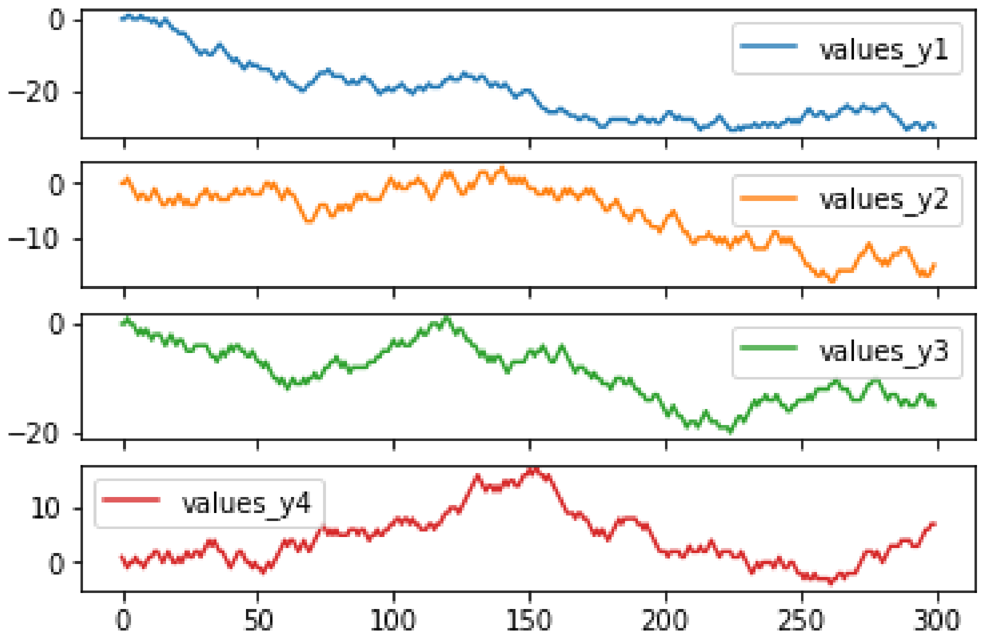
```python
for ax, y in zip(axs_flat, ys):
    ax.plot(x, y)
```



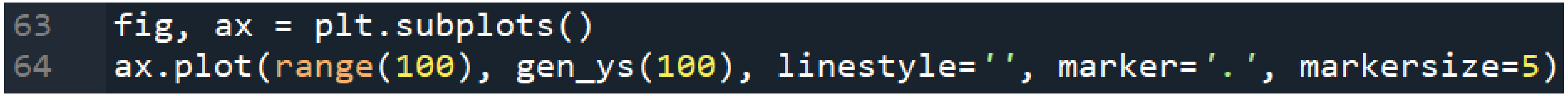
Multiple subplots: Pandas



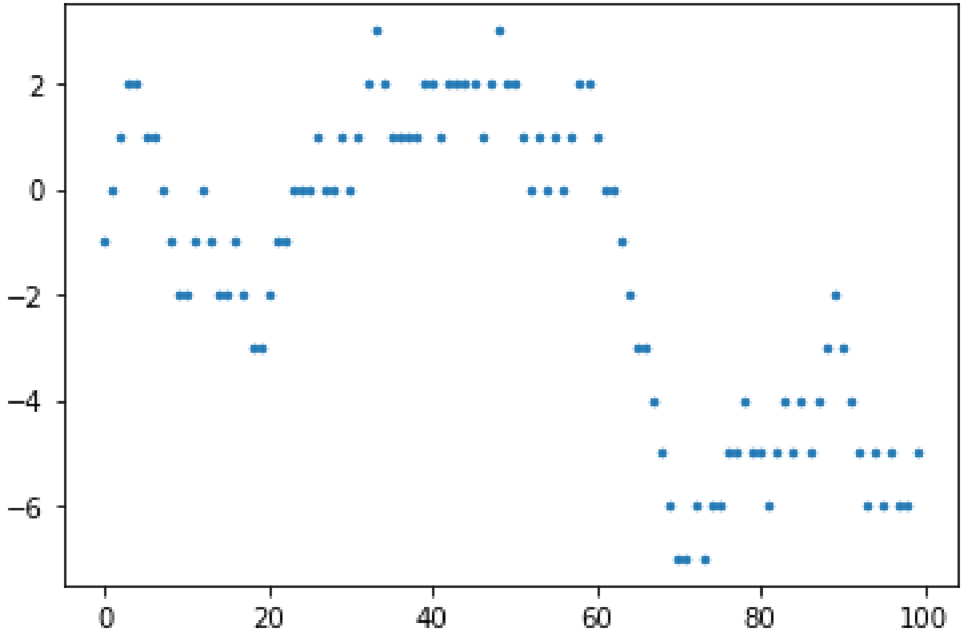
```python
df = pd.DataFrame({'values_y1':gen_ys(300), 'values_y2':gen_ys(300),
                  'values_y3':gen_ys(300), 'values_y4':gen_ys(300)})
df.plot(subplots=True)
```



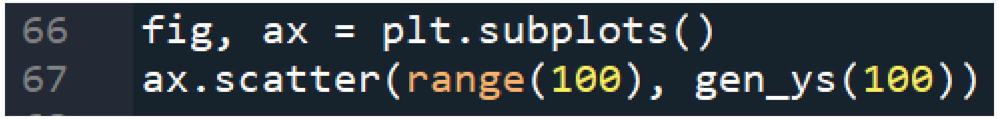
Other plot types: scatter



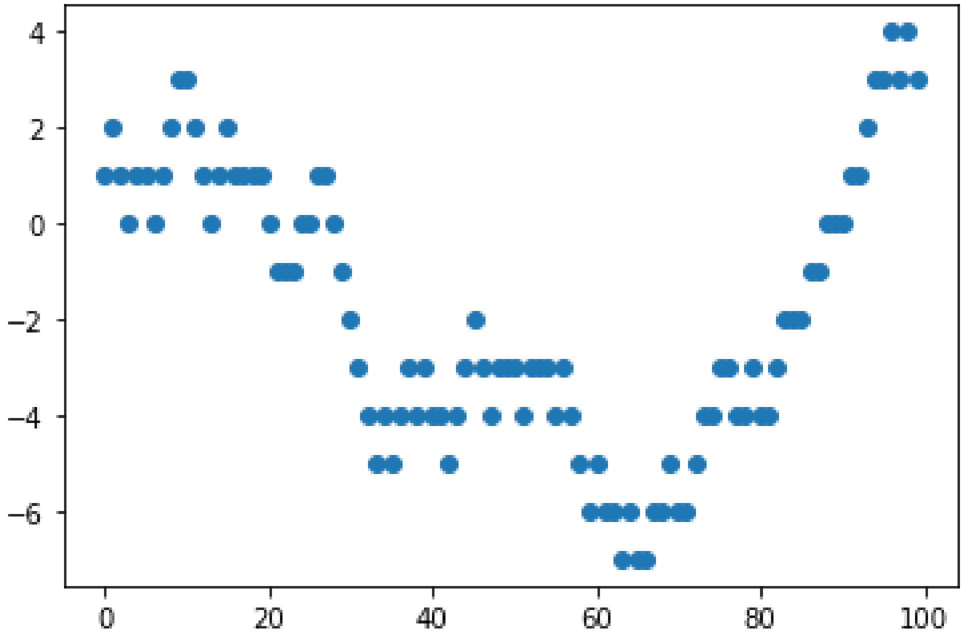
```python
fig, ax = plt.subplots()
ax.plot(range(100), gen_ys(100), linestyle='', marker='.', markersize=5)
```



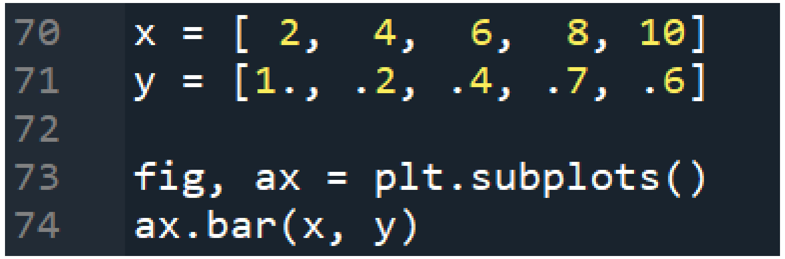
Other plot types: scatter



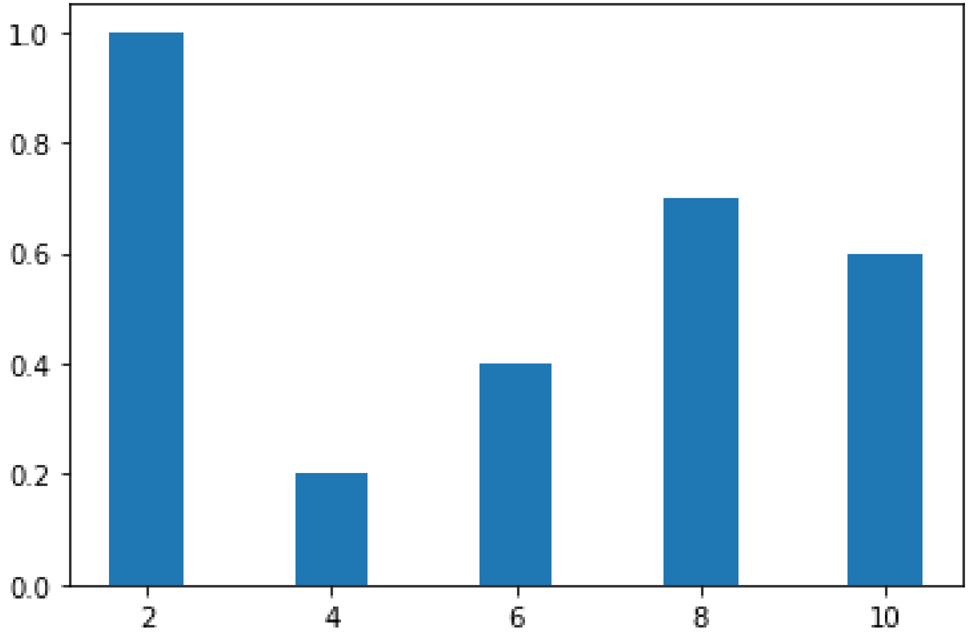
```
fig, ax = plt.subplots()
ax.scatter(range(100), gen_ys(100))
```



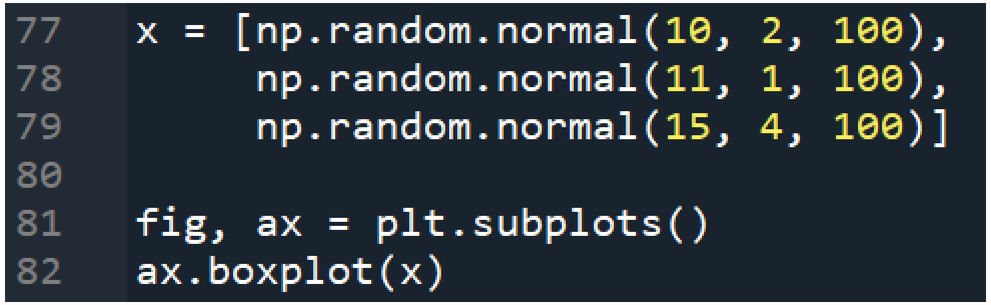
Other plot types: bar



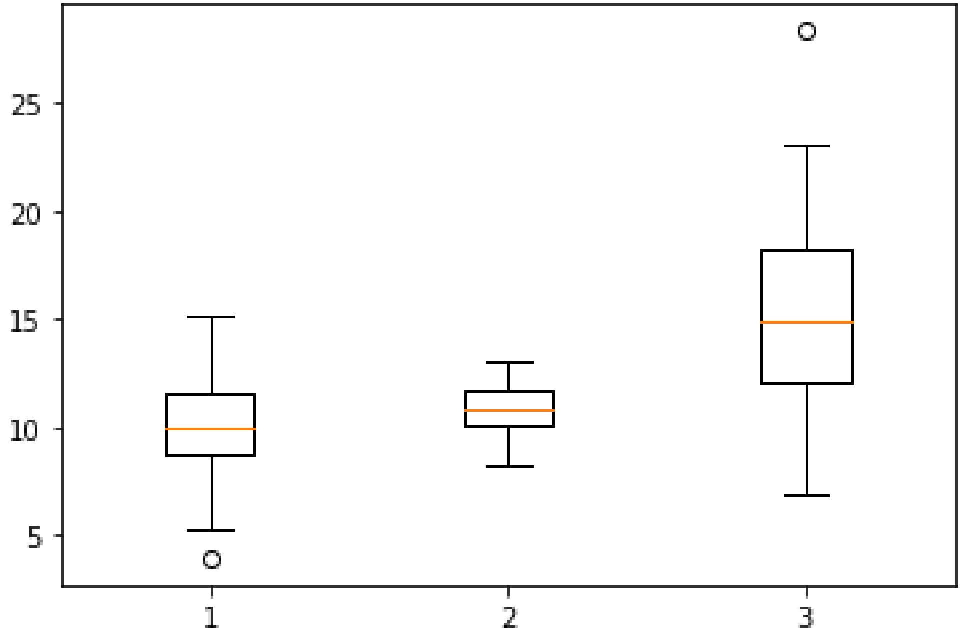
```
x = [ 2,  4,  6,  8, 10]
y = [1., .2, .4, .7, .6]

fig, ax = plt.subplots()
ax.bar(x, y)
```



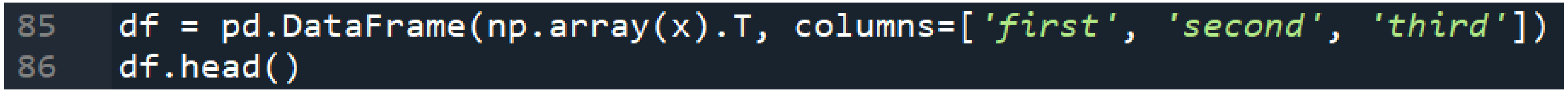
Other plot types: box



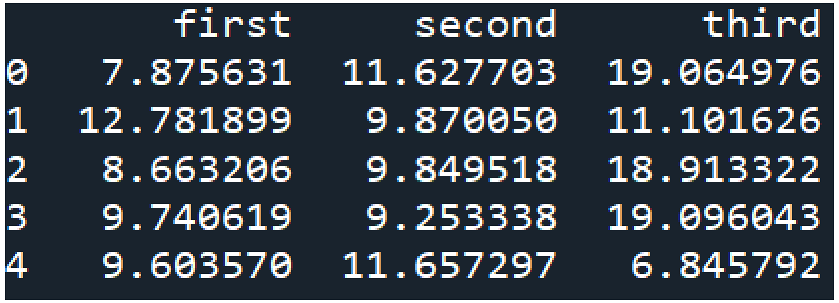
```
x = [np.random.normal(10, 2, 100),
     np.random.normal(11, 1, 100),
     np.random.normal(15, 4, 100)]

fig, ax = plt.subplots()
ax.boxplot(x)
```



Seaborn: box plots

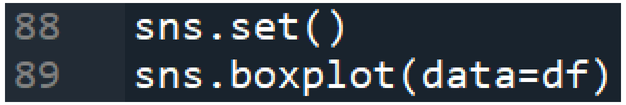


```python
df = pd.DataFrame(np.array(x).T, columns=['first', 'second', 'third'])
df.head()
```

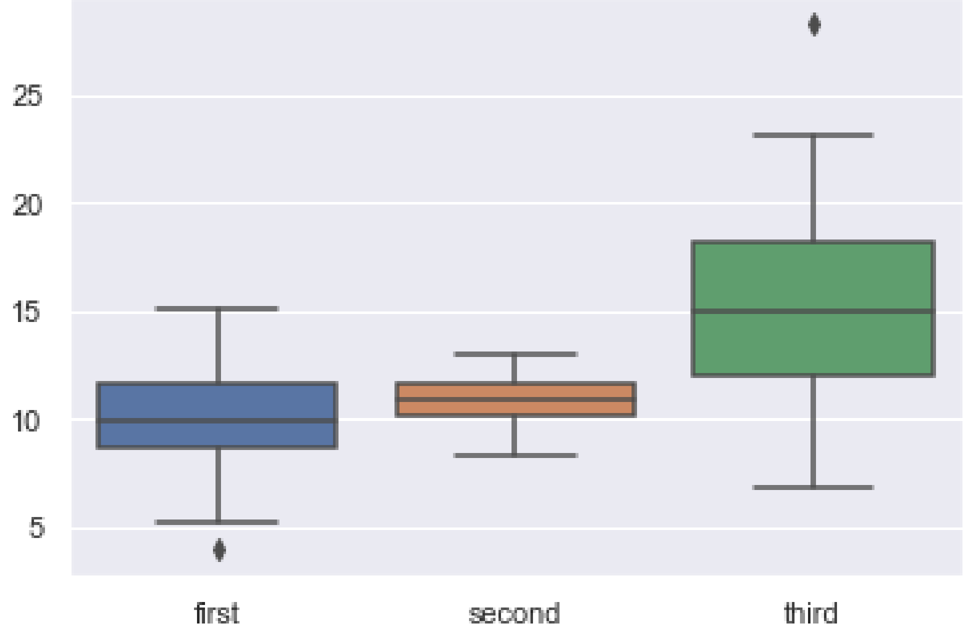


```
       first     second      third
0   7.875631  11.627703  19.064976
1  12.781899   9.870050  11.101626
2   8.663206   9.849518  18.913322
3   9.740619   9.253338  19.096043
4   9.603570  11.657297   6.845792
```

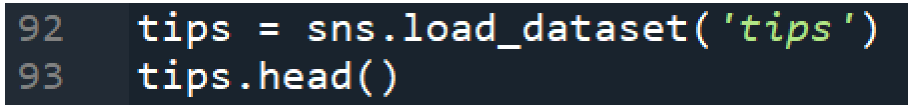
Seaborn: box plots



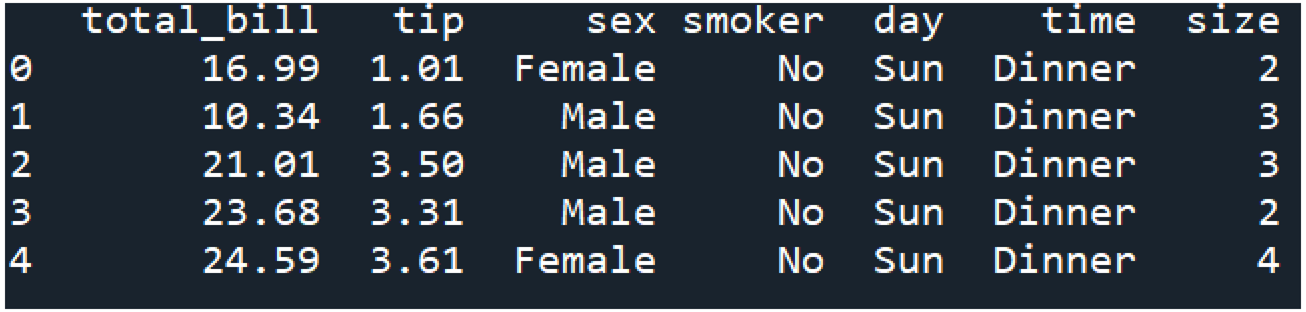
```
sns.set()
sns.boxplot(data=df)
```



Seaborn: grouped box plots



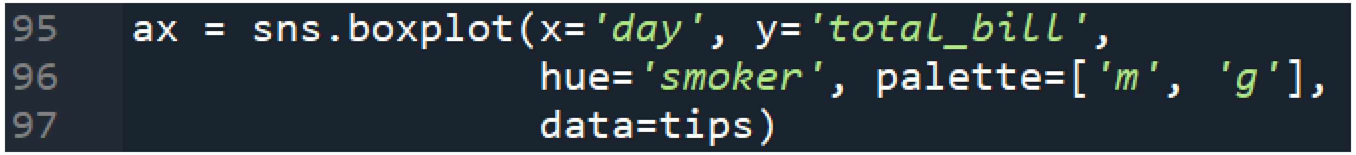
```python
tips = sns.load_dataset('tips')
tips.head()
```



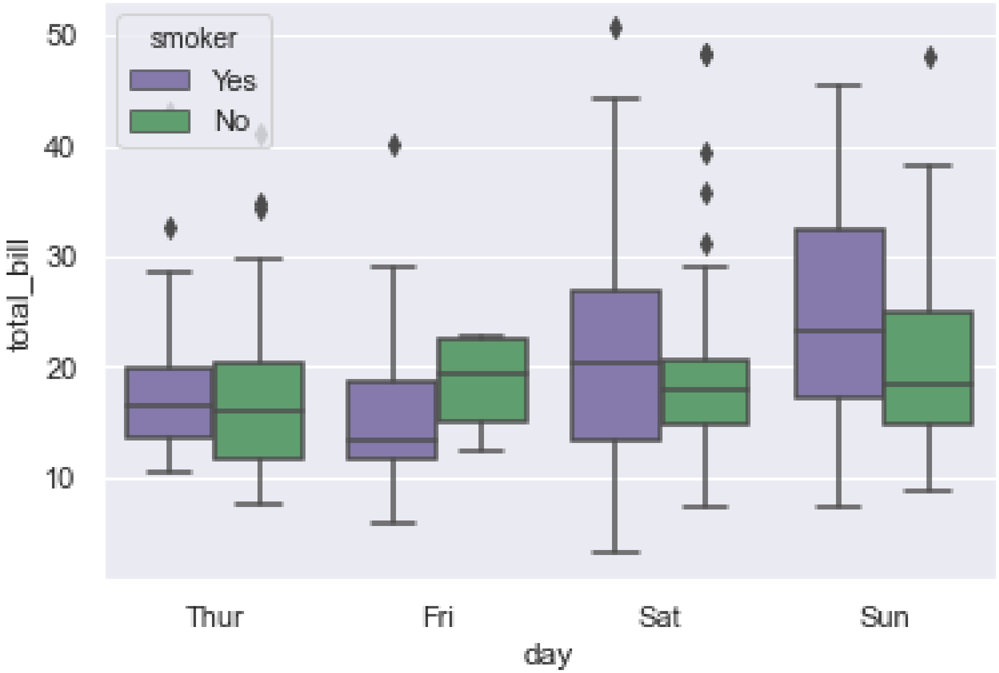
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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

https://seaborn.pydata.org/examples/grouped_boxplot.html

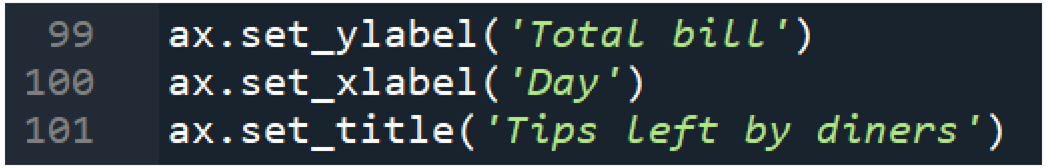
Seaborn: grouped box plots



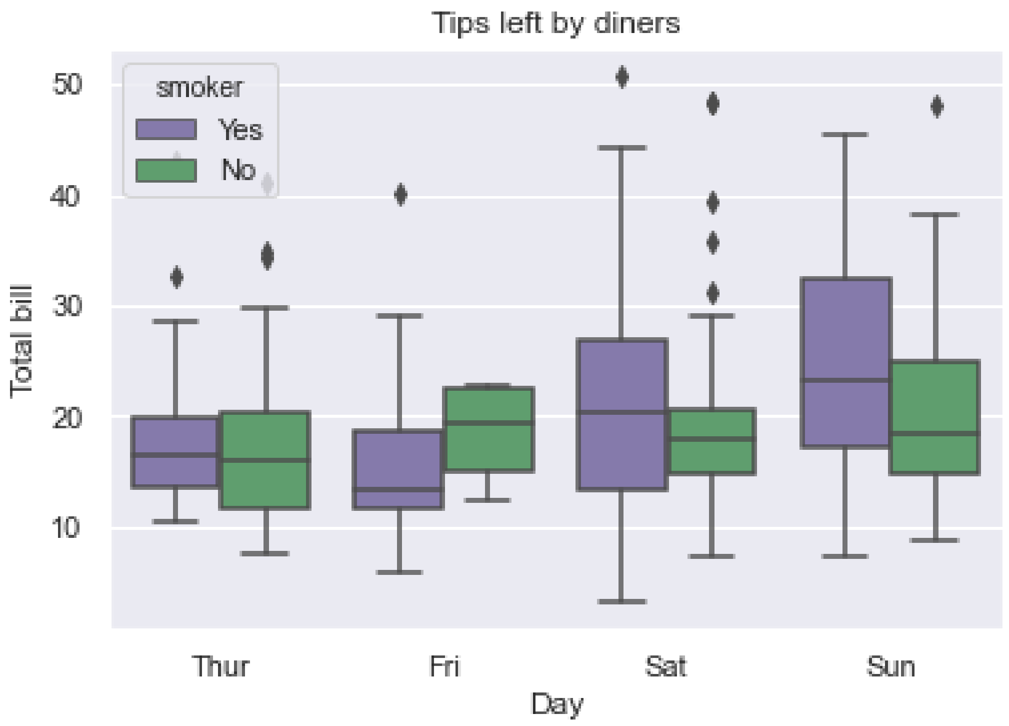
```python
ax = sns.boxplot(x='day', y='total_bill',
                 hue='smoker', palette=['m', 'g'],
                 data=tips)
```



Seaborn and MatPlotLib



```
ax.set_ylabel('Total bill')
ax.set_xlabel('Day')
ax.set_title('Tips left by diners')
```


Tips left by diners
smoker
🟪 Yes
🟩 No
Total bill
Thur
Fri
Sat
Sun
Day