

# 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).

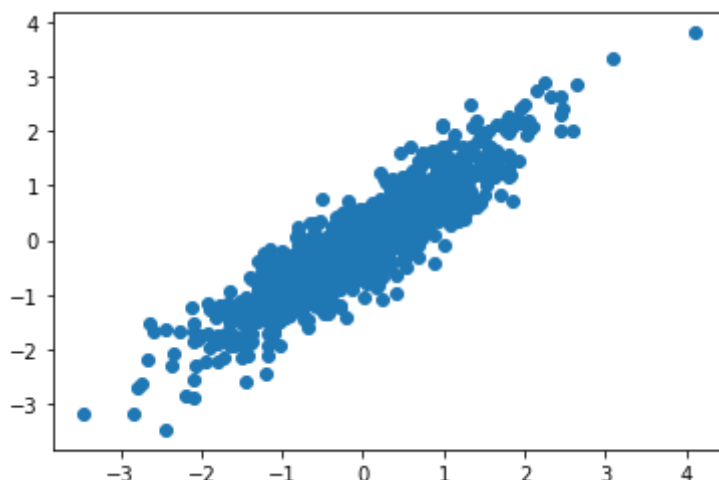
Here after, you will be asked to look at a block of code (that you might or might have not seen before) and describe in words what the code is doing.

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 [4]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from numpy.random import multivariate_normal
```

```
In [5]: from numpy.random import multivariate_normal
mu1 = 0;
mu2 = 0;
var1 = 1;
var2 = 1;
cov = 0.9;
cov_m = [[var1, cov],
         [cov, var2]]
data = multivariate_normal([mu1, mu2], cov_m, size=1000)
plt.scatter(data[:,0], data[:,1])
```

```
Out[5]: <matplotlib.collections.PathCollection at 0x7fac189321c0>
```



Explain in your words what the code is doing. Do that by commenting each line of the code. The description should cover 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 paragraphs of description max.

Type *Markdown* and LaTeX:  $\alpha^2$

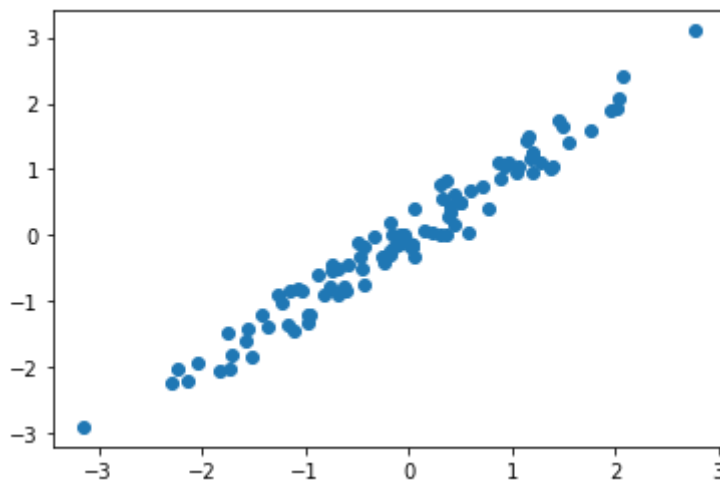
## Create a series of correlated datasets using for loops

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

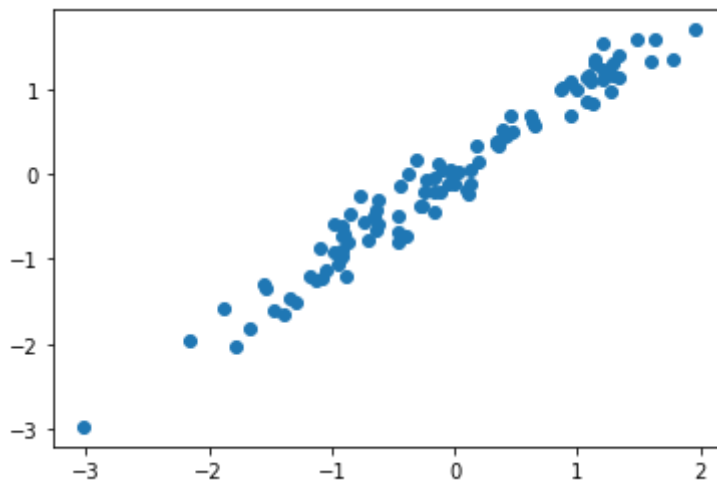
```

In [6]: counter = 0;
m = 5; #number of datasets
n = 100;
scaling = 0.2
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.")

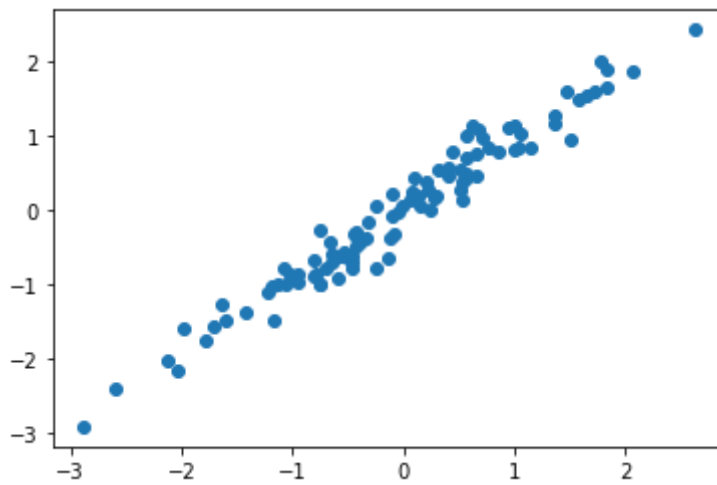
```



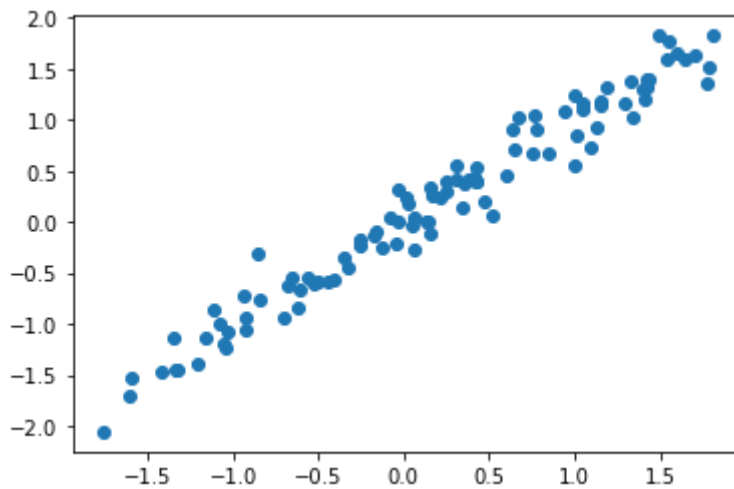
We are plotting because we are INSIDE the while loop.



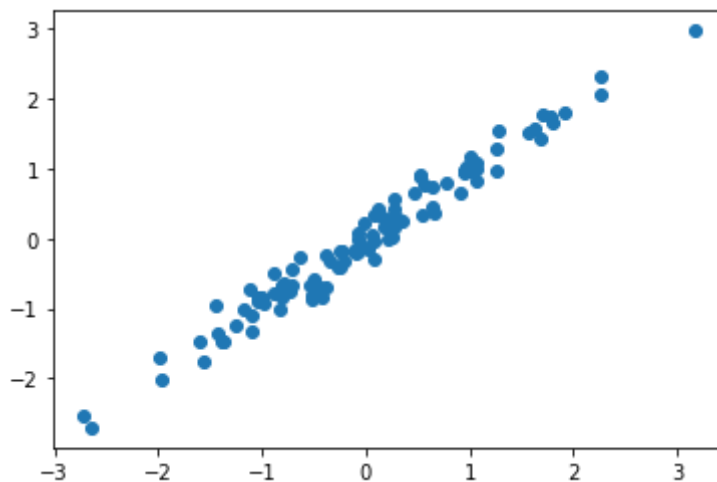
We are plotting because we are INSIDE the while loop.



We are plotting because we are INSIDE the while loop.



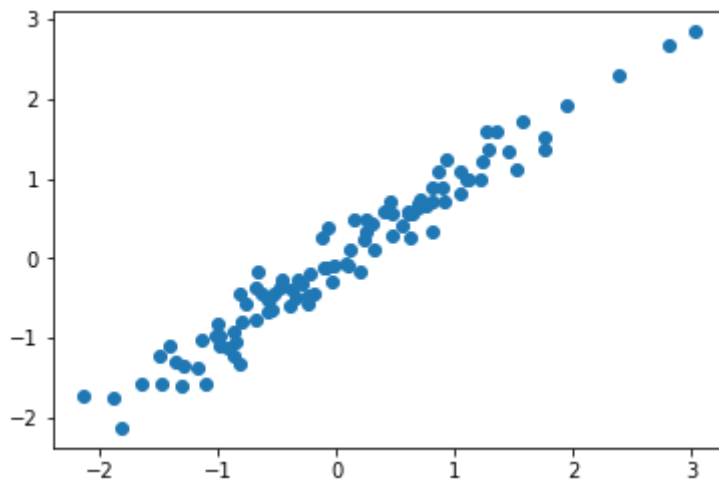
We are plotting because we are INSIDE the while loop.



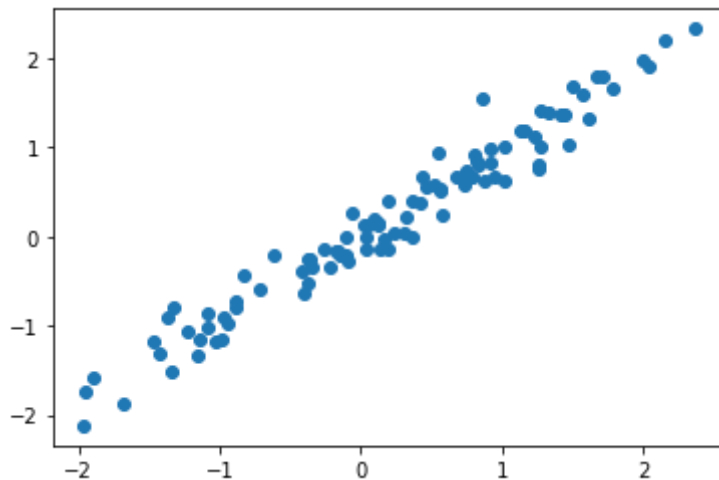
We are plotting because we are INSIDE the while loop.  
We are NOT plotting because we are OUTSIDE the while loop.

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.

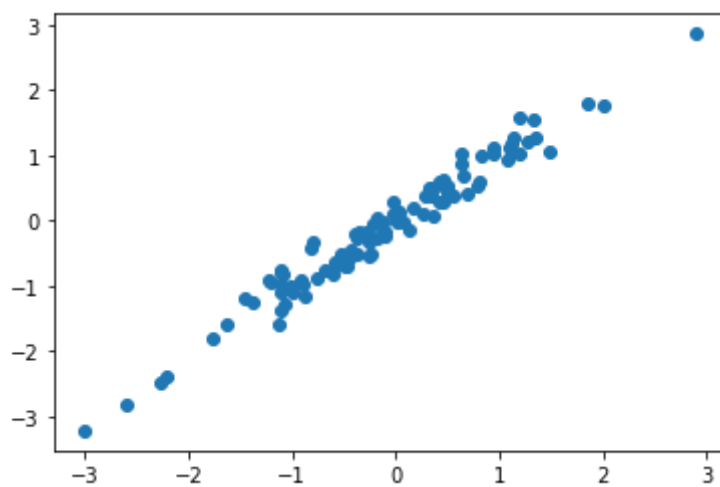
```
In [7]: m = 5
n = 100
scaling = 0.2
for i in range (m):
    y = np.random.randn(n,1)
    x = y + scaling*np.random.randn(n,1)
    plt.scatter(x, y)
    plt.show()
    print("We are plotting because we are INSIDE the for loop.")
    plt.pause(0.05)
```



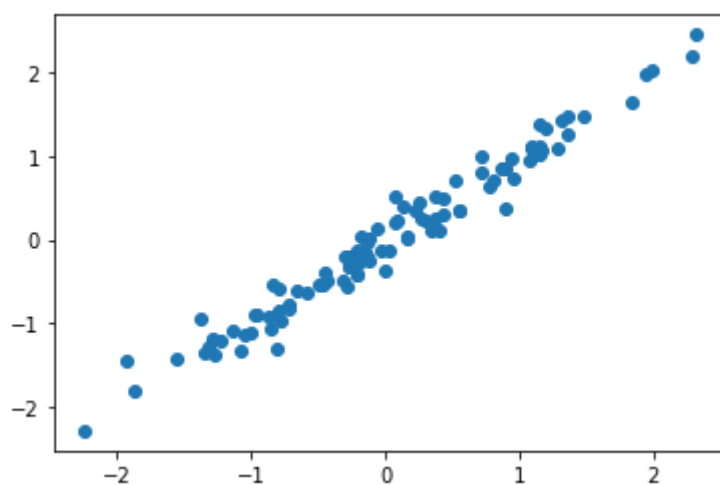
We are plotting because we are INSIDE the for loop.



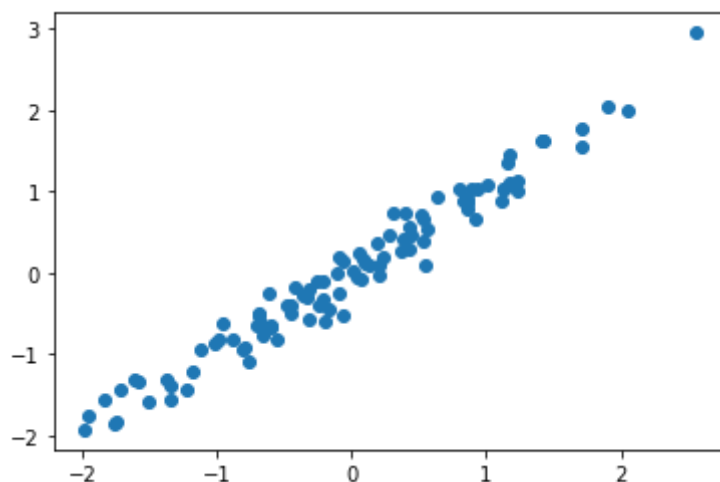
We are plotting because we are INSIDE the for loop.



We are plotting because we are INSIDE the for loop.



We are plotting because we are INSIDE the for loop.



We are plotting because we are INSIDE the for loop.

## 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.subplots](https://matplotlib.org/3.5.1/api/_as_gen/matplotlib.pyplot.subplots.html) ([https://matplotlib.org/3.5.1/api/\\_as\\_gen/matplotlib.pyplot.subplots.html](https://matplotlib.org/3.5.1/api/_as_gen/matplotlib.pyplot.subplots.html)) 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 around the axis (good plots only have two axis not 4!)
  - Add labels
  - Add titles
  - etc

```
In [5]: from numpy.random import uniform
mu1 = 1;
mu2 = 1;
var1 = 1;
var2 = 3;
rows, cols = 3, 3
fig, ax = plt.subplots(rows, cols, figsize = (10, 10))
for r in range (rows):
    for c in range (cols):
        cov = uniform(0.3, 0.9)
        cov_m = [[var1, -cov],
                  [-cov, var2]]
        data = multivariate_normal([mu1, mu2], cov_m, size=1000)
        ax[r, c].scatter(data[:,0], data[:,1])
```

