Step 1: Import the digits dataset from sklearn and split it into train and test sets. ¶

Digits dataset in sklearn.datasets:

This dataset is made up of 1797 8x8 images.

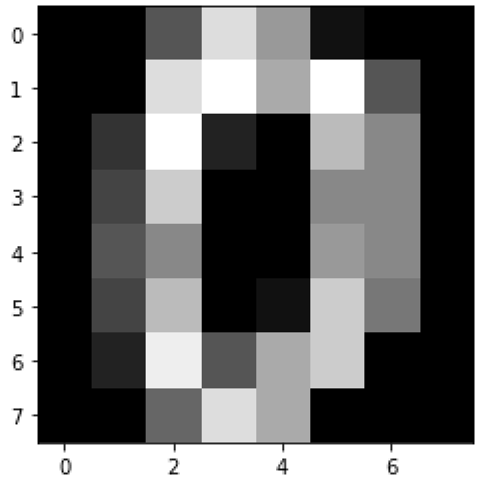
- Number of classes: 10 --> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Number of samples: 1797
- Number of features: 64
- Each sample represents the 64 pixel values of a 8x8 image.
- Range of each pixel value is 0 - 16.

In [1]:
```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

dataset = load_digits()
print("Feature Names: ")
print("----------------")
print(dataset.feature_names)
print()
print("Target Classes: ")
print("----------------")
print(dataset.target_names)
x = dataset.data
y = dataset.target
print(x[0])
plt.imshow(x[0].reshape((8, 8)), cmap = "gray")
plt.show()
print(y[0])
print(x.shape, y.shape)
```

```
Feature Names:
----------------
['pixel_0_0', 'pixel_0_1', 'pixel_0_2', 'pixel_0_3', 'pixel_0_4', 'pixel_0_
5', 'pixel_0_6', 'pixel_0_7', 'pixel_1_0', 'pixel_1_1', 'pixel_1_2', 'pixel_1
_3', 'pixel_1_4', 'pixel_1_5', 'pixel_1_6', 'pixel_1_7', 'pixel_2_0', 'pixel_
2_1', 'pixel_2_2', 'pixel_2_3', 'pixel_2_4', 'pixel_2_5', 'pixel_2_6', 'pixel
_2_7', 'pixel_3_0', 'pixel_3_1', 'pixel_3_2', 'pixel_3_3', 'pixel_3_4', 'pixe
l_3_5', 'pixel_3_6', 'pixel_3_7', 'pixel_4_0', 'pixel_4_1', 'pixel_4_2', 'pix
el_4_3', 'pixel_4_4', 'pixel_4_5', 'pixel_4_6', 'pixel_4_7', 'pixel_5_0', 'pi
xel_5_1', 'pixel_5_2', 'pixel_5_3', 'pixel_5_4', 'pixel_5_5', 'pixel_5_6', 'p
ixel_5_7', 'pixel_6_0', 'pixel_6_1', 'pixel_6_2', 'pixel_6_3', 'pixel_6_4',
'pixel_6_5', 'pixel_6_6', 'pixel_6_7', 'pixel_7_0', 'pixel_7_1', 'pixel_7_2',
'pixel_7_3', 'pixel_7_4', 'pixel_7_5', 'pixel_7_6', 'pixel_7_7']

Target Classes:
----------------
[0 1 2 3 4 5 6 7 8 9]
[ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
  0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
  0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]
```




```
0
(1797, 64) (1797,)
```

In [2]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25)

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(1347, 64) (1347,)
(450, 64) (450,)
```

Step 2: Build the model

- Import the class SVC from sklearn.svm module
- Create a model objet using SVC class
- Use fit method to train the model with training dataset.

In [3]:
```
from sklearn.svm import SVC

model = SVC()
model.fit(x_train, y_train)
```

Out[3]:
```
SVC()
```

Step 3: Test and Evaluate the model.

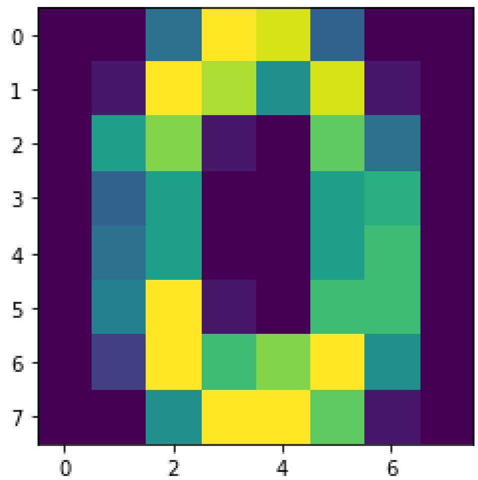
- Use predict method to predict the class for unseen data samples.
- Import accuracy_score function from sklearn.metrics module for evaluation

In [4]:
```
from sklearn.metrics import accuracy_score
y_pred = model.predict(x_test)
print(accuracy_score(y_test, y_pred))
```

```
0.9911111111111112
```

In [5]:
```
plt.imshow(x_test[0].reshape((8, 8)))
print(model.predict(x_test[0].reshape(1,-1)))
```

```
[0]
```



Step 4: Evaluate the model with different kernels

- SVC class takes a parameter kernel which can take one of linear / poly / rbf / sigmoid.
- rbf (Radial Basis Function) is the default value.

In [6]:
```
model1 = SVC(kernel = "linear")
model1.fit(x_train, y_train)
y1_pred = model1.predict(x_test)
print(accuracy_score(y_test, y1_pred))
```

```
0.9822222222222222
```

In [7]:
```
model2 = SVC(kernel = "poly")
model2.fit(x_train, y_train)
y2_pred = model2.predict(x_test)
print(accuracy_score(y_test, y2_pred))
```

```
0.9955555555555555
```

In [10]:
```
model3 = SVC(kernel = "sigmoid")
model3.fit(x_train, y_train)
y3_pred = model3.predict(x_test)
print(accuracy_score(y_test, y3_pred))
```

```
0.9133333333333333
```