Exercise for Tutorial 14

Learn how to understand and describe code with words!

So far we have read text that describes code. We have looked at code and tried to understand what it does, with the help of a verbal description (in most cases).

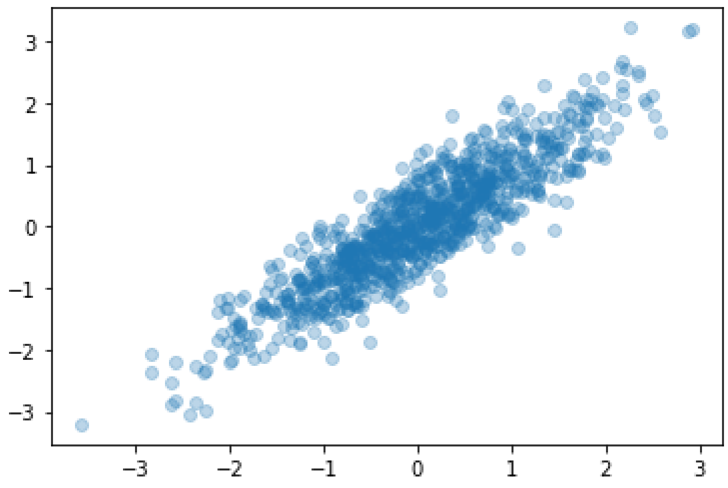
Here after, you will be asked to look at a block of code (that you might or might have not seen before) and describe in words what the code is doing.

You are welcome to look at the documentation of the methods, functions and libraries used but it would be asked to not use Google the answer in this case (besides using Google to find the documentation of the function on the python libraries web pages).

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [6]:
```
from  numpy.random import multivariate_normal # Importing the program that can c
# Defining the parameters
mu1 = 0;  # Defining the mean of our data sets
mu2 = 0;
var1 = 1; # Defining the variance of our data sets
var2 = 1;
cov = 0.9; # Defining the co-variance between our two datasets
cov_m = [[var1, cov],     # Making the covariance matrix
         [cov, var2]]
data = multivariate_normal([mu1, mu2], cov_m, size=1000) # Defining the 2 x1000
plt.scatter(data[:,0], data[:,1], alpha = 0.3) # Plotting the two data sets into
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x127f3ae80>
```



Explain in your words what the code is doing. Do that by commenting each line of the code. The description should cober each parameter (say the variables initialized at the beginning) but also each operation in the code, for example the calls to a function should be described as well and the indexing operations. For the indexing please make sure to be explicit about which dimensions of the array are being addressed and also report the full dimensions of the array as you describe the indexing operation. We expect 1-3 pragraphs of description max.

Graph 1 Description

With this first set of code we are creating a multivariate matrix in order to compare two correlated sets of data. We begin by defining our important variabels (mu1&2 = mean, var1&2 = the variance among all the points). These are lines 3 through 6. We set the means for each data set to be equal to 0, and the variance to be equal to 1. This should essentially create a a normal distribution for each data set in which the data is centered around a mean of 0, and varies by about 1 to 3 standard deviations from the point (i.e. can get points such as 0, 1, 2, or 3). The reason for 3 standard deviations is because this holds essentially all the data in a normal distribution. We also define the co-variance between the two data sets with the variable cov, which we se to 0.9. This is essentially how strongly our two data sets will correlate with each other. So in this case since the cov is 0.9 and the max is 1, these two datasets should be highly correlated with one another.

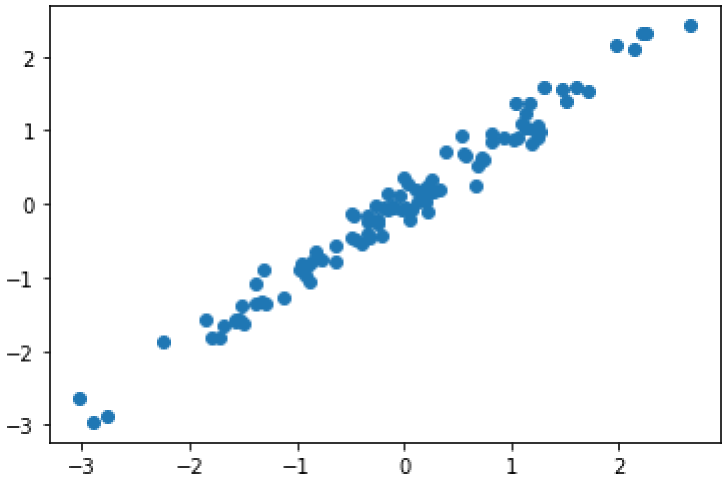
The last section of our code focuses creating our multivariate matrix. We began the code by importing numpy.random as as multivariate normal. This will allow us to create our matrix by creating two randomly generated sets of data each sorted into an array of 1000 x 1, each with a mean of 0 and variance of 1. The cov_m is how we organize our variances and covariance to be input into the multivariate_normal set of code. Next we organize all of this into the data index on the secon to last line. We use the multivariate_normal and proceed to tell it what the means of our two datasets are, input the variances and covariance we made with our cov_m line, and set the size of the data equal to 1000. This will essentially the multivariate_normal code to randomly generate two datasets with a sample size of 1000. Finally, we plot the code in a scatter plot, plotting all the rows (delineated with the : in [:,0] and [:,1]). We also set the alpha = 0.3 so that we can more easily see the distribution of data.

Create a series of correlated datasets using `for loops`

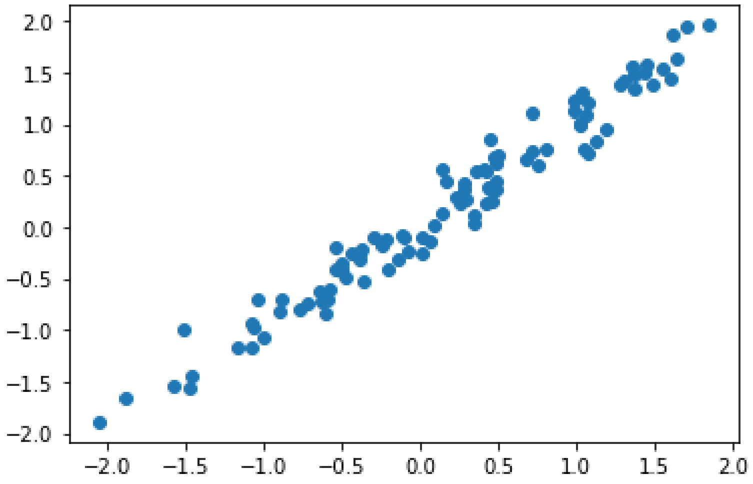
The code below, shows how to create and plot a series of correlated datasets.

In [4]:
```
counter = 0; # Defining the starting counter for our while loop
m = 5; #number of datasets
n = 100; # Setting the size of each of data sets, essentially a data array of 10
scaling = 0.2 # Assigning our amount of noise for our comparison dataset
while counter < m : #
    y = np.random.randn(n,1)
    x = y + scaling*np.random.randn(n,1)
    plt.scatter(x, y)
    plt.show()
    print("We are plotting because we are INSIDE the while loop.")
    plt.pause(0.05)
    counter = counter + 1
```

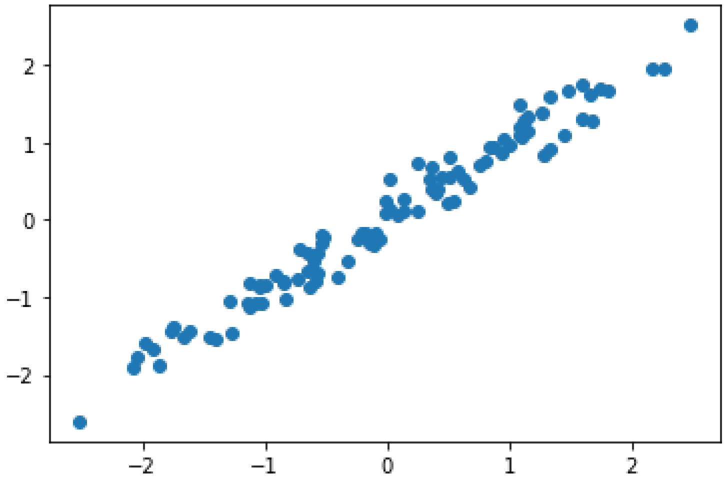
```
else:
    print("We are NOT plotting because we are OUTSIDE the while loop.")
```



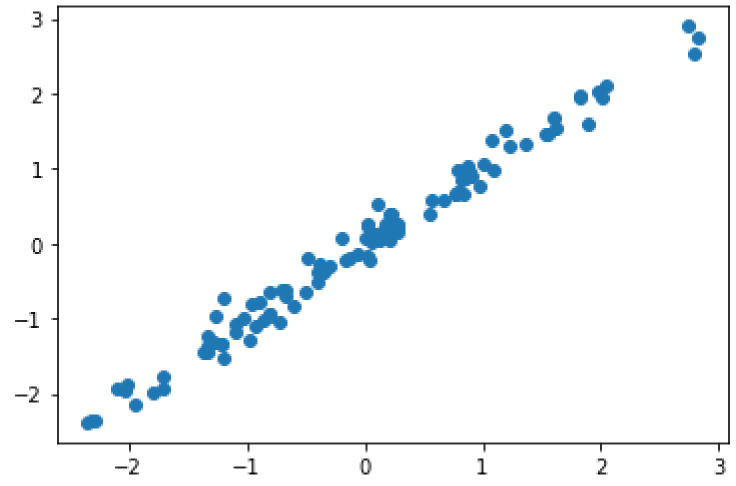
```
We are plotting because we are INSIDE the while loop.
```



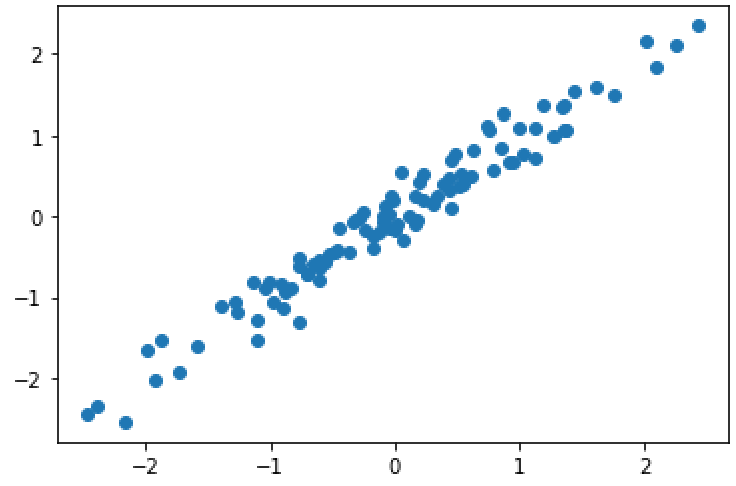
```
We are plotting because we are INSIDE the while loop.
```



```
We are plotting because we are INSIDE the while loop.
```



```
We are plotting because we are INSIDE the while loop.
```

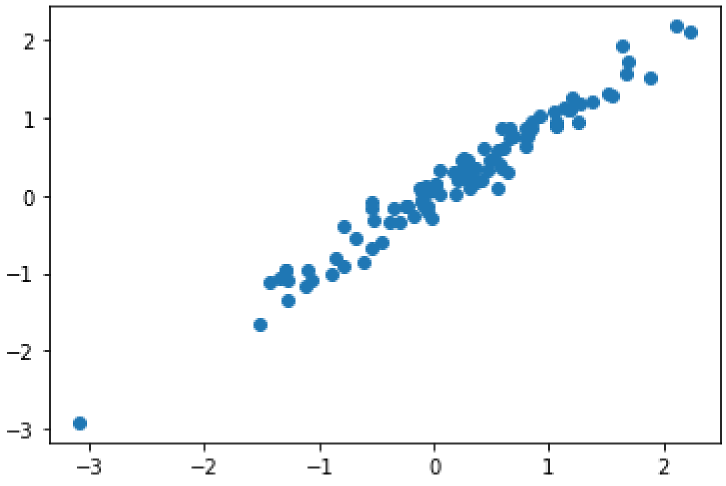


```
We are plotting because we are INSIDE the while loop.
We are NOT plotting because we are OUTSIDE the while loop.
```

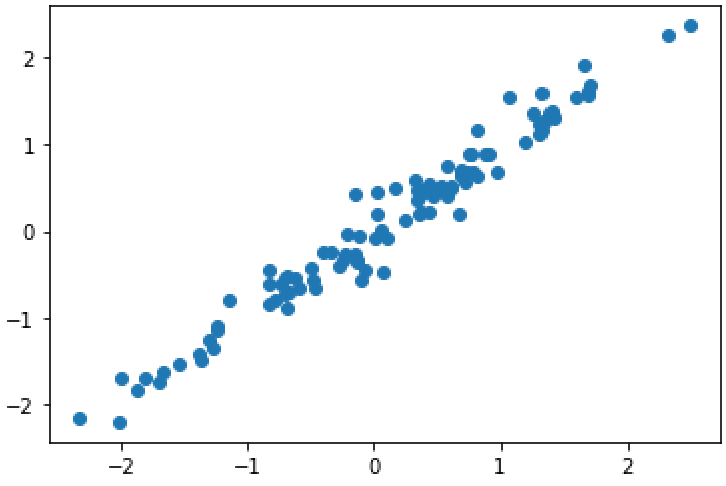
You are tasked to write similar code (that generates and plots correlated datasets). But, your code should use a `for` loop instead of the `while` loop shown in the example.

In [63]:

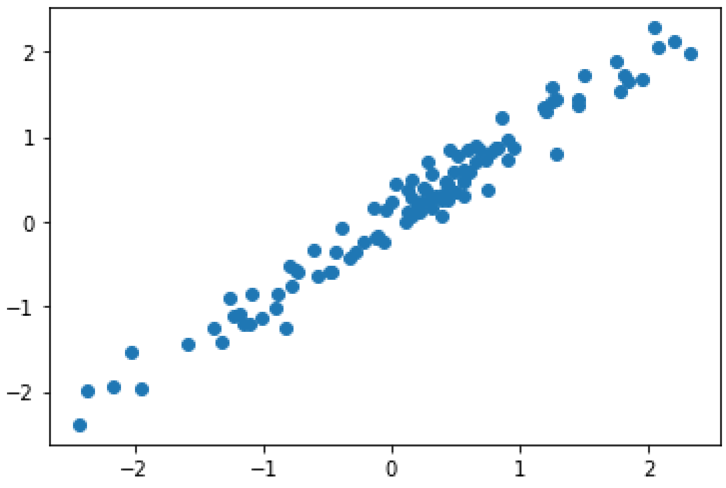
```python
counter = 0; # Defining the starting counter for our while loop
m = 5; #number of datasets
n = 100; # Setting the size of each of data sets, essentially a data array of 10
scaling = 0.2 # Assigning our amount of noise for our comparison dataset

for i in range(m):
    y = np.random.randn(n,1)
    x = y + scaling*np.random.randn(n,1)
    plt.scatter(x, y)
    plt.show()
    print("Here is one of our datasets")
```



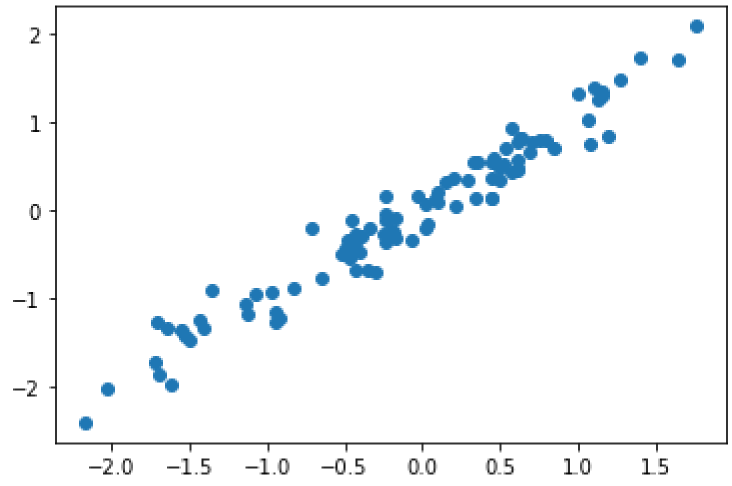
```
Here is one of our datasets
```



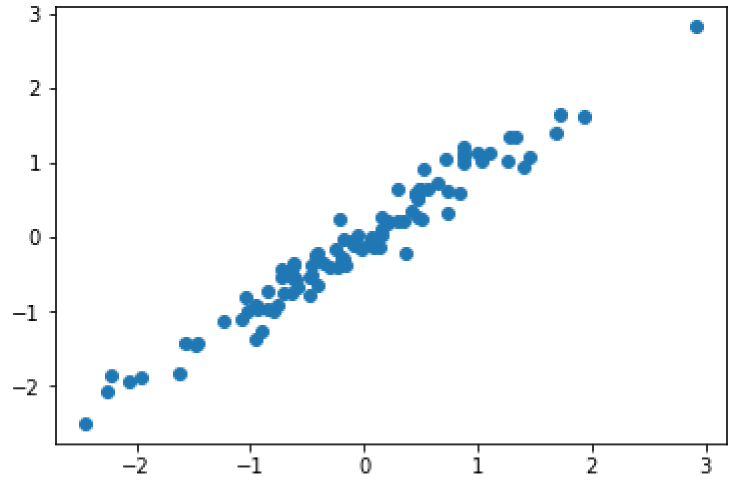
```
Here is one of our datasets
```



```
Here is one of our datasets
```



```
Here is one of our datasets
```



```
Here is one of our datasets
```

Make pretty plots

A majority of data science tasks, will not be a simple plug and play of previously learned tools and code snippet. Instead, they will require learning new skills on the job. Here we ask you to make a few plots based on the previously learned functions. Yet, we ask that you go a little beyond that, by finding ways to improve the visuals of the plots. You can do this by learning about the functionality of `scatter` and `plot` online.

What does it mean to improve the visuals of the plots? Improving the visuals means that the plots look simple, slick, elegant the colors are not the default but are personalized and well chosen. You are free to do the customization of the plots as you prefer, have fun with it and see what comes out of the fun!

This is the plot we ask you to do: Use plt.subplot to make

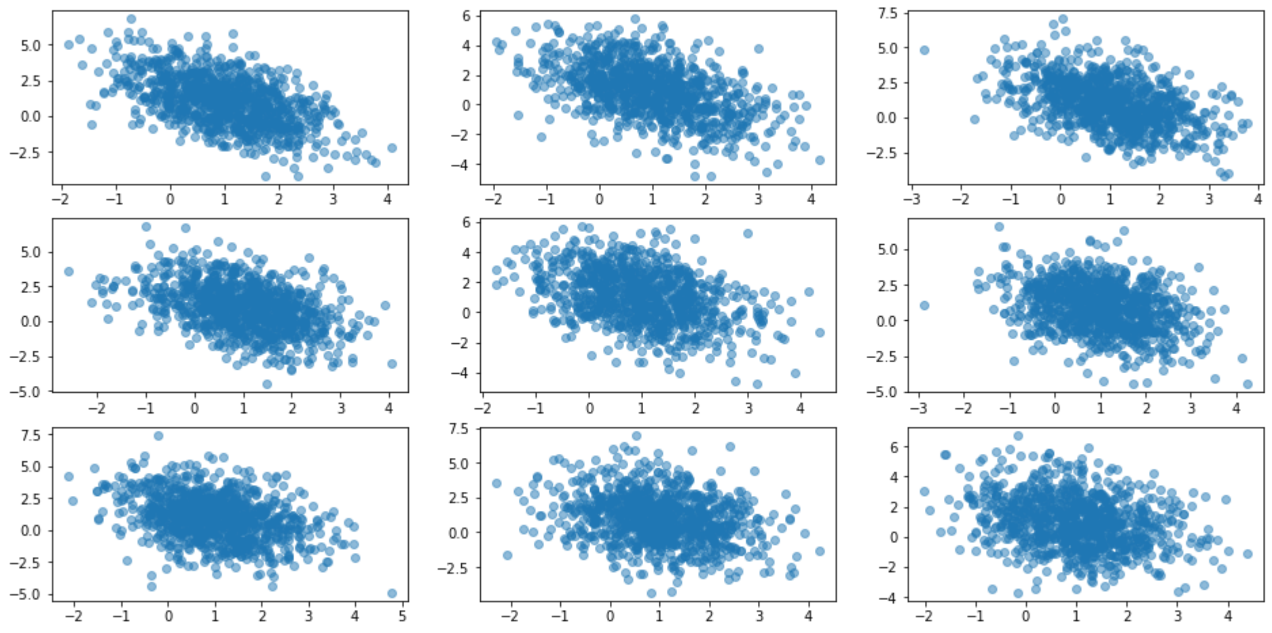
- a 3 x 3 array of scatter plots of negatively correlated datasets
- the correlated datasets should have
  - Dataset 1: mu = 1, variance = 1
  - Dataset 2: mu = 1, variance = 3
  - covariances should be between 0.3 and 0.9

- Make the plots as pretty as possible, for example:
    - change the colors
    - Removing any superfluous visuals, such as boxes aroung the axis (good plots only have two axis not 4!)
    - Add labels
    - Add titles
    - etc

In [26]:
```
steps = (0.9-0.3)/9
cov = np.arange(-0.9,-0.3,steps)
cov2d = np.reshape(cov,(3,3))
cov2d
```

Out[26]:
```
array([[-0.9       , -0.83333333, -0.76666667],
       [-0.7       , -0.63333333, -0.56666667],
       [-0.5       , -0.43333333, -0.36666667]])
```

In [60]:
```
#Describing the
fig, axs = plt.subplots(3,3, figsize = (16,8))
mu1  = 1
mu2  = 1
var1  = 1
var2  = 3
var3  = 2
for i in range(len(cov2d)):
    for j in range(len(cov2d[i])):
        cov_m = [[var1,cov2d[i,j]],
                [cov2d[i,j],var2]]
        data = multivariate_normal([mu1, mu2], cov_m, size=1000)
        ax = axs[i,j].scatter(data[:,0], data[:,1], alpha = 0.5)
```



In [ ]: