tu08_re_MatPlotLib_HW

February 9, 2023

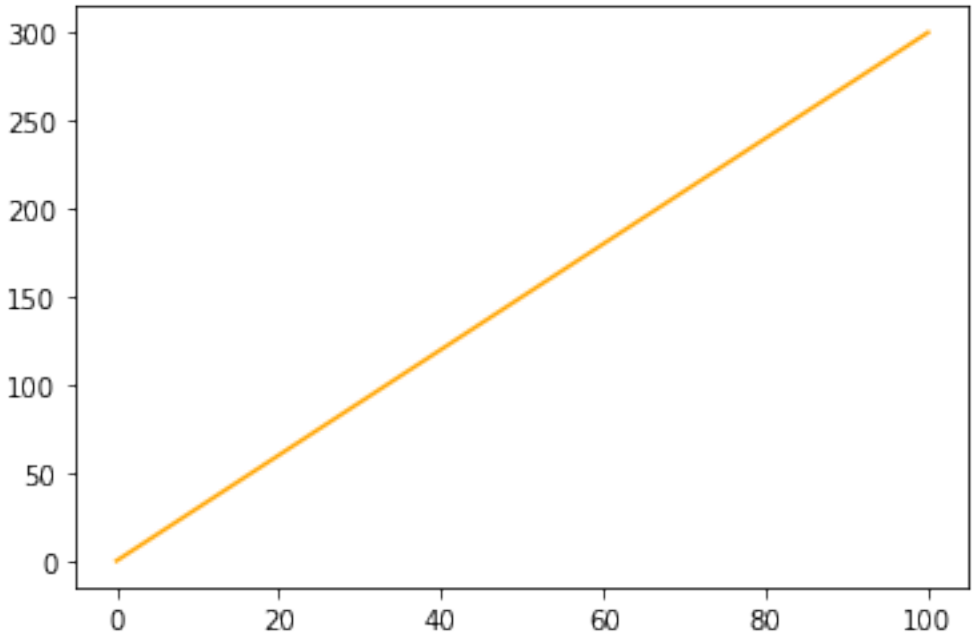
1 matplotlib homework

1. Make a plot of a straight line. Use `linspace()` to create the x values and the formula of a straight line, $y = a + bx$, *to create the* y* values (use an a and b of your choosing). You can pretend x and y are anything you like (x = time, y = international piracy or whatever).

[1]:
```
# Library
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# X-value
x = np.linspace(0, 100)

# Y-value (slope = 3)
y = 0 + 3*x

# Graph
plt.plot(x,y,'orange')
```

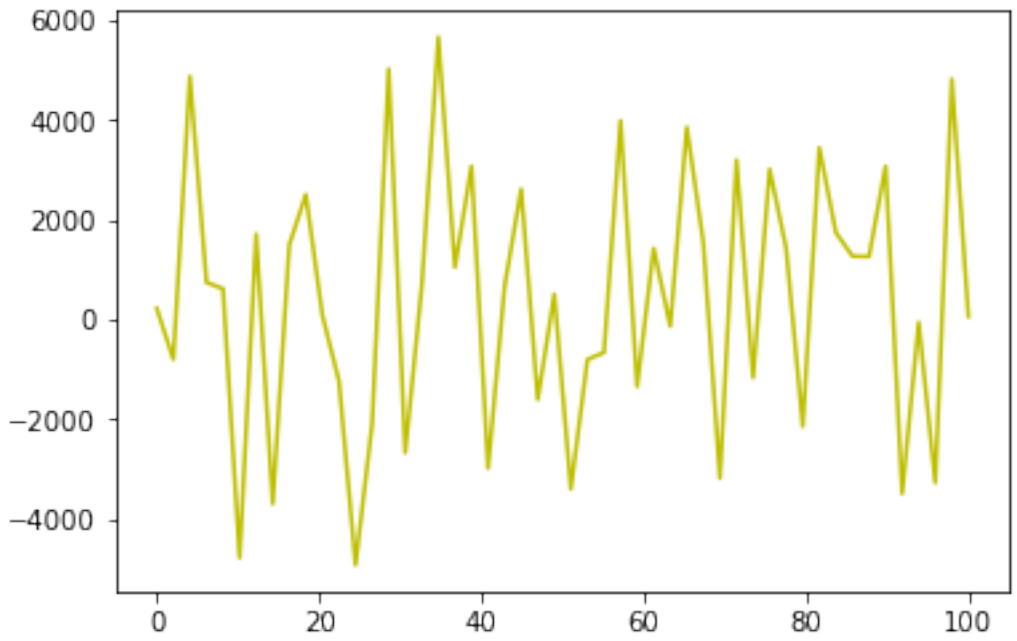
[2]:
```
[<matplotlib.lines.Line2D at 0x7f7c3fed2c40>]
```



2. Make some data that are straight line values from the same straight line relationship as in 1. plus random noise. Plot these data.

[3]:
```
# Random Noise
dat_x = x
dat_y = (np.cos(dat_x) + 10*np.random.randn(len(dat_x)))*300

# Both graph together
plt.plot(dat_x,dat_y, 'y')
```

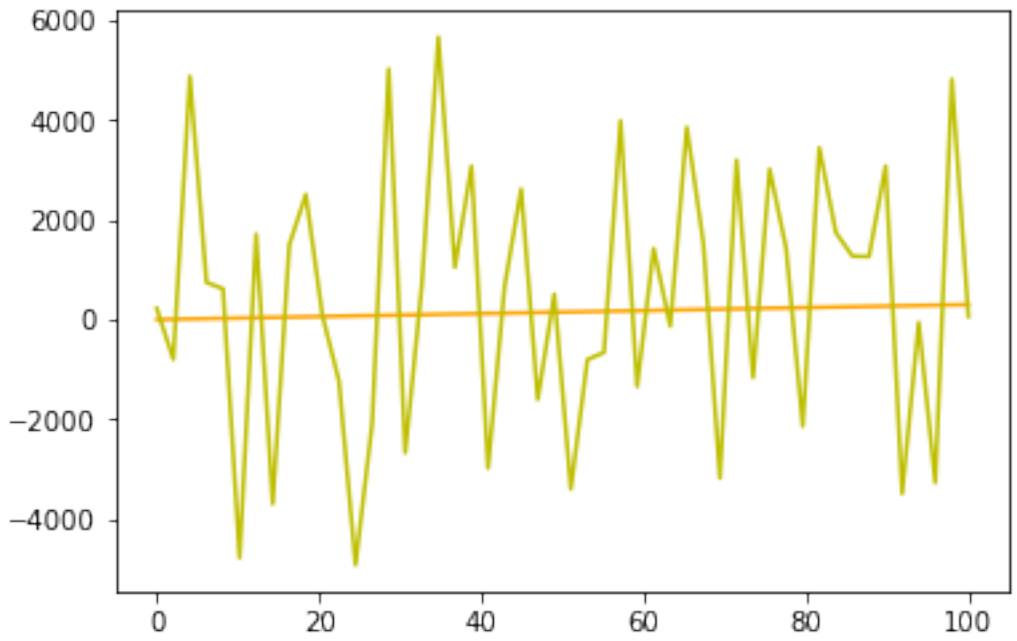
[3]:
```
[<matplotlib.lines.Line2D at 0x7f7c3fffe4f0>]
```



3. Plot the straight line from 1. and the data from 2. on the same graph. Make sure to add the standard annotations, including a legend.

[4]:
```
# Combine graph
plt.plot(x,y,'orange')
plt.plot(dat_x,dat_y, 'y')
```

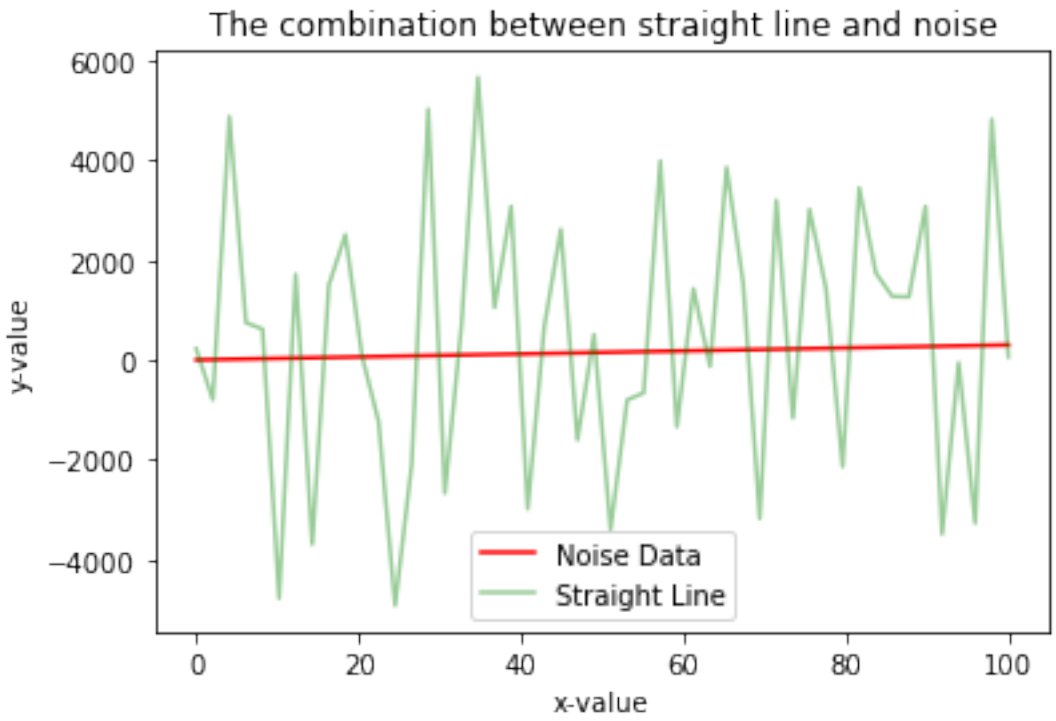
[4]:
```
[<matplotlib.lines.Line2D at 0x7f7c401291c0>]
```



4. Tinker around with your plot (colors, symbols, marker sizes, etc.) until you have a plot you would be happy to use in a presentation.

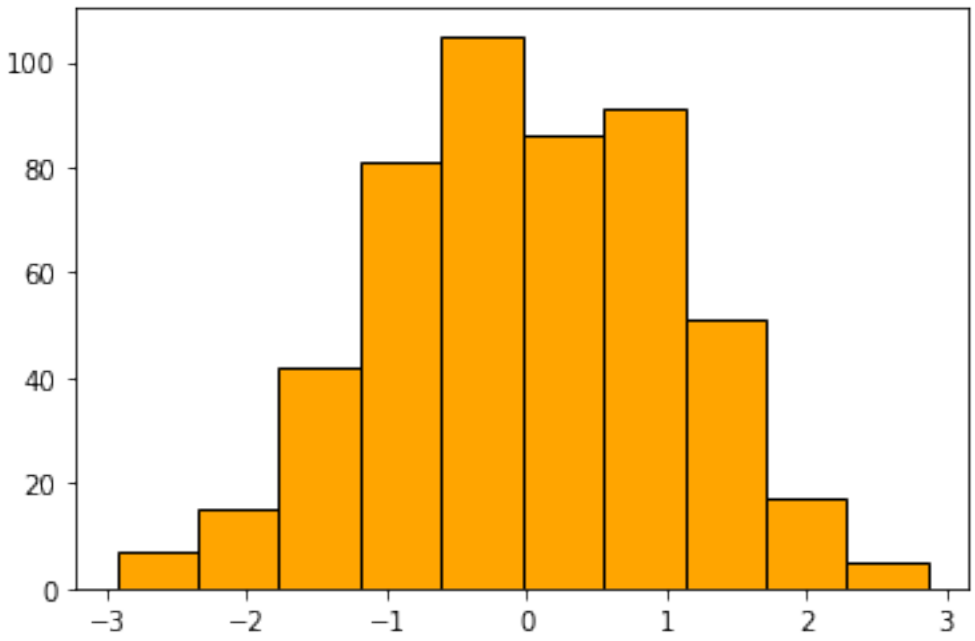
[5]:
```
plt.plot(x,y,label = 'Noise Data',color = 'red')
plt.plot(dat_x,dat_y, label = 'Straight Line', color = 'green', alpha = 0.4)

plt.xlabel('x-value')
plt.ylabel('y-value')
plt.title('The combination between straight line and noise')
plt.legend()
```

[5]:
```
<matplotlib.legend.Legend at 0x7f7c401b9b20>
```



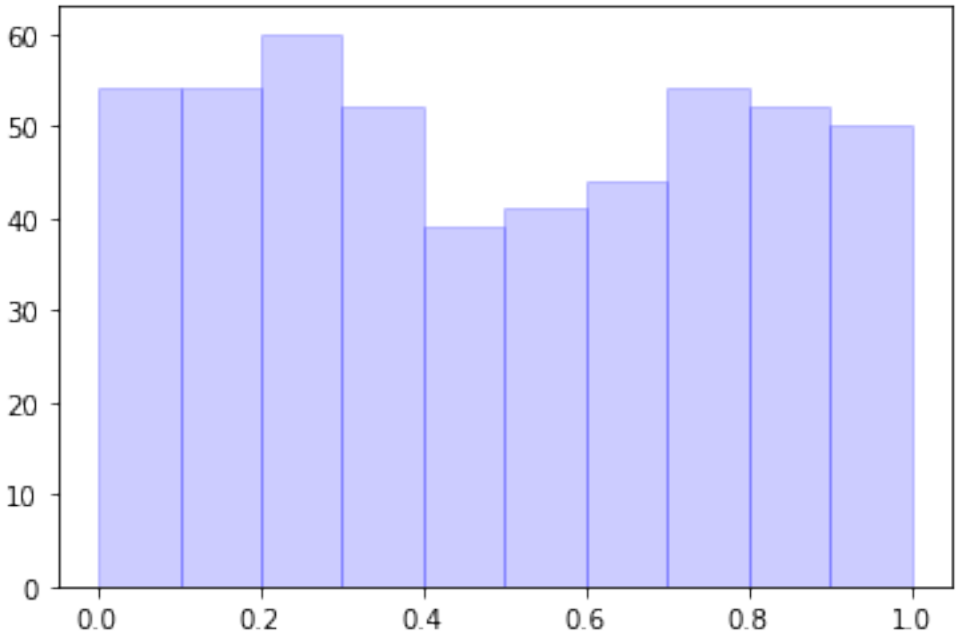
5. Make 500 ***normally*** distributed random numbers and make a histogram of them.

[6]:
```
data = np.random.randn(500)
plt.hist(data, bins = 10, color = 'orange', edgecolor = 'k', alpha = 1);
```



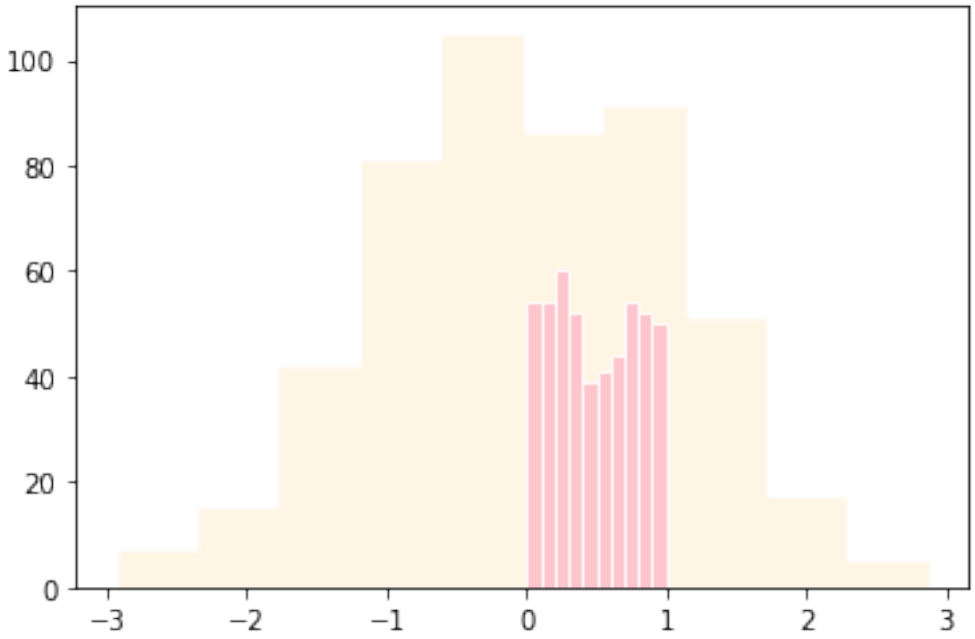
6. Make 500 ***uniformly*** distributed random numbers (use `...rand()` instead of `...randn()`) and make a histogram of them.

[7]:
```
data2 = np.random.rand(500)
plt.hist(data2, bins = 10, color = 'blue', edgecolor = 'blue', alpha = 0.2);
```



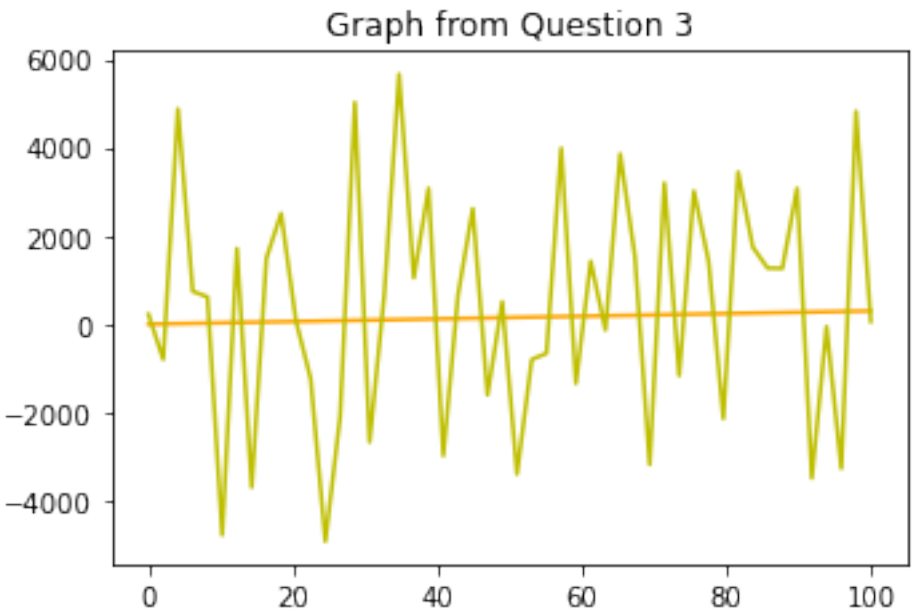
7. Plot the histograms from 5. and 6. in the same axes to compare the two distributions. Tinker around with the `color =` and `alpha =` arguments to `plt.hist()` until you're happy with your figure. Don't forget the axis labels and a legend!

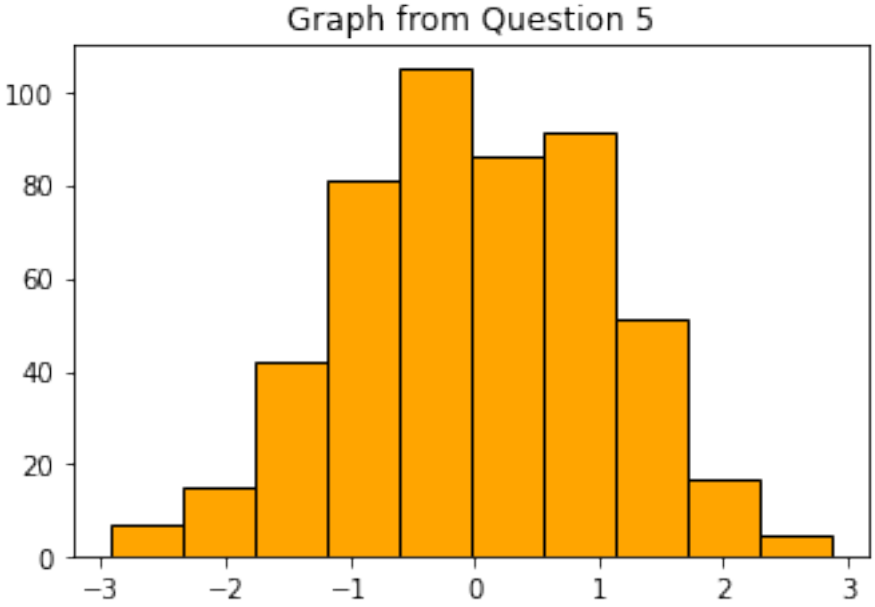
[8]:
```
plt.hist(data, bins = 10, color = 'orange', edgecolor = 'white', alpha = 0.1);
plt.hist(data2, bins = 10, color = 'pink', edgecolor = 'white', alpha = 0.9);
```

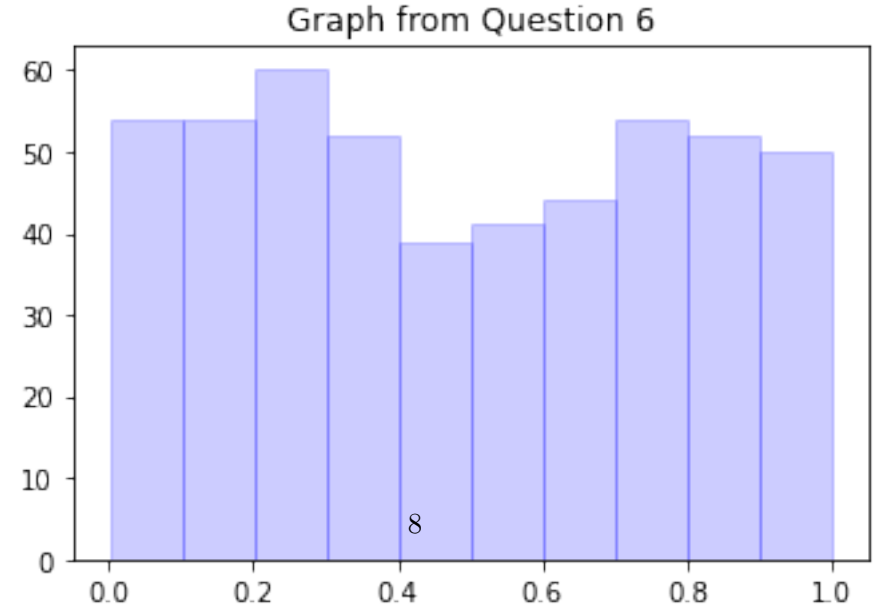


8. Make a figure with 3 subplots, the first containing the plot of the data with a straight line (from 3.), and the second and third containing each of the 2 histograms created in 5. and 6. Try a 3x1 and 1x3 layout and show your favorite.

[9]:
```
# Figure size of each graph
plt.figure(figsize=(5, 10))

# Graph from Question 3
plt.subplot(3,1,1)
plt.plot(x,y,'orange')
plt.plot(dat_x,dat_y, 'y')
plt.title('Graph from Question 3')


# Graph from Question 3
plt.subplot(3,1,2)
plt.hist(data, bins = 10, color = 'orange', edgecolor = 'k', alpha = 1);
plt.title('Graph from Question 5')

# Graph from Question 6
plt.subplot(3,1,3)
plt.hist(data2, bins = 10, color = 'blue', edgecolor = 'blue', alpha = 0.2);
plt.title('Graph from Question 6')

# Make graph fit
plt.tight_layout()
```


Graph from Question 3
6000
4000
2000
0
−2000
−4000
0
20
40
60
80
100


Graph from Question 5
100
80
60
40
20
0
−3
−2
−1
0
1
2
3


Graph from Question 6
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0