PRACTICAL IMPLEMENTATION OF STATISTICS

Measures of Central Tendency

1. Mean
2. Meadian
3. Mode

In [7]:
```
ages = [23,24,32,45,12,43,67,45,32,56,32,120]
```

In [8]:
```
import numpy as np
```

In [9]:
```
print(np.mean(ages))
print(np.median(ages))
```

```
44.25
37.5
```

In [10]:
```
import statistics
print(statistics.mean(ages))
print(statistics.median(ages))
```

```
44.25
37.5
```

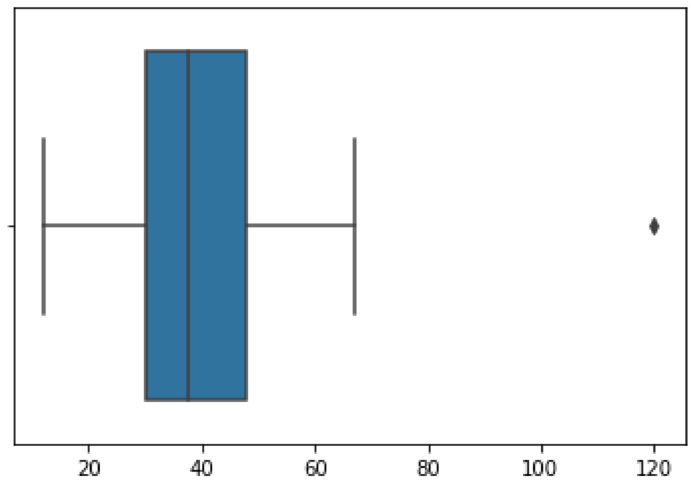
In [11]:
```
statistics.mode(ages)
```

Out[11]:
```
32
```

In [12]:
```
import seaborn as sns
sns.boxplot(ages)
```

```
C:\Users\Sonali Thakur\Documents\Python Scripts\lib\site-packages\seaborn\_deco
rators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. F
rom version 0.12, the only valid positional argument will be `data`, and passin
g other arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```

Out[12]: <AxesSubplot:>



5 NUMBER SUMMARY

In [13]:
```
q1,q3=np.percentile(ages,[25,75])
```

In [16]:
```
q1,q3
```

Out[16]: (30.0, 47.75)

In [17]:
```
# to check outlier(Lower Fence,Higher Fence)

IQR=q3-q1
Lower_fence = q1 - 1.5*(IQR)
Higher_fence = q3 + 1.5*(IQR)
print(Lower_fence,Higher_fence)
```

```
3.375 74.375
```

In [18]:
```
# the values which are outside the range of (Lower_fence,Higher_fence) are consic
```

MEASURES OF DISPERSION

1. VARIANCE
2. STANDARD DEVIATION

In [19]:
```
statistics.variance(ages)   #statistics.variance(ages) gives the variance of samp
```

Out[19]:
```
795.2954545454545
```

In [31]:
```
statistics.pvariance(ages)   #population variance
```

Out[31]:
```
729.0208333333334
```

In [33]:
```
import math
math.sqrt(statistics.pvariance(ages))  #standard deviation
```

Out[33]:
```
27.000385799712813
```

In [20]:
```
np.var(ages,axis=0)     #np.var gives the variance of population data in which th
```

Out[20]:
```
729.0208333333334
```

In [27]:
```
## Population Variance**
def variance(data):
    n=len(ages)
    ## mean of the data
    mean = sum(data)/n
    ##variance of the data
    deviation = [(x-mean)**2 for x in data]
    variance = sum(deviation)/n
    return variance
print(variance(ages))
```

```
729.0208333333334
```

In [28]:
```
## Sample Variance**
def variance(data):
    n=len(ages)
    ## mean of the data
    mean = sum(data)/n
    ##variance of the data
    deviation = [(x-mean)**2 for x in data]
    variance = sum(deviation)/n-1
    return variance
print(variance(ages))
```

```
728.0208333333334
```

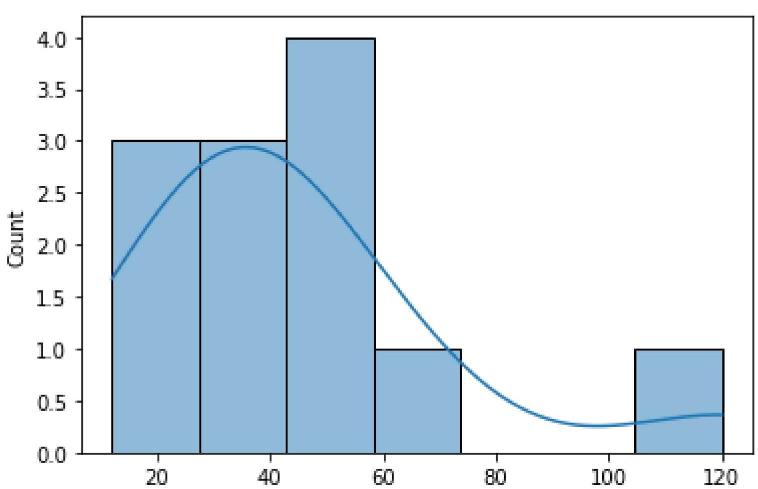
In [30]:
```
## Sample Variance** and Population Variance by giving degree of freedom

def variance(data,dof=0):
    n=len(ages)
    ## mean of the data
    mean = sum(data)/n
    ##variance of the data
    deviation = [(x-mean)**2 for x in data]
    variance = sum(deviation)/(n-dof)
    return variance
print(variance(ages))


def variance(data,dof=1):
    n=len(ages)
    ## mean of the data
    mean = sum(data)/n
    ##variance of the data
    deviation = [(x-mean)**2 for x in data]
    variance = sum(deviation)/(n-dof)
    return variance
print(variance(ages))
```

```
729.0208333333334
795.2954545454545
```

HISTOGRAMS AND PDF

In [36]:
```
import seaborn as sns
sns.histplot(ages,kde=True)  ## kde is Kernel Density Estimator which is used to
                             ## probability density function
```

Out[36]: <AxesSubplot:ylabel='Count'>



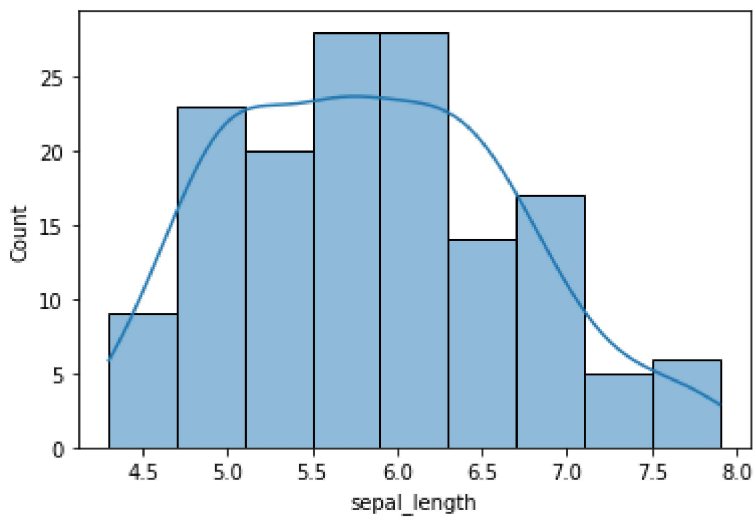
In [38]:
```
df=sns.load_dataset('iris')
df.head()
```

Out[38]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

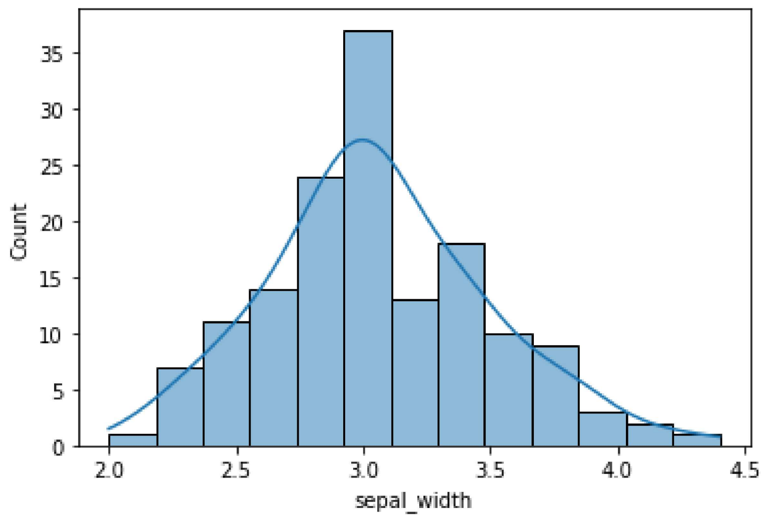
In [40]:
```
sns.histplot(df['sepal_length'],kde=True)
```

Out[40]:
```
<AxesSubplot:xlabel='sepal_length', ylabel='Count'>
```



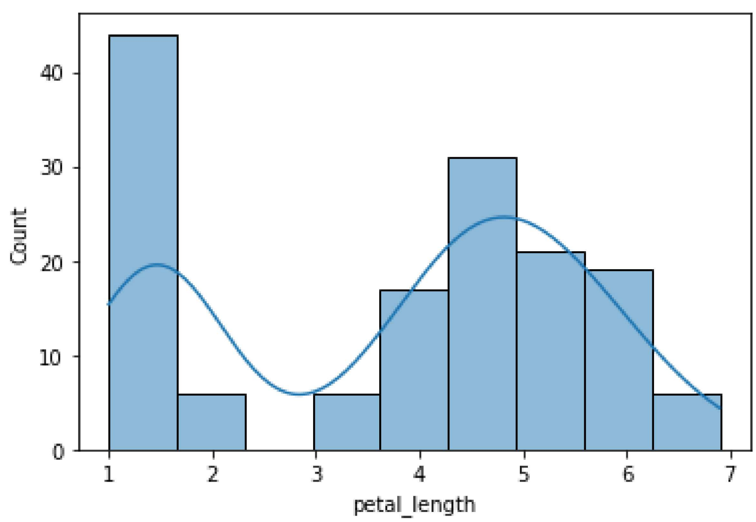
In [41]:
```
sns.histplot(df['sepal_width'],kde=True)
```

Out[41]: <AxesSubplot:xlabel='sepal_width', ylabel='Count'>



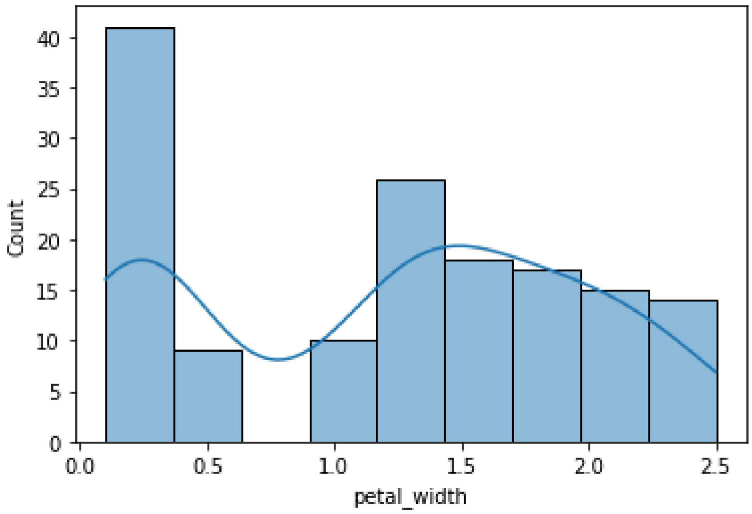
In [42]:
```
sns.histplot(df['petal_length'],kde=True)
```

Out[42]: <AxesSubplot:xlabel='petal_length', ylabel='Count'>



In [43]:
```
sns.histplot(df['petal_width'],kde=True)
```

Out[43]:
```
<AxesSubplot:xlabel='petal_width', ylabel='Count'>
```



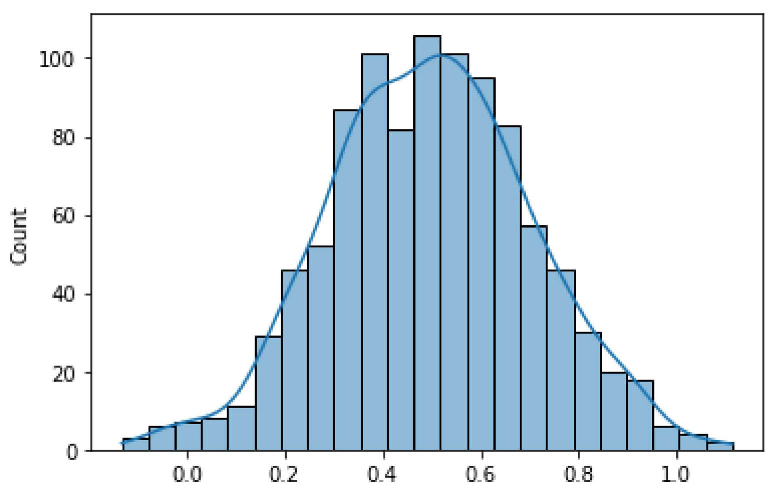
In [47]:
```
## Create a normal distributed dataset

s= np.random.normal(0.5,0.2,1000)  ## 0.5 mean , 0.2 standard deviation , 1000 da
```

In [48]:
```
s
```

Out[48]:
```
array([ 0.46885393,  0.53755175,  0.56345887,  0.41866171,  0.33502577,
        0.46821331,  0.71896254,  0.66810085, -0.04176154,  0.51919745,
        0.55863061,  0.29416849,  0.27172302,  0.86423213,  0.58134788,
        0.61993206,  0.57360757,  0.49643549,  0.77985264,  0.59125251,
        0.28007964,  0.79996458,  0.90346038,  0.57371926,  0.40871836,
        0.56387813,  0.76805852,  0.34656805,  0.13995148,  0.38682506,
        0.69645144,  0.65667529,  0.65280653,  0.7133452 ,  0.34551359,
        0.35679306,  0.31404002,  0.14319628,  0.52273395,  0.65184255,
        0.39547089,  0.37970984,  0.63267257,  0.59539068,  0.49566071,
        0.01793687,  0.62482541,  0.63753013,  0.2116782 ,  0.27721045,
        0.2572147 ,  0.16170782,  0.82366648,  0.13308887,  0.31182152,
        0.66037292,  0.06749542,  0.33217543,  0.79389075,  0.20602317,
        0.4067727 ,  0.35727558,  0.27135679,  0.2371439 ,  0.29443231,
        0.30786745,  0.40356238,  0.29043287,  0.35625636,  0.42836788,
        0.83677502,  0.3969728 ,  0.12134964,  0.49672749,  0.03858746,
        0.47782967,  0.38178751,  0.76692281,  0.49609255,  0.61627399,
        0.95071726,  0.74659952,  0.48390316,  0.46420526,  0.23055395,
       -0.048203  ,  0.70411953,  0.67288244,  0.30997459,  0.17265746,
        0.27568854,  0.63030711, -0.13269191,  0.49859373,  0.26183776,
```

In [49]:
```
sns.histplot(s,kde=True)
```

Out[49]:
```
<AxesSubplot:ylabel='Count'>
```



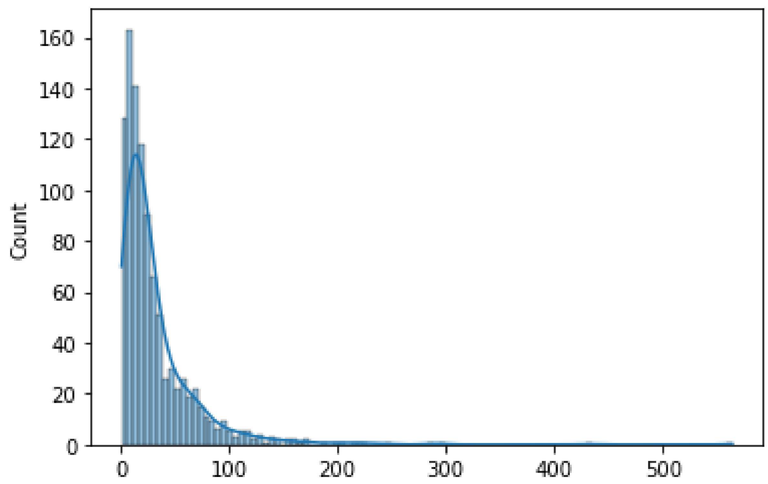
OTHER DISTRIBUTION

LOG NORMAL DISTRIBUTION,POWER LAW DISTRIBUTION

In [54]:
```
mu,sigma = 3,1 ## mean and standard deviation
s = np.random.lognormal(mu,sigma,1000)
```

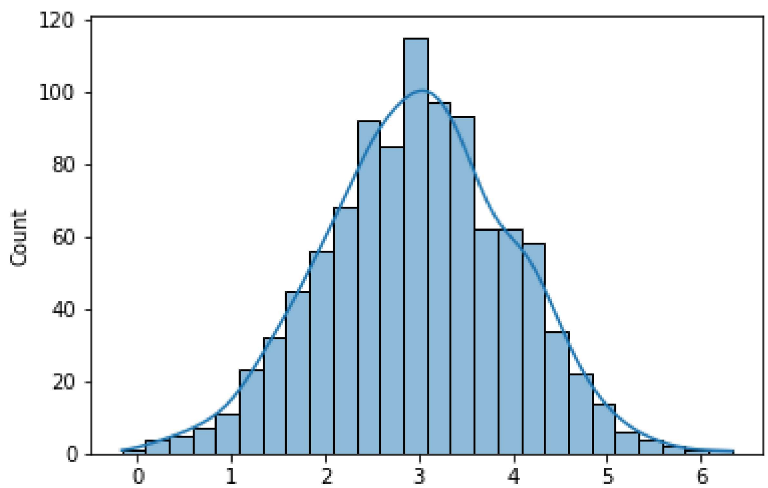
In [55]:
```
sns.histplot(s,kde=True)
```

Out[55]: <AxesSubplot:ylabel='Count'>



In [56]:
```
sns.histplot(np.log(s),kde=True)
```

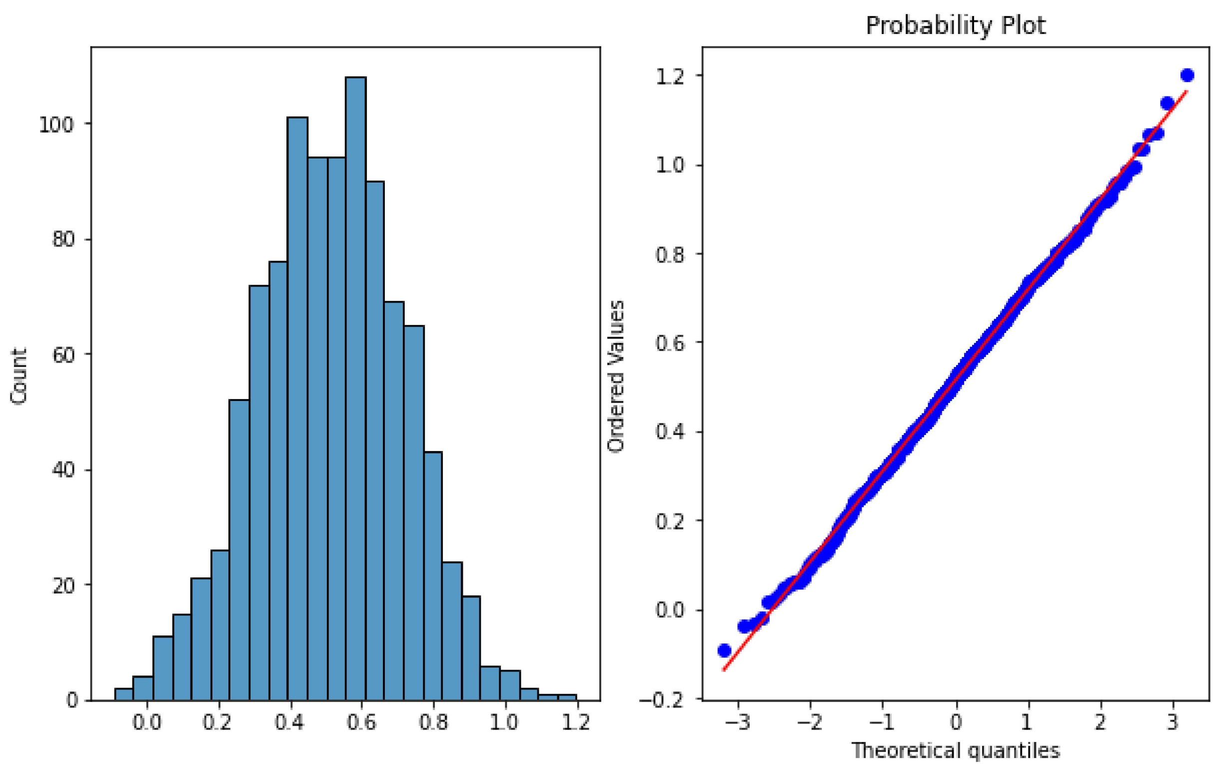
Out[56]: <AxesSubplot:ylabel='Count'>



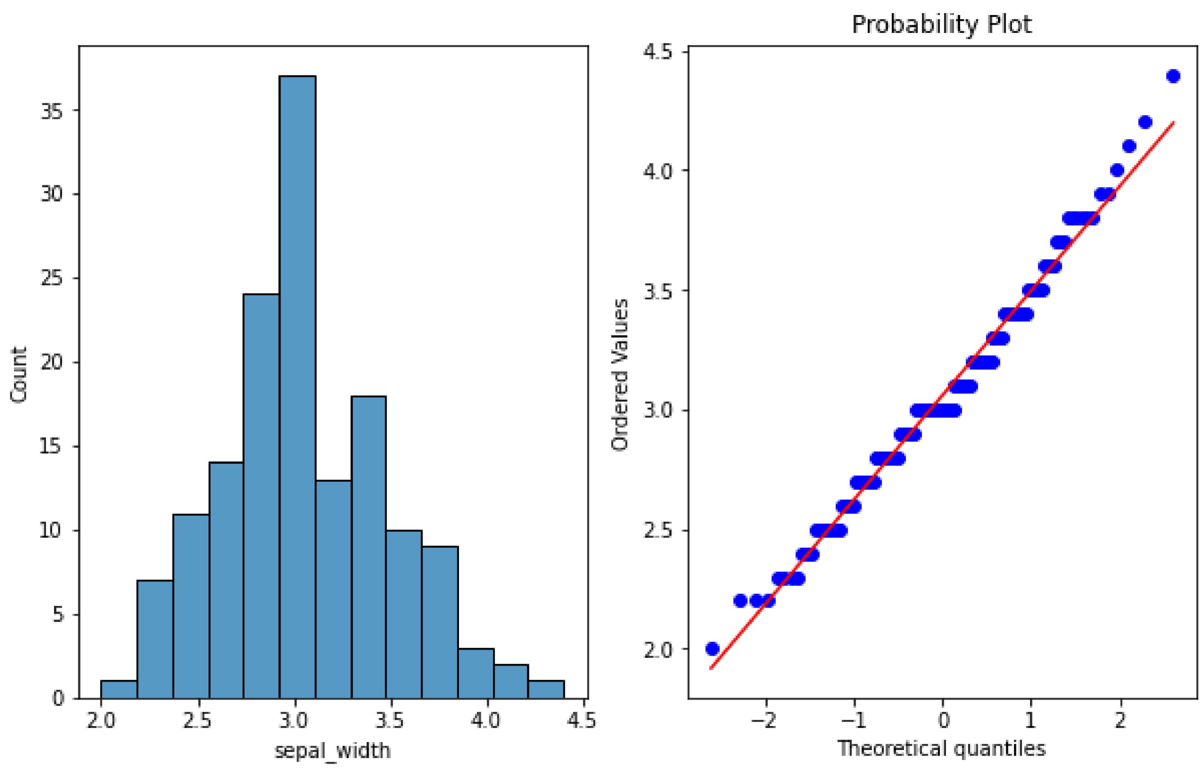
CHECK WEATHER DISTRIBUTION IS NORMAL DISTRIBUTION

In [68]:
```
### If you want to check weather feature is normal or guassian distributed
### Q-Q plot

import matplotlib.pyplot as plt
import scipy.stats as stat
import pylab
def plot_data(sample):
    plt.figure(figsize=(10,6))
    plt.subplot(1,2,1)
    sns.histplot(sample)
    plt.subplot(1,2,2)
    stat.probplot(sample,dist='norm',plot=pylab)
    plt.show()
```

In [69]:
```
## Create a normal distributed dataset
s=np.random.normal(0.5,0.2,1000)
plot_data(s)
```



In [70]:
```
plot_data(df['sepal_width'])
```



PEARSON AND SPERMAN RANK CORRELATION

In [80]:
```
df = sns.load_dataset('tips')
```

In [81]:
```
df.head()
```

Out[81]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [82]:
```
df.corr()
```

Out[82]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [83]:
```
sns.pairplot(df)
```

Out[83]: <seaborn.axisgrid.PairGrid at 0x2207a64dcd0>