Assignment 1

1. Question:

Describe in one or two sentences the output of following program.

```
import torch

xs = torch.randn(30)
```

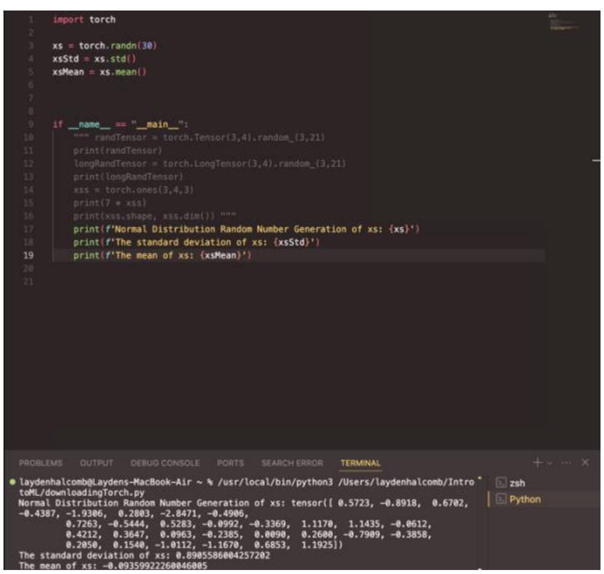
Answer:

a.) It is a tensor of thirty random numbers generated between 0 and 1.
b.) I did some research, and I found that the reason that the “randn” method changes the output each time the program is ran is because it generates pseudo random numbers based on Gaussian distribution.

2. Question:

Add lines to the program above that print the mean and standard deviation of the 30 numbers held in xs. Hint: PyTorch tensors implement methods mean() and std(). Include a screenshot of your program and its output in your document for this assignment.

Screenshot:



Answer:

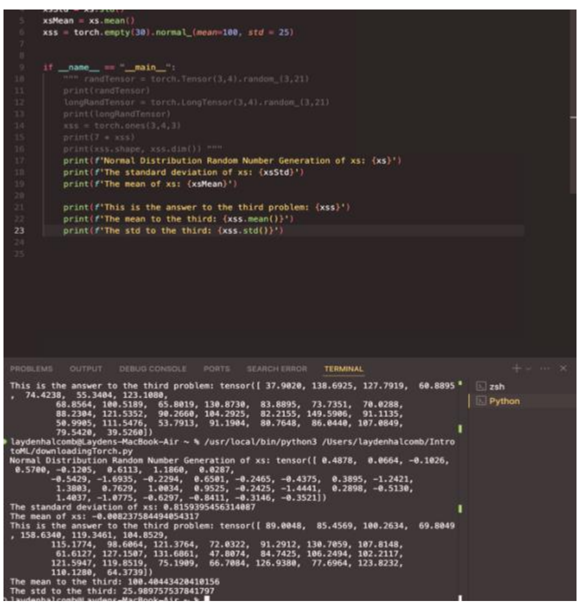
Is the mean exactly equal to zero? If it is not, why not.

No, it is not exactly equal to zero. If the mean were exactly zero, then all our numbers in the tensor would have been zero.

Is the standard deviation exactly equal to one? If it is not, why not

The standard deviation is not exactly equal to one. For the standard deviation to be equal to one, the data must be spread so evenly from the mean such that the square root of the variance is equal to one.

3. Modify your program so that it generates and prints 30 numbers from the normal distribution with mean 100 and standard deviation 25. (Note: we saw in class that there is more than one way to do this in PyTorch!) Include a screenshot of your program and its output in your document for this assignment.



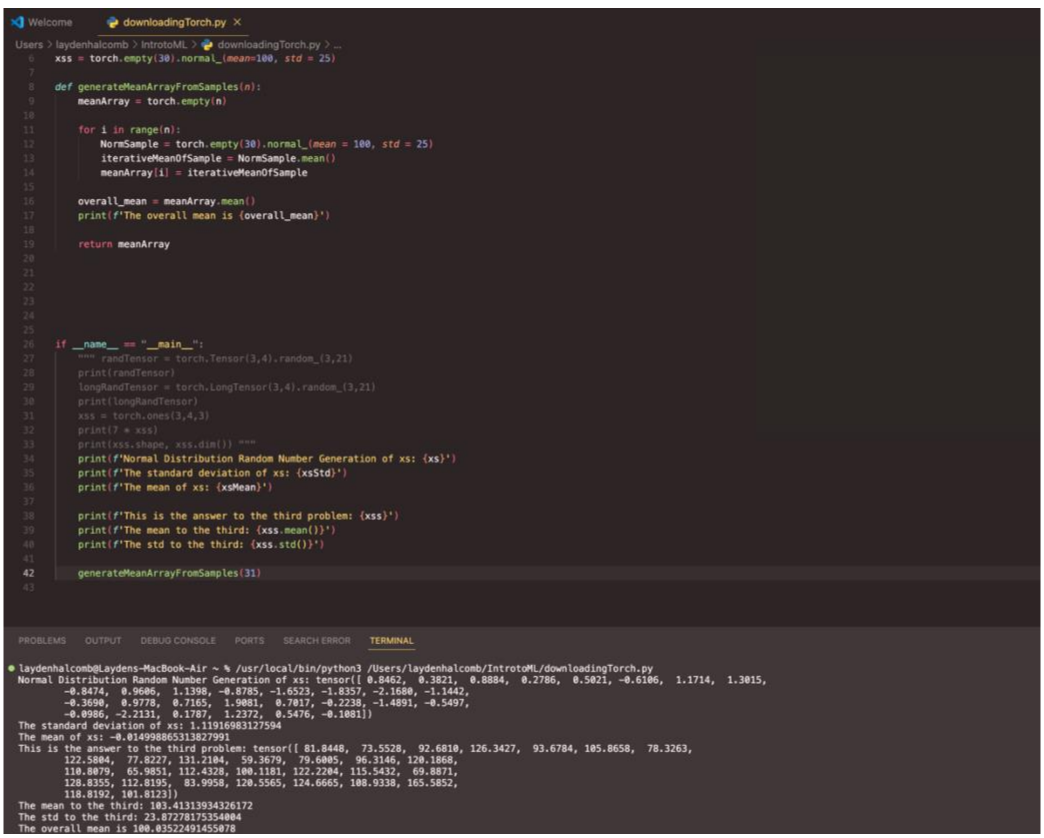
4. If you use your program in the last exercise to generate a single sample of size 30, it is highly unlikely that the mean of the sampled numbers will be exactly 100. In fact, you would routinely get numbers in the low 90s or around 110. However, suppose that you draw many samples of size 30; and that for each such sample you

compute and record its mean. What would you expect the mean of those means to be? Write a program do this and see if your prediction is correct? (Screenshot your code and its output and include it in your document).

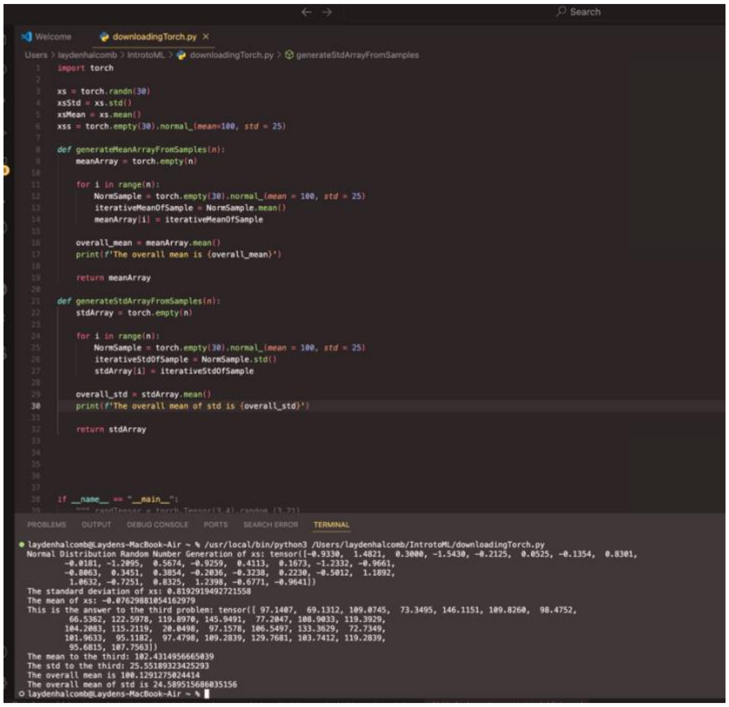
Prediction:
I predict that at some point, continuously taking the mean of these many samples would cause the mean to converge on a certain integer.

Answer:
My prediction ended up being wrong. I did a little research and there is something significant about a sample size of 30 in mathematical statistics. It has something to do with the normal distribution.

ScreenShot:



5. Question: Do the same as in the last exercise but for the standard deviation. Does the mean of the standard deviations over very many samples target the correct value? As always, include your code and its output.
Answer: It is something very similar, but I am not sure what is happening. I think the mean of the standard deviations are targeting close to the desired value.



6.

Question:

What happens if you sample from the uniform distribution on [0,1] instead of a normal distribution? What value does the mean of the means of samples of size 30 appear to target? What about the mean of the standard deviations of many samples of size 30? Modify your program above and include screenshots in your solution document.

Answer:

6a.) If you sample a uniform distribution on [0, 1] instead of a normal distribution, you get numbers in between 0 and 1 instead of spreading out amongst the normal distribution curve.

6b.) The value that the mean of the means appears to target around 0.5 with 30 samples of 30 sample sized uniform distributions between 0 and 1. For such a scale between 100 and 125 like previously shown, it is about 112.5.

6c.) The overall mean of the samples of standard deviations fell around 0.28 something for a distribution between 0 and 1. And for a distribution between 100 and 125 like mentioned in previous problems fell around 7.

ScreenShots:

Search

Welcome   downloadingTorch.py ×

Users > laydenhalcomb > IntrotoML > downloadingTorch.py > genUniformStdFromSamples

```python
def genUniformMeanArrayFromSamples(n):
    meanArray = torch.empty(n)

    for i in range(n):
        NormSample = torch.empty(30).uniform_(100,125)
        iterativeMeanOfSample = NormSample.mean()
        meanArray[i] = iterativeMeanOfSample
    overall_mean = meanArray.mean()

    print(f'The overall mean of the uniform means is {overall_mean}')

def genUniformStdFromSamples(n):
    stdArray = torch.empty(n)

    for i in range(n):
        NormSample = torch.empty(30).uniform_(100, 125)
        iterativeStdOfSample = NormSample.std()
        stdArray[i] = iterativeStdOfSample
    overall_mean = stdArray.mean()

    print(f'The overall mean of the uniform std is {overall_mean}')
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   PORTS   SEARCH ERROR   TERMINAL

```
        121.2009,  92.1132, 111.4801,  82.0968,  70.4425, 100.0953, 148.5762,
        129.8192, 132.8951,  91.2678, 114.7370,  84.8745, 100.1204,  98.1046,
         96.1898, 130.0782])
The mean to the third: 102.1694564819336
The std to the third: 18.99201202392578
tensor([7.5214e-01, 3.5509e-01, 1.3497e-01, 7.9467e-01, 4.4025e-01, 3.2445e-01,
        2.2169e-01, 7.5327e-01, 4.7540e-01, 3.9379e-01, 5.9724e-01, 4.7207e-01,
        5.4502e-01, 7.7266e-01, 5.6366e-01, 1.3664e-01, 9.7957e-01, 4.3727e-01,
        6.6330e-01, 2.4441e-01, 7.4531e-01, 1.9997e-04, 1.7168e-01, 9.5230e-01,
        3.6798e-01, 5.8667e-01, 2.0806e-01, 8.3502e-02, 1.4676e-01, 7.2931e-01])
The overall mean is 99.5029296875
The overall mean of std is 23.95519256591797
The overall mean of the uniform means is 112.29232788085938
The overall mean of the uniform std is 7.079341411590576
laydenhalcomb@Laydens-MacBook-Air ~ % /usr/local/bin/python3 /Users/laydenhalcomb/IntrotoML/downloadingTorch.py
Normal Distribution Random Number Generation of xs: tensor([-1.3584,  0.5039, -0.8776,  0.2618, -0.3361,  1.1828,  0.5024, -1.0090,
        -0.2336, -1.0152, -1.0249,  0.1562,  0.4161,  0.5470,  0.7455, -1.0176,
         0.6265, -0.0397, -0.7875,  0.3805, -0.8828, -1.5615, -0.2679, -1.9031,
        -0.3614,  0.3939,  1.9145,  0.9735,  1.8451,  1.5222])
The standard deviation of xs: 0.9933941960334778
The mean of xs: -0.023471934720873833
This is the answer to the third problem: tensor([124.1056, 103.3444,  97.8522, 118.3048,  70.7105,  95.8080, 111.0914,
        149.8903, 104.1642, 115.0855,  87.3565, 131.5374, 138.3309,  90.3758,
         59.6538, 106.3383,  99.2670, 105.9412,  83.8455, 100.8568, 101.0316,
         83.2329, 109.0162,  98.4115, 112.2144, 130.3457,  88.6708,  87.2395,
         86.5222, 120.9796])
The mean to the third: 103.71748352050781
The std to the third: 19.722824096679688
tensor([0.3403, 0.3075, 0.4045, 0.8899, 0.9087, 0.0455, 0.0690, 0.0951, 0.3524,
        0.6398, 0.7800, 0.8703, 0.5612, 0.3070, 0.4484, 0.1965, 0.7495, 0.9532,
        0.7764, 0.4808, 0.3378, 0.3079, 0.7771, 0.6528, 0.2527, 0.6558, 0.8789,
        0.9163, 0.1020, 0.6323])
The overall mean is 99.77957153320312
The overall mean of std is 24.864816665649414
The overall mean of the uniform means is 112.66967010498047
The overall mean of the uniform std is 7.2033772468566895
```