Crash Course Review Exercises

Import numpy,pandas,matplotlib,and sklearn. Also set visualizations to be shown inline in the notebook.

I'm also importing the train-test split, so as to have all imorts here.

In [1]:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

%matplotlib inline
```

Set Numpy's Random Seed to 101

In [2]:

```
np.random.seed(101)
```

Create a NumPy Matrix of 100 rows by 5 columns consisting of random integers from 1-100. (Keep in mind that the upper limit may be exclusive.

In [3]:

```
my_matrix = np.random.randint(1, 101, (100, 5))
```

In [4]:

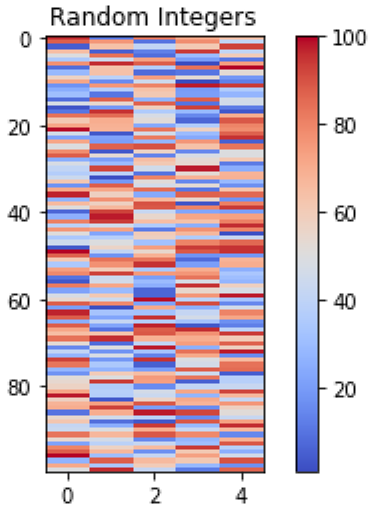
```
my_matrix
```

```
Out[4]: array([[ 96,  12,  82,  71,  64],
               [ 88,  76,  10,  78,  41],
               [  5,  64,  41,  61,  93],
               [ 65,   6,  13,  94,  41],
               [ 50,  84,   9,  30,  60],
               [ 35,  45,  73,  20,  11],
               [ 77,  96,  88,   1,  74],
               [  9,  63,  37,  84, 100],
               [ 29,  64,   8,  11,  53],
               [ 57,  39,  74,  53,  19],
               [ 72,  16,  45,   1,  13],
               [ 18,  76,  80,  98,  94],
               [ 25,  37,  64,  20,  36],
               [ 31,  11,  61,  21,  28],
               [  9,  87,  27,  88,  47],
               [ 48,  55,  87,  10,  46],
               [  3,  19,  59,  93,  12],
               [ 11,  95,  36,  29,   4],
               [ 84,  85,  48,  15,  70],
               [ 61,  70,  52,   7,  89],
               [ 72,  69,  24,  36,  80],
               [ 99,  68,  83,  58,  78],
               [ 47,   4,  47,  30,  87],
               [ 22,  22,  82,  24,  95],
               [ 72,  21,  28,  76,   6],
               [ 50,  87,  90,  64,  83],
               [ 78,   4,  57,  15,  50],
               [ 88,  53,  14,  48,  50],
               [ 25,  21,  65,  53,  61],
               [ 48,  30,  61,  54,  12],
               [ 41,  92,  46,  98,  25],
               [ 37,  39,  10,  53,  68],
               [ 44,   2,  80,  69,  69],
               [ 62,  19,  52,  15,  29],
               [ 18,  88,  47,  53,  17],
               [ 71,  72,  85,  11,  63],
               [ 97,  58,  24,  87,  86],
               [ 27,  77,  67,  55,  18],
               [ 66,  58,  90,   3,  81],
               [ 51,  67,  89,  80,  94],
               [  7,  93,  43,  23,  21],
               [ 26,  98,  55,  72,  73],
               [ 81,  94,  65,  64,  81],
               [ 39,  46,  36,  26,  96],
               [ 76,  73,  12,  77,  80],
               [ 51,  23,  60,  67,   2],
               [ 35,  38,  58,  36,  43],
               [ 45,  50,  32,  80,  86],
               [  4,  56,  74,  94,  95],
               [100,  41,  55,  89,  95],
               [ 87,  18,  69,  18,  19],
               [ 61,  84,  83,   8,  68],
               [ 35,  77,  95,  21,  70],
               [ 74,  60,  35,  70,  26],
               [ 79,  93,  75,  76,  34],
               [ 10,  44,  21,  83,  31],
               [  4,  47,  30,  48,  28],
               [ 82,  72,  26,  95,  58],
               [ 22,  30,   7,  55,  48],
               [ 48,  61,   7,  76,  98],
               [ 54,  45,  99,  40,  33],
               [ 88,  79,  22,  91,  15],
               [ 21,   2,  71,  26,  46],
               [ 97,  33,  32,  42,  80],
               [ 88,  23,  95,  47,  72],
               [ 25,  42,  37,  32,  17],
               [ 88,  23,  97,   4,  13],
               [ 72,  10,  88,  96,  40],
               [ 65,  63,  89,  77,  94],
               [ 84,  96,  69,  70,  60],
               [ 53,   8,  41,  74,  87],
               [ 15,  50,  98,  26,  58],
               [ 41,  18,  33,  84,  98],
               [ 28,  48,  14,  71,  16],
               [ 93,  19,  95,  49,  66],
               [ 83,  35,   6,  47,  84],
               [ 28,  27,  21,  88,  85],
               [ 18,  60,  65,  45,   5],
               [ 52,  50,  75,  83,  38],
               [ 54,  94,  74,   6,  38],
               [ 57,  36,  16,  41,  43],
               [ 72,  38,  47,  72,  92],
               [ 98,  37,  44,  28,  67],
               [ 58,   4,  56,  71,  42],
               [ 68,  73,  89,  68,  76],
               [ 70,  93,  21,  16,  58],
               [ 10,  70,  98,  92,  52],
               [ 55,  46,  39,  16,  43],
               [ 62,   9,   4,  89,  73],
               [ 42,  25,  94,  29,  96],
               [ 44,  49,  70,  43,  67],
               [ 83,  67,  89,  79,  15],
               [ 54,  47,  15,  28,  69],
               [ 22,  39,  43,  31,  89],
               [ 80,  57,  66,  94,  38],
               [ 88,  67,  17,  61,  26],
               [100,  31,  42,  73,  46],
               [ 27,  88,  66,  61,  90],
               [ 71,  34,  60,  29,  17],
               [ 50,  96,  42,  12,  87]])
```

Create a 2-D visualization using plt.imshow of the numpy matrix with a colorbar. Add a title to your plot. Bonus: Figure out how to change the *aspect* (https://stackoverflow.com/questions/10540929/figure-of-imshow-is-too-small) of the imshow() plot.

In [5]:
```
plt.imshow(my_matrix, aspect=0.1, cmap="coolwarm")
plt.colorbar()
plt.title("Random Integers")
```

Out[5]:
```
<matplotlib.text.Text at 0x18bd69a1ba8>
```



Now use pd.DataFrame() to read in this numpy array as a dataframe. Simple pass in the numpy array into that function to get back a dataframe. Pandas will auto label the columns to 0-4

```
In [6]: my_df = pd.DataFrame(my_matrix)
```

In [7]:

```
my_df
```

Out[7]:

	0	1	2	3	4
0	96	12	82	71	64
1	88	76	10	78	41
2	5	64	41	61	93
3	65	6	13	94	41
4	50	84	9	30	60
5	35	45	73	20	11
6	77	96	88	1	74
7	9	63	37	84	100
8	29	64	8	11	53
9	57	39	74	53	19
10	72	16	45	1	13
11	18	76	80	98	94
12	25	37	64	20	36
13	31	11	61	21	28
14	9	87	27	88	47
15	48	55	87	10	46
16	3	19	59	93	12
17	11	95	36	29	4
18	84	85	48	15	70
19	61	70	52	7	89
20	72	69	24	36	80
21	99	68	83	58	78
22	47	4	47	30	87
23	22	22	82	24	95
24	72	21	28	76	6
25	50	87	90	64	83

	0	1	2	3	4
26	78	4	57	15	50
27	88	53	14	48	50
28	25	21	65	53	61
29	48	30	61	54	12
...	...	...	...	...	...
70	53	8	41	74	87
71	15	50	98	26	58
72	41	18	33	84	98
73	28	48	14	71	16
74	93	19	95	49	66
75	83	35	6	47	84
76	28	27	21	88	85
77	18	60	65	45	5
78	52	50	75	83	38
79	54	94	74	6	38
80	57	36	16	41	43
81	72	38	47	72	92
82	98	37	44	28	67
83	58	4	56	71	42
84	68	73	89	68	76
85	70	93	21	16	58
86	10	70	98	92	52
87	55	46	39	16	43
88	62	9	4	89	73
89	42	25	94	29	96
90	44	49	70	43	67
91	83	67	89	79	15

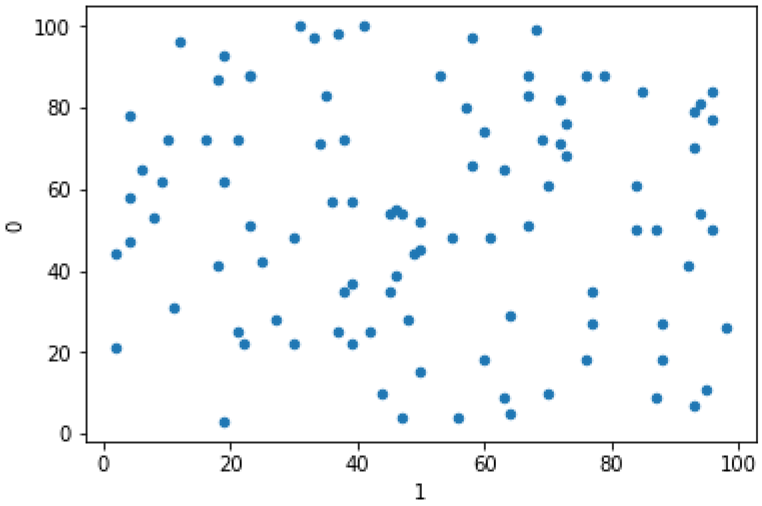
	0	1	2	3	4
92	54	47	15	28	69
93	22	39	43	31	89
94	80	57	66	94	38
95	88	67	17	61	26
96	100	31	42	73	46
97	27	88	66	61	90
98	71	34	60	29	17
99	50	96	42	12	87

100 rows × 5 columns

Now create a scatter plot using pandas of the 0 column vs the 1 column.

```
In [8]: y_column = 0
        x_column = 1
        my_df.plot(x_column, y_column, kind="scatter") #  I disagree with his
                                                       #+ saying "0 vs. 1",
                                                       #+ then plotting it as
                                                       #+ x=0 and y=1.
                                                       #+ As I told my
                                                       #+ students, it's
                                                       #+ always y vs. x
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x18bce620c18>
```

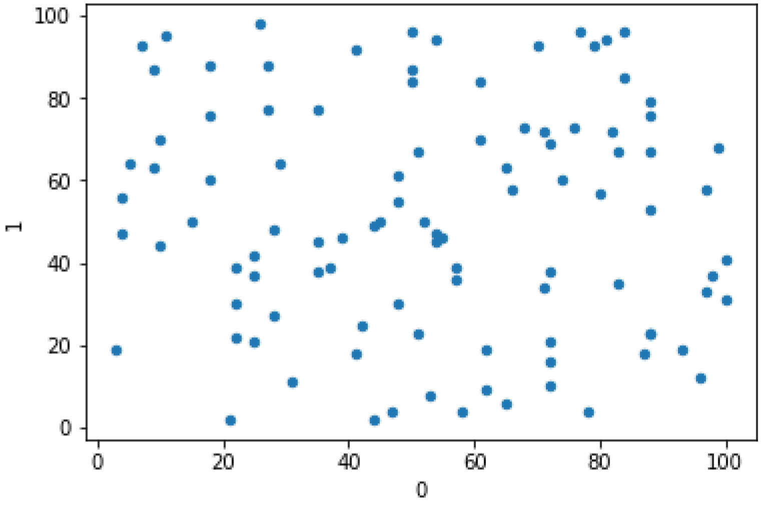


In [9]:

```
# Jose's version, for comparison
my_df.plot(x=0, y=1, kind='scatter')
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18bd6abc400>
```



Now scale the data to have a minimum of 0 and a maximum value of 1 using scikit-learn.

In [10]:

```
my_scaler = MinMaxScaler()
```

In [11]:

```
scaled_data = my_scaler.fit_transform(my_df) #  The number of cells suggests
    #+ the combined commarnd. In
    #+ practice, we'll usually
    #+ have separate training and
    #+ test data. We'll fit to the
                                          #+ training data only and use
                                          #+ the resulting fit to
                                          #+ transform the test data
                                          #+ after transforming the
                                          #+ training data in order to
                                          #+ train with it.
```

In [12]:

```
scaled_data
```

```
Out[12]: array([[0.95876289, 0.10416667, 0.82105263, 0.72164948, 0.63265306],
                [0.87628866, 0.77083333, 0.06315789, 0.79381443, 0.39795918],
                [0.02061856, 0.64583333, 0.38947368, 0.6185567 , 0.92857143],
                [0.63917526, 0.04166667, 0.09473684, 0.95876289, 0.39795918],
                [0.48453608, 0.85416667, 0.05263158, 0.29896907, 0.59183673],
                [0.32989691, 0.44791667, 0.72631579, 0.19587629, 0.09183673],
                [0.7628866 , 0.97916667, 0.88421053, 0.        , 0.73469388],
                [0.06185567, 0.63541667, 0.34736842, 0.8556701 , 1.        ],
                [0.26804124, 0.64583333, 0.04210526, 0.10309278, 0.52040816],
                [0.55670103, 0.38541667, 0.73684211, 0.53608247, 0.17346939],
                [0.71134021, 0.14583333, 0.43157895, 0.        , 0.1122449 ],
                [0.15463918, 0.77083333, 0.8       , 1.        , 0.93877551],
                [0.22680412, 0.36458333, 0.63157895, 0.19587629, 0.34693878],
                [0.28865979, 0.09375   , 0.6       , 0.20618557, 0.26530612],
                [0.06185567, 0.88541667, 0.24210526, 0.89690722, 0.45918367],
                [0.46391753, 0.55208333, 0.87368421, 0.09278351, 0.44897959],
                [0.        , 0.17708333, 0.57894737, 0.94845361, 0.10204082],
                [0.08247423, 0.96875   , 0.33684211, 0.28865979, 0.02040816],
                [0.83505155, 0.86458333, 0.46315789, 0.1443299 , 0.69387755],
                [0.59793814, 0.70833333, 0.50526316, 0.06185567, 0.8877551 ],
                [0.71134021, 0.69791667, 0.21052632, 0.36082474, 0.79591837],
                [0.98969072, 0.6875    , 0.83157895, 0.58762887, 0.7755102 ],
                [0.45360825, 0.02083333, 0.45263158, 0.29896907, 0.86734694],
                [0.19587629, 0.20833333, 0.82105263, 0.2371134 , 0.94897959],
                [0.71134021, 0.19791667, 0.25263158, 0.77319588, 0.04081633],
                [0.48453608, 0.88541667, 0.90526316, 0.64948454, 0.82653061],
                [0.77319588, 0.02083333, 0.55789474, 0.1443299 , 0.48979592],
                [0.87628866, 0.53125   , 0.10526316, 0.48453608, 0.48979592],
                [0.22680412, 0.19791667, 0.64210526, 0.53608247, 0.60204082],
                [0.46391753, 0.29166667, 0.6       , 0.54639175, 0.10204082],
                [0.39175258, 0.9375    , 0.44210526, 1.        , 0.23469388],
                [0.35051546, 0.38541667, 0.06315789, 0.53608247, 0.67346939],
                [0.42268041, 0.        , 0.8       , 0.70103093, 0.68367347],
                [0.60824742, 0.17708333, 0.50526316, 0.1443299 , 0.2755102 ],
                [0.15463918, 0.89583333, 0.45263158, 0.53608247, 0.15306122],
                [0.70103093, 0.72916667, 0.85263158, 0.10309278, 0.62244898],
                [0.96907216, 0.58333333, 0.21052632, 0.88659794, 0.85714286],
                [0.24742268, 0.78125   , 0.66315789, 0.55670103, 0.16326531],
                [0.64948454, 0.58333333, 0.90526316, 0.02061856, 0.80612245],
                [0.49484536, 0.67708333, 0.89473684, 0.81443299, 0.93877551],
                [0.04123711, 0.94791667, 0.41052632, 0.22680412, 0.19387755],
                [0.2371134 , 1.        , 0.53684211, 0.73195876, 0.7244898 ],
                [0.80412371, 0.95833333, 0.64210526, 0.64948454, 0.80612245],
                [0.37113402, 0.45833333, 0.33684211, 0.25773196, 0.95918367],
                [0.75257732, 0.73958333, 0.08421053, 0.78350515, 0.79591837],
                [0.49484536, 0.21875   , 0.58947368, 0.68041237, 0.        ],
                [0.32989691, 0.375     , 0.56842105, 0.36082474, 0.41836735],
                [0.43298969, 0.5       , 0.29473684, 0.81443299, 0.85714286],
                [0.01030928, 0.5625    , 0.73684211, 0.95876289, 0.94897959],
                [1.        , 0.40625   , 0.53684211, 0.90721649, 0.94897959],
                [0.86597938, 0.16666667, 0.68421053, 0.17525773, 0.17346939],
                [0.59793814, 0.85416667, 0.83157895, 0.07216495, 0.67346939],
                [0.32989691, 0.78125   , 0.95789474, 0.20618557, 0.69387755],
                [0.73195876, 0.60416667, 0.32631579, 0.71134021, 0.24489796],
                [0.78350515, 0.94791667, 0.74736842, 0.77319588, 0.32653061],
                [0.07216495, 0.4375    , 0.17894737, 0.84536082, 0.29591837],
                [0.01030928, 0.46875   , 0.27368421, 0.48453608, 0.26530612],
                [0.81443299, 0.72916667, 0.23157895, 0.96907216, 0.57142857],
                [0.19587629, 0.29166667, 0.03157895, 0.55670103, 0.46938776],
                [0.46391753, 0.61458333, 0.03157895, 0.77319588, 0.97959184],
                [0.5257732 , 0.44791667, 1.        , 0.40206186, 0.31632653],
                [0.87628866, 0.80208333, 0.18947368, 0.92783505, 0.13265306],
                [0.18556701, 0.        , 0.70526316, 0.25773196, 0.44897959],
                [0.96907216, 0.32291667, 0.29473684, 0.42268041, 0.79591837],
                [0.87628866, 0.21875   , 0.95789474, 0.4742268 , 0.71428571],
                [0.22680412, 0.41666667, 0.34736842, 0.31958763, 0.15306122],
                [0.87628866, 0.21875   , 0.97894737, 0.03092784, 0.1122449 ],
                [0.71134021, 0.08333333, 0.88421053, 0.97938144, 0.3877551 ],
                [0.63917526, 0.63541667, 0.89473684, 0.78350515, 0.93877551],
                [0.83505155, 0.97916667, 0.68421053, 0.71134021, 0.59183673],
                [0.51546392, 0.0625    , 0.38947368, 0.75257732, 0.86734694],
                [0.12371134, 0.5       , 0.98947368, 0.25773196, 0.57142857],
                [0.39175258, 0.16666667, 0.30526316, 0.8556701 , 0.97959184],
                [0.25773196, 0.47916667, 0.10526316, 0.72164948, 0.14285714],
                [0.92783505, 0.17708333, 0.95789474, 0.49484536, 0.65306122],
                [0.82474227, 0.34375   , 0.02105263, 0.4742268 , 0.83673469],
                [0.25773196, 0.26041667, 0.17894737, 0.89690722, 0.84693878],
                [0.15463918, 0.60416667, 0.64210526, 0.45360825, 0.03061224],
                [0.50515464, 0.5       , 0.74736842, 0.84536082, 0.36734694],
                [0.5257732 , 0.95833333, 0.73684211, 0.05154639, 0.36734694],
                [0.55670103, 0.35416667, 0.12631579, 0.41237113, 0.41836735],
                [0.71134021, 0.375     , 0.45263158, 0.73195876, 0.91836735],
                [0.97938144, 0.36458333, 0.42105263, 0.27835052, 0.66326531],
                [0.56701031, 0.02083333, 0.54736842, 0.72164948, 0.40816327],
                [0.67010309, 0.73958333, 0.89473684, 0.69072165, 0.75510204],
                [0.69072165, 0.94791667, 0.17894737, 0.15463918, 0.57142857],
                [0.07216495, 0.70833333, 0.98947368, 0.93814433, 0.51020408],
                [0.53608247, 0.45833333, 0.36842105, 0.15463918, 0.41836735],
                [0.60824742, 0.07291667, 0.        , 0.90721649, 0.7244898 ],
                [0.40206186, 0.23958333, 0.94736842, 0.28865979, 0.95918367],
                [0.42268041, 0.48958333, 0.69473684, 0.43298969, 0.66326531],
                [0.82474227, 0.67708333, 0.89473684, 0.80412371, 0.13265306],
                [0.5257732 , 0.46875   , 0.11578947, 0.27835052, 0.68367347],
                [0.19587629, 0.38541667, 0.41052632, 0.30927835, 0.8877551 ],
                [0.79381443, 0.57291667, 0.65263158, 0.95876289, 0.36734694],
                [0.87628866, 0.67708333, 0.13684211, 0.6185567 , 0.24489796],
                [1.        , 0.30208333, 0.4       , 0.74226804, 0.44897959],
                [0.24742268, 0.89583333, 0.65263158, 0.6185567 , 0.89795918],
                [0.70103093, 0.33333333, 0.58947368, 0.28865979, 0.15306122],
                [0.48453608, 0.97916667, 0.4       , 0.11340206, 0.86734694]])
```

Using your previously created DataFrame, use [df.columns = [...] (https://stackoverflow.com/questions/11346283/renaming-columns-in-pandas)](https://stackoverflow.com/questions/11346283/renaming-columns-in-pandas) to rename the pandas columns to be ['f1','f2','f3','f4','label']. Then perform a train/test split with scikitlearn.

In [13]:

```python
my_df.columns = ['f1', 'f2', 'f3', 'f4', 'label']
```

In [14]:

```python
#  I already imported train_test_split. I will mention that
#+ I'll type in 'train_test_split', then use [Shift] + [Tab]
#+ to get the basic command I want. I'm going to comment
#+ out my more-complicated command and put in Jose's
#+ bare-bones command
```

In [15]:

```python
X = my_df[['f1', 'f2', 'f3', 'f4']]
y = my_df['label']
```

In [16]:

```python
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

I want to check it out. I don't know how to look at the `random state` right away, so I won't worry about it. However, I can determine the `test_size` .

In [17]:

```
print("X_train.shape = " + str(X_train.shape))
print("X_test.shape = " + str(X_test.shape))
test_size_ratio = X_test.shape[0] / (X_test.shape[0] + X_train.shape[0])
print("test_size_ratio = " + str(test_size_ratio))
```

```
X_train.shape = (75, 4)
X_test.shape = (25, 4)
test_size_ratio = 0.25
```

In [18]:

```
print("y_train.shape = " + str(y_train.shape))
print("y_test.shape = " + str(y_test.shape))
test_size_ratio_from_y = y_test.shape[0] / (y_test.shape[0] + y_train.shape[0])
print("test_size_ratio_from_y = " + str(test_size_ratio_from_y))
```

```
y_train.shape = (75,)
y_test.shape = (25,)
test_size_ratio_from_y = 0.25
```

Great Job!