Statistics

In [1]:
```
# Import Libraries
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import statistics
```

In [3]:
```
# compute mean.median,mode
# Load dataset tips
df = sns.load_dataset('tips')
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
# mean
np.mean(df['total_bill'])
```

Out[5]:
```
19.785942622950824
```

In [6]:
```
# median
np.median(df['total_bill'])
```

Out[6]:
```
17.795
```

In [7]:
```
# differences in the values of mean and median is due to the presence of outliers
```

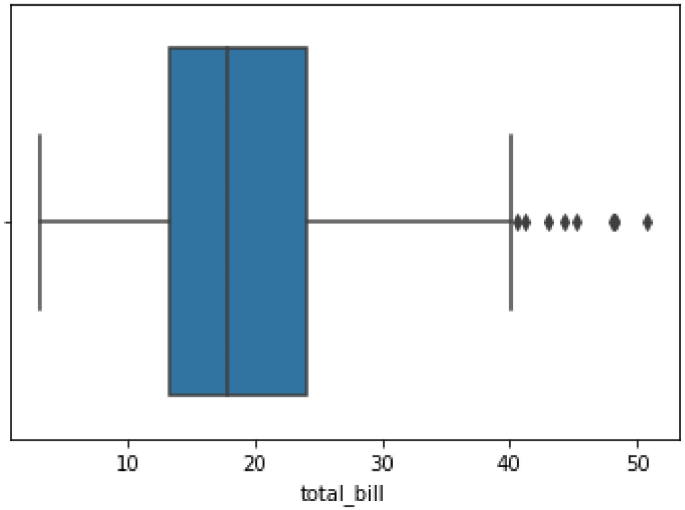
In [8]:
```
# mode
statistics.mode(df['total_bill'])
```

Out[8]:
```
13.42
```

In [9]:
```
# boxplot to see outliers
sns.boxplot((df['total_bill']))
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a ke
yword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:
```
<AxesSubplot:xlabel='total_bill'>
```

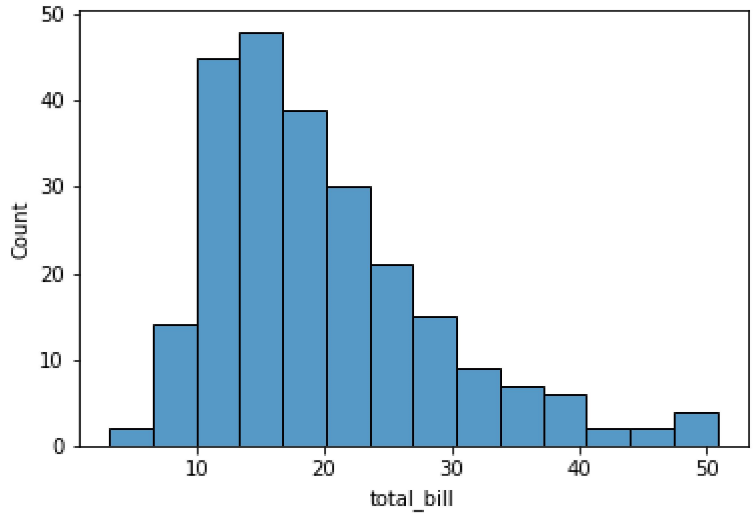


In [10]:
```
# values of drawing boxplot can be find using describe()
df['total_bill'].describe()
```

Out[10]:
```
count    244.000000
mean      19.785943
std        8.902412
min        3.070000
25%       13.347500
50%       17.795000
75%       24.127500
max       50.810000
Name: total_bill, dtype: float64
```

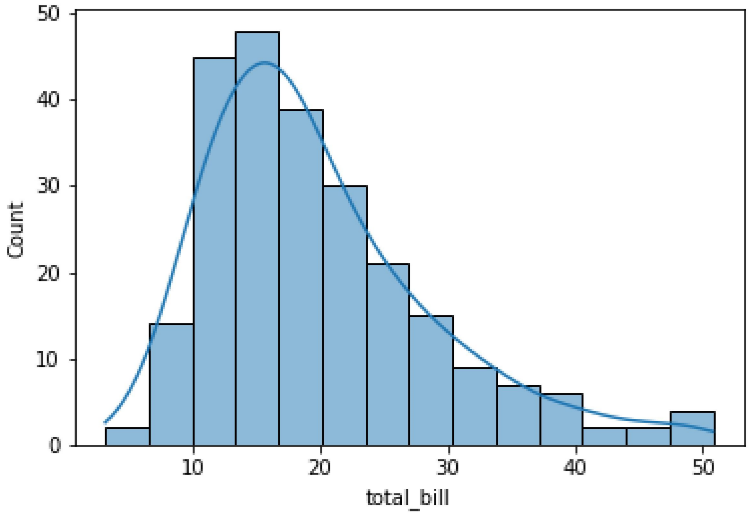
In [11]:
```
# data is not normally distributed
sns.histplot(df['total_bill'])
```

Out[11]:
```
<AxesSubplot:xlabel='total_bill', ylabel='Count'>
```



In [12]:
```
# smoothening the curve we understand
# data is not normally distributed
# it is right skewed
sns.histplot(df['total_bill'],kde = True)
```

Out[12]: <AxesSubplot:xlabel='total_bill', ylabel='Count'>



Checking another Dataset

In [13]:
```
df1=sns.load_dataset('iris')
```

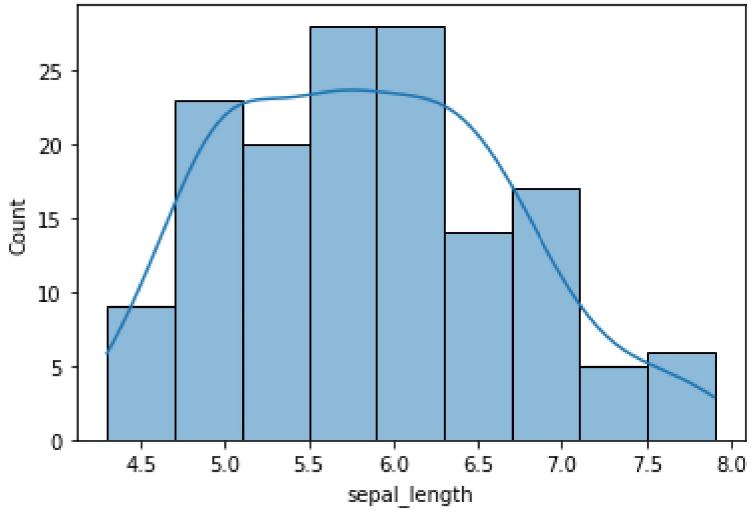
In [14]:
```
df1.head()
```

Out[14]:

	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

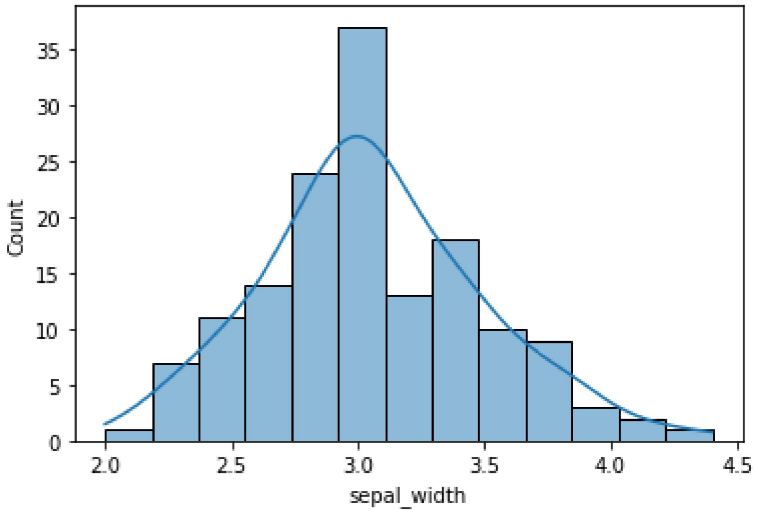
In [15]:
```
sns.histplot(df1['sepal_length'],kde = True)
```

Out[15]:
```
<AxesSubplot:xlabel='sepal_length', ylabel='Count'>
```



In [16]:
```
# Exactly following Gaussian Distribution
sns.histplot(df1['sepal_width'],kde = True)
```

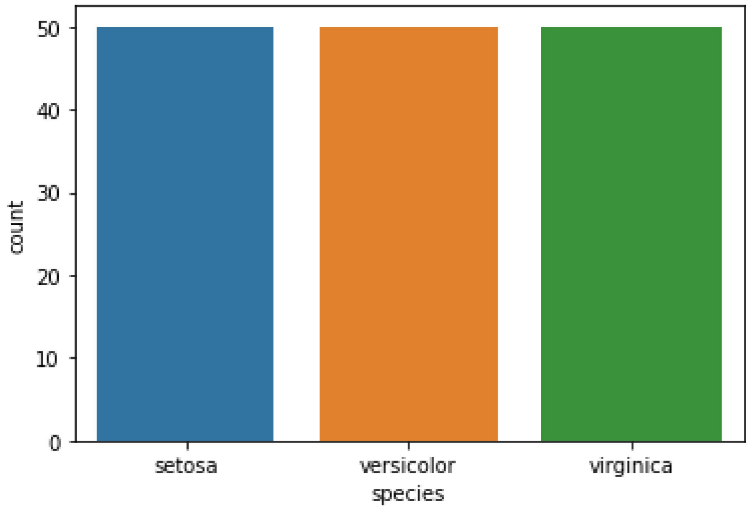
Out[16]: <AxesSubplot:xlabel='sepal_width', ylabel='Count'>



In [17]:
```
# Bar Graph on count
sns.countplot(df1['species'])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a ke
yword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]: <AxesSubplot:xlabel='species', ylabel='count'>



In [18]:
```
# checking 25 and 75 percentile of sepal length
np.percentile(df1['sepal_length'],[25,75])
```

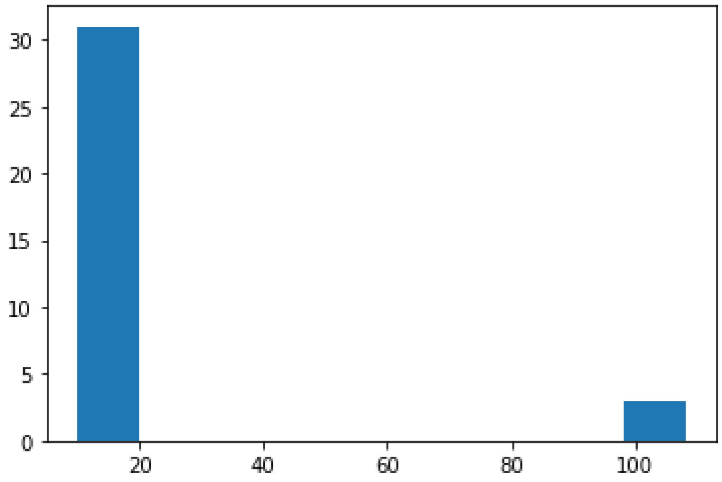
Out[18]:
```
array([5.1, 6.4])
```

Outliers

In [19]:
```
## Define a dataset
dataset = [11,10,12,14,12,15,14,13,15,102,12,14,17,19,107,10,13,12,14,12,108,12,11,14,13,15,10,15,12,10,14,13,15,10]
```

In [20]:
```
plt.hist(dataset)
```

Out[20]:
```
(array([31.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  3.]),
 array([ 10. ,  19.8,  29.6,  39.4,  49.2,  59. ,  68.8,  78.6,  88.4,
         98.2, 108. ]),
 <BarContainer object of 10 artists>)
```



In [21]:
```
## outliers using Z score

outliers=[]
def detect_outlier(dataset):
    threshold=3 ## 3rd std deviation
    mean=np.mean(dataset)
    std=np.std(dataset)

    for i in dataset:
        z_score = (i-mean)/std
        if np.abs(z_score)>threshold:
            outliers.append(i)

    return outliers
```

In [22]:
```
detect_outlier(dataset)
```

Out[22]:
```
[102, 107, 108]
```

IQR - Interquartile Range

1. Sort the data
2. Calculate Q1 and Q3
3. IQR(Q3-Q1)
4. Find the lower fence(q1-1.5(IQR))
5. Find the lower fence(q3+1.5(IQR))

In [23]:
```
dataset=sorted(dataset)
```

In [24]:
```
dataset
```

Out[24]:
```
[10,
 10,
 10,
 10,
 10,
 10,
 11,
 11,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 12,
 13,
 13,
 13,
 13,
 13,
 14,
 14,
 14,
 14,
 14,
 14,
 14,
 15,
 15,
 15,
 15,
 15,
 15,
 17,
 19,
 102,
 107,
 108]
```

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

In [26]:
```
print(q1,q3)
```

```
12.0 15.0
```

In [27]:
```
iqr = q3-q1
```

In [28]:
```
print(iqr)
```

```
3.0
```

In [29]:
```
## Find lower fence and Higher fence
lower_fence=q1-(1.5*iqr)
higher_fence=q1+(1.5*iqr)
```

In [30]:
```
print(lower_fence)
```

```
7.5
```

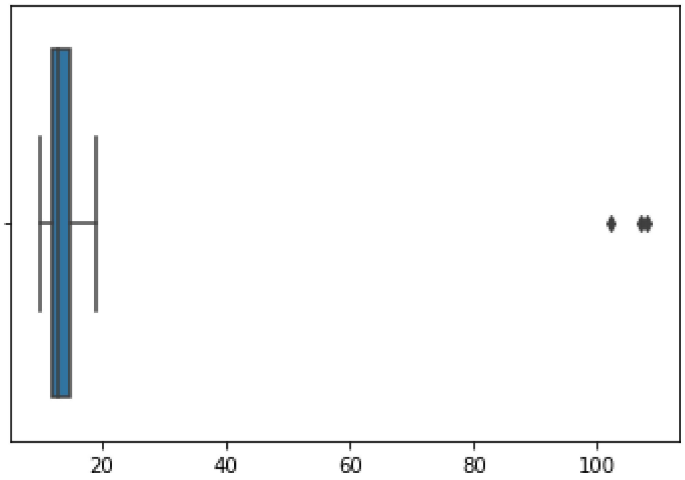
In [31]:
```
print(higher_fence)
```

```
16.5
```

In [32]:
```
sns.boxplot(dataset)
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a ke
yword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[32]: <AxesSubplot:>



Statistical Hypothesis Testing

z-test

Suppose the IQ in a certain population is normally distributed with a mean μ = 100 and standard deviation of σ = 15.

A researcher wants to know if a new drug affects IQ levels,so he recruits 20 patients to try it and records their IQ levels.

The following code shows how to perform a one sample z-test in python to determine if the new drug cause a significant difference n IQ levels:

In [33]:
```
from statsmodels.stats.weightstats import ztest as ztest

#enter IQ level of patients

data = [88, 92 , 94, 94, 96, 97, 97, 97, 99, 99, 105, 109, 109, 109, 110, 112, 112, 113, 114, 115]

ztest(data,value = 100)
```

Out[33]: (1.5976240527147705, 0.1101266701438426)

In [34]:
```
# first value is z-test value and second value is p-value
# if p value is not less than alpha value, got rejected , 0.11>0.05 so got rejected
```

t-test

In [35]:
```
ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,30,28,14,24,16,17,32,35,26,27,65,18,43,23,21,20,19,70]
```

In [36]:
```
ages_mean=np.mean(ages)
ages_mean
```

Out[36]: 30.34375

In [37]:
```
# here we don't know population standard deviation
# considering sample size 10
```

In [38]:
```
sample_size = 10
age_sample = np.random.choice(ages,sample_size)
```

In [39]:
```
age_sample
```

Out[39]:
```
array([21, 21, 55, 16, 24, 30, 19, 30, 40, 24])
```

In [40]:
```
np.mean(age_sample)
```

Out[40]:
```
28.0
```

In [41]:
```
from scipy.stats import ttest_1samp
```

In [42]:
```
ttest_1samp(age_sample,30)
```

Out[42]:
```
Ttest_1sampResult(statistic=-0.5396870722208659, pvalue=0.6025076250343979)
```

In [43]:
```
_,p_value = ttest_1samp(age_sample,30)
```

In [44]:
```
print(p_value)
```

```
0.6025076250343979
```

In [45]:
```
# if p value is greater than alpha value, got rejected
if p_value < 0.05:    # alpha value is 0.05 or 5%
    print(" we are rejecting null hypothesis")
else:
    print("we are accepting null hypothesis")
```

```
we are accepting null hypothesis
```

In [46]:
```
# where alpha = 0.05
# if p_value<=0.05, we will reject null hypothesis
# here 5% probability the null hypothesis is correct
# if p_value>=0.05, we will accept null hypothesis
```

In [47]:
```
# another example

# ages of the college students(poppulation)
#1 class student mean of all the ages
```

In [48]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import math
np.random.seed(6)
school_ages = stats.poisson.rvs(loc=18,mu=35,size=1500)
classA_ages = stats.poisson.rvs(loc=18,mu=30,size=60)
```

In [49]:
```
school_ages
```

Out[49]:
```
array([62, 59, 44, ..., 45, 52, 50])
```

In [50]:
```
classA_ages
```

Out[50]:
```
array([52, 46, 40, 40, 47, 50, 51, 45, 44, 52, 46, 53, 43, 44, 51, 50, 54,
       42, 54, 45, 61, 53, 49, 46, 47, 41, 45, 51, 43, 45, 48, 50, 40, 52,
       44, 55, 54, 40, 45, 46, 54, 42, 46, 35, 51, 51, 46, 48, 47, 35, 52,
       52, 39, 44, 48, 40, 42, 46, 47, 45])
```

In [51]:
```
classA_ages.mean()
```

Out[51]:
```
46.9
```

In [52]:
```
ttest_1samp(classA_ages,popmean=school_ages.mean())
```

Out[52]:
```
Ttest_1sampResult(statistic=-9.604796510704091, pvalue=1.139027071016194e-13)
```

In [53]:
```
_,p_value=ttest_1samp(classA_ages,popmean=school_ages.mean())
```

In [54]:
```
school_ages.mean()
```

Out[54]:
```
53.303333333333335
```

In [55]:
```
if p_value<=0.05:
    print("Reject H0 ")
else:
    print("Accept H0")
```

```
Reject H0
```

Correlation

In [56]:
```
import seaborn as sns
```

In [57]:
```
df = sns.load_dataset('iris')
df.head()
```

Out[57]:

	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

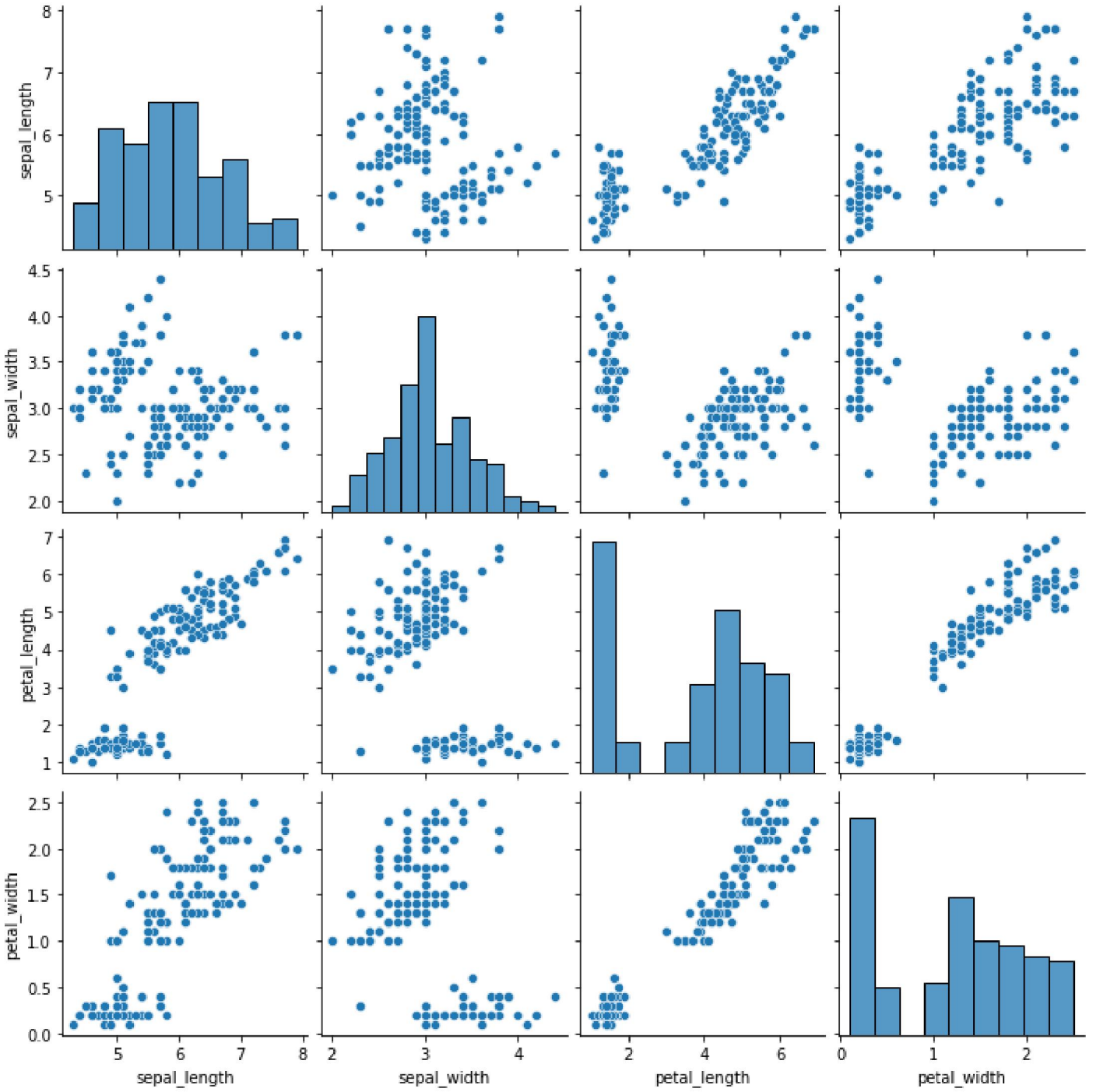
In [58]:
```
df.corr()
```

Out[58]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

In [59]:
```
sns.pairplot(df)
```

Out[59]:
```
<seaborn.axisgrid.PairGrid at 0x21269c82ac0>
```


sepal_length
sepal_width
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width

Chi-Square Test-

The test is applied when you have two categorical variables from a single population. It is used to determine whether there is a significant association between the two variables.

In [60]:
```
import scipy.stats as stats
```

In [61]:
```
dataset=sns.load_dataset('tips')
dataset.head()
```

Out[61]:

	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 [62]:
```
dataset_table=pd.crosstab(dataset['sex'],dataset['smoker'])
print(dataset_table)
```

```
smoker  Yes  No
sex
Male     60  97
Female   33  54
```

In [63]:
```
dataset_table.values
```

Out[63]:
```
array([[60, 97],
       [33, 54]], dtype=int64)
```

In [64]:
```
#Observed Values
Observed_Values = dataset_table.values
print("Observed Values :-\n",Observed_Values)
```

```
Observed Values :-
 [[60 97]
 [33 54]]
```

In [65]:
```
val=stats.chi2_contingency(dataset_table)
```

In [66]:
```
val
```

Out[66]:
```
(0.0,
 1.0,
 1,
 array([[59.84016393, 97.15983607],
        [33.15983607, 53.84016393]]))
```

In [67]:
```
Expected_Values=val[3]
```

In [68]:
```
no_of_rows=len(dataset_table.iloc[0:2,0])
no_of_columns=len(dataset_table.iloc[0,0:2])
ddof=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",ddof)
alpha = 0.05
```

```
Degree of Freedom:- 1
```

In [69]:
```
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
```

In [70]:
```
print("chi-square statistic:-",chi_square_statistic)
```

```
chi-square statistic:- 0.001934818536627623
```

In [71]:
```
critical_value=chi2.ppf(q=1-alpha,df=ddof)
print('critical_value:',critical_value)
```

```
critical_value: 3.841458820694124
```

In [72]:
```
#p-value
p_value=1-chi2.cdf(x=chi_square_statistic,df=ddof)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',ddof)
print('p-value:',p_value)
```

```
p-value: 0.964915107315732
Significance level:  0.05
Degree of Freedom:  1
p-value: 0.964915107315732
```

In [73]:
```
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
```

```
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
```

Anova Test(F-Test)

The t-test works well when dealing with two groups, but sometimes we want to compare more than two groups at the same time.

For example, if we wanted to test whether petal_width age differs based on some categorical variable like species, we have to compare the means of each level or group the variable

One Way F-test(Anova) :-

It tell whether two or more groups are similar or not based on their mean similarity and f-score.

Example : there are 3 different category of iris flowers and their petal width and need to check whether all 3 group are similar or not

In [74]:
```
import seaborn as sns
df1=sns.load_dataset('iris')
```

In [75]:
```
df1.head()
```

Out[75]:

	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 [76]:
```
df_anova = df1[['petal_width','species']]
```

In [77]:
```
grps = pd.unique(df_anova.species.values)
```

In [78]:
```
grps
```

Out[78]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [79]:
```
d_data = {grp:df_anova['petal_width'][df_anova.species == grp] for grp in grps}
```

In [80]:
```
F, p = stats.f_oneway(d_data['setosa'], d_data['versicolor'], d_data['virginica'])
```

In [81]:
```
print(p)
```

```
4.169445839443116e-85
```

In [82]:
```
if p<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
reject null hypothesis
```