Application of Stats in Python using Jupyter note book

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stat
import pylab as pl
import statistics as st
import math
```

In [9]:
```
age = [23,24,32,45,12,43,67,45,54,32,24,56,28,120]
```

In [10]:
```
np.mean(age)    # numpy module used
```

Out[10]:
```
43.214285714285715
```

In [11]:
```
np.median(age)
```

Out[11]:
```
37.5
```

In [12]:
```
st.mean(age)  # statistics module used
```

Out[12]:
```
43.214285714285715
```

In [13]:
```
st.median(age)
```

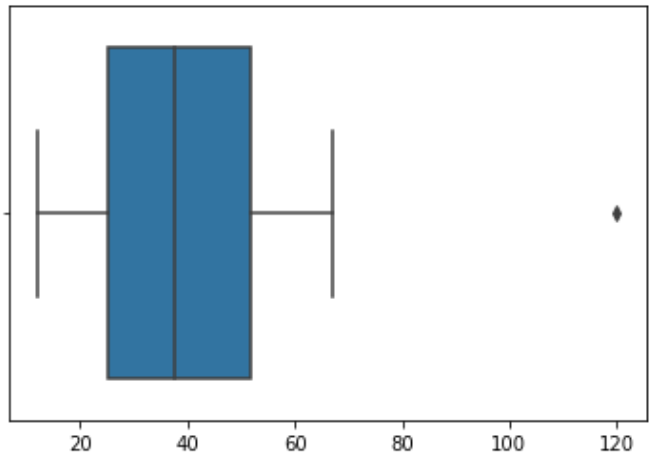
Out[13]:
```
37.5
```

In [14]:
```
st.mode(age)
```

Out[14]:
```
24
```

In [16]:
```
sns.boxplot(age)  # box plot helps finding outliers in data
```

Out[16]:
```
C:\Users\annah\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resul
t in an error or misinterpretation.
  warnings.warn(
<AxesSubplot:>
```



5 Number Summary

In [20]:
```
q1,q3 = np.percentile(age,[25,75])
```

In [21]:
```
IQR = q3-q1
lower_fence = q1 - (1.5*IQR)
upper_fence = q3 + (1.5*IQR)
```

In [28]:
```
# 5 Number summary:
lower_fence, min(age),q1, np.median(age),q3,max(age),upper_fence
```

Out[28]:
```
(-15.125, 12, 25.0, 37.5, 51.75, 120, 91.875)
```

In [29]:
```
# values that lie outside of either lower_fence or upper_fence are outliers
```

Measure of Dispersion

In [31]:
```
st.variance(age)   # returns sample variance of the data
```

Out[31]:
```
719.4120879120878
```

In [33]:
```
st.pvariance(age)  # returns population variance of the data
```

Out[33]:
```
668.0255102040816
```

In [36]:
```
np.var(age,axis = 0)  # returns population variance of the data
```

Out[36]:
```
668.0255102040816
```

In [34]:
```
math.sqrt(st.pvariance(age))  # retuns standard deviation of population
```

Out[34]:
```
25.846189471643235
```

In [38]:
```
## population variance

def variance(data):
```

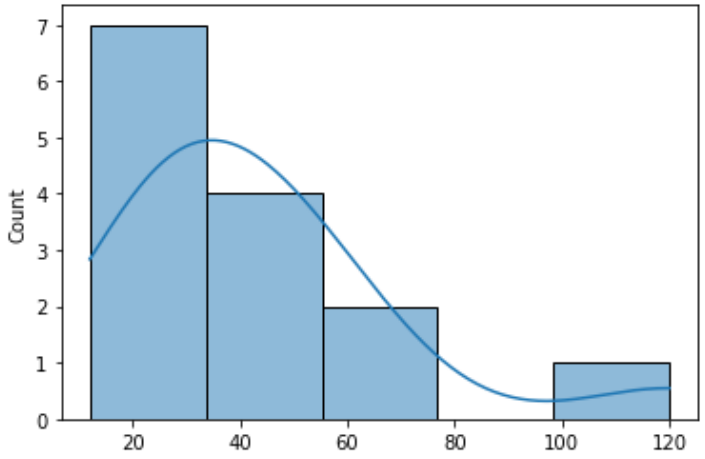
```
    n = len(data)
    # mean of data
    mean = sum(data)/n
    # Variance of data
    deviation = [ (x-mean)**2 for x in data ]
    var =  sum(deviation)/n
    return var

variance(age)

## for sample variance use ' n-1 '
```

Out[38]:
```
668.0255102040816
```

Histogram & PDF

In [40]:
```
sns.histplot(age,kde = True)  # kde is uded to estimate pdf
```

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



working on 'IRIS' data set

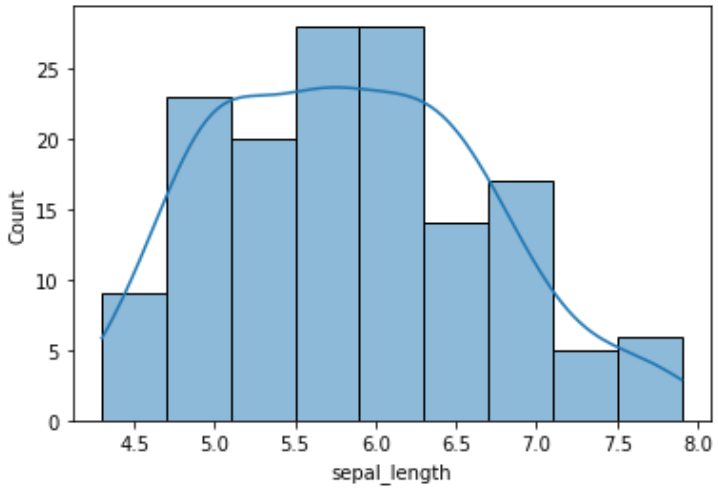
In [41]:
```
df = sns.load_dataset('iris')
df.head()
```

Out[41]:

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

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

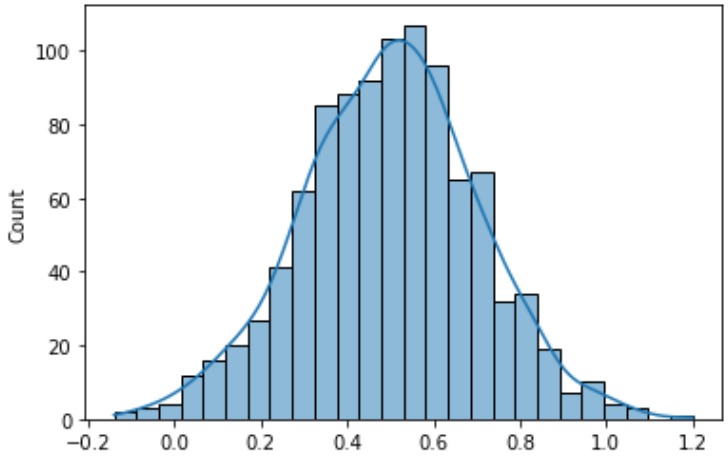


Create a Normal Distribution dataset

In [47]:
```
h = np.random.normal( 0.5, 0.2, 1000) # mean 0.5, standard deviation 0.2 , no of values =
```

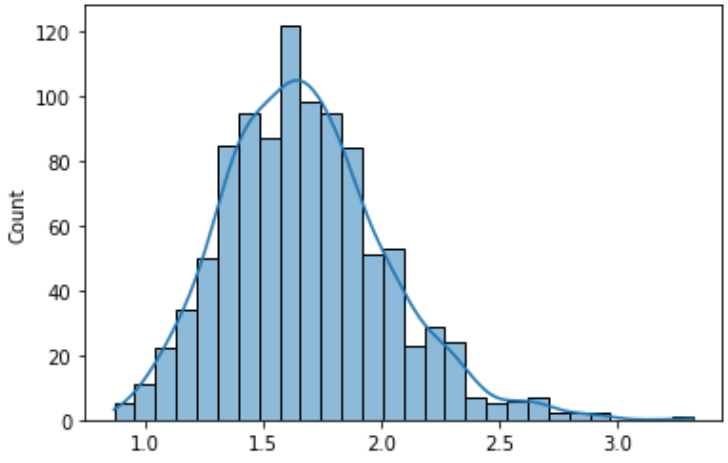
In [49]:
```
sns.histplot(h,kde = True)
```

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



In [52]:
```
sns.histplot( np.exp(h), kde = True)   # coverting normal sitribution to log normal
```

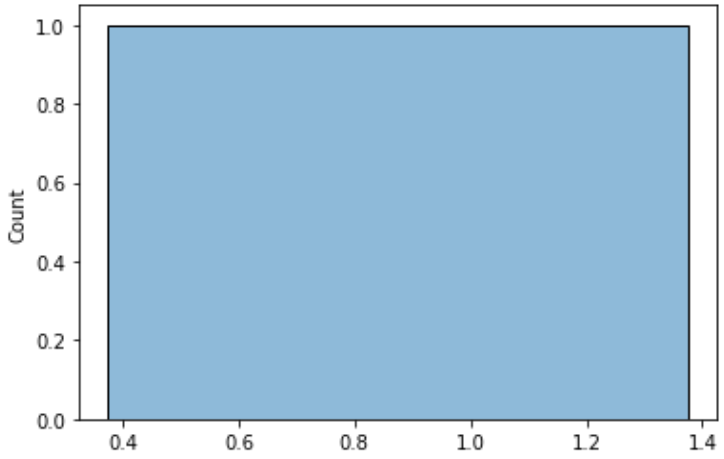
Out[52]:
```
<AxesSubplot:ylabel='Count'>
```



Log normal & Power law Distribution

In [81]:
```
p = np.random.power(0.5,1)  # power distribution
sns.histplot(p, kde = True)
```

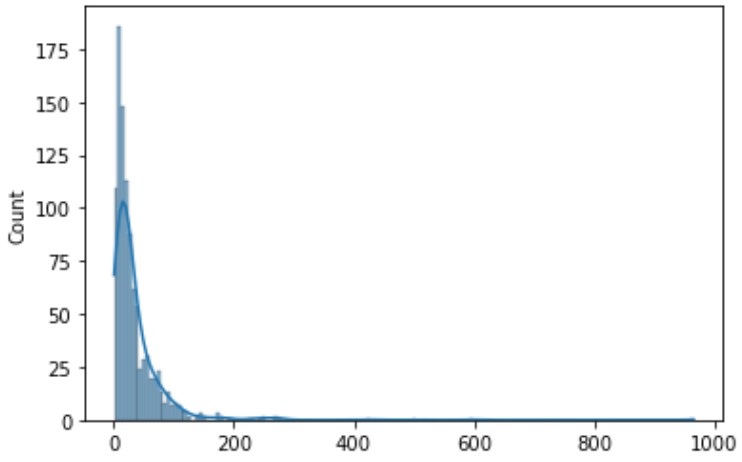
Out[81]:
```
<AxesSubplot:ylabel='Count'>
```



In [57]:
```
mu, sigma = 3, 1
s = np.random.lognormal(mu,sigma,1000)
```

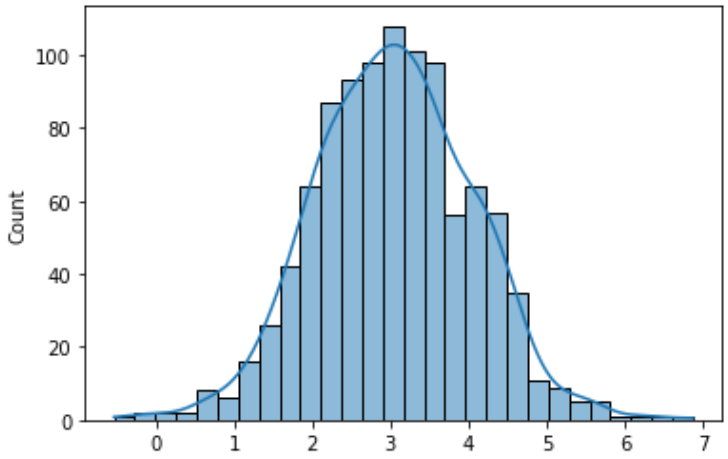
In [58]:
```
sns.histplot(s, kde = True)
```

Out[58]:
```
<AxesSubplot:ylabel='Count'>
```



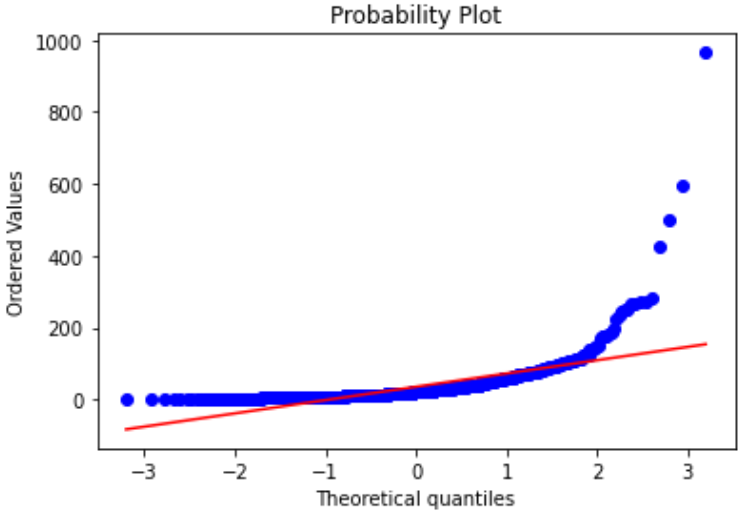
In [59]:
```
# Converting log- normal to normal distribution

sns.histplot(np.log(s),kde = True)
```

Out[59]:
```
<AxesSubplot:ylabel='Count'>
```

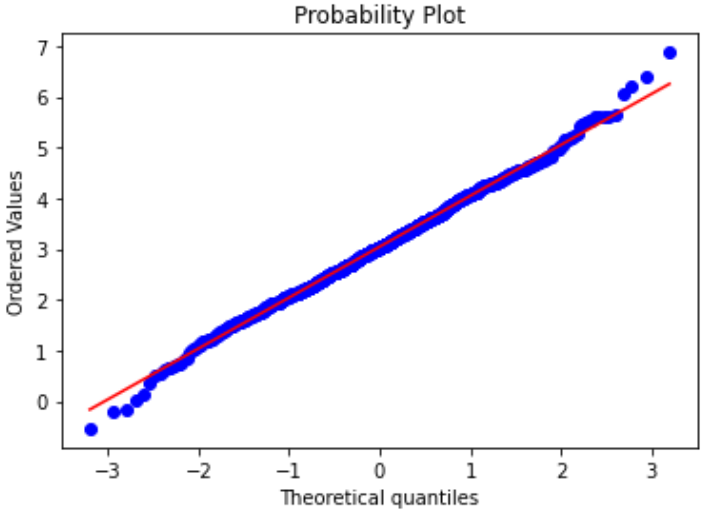


Using Q Q plot to find weather distribution is normal or not

In [68]:
```
stat.probplot(s,plot = pl)    # here s is log normal data set
plt.show()
```



In [69]:
```
stat.probplot(np.log(s),plot = pl)  # s(data set)  converted to normal distribution from l
plt.show()
```



Pearson & Spearman rank correlation

In [82]:
```
df = sns.load_dataset('tips')   # loading tips dataset from seaborn
```

In [83]:
```
df.head()
```

Out[83]:

	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

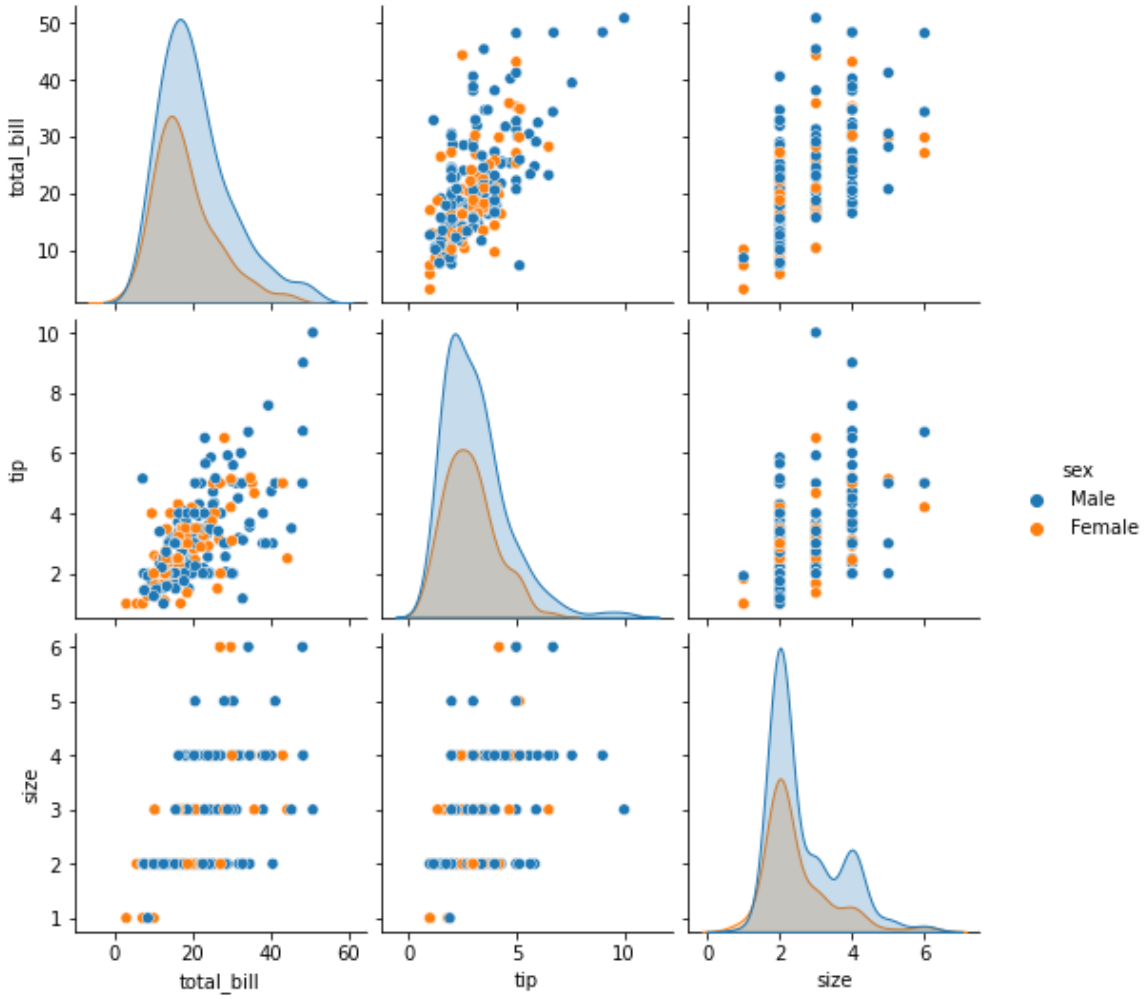
In [84]:
```
df.corr()   # finding correcation
```

Out[84]:

	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 [88]:
```
sns.pairplot(df, hue = 'sex')  # based on gender classification
```

Out[88]:
```
<seaborn.axisgrid.PairGrid at 0x152c496fa00>
```


total_bill
tip
size
sex
Male
Female