# assignment1final

April 13, 2023

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     from scipy.stats import norm
```

## 1 1a

```
[ ]: a = np.array([1,1,1])
     b = np.array([2,2,2])
     c = np.array([3,3,3])
     def three_array(a,b,c):
         return a+b-c

     print(three_array(a,b,c))
```

```
[0 0 0]
```

## 2 1b

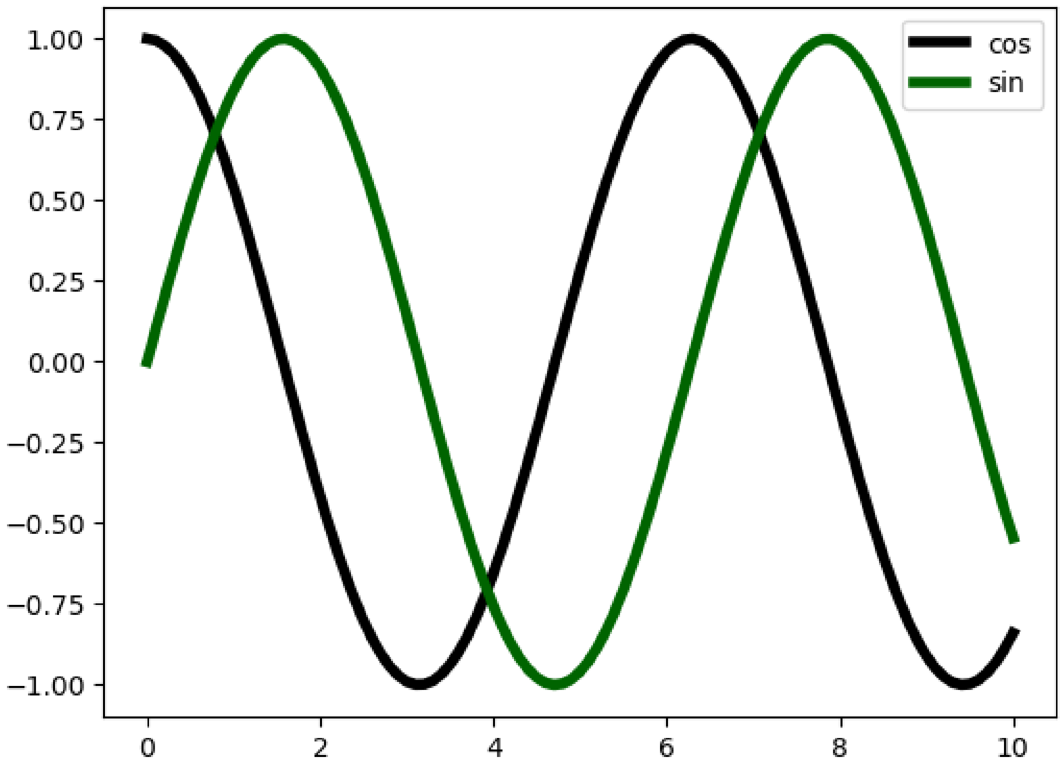
```
[ ]: def add_array(a,b):
         # check if dimension is correct
         if a.shape != b.shape:
             return "Dimension Error"
         else:
             return a+b

     array_1 = np.array([3,5,7])
     array_2 = np.array([1,1,1])
     array_3 = np.array([2,2,2,2])

     # testing cases
     print(add_array(array_1,array_2))
     print(add_array(array_1,array_3))
```

```
[4 6 8]
Dimension Error
```

## 3 1c

```
[ ]: #fig, ax = plt.subplots(figsize=(10,8), nrows=1, ncols = 2)
     #time = np.linspace(0,10,101)
     #cos_y = np.cos(time)
     #sin_y = np.sin(time)
     #ax[0,0].plot(time, sin_y)
     #ax[1,1].plot(time, cos_y)
```

```
[ ]: time = np.linspace(0,10,101)
     # create sin and cos data
     cos_y = np.cos(time)
     sin_y = np.sin(time)
     # simple plotting
     plt.plot(time, cos_y, color ="black", lw=4, label = "cos")
     plt.plot(time,sin_y, color = "darkgreen", lw=4, label="sin")
     plt.legend()
     plt.show()
```



## 4 1d

```
[ ]: A = np.array([[1,0,0],[0,1,0],[0,0,1]])
     def inver(A):
         # find the inverse
         try:
             A_inv = np.linalg.inv(A)
             return A_inv
         # catch non invertible exception
         except:
             det = np.linalg.det(A)
             print("Error: this matrix is not invertible with a determinant of " +␣
      ↪str(det))

     output = inver(A)
     print(np.dot(A, output))
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

## 5 2a

```
[ ]: def generate(a,b,N):
         sample_rand =  np.random.rand(1,N) * (b-a) + a
         return sample_rand

     a = 500
     b = 850
     N = 100
     sample = generate(a,b,N)
     print("Size is: " + str(sample.size))
     print(sample)
```

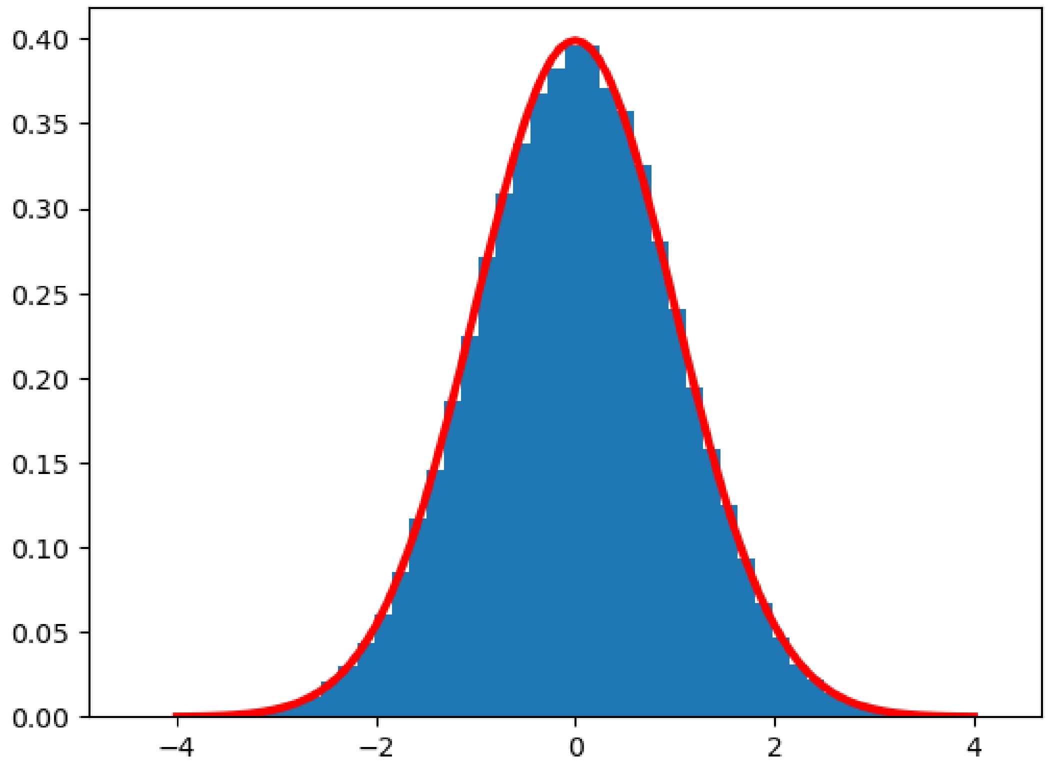
```
Size is: 100
[[804.00007385 519.32660988 543.21836841 636.02437206 656.74893761
  549.2322158  751.65853457 539.28531954 562.9153935  587.29184186
  704.58661805 823.00721452 563.6478545  622.1245086  685.82481784
  679.08934006 790.58418414 650.67912179 626.23726773 745.41811395
  767.4447916  553.28520303 582.57697716 534.16448509 764.18796665
  776.04158278 709.75459825 617.56028544 810.36868832 628.74736498
  737.41256854 506.71592854 526.69956955 814.48675417 594.45856284
  751.86803583 630.74123717 701.58895959 590.13733338 611.86338373
  812.54932038 537.33860327 808.97876555 666.89091405 626.54072514
  809.33576705 752.32412783 599.8778402  776.00375111 760.46383805
  621.2535095  649.23991705 523.12361999 761.54241491 501.50299721
  672.98465857 640.02124831 840.51926884 543.11273961 585.7073307
```

```
 819.43029756 607.11179955 843.78518374 826.88913269 539.40752791
 730.31191848 697.92820868 826.71211281 747.28162577 506.01836242
 539.5521879  571.2108153  646.40309035 631.66146432 797.36043908
 723.41933662 627.54856771 578.62065461 793.39168053 598.43781141
 785.4022629  719.07519952 742.67841916 591.77318554 548.84191805
 782.18706825 648.43560996 646.16641449 576.80965769 782.20916534
 504.53525215 600.85520829 661.53138066 643.03200227 527.0563693
 528.40212261 541.07932226 794.36173836 553.02280768 506.48294788]]
```

## 6 2b

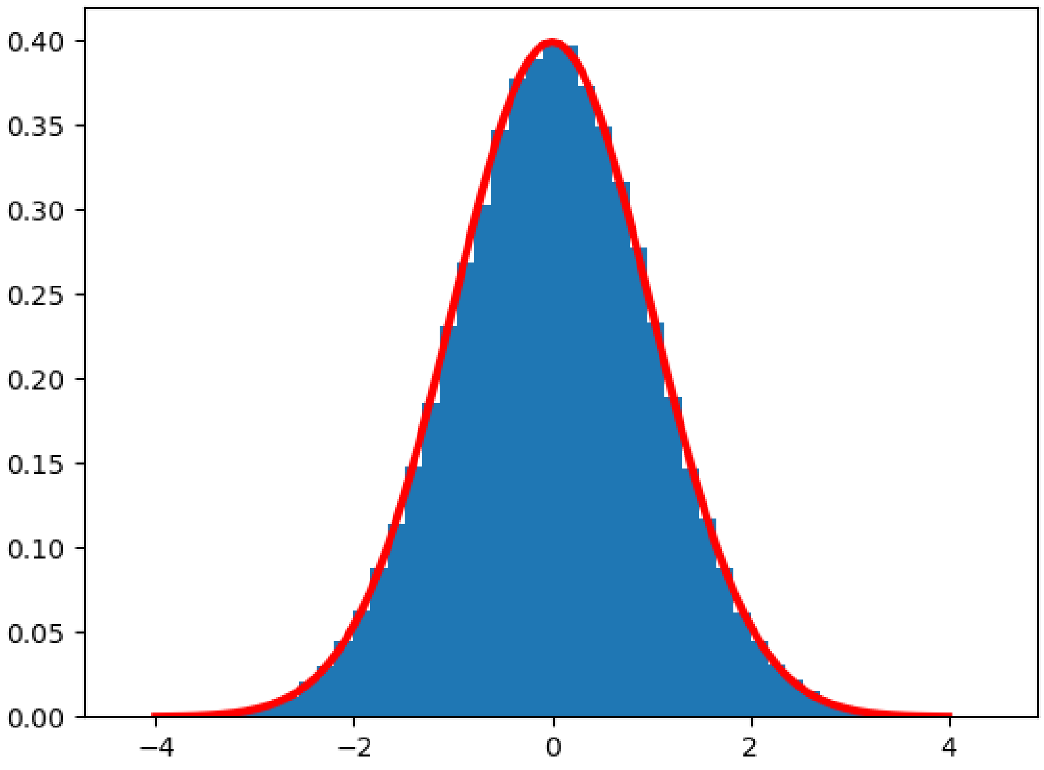
There are two ways to draw the normal distribution curve. The first way is to use it from the definition.

```
[ ]: data = np.random.randn(100000,1)
     x = np.linspace(-4,4,101)
     gaus = 1/(np.sqrt(2*np.pi))*np.e**(-1/2*(x)**2) # directly from gaussian␣
      ↪formulation
     plt.hist(data,bins=50, density=True) # histogram
     plt.plot(x, gaus, lw = 3, color = "red") # plotting actual function
     plt.show()
```



The second method is to use the scipy method:

```
[ ]: data = np.random.randn(100000,1)
     x = np.linspace(-4,4,101)
     plt.hist(data,bins=50, density=True)
     plt.plot(x, norm.pdf(x, 0, 1), lw = 3, color = "red") # plot x versus the␣
      ↪normal distribution sample from scipy
     plt.show()
```



## 7 2c

To calculate the expectation value, we evaluate the following integral

$$\langle x \rangle = \int_0^\infty xae^{-ax}dx = \frac{1}{a}$$

Then, we want to calculate the variance of the pdf. The variance is given by the following formula:

$$var = \langle x^2 \rangle - \langle x \rangle^2$$

This expectation of mean squared is given by the following integral:

$$\langle x^2 \rangle = \int_0^\infty x^2ae^{-ax}dx = \frac{2}{a^2}$$

Combining the result:

$$var = \langle x^2 \rangle - \langle x \rangle^2 = \frac{2}{a^2} - (\frac{1}{a})^2 = \frac{1}{a^2}$$

## 8 2d

i): we can only get 12 if both of the dice are 6. Therefore, the probability is given by

$$\frac{1}{6} * \frac{1}{6} = \frac{1}{36}$$

ii): the sum of 10 can be achieved in three ways:

$$5 + 5 = 10$$

$$6 + 4 = 10$$

$$4 + 6 = 10$$

each configuration has a probability of 1 over 36, so the final probability is the sum:

$$\frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{1}{12}$$

iii): the sum of 6 can be achieved in five ways:

$$1 + 5 = 6$$

$$2 + 4 = 6$$

$$3 + 3 = 6$$

$$4 + 2 = 6$$

$$5 + 1 = 6$$

each configuration has a probability of 1 over 36, so the final probability is:

$$\frac{1}{36} * 5 = \frac{5}{36}$$

[ ]:
```
# testing for above

values = [1,2,3,4,5,6]
probaility = [1/6,1/6,1/6,1/6,1/6,1/6]
test1 = np.random.choice(values, 100000, p=probaility)
test2 = np.random.choice(values, 100000, p=probaility)
test = test1 + test2
count_12 = np.count_nonzero(test==12)
count_10 = np.count_nonzero(test==10)
count_6 = np.count_nonzero(test==6)
print(count_12/100000)
print(1/36)
print(count_10/100000)
print(3/36)
print(count_6/100000)
print(5/36)
```

```
0.02744
0.027777777777777776
0.08388
0.08333333333333333
0.13944
0.1388888888888889
```