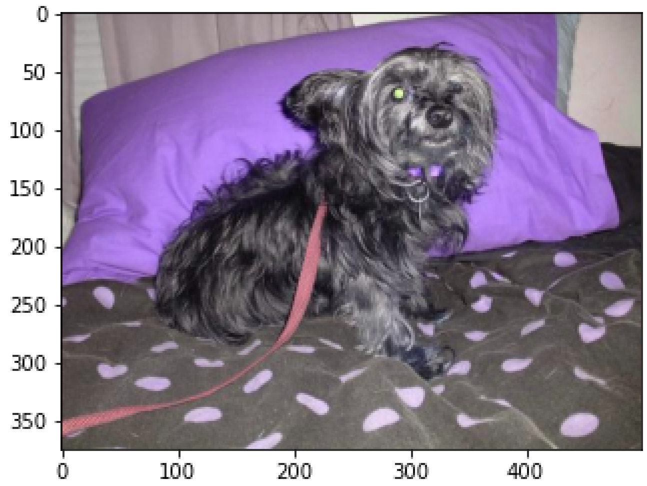
In [1]:

```
# Noah Manz
# 09/26/2022
# Simple CNN to classify dogs and cats

#Import libraries
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import random
import pickle
import tensorflow as tf
from tensorflow import keras
```

In [25]:

```
#Import data and define the 2 categories
DATADIR = 'C:/Users/Noah/Desktop/PetImages'
CATEGORIES = ['Dog','Cat']

#Iterate thru the dog/cat categories and create the labels for each image in the datafile
for category in CATEGORIES:
    path = os.path.join(DATADIR,category)
    #Convert to grayscale. Not sure what else happens here
    for img in os.listdir(path):
        #img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE) # For grayscale images
        img_array = cv2.imread(os.path.join(path,img)) #For color images
        plt.imshow(img_array,cmap='gray')
        plt.show()
        print(img_array.shape)
        print()
        print(img_array)
        break
    break
```

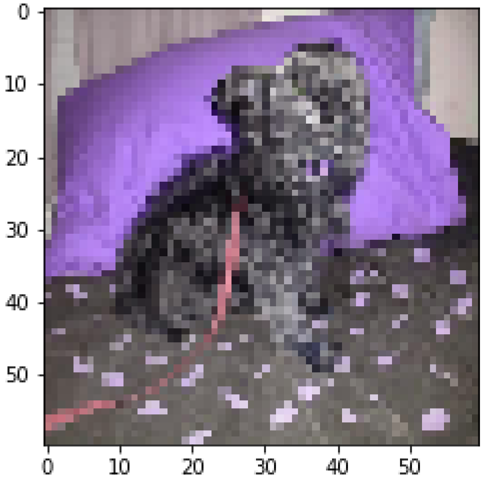

0
50
100
150
200
250
300
350
0
100
200
300
400

```
(375, 500, 3)

[[[126 115 117]
  [126 115 117]
  [130 117 119]
  ...
  [142 132 132]
  [141 131 131]
  [141 131 131]]

 [[127 116 118]
  [126 115 117]
  [130 117 119]
  ...
  [144 134 134]
  [143 133 133]
  [143 133 133]]

 [[128 117 119]
  [127 116 118]
  [131 118 120]
  ...
  [146 136 136]
  [145 135 135]
  [145 135 135]]

 ...

 [[ 84  80  75]
  [ 79  75  70]
  [ 78  74  69]
  ...
  [ 85  82  74]
  [ 81  78  70]
  [ 78  75  67]]

 [[ 83  79  74]
  [ 77  73  68]
  [ 74  70  65]
  ...
  [ 77  74  66]
  [ 78  75  67]
  [ 79  76  68]]

 [[ 79  75  70]
```

```
  [ 76  72  67]
  [ 75  71  66]
  ...
  [ 80  77  69]
  [ 78  75  67]
  [ 76  73  65]]]
```

In [26]:

```
#Resize images to be 50x50
IMG_SIZE=60
new_array=cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
#CV2 does BRG instead of RBG. Need to deal w/ this for my images?
plt.imshow(new_array,cmap='gray')
plt.show()
print(new_array.shape)
```



```
(60, 60, 3)
```

In [29]:

```
#Create the training data
training_data=[]

def create_training_data():
    for category in CATEGORIES:
        path=os.path.join(DATADIR,category)
        class_num=CATEGORIES.index(category)

        for img in os.listdir(path)[:5]:
            try:
                #img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE) # For grayscale images
                img_array=cv2.imread(os.path.join(path,img)) #For color images
                new_array=cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
                training_data.append([new_array,class_num])
            except Exception as e:
                pass

create_training_data()
print(len(training_data))
```

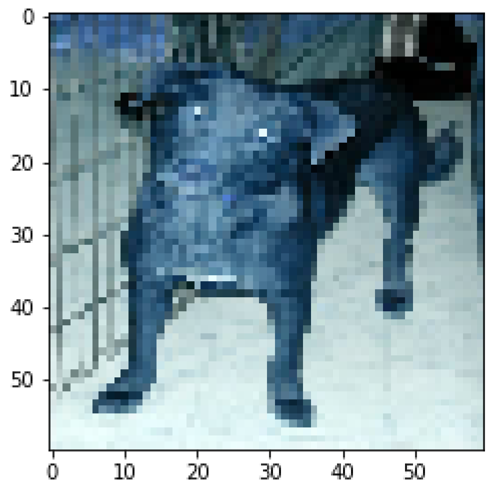
```
10
```

In [30]:

```
plt.imshow(training_data[1][0])
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x14c5c132040>
```



In [22]:

```
np.array(training_data).shape
```

Out[22]:

```
(4, 60, 60, 3)
```

In [31]:

```
#Shuffle the training data so that not all dogs are presented first followed by all cats
random.shuffle(training_data)
```

In [32]:

```
#Define empty arrays for training data
X=[]
y=[]

#Split training data into features and labels data
for features,label in training_data:
    X.append(features)
    y.append(label)
```

```
#Convert X to np.array and resize. -1 means any number (allows for all our images), IMG_SIZE corresponds to the
    #dimensions of the image and the final 1 refers to the size of the data w/in each cell. ie. 1 for grayscale values
        #If change the 1 to a 3, can have CNN that does color w/ 3 channel RBG
X=np.array(X).reshape(-1,IMG_SIZE,IMG_SIZE,3) #changed the 1 to a 3
y=np.array(y)

X=X/255.0

print(len(X))
print(len(y))
```

```
10
10
```

In [33]:

```
y
```

Out[33]:

```
array([1, 1, 0, 0, 1, 1, 1, 0, 0, 0])
```

In [66]:

```
X.shape
```

Out[66]:

```
(4, 60, 60, 3)
```

In [81]:

```
#Save the processed training data using pickle
#pickle_out=open('X.pickle','wb')
#pickle.dump(X,pickle_out)
#pickle_out.close()

#pickle_out=open('y.pickle','wb')
#pickle.dump(X,pickle_out)
#pickle_out.close()
```

In [11]:

```
#Create the model
model=tf.keras.models.Sequential()
model.add(keras.layers.Conv2D(64,(3,3),input_shape=X.shape[1:],activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(64))
model.add(keras.layers.Dense(1,activation='sigmoid'))

#Compile the model
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=['accuracy'])
```

In [12]:

```
#Train the model
model.fit(X,y,batch_size=32,epochs=3,validation_split=0.1)
```

```
Epoch 1/3
702/702 [==============================] - 12s 11ms/step - loss: 0.6111 - accuracy: 0.6606 - val_loss: 0.5382 - val_acc
uracy: 0.7194
Epoch 2/3
702/702 [==============================] - 7s 11ms/step - loss: 0.5041 - accuracy: 0.7560 - val_loss: 0.4709 - val_accu
racy: 0.7852
Epoch 3/3
702/702 [==============================] - 7s 10ms/step - loss: 0.4589 - accuracy: 0.7846 - val_loss: 0.5018 - val_accu
racy: 0.7723
```

Out[12]:

```
<keras.callbacks.History at 0x1905898f8e0>
```