BETA DISTRIBUTION

CODE FOR VARYING BETA CONSTANT ALPHA

```
def factorial(n):
    if n==0:
        return 1
    return n*factorial(n-1)

x=np.arange(0,1, 0.001)
a=1
b=1
for i in range(4):
    const = factorial(a+b-1)/(factorial(a-1)*factorial(b-1))
    fx = const * ( x**(a-1) * (1-x)**(b-1) )
    plt.plot(x,fx,label='a=1 b= {}'.format(b))
    b=b+1
plt.ylabel('beta distribution')
plt.legend()
```

CODE FOR VARYING ALPHA CONSTANT BETA

```
def factorial(n):
    if n==0:
        return 1
    return n*factorial(n-1)

x=np.arange(0,1, 0.001)
a=1
b=1
for i in range(4):
    const = factorial(a+b-1)/(factorial(a-1)*factorial(b-1))
    fx = const * ( x**(a-1) * (1-x)**(b-1) )
    plt.plot(x,fx,label='a={} b=1'.format(a))
    a=a+1
plt.ylabel('beta distribution')
plt.legend()
```

CODE FOR MEAN AND VARIANCE VS ALPHA

```
a=np.arange(0.1,10, 0.01)
b=1
mean=a/(a+b);
var=(a*b)/((a+b)*(a+b)*(a+b+1));
plt.plot(a,mean,label='beta=1 ,mean');
plt.plot(a,var,label='beta=1 , varinace');

plt.legend()
```

CODE FOR MEAN AND VARIANCE VS BETA

```
b=np.arange(0.1,10, 0.01)
a=1
mean=a/(a+b);
var=(a*b)/((a+b)*(a+b)*(a+b+1));
plt.plot(b,mean,label='alpha=1 ,mean');
plt.plot(b,var,label='alpha=1 , varinace');

plt.legend()
```

CENTRAL LIMIT THEOREM VERIFICATION

```
a=1
b=1
n=40
ns=10
for i in range(4):

    samplemean=[]
    for j in range(ns):
        sum=0

        x = np.random.beta(1,1,n)
        for k in x:
            sum=sum+k
        samplemean.append(sum/n)
    fig, ax = plt.subplots(figsize =(10, 7))
    ax.hist(samplemean, bins ='auto')

    plt.title("NUMBER OF SAMPLES ={}".format(ns))
    plt.show()
    ns=ns*10
```