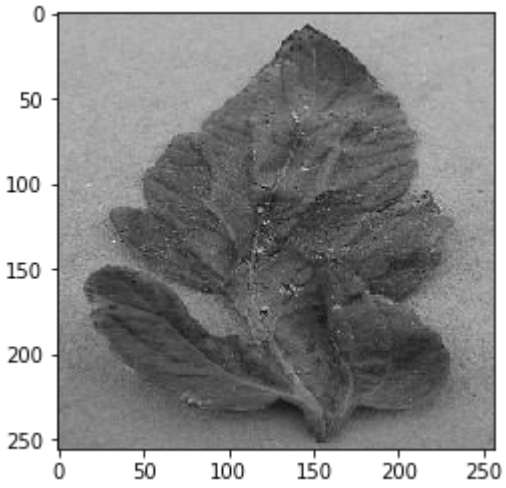
In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2

DATADIR = "C:/Users/admin/Desktop/test"
CATEGORIES = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]

for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for x in range(1,9):
        for imp in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,imp), cv2.IMREAD_GRAYSCALE)
            plt.imshow(img_array, cmap="gray")
            plt.show()
            break
        break
    break
```



In [10]:

```
print(img_array)
```

```
[[155 154 152 ... 157 155 152]
 [155 154 152 ... 158 155 152]
 [155 154 152 ... 158 155 152]
 ...
 [115 114 113 ... 106 122 124]
 [117 115 114 ... 124 134 124]
 [118 117 116 ... 132 136 115]]
```

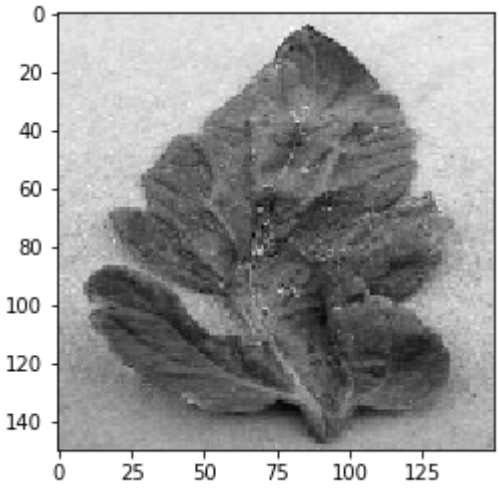
In [11]:

```
print(img_array.size)
print(img_array.shape)
```

```
65536
(256, 256)
```

In [15]:

```
IMG_SIZE = 150

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array, cmap = 'gray')
plt.show()
```



In [16]:

```
training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for imp in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,imp), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
                #print(training_data[10])
            except Exception as e:
                pass

create_training_data()

print(len(training_data))
```

75

In [17]:

```
import random

random.shuffle(training_data)
```

In [18]:

```
classify = []
for sample in training_data[:10]:
    #print(sample[1])
    classify.append(sample[1])
#print("new")
for classify in training_data[:20]:
    #print(classify[1])
    if classify[1] == 0:
        print(classify[1], "Bacterial_spot")
    elif classify[1] == 1:
        print(classify[1], "Early_Blight")
    elif classify[1] == 2:
        print(classify[1], "Healthy")
    elif classify[1] == 3:
        print(classify[1], "Late_Blight")
    else:
        print(classify[1], "Leaf_Mold")
```

```
4 Leaf_Mold
3 Late_Blight
3 Late_Blight
2 Healthy
2 Healthy
3 Late_Blight
1 Early_Blight
2 Healthy
3 Late_Blight
2 Healthy
2 Healthy
1 Early_Blight
0 Bacterial_spot
4 Leaf_Mold
0 Bacterial_spot
0 Bacterial_spot
1 Early_Blight
1 Early_Blight
3 Late_Blight
4 Leaf_Mold
```

In [20]:

```
X = []
y = []
for features, label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
```

In [21]:

```
import pickle

pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()
```

In [22]:

```
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_inn = open("y.pickle", "rb")
y = pickle.load(pickle_inn)
```

In [23]:

```
X[1]
```

Out[23]:

```
array([[[117],
        [116],
        [116],
        ...,
        [134],
        [127],
        [124]],

       [[118],
        [119],
        [120],
        ...,
        [128],
        [134],
        [130]],

       [[117],
        [119],
        [120],
        ...,
        [129],
        [126],
        [126]],

       ...,

       [[113],
        [114],
        [114],
        ...,
        [ 74],
        [ 76],
        [ 78]],

       [[107],
        [112],
        [114],
        ...,
        [ 72],
        [ 70],
        [ 71]],

       [[111],
        [109],
        [109],
        ...,
        [ 73],
        [ 68],
        [ 68]]], dtype=uint8)
```

In [24]:

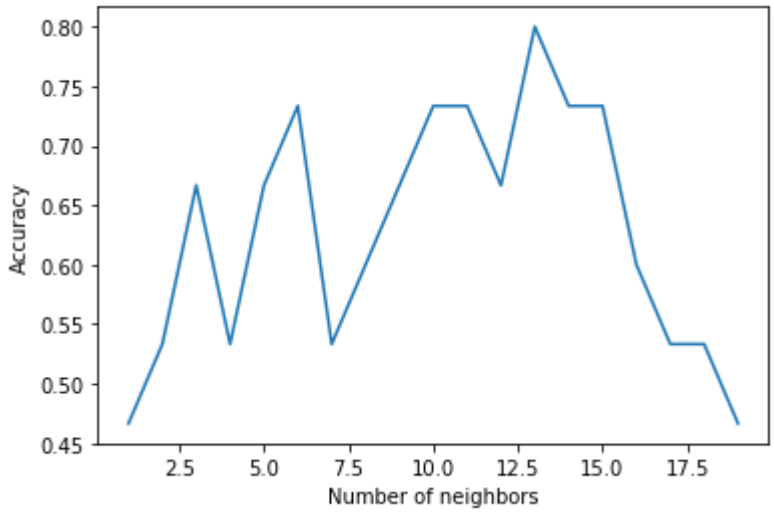
```
from sklearn import preprocessing, neighbors
from sklearn.model_selection import train_test_split
import pandas as pd

nsamples,nx,ny,nz=X.shape
XX=X.reshape(nsamples,nx*ny*nz)
X_train, X_test, y_train, y_test =train_test_split(XX, y, test_size=0.20)
clf = neighbors.KNeighborsClassifier()
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)
print(accuracy)
```

```
0.6666666666666666
```

In [25]:

```
print("Using SKLEARN")
lix = []
liy = []
index=1
acc=0
from sklearn.neighbors import KNeighborsClassifier
for k in range(1, 20):
    neigh = KNeighborsClassifier(n_neighbors=k)
    neigh.fit(X_train, y_train)
    liy.append(neigh.score(X_test, y_test))
    if liy[k-1]>acc:
        acc=liy[k-1]
        index=k-1
    lix.append(k)

plt.plot(lix, liy)
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
print("max acc at k="+str(index+1)+" acc of "+str(acc))
```

```
Using SKLEARN
```



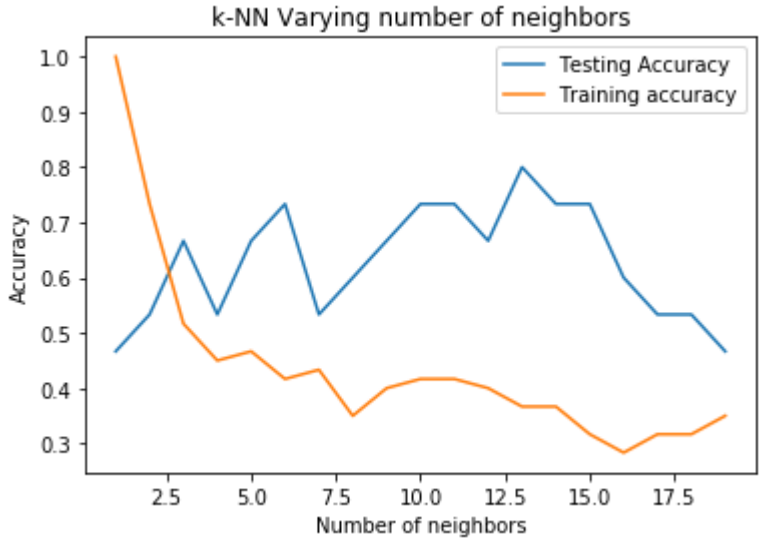

```
max acc at k=13 acc of 0.8
```

In [30]:

```
from sklearn.neighbors import KNeighborsClassifier

#Setup arrays to store training and test accuracies
neighbors = np.arange(1,20)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```

In [32]:

```
plt.title('k-NN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
#print(accuracy)
plt.show()
```



In [35]:

```
knn = KNeighborsClassifier(n_neighbors=11)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[35]:

```
0.7333333333333333
```

In [36]:

```
knn = KNeighborsClassifier(n_neighbors=13)
knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[36]:

```
0.8
```

In [37]:

```
from sklearn.metrics import confusion_matrix
#let us get the predictions using the classifier we had fit above
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
```

Out[37]:

```
array([[5, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 1, 2, 0],
       [1, 0, 1, 0, 1]], dtype=int64)
```

In [38]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      1.00      1.00         1
           2       0.60      1.00      0.75         3
           3       1.00      0.67      0.80         3
           4       1.00      0.33      0.50         3

    accuracy                           0.80        15
   macro avg       0.89      0.80      0.79        15
weighted avg       0.86      0.80      0.78        15
```

In [43]:

```
class KNearestNeighbor(object):
    def __init__(self):
        pass

    def predict_label(self, dists, k=3):
        num_test = dists.shape[0]
        y_pred = np.zeros(num_test)
        for i in range(num_test):
            closest_y = []
            closest_y = self.y_train[np.argsort(dists[i])][0:k]
            y_pred[i] = np.bincount(closest_y).argmax()
        return y_pred
    def train(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X, k=3):
        dists = self.compute_distances_no_loops(X)

        return self.predict_labels(dists, k=k)

    def compute_distances_no_loops(self, X):
        num_test = X.shape[0]
        num_train = self.X_train.shape[0]
        dists = np.zeros((num_test, num_train))
        dists = np.sqrt((X ** 2).sum(axis=1, keepdims=1) + (self.X_train ** 2).sum(axis
=1) - 2 * X.dot(self.X_train.T))

        return dists

    def predict_labels(self, dists, k=3):
        num_test = dists.shape[0]
        y_pred = np.zeros(num_test)
        for i in range(num_test):
            closest_y = []
            closest_y = self.y_train[np.argsort(dists[i])][0:k]
            closest_y = closest_y.astype(int)
            y_pred[i] = np.bincount(closest_y).argmax()
        return y_pred
```

In [44]:

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/eb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

```
Predicting custom image
EARLY_BLIGHT
```

In [45]:

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/bs1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

```
Predicting custom image
BACTERIAL_SPOT
```

In [46]:

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/h1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

```
Predicting custom image
HEALTHY
```

In [47]:

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/lb1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

```
Predicting custom image
LATE_BLIGHT
```

In [48]:

```
print("Predicting custom image")
labels = ["BACTERIAL_SPOT", "EARLY_BLIGHT", "HEALTHY", "LATE_BLIGHT", "LEAF_MOLD"]
img = cv2.imread(r"C:/Users/admin/Desktop/lm1.JPG")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_pred = cv2.resize(img, (150, 150))
img_pred = np.reshape(img_pred, (1, img_pred.shape[0]*img_pred.shape[1]))
e=knn.predict(img_pred)
print(labels[(e[0])])
```

```
Predicting custom image
BACTERIAL_SPOT
```

In [ ]:

```

```