Central Limit Theorem

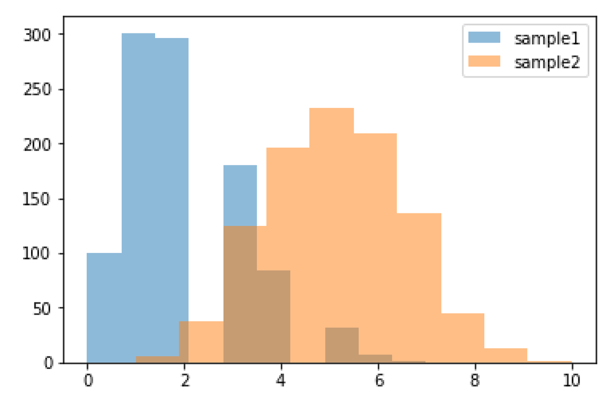
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy
%matplotlib inline
```

In [35]:
```
pop1 = np.random.binomial(10, 0.2, 10000)
pop2 = np.random.binomial(10, 0.5, 10000)

sample1 = np.random.choice(pop1, 100, replace = True)
sample2 = np.random.choice(pop2, 100, replace = True)

from scipy.stats import ttest_ind
print(ttest_ind(sample2, sample1, equal_var = False))
```

```
Ttest_indResult(statistic=12.954089689032953, pvalue=4.219245516493207e-28)
```

In [20]:
```
print('Sample 1 mean is', sample1.mean())
print('Sample 1 standard deviation is', sample1.std())
print('Sample 2 mean is', sample2.mean())
print('Sample 2 standard deviation is', sample2.std())
print('Difference in means is', sample2.mean() - sample1.mean())

plt.hist(sample1, alpha = 0.5, label = 'sample1')
plt.hist(sample2, alpha = 0.5, label = 'sample2')
plt.legend(loc = 'upper right')
plt.show()
```

```
Sample 1 mean is 1.973
Sample 1 standard deviation is 1.2776036161501736
Sample 2 mean is 5.091
Sample 2 standard deviation is 1.585155828302063
Difference in means is 3.1180000000000003
```



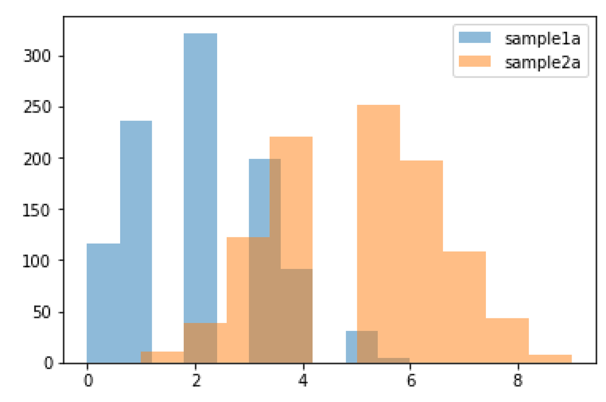
Question 1

Increase the size of your samples from 100 to 1000, then calculate the means and standard deviations for your new samples and create histograms for each. Repeat this again, decreasing the size of your samples to 20. What values change, and what remain the same?

I believe when we increase the sample size the means and standard deviations should be closer to the expected difference and when we decrease the sample size to 20, they will be farther apart.

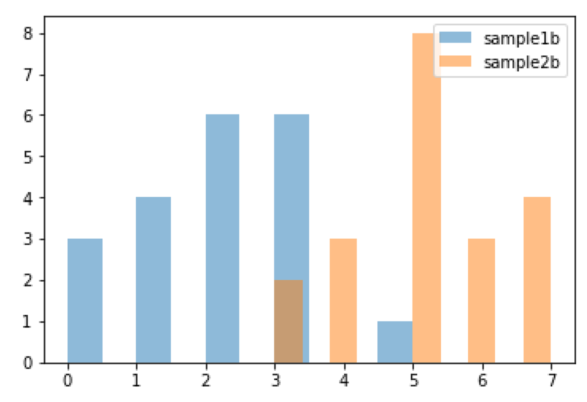
In [19]:
```
#Increasing sample size to 1000
sample1a = np.random.choice(pop1, 1000, replace = True)
sample2a = np.random.choice(pop2, 1000, replace = True)

print('Sample 1a mean is', sample1a.mean())
print('Sample 1a standard deviation is', sample1a.std())
print('Sample 2a mean is', sample2a.mean())
print('Sample 2a standard deviation is', sample2a.std())
print('Difference in means is', sample2a.mean() - sample1a.mean())

plt.hist(sample1a, alpha = 0.5, label = 'sample1a')
plt.hist(sample2a, alpha = 0.5, label = 'sample2a')
plt.legend(loc = 'upper right')
plt.show()
```

```
Sample 1a mean is 2.023
Sample 1a standard deviation is 1.274547370637906
Sample 2a mean is 4.951
Sample 2a standard deviation is 1.5429189868557585
Difference in means is 2.9279999999999995
```



In [22]:
```
#Decreasing sample size to 20
sample1b = np.random.choice(pop1, 20, replace = True)
sample2b = np.random.choice(pop2, 20, replace = True)

print('Sample 1b mean is', sample1b.mean())
print('Sample 1b standard deviation is', sample1b.std())
print('Sample 2b mean is', sample2b.mean())
print('Sample 2b standard deviation is', sample2b.std())
print('Difference in means is', sample2b.mean() - sample1b.mean())

plt.hist(sample1b, alpha = 0.5, label = 'sample1b')
plt.hist(sample2b, alpha = 0.5, label = 'sample2b')
plt.legend(loc = 'upper right')
plt.show()
```

```
Sample 1b mean is 1.95
Sample 1b standard deviation is 1.2439855304624727
Sample 2b mean is 5.2
Sample 2b standard deviation is 1.2083045973594573
Difference in means is 3.25
```



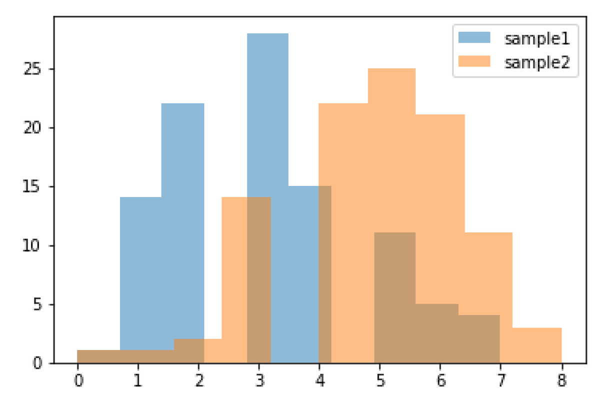
Question 2

Change the probability value (p in the NumPy documentation) for pop1 to 0.3, then take new samples and compute the t-statistic and p-value. Then change the probability value p for group 1 to 0.4, and do it again. What changes, and why?

I believe that the t-value will decrease as the distributions get closer together and p-value will stay close to the same.

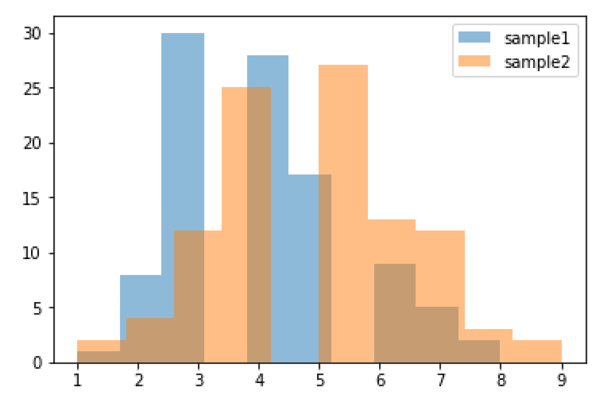
In [43]:
```
#Change probability value to 0.3 for pop1
pop1a = np.random.binomial(10, 0.3, 10000)
pop2a = np.random.binomial(10, 0.5, 10000)

sample1 = np.random.choice(pop1a, 100, replace = True)
sample2 = np.random.choice(pop2a, 100, replace = True)

from scipy.stats import ttest_ind
print(ttest_ind(sample2, sample1, equal_var = False))

print('Sample 1b mean is', sample1.mean())
print('Sample 1b standard deviation is', sample1.std())
print('Sample 2b mean is', sample2.mean())
print('Sample 2b standard deviation is', sample2.std())
print('Difference in means is', sample2.mean() - sample1.mean())

plt.hist(sample1, alpha = 0.5, label = 'sample1')
plt.hist(sample2, alpha = 0.5, label = 'sample2')
plt.legend(loc = 'upper right')
plt.show()
```

```
Ttest_indResult(statistic=7.810849750716722, pvalue=3.280845187158905e-13)
Sample 1b mean is 3.15
Sample 1b standard deviation is 1.5898113095584647
Sample 2b mean is 4.87
Sample 2b standard deviation is 1.507680337472105
Difference in means is 1.7200000000000002
```



In [42]:
```
#Change probability value to 0.4 for pop1
pop1b = np.random.binomial(10, 0.4, 10000)
pop2b = np.random.binomial(10, 0.5, 10000)

sample1 = np.random.choice(pop1b, 100, replace = True)
sample2 = np.random.choice(pop2b, 100, replace = True)

from scipy.stats import ttest_ind
print(ttest_ind(sample2, sample1, equal_var = False))

print('Sample 1b mean is', sample1.mean())
print('Sample 1b standard deviation is', sample1.std())
print('Sample 2b mean is', sample2.mean())
print('Sample 2b standard deviation is', sample2.std())
print('Difference in means is', sample2.mean() - sample1.mean())

plt.hist(sample1, alpha = 0.5, label = 'sample1')
plt.hist(sample2, alpha = 0.5, label = 'sample2')
plt.legend(loc = 'upper right')
plt.show()
```

```
Ttest_indResult(statistic=3.529825481585233, pvalue=0.000518849171598611)
Sample 1b mean is 4.09
Sample 1b standard deviation is 1.4148851543499918
Sample 2b mean is 4.85
Sample 2b standard deviation is 1.608570794214541
Difference in means is 0.7599999999999998
```



Question 3

Change the distribution of your populations from binomial to a distribution of your choice. Do the sample mean values still accurately represent the population values?

The sample means should accurately represent the population values as long as the sample sizes are large enough. A more skewed distribution would require greater sample sizes, but should eventually approach the true population mean.

In [41]:
```
pop1_pos = np.random.poisson(2, 10000)
pop2_pos = np.random.poisson(5, 10000)

sample_pos1 = np.random.choice(pop1_pos, 100, replace = True)
sample_pos2 = np.random.choice(pop2_pos, 100, replace = True)

from scipy.stats import ttest_ind
print(ttest_ind(sample_pos2, sample_pos1, equal_var = False))
```

```
Ttest_indResult(statistic=10.725120547942884, pvalue=5.033923124871127e-21)
```

In [40]:
```
print('Sample 1 mean is', sample_pos1.mean())
print('Sample 1 standard deviation is', sample_pos1.std())
print('Sample 2 mean is', sample_pos2.mean())
print('Sample 2 standard deviation is', sample_pos2.std())
print('Difference in means is', sample_pos2.mean() - sample_pos1.mean())

plt.hist(sample_pos1, alpha = 0.5, label = 'sample1')
plt.hist(sample_pos2, alpha = 0.5, label = 'sample2')
plt.legend(loc = 'upper right')
plt.show()
```

```
Sample 1 mean is 2.0
Sample 1 standard deviation is 1.4422205101855958
Sample 2 mean is 5.36
Sample 2 standard deviation is 1.9975985582694036
Difference in means is 3.3600000000000003
```