## Imports (do NOT change/add)

Toggle code

## Constants (do NOT change/add)

## 1. Basic Stats

- Implement the mean and covariance functions

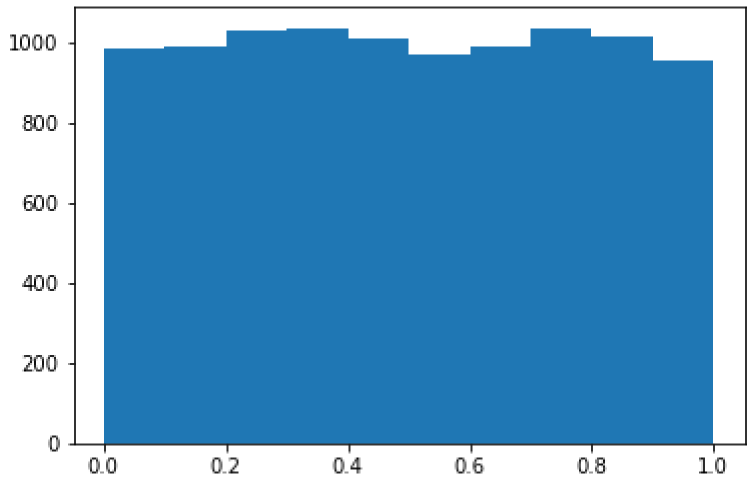
```
mean([])=0.00000, expected=0.00000
mean([1])=1.00000, expected=1.00000
mean([1, 2, 3])=2.00000, expected=2.00000
mean([1, 2, 3, 4])=2.50000, expected=2.50000
mean([0.607, 0.094, 0.314, 0.491, 0.664, 0.282, 0.914, 0.517, 0.731, 0.
804])=0.54180, expected=0.54180
```

```
svariance([1, 2, 3])=1.00000, expected=1.00000
svariance([1, 2, 3, 4])=1.66667, expected=1.66667
svariance([600, 470, 170, 430, 300])=27130.00000, expected=27130.00000
scovariance([2.1, 2.5, 3.6, 4.0], [8, 10, 12, 14])=2.26667/2.26667, exp
ected=2.66667
```

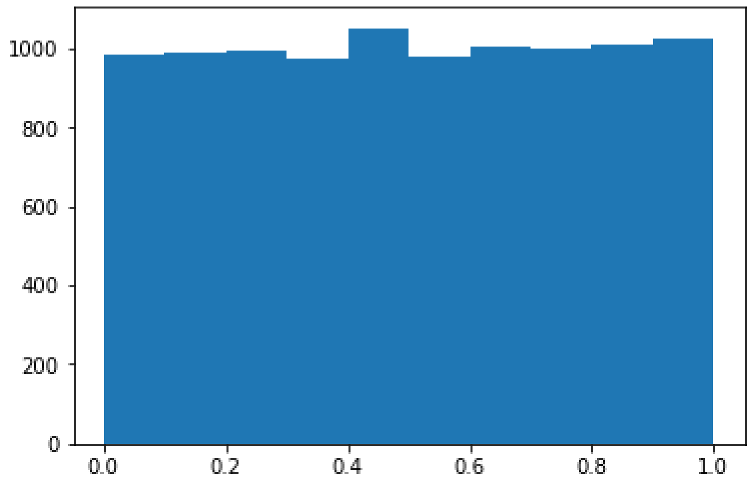
## 2. Generate Uniform Random Samples

- Generates three lists of N numbers, where each value is i.i.d. in [0,1]
- Respond to the question below

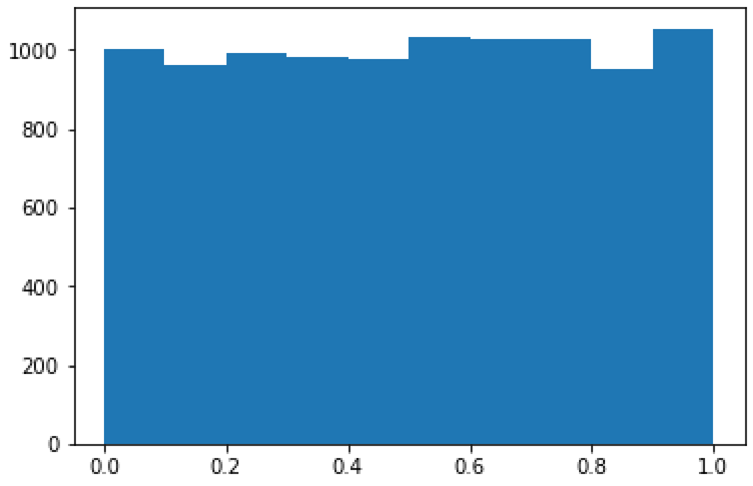
```
sample1 mean: 0.49878
```



```
sample2 mean: 0.50244
```



```
sample3 mean: 0.50344
```



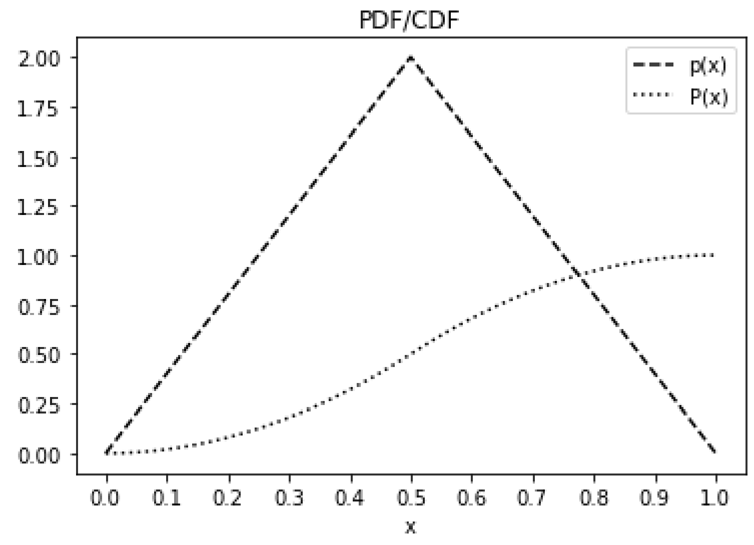
## Q: Are these "good" samples? How do you know?

TODO: respond here

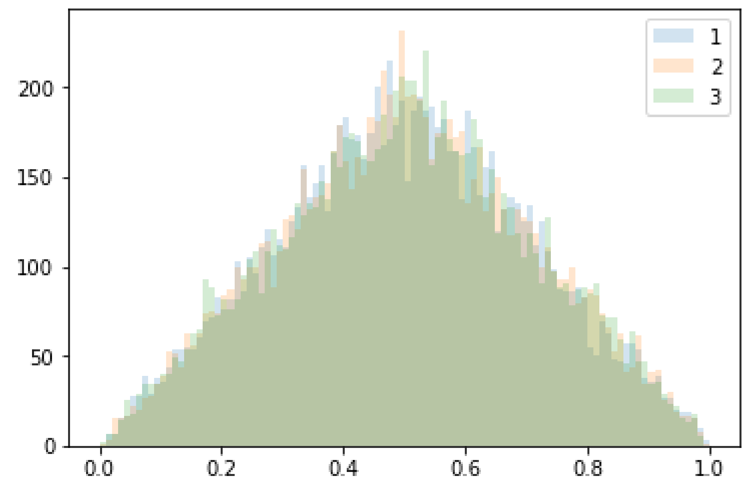
## 3. Inverse Transform Sampling

- Implement the CDF for a given PDF
- Implement the corresponding inverse-CDF

```
<matplotlib.legend.Legend at 0x11a0933c8>
```



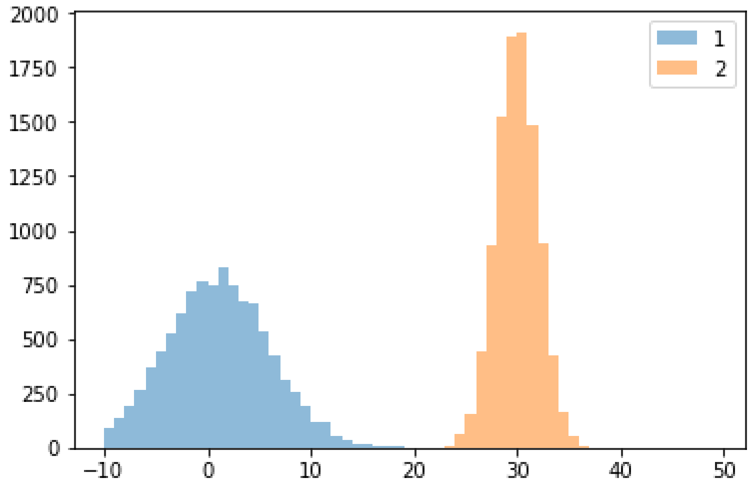
```
<matplotlib.legend.Legend at 0x119fa5a20>
```



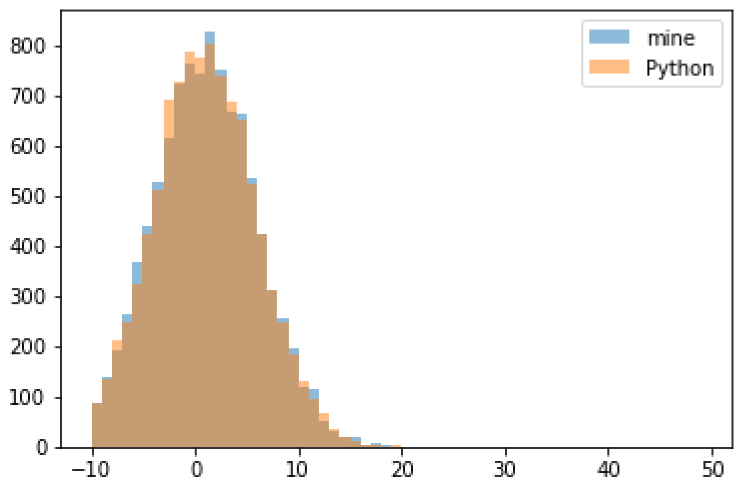
## 4. Generate Normally Distributed Numbers via Box–Muller

- Implement the Box-Muller transform
- Implement function to shift/scale normally distributed values

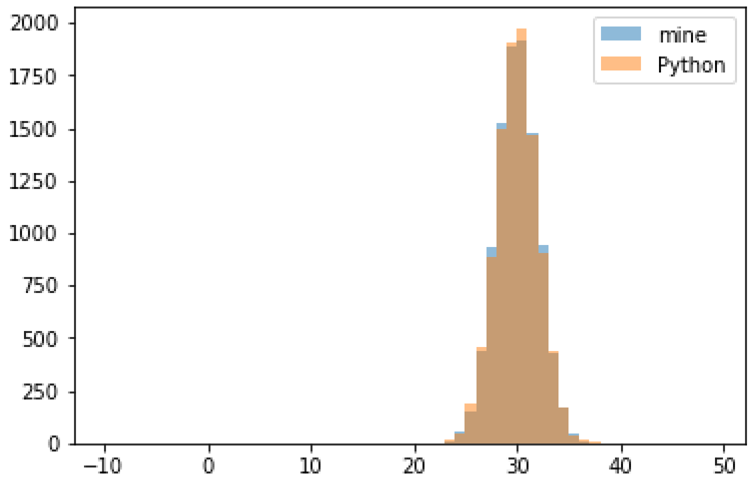
```
Gaussian 1: mean=0.98692, variance=25.08427
Gaussian 2: mean=29.99661, variance=3.98527
```



```
Python Gaussian1: mean=0.9626797664670734, variance=24.81399842067043
```



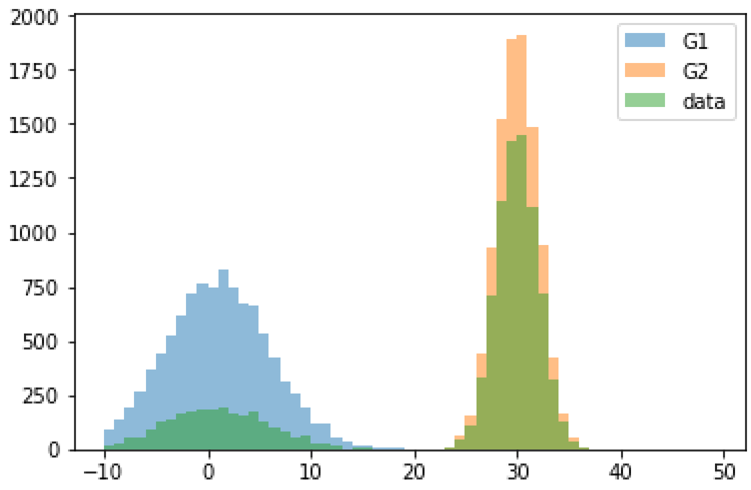
```
Python Gaussian2: mean=29.981038509019378, variance=4.004938950448199
```



## 5. Sampling a 1D Gaussian Mixture Model

- Implement the function to sample a 1D 2-Gaussian mixture model
- Respond to the question below

```
<matplotlib.legend.Legend at 0x11aefc780>
```



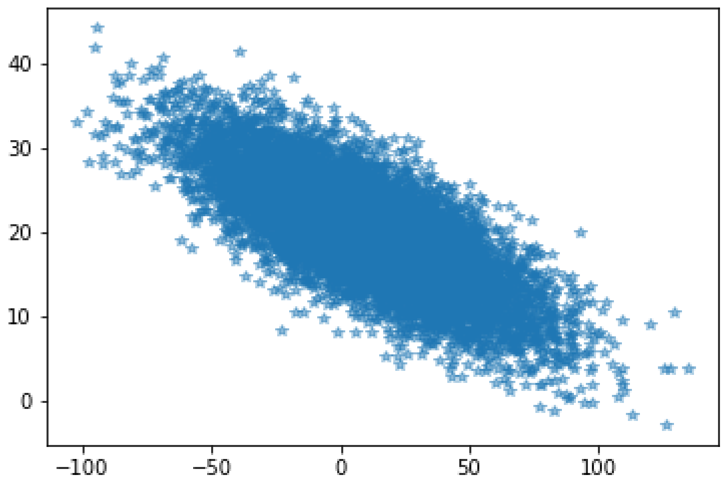
### Q: As you change the split variable, what happens to the resulting distribution? What are extreme values and what happens at these extremes?

TODO: Respond here

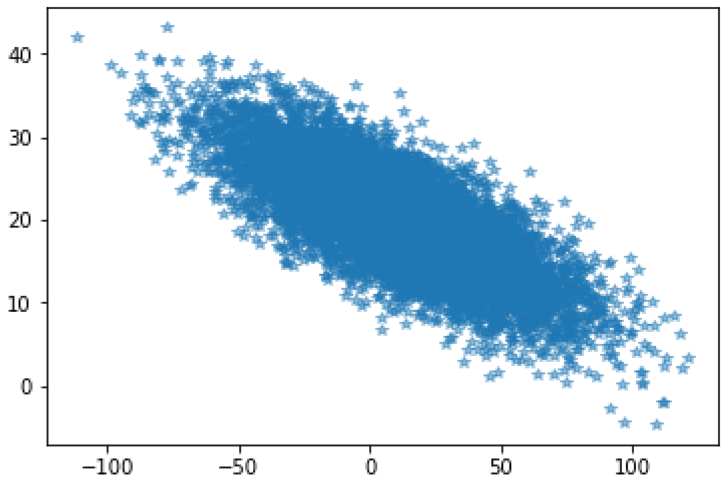
## 6. Generate a 2D Gaussian Distribution

- Implement function to generate a 2D Gaussian given two sets of uniformly random values in [0,1]

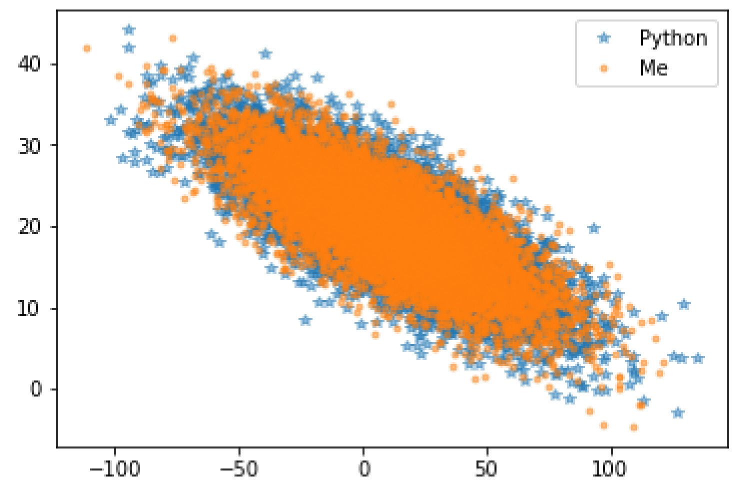
```
Python Gaussian mean: (9.547020658806519, 20.01003776715308)
Python Gaussian cov: x=1004.9496474113697, y=36.94117819446662 xy=-145.
99243102125416
```



```
My Gaussian mean: (9.917252684970098, 20.005288321150953)
My Gaussian cov: x=1003.3707530613519, y=36.329206484529024 xy=-145.580
93473979082
```



```
<matplotlib.legend.Legend at 0x11ab2a780>
```



## Fin