In [7]:
```
import pandas as pd
import seaborn as sns
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
```

In [3]:
```
ram=load_digits()
```

In [4]:
```
dir(ram) # the model contain data and images as well
```

Out[4]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_names']
```

In [5]:
```
ram.data[0] # this is the data of first image
```

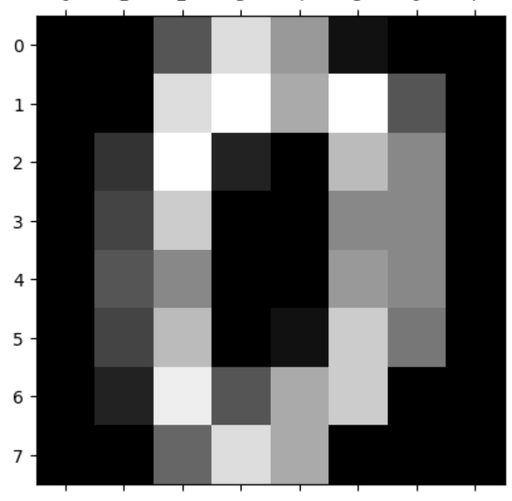
Out[5]:
```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

In [6]:
```
ram.images[0]
```

Out[6]:
```
array([[ 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.]])
```

In [9]:
```
plt.gray()
plt.matshow(ram.images[0])  # here we print the first image
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x26dc5218e10>
<Figure size 640x480 with 0 Axes>
```



In [10]:
```
from sklearn.model_selection import train_test_split
```

In [18]:
```
x_train,x_test,y_train,y_test=train_test_split(ram.data,ram.target,test_size=0.2)
# here is the good concept to learn the data is the numpy array and target is the predicted value
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
reg=LogisticRegression () # here is the model of our own
```

In [21]:
```
reg.fit(x_train,y_train)
```

```
C:\Users\Asus\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[21]:
```
LogisticRegression
LogisticRegression()
```

In [22]:
```
reg.predict(x_test)
```

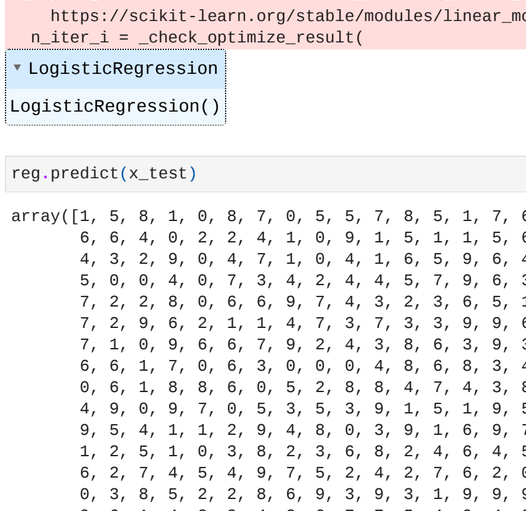
Out[22]:
```
array([1, 5, 8, 1, 0, 8, 7, 0, 5, 5, 7, 8, 5, 1, 7, 6, 7, 7, 6, 9, 8, 7,
       6, 6, 4, 0, 2, 2, 4, 1, 0, 9, 1, 5, 1, 1, 5, 6, 6, 7, 1, 0, 5, 9,
       4, 3, 2, 9, 0, 4, 7, 1, 0, 4, 1, 6, 5, 9, 6, 4, 1, 2, 5, 5, 3, 7,
       5, 0, 0, 4, 0, 7, 3, 4, 2, 4, 4, 5, 7, 9, 6, 3, 9, 7, 1, 3, 4, 5,
       7, 2, 2, 8, 0, 6, 6, 9, 7, 4, 3, 2, 3, 6, 5, 1, 8, 2, 3, 9, 4, 0,
       7, 2, 9, 6, 2, 1, 1, 4, 7, 3, 7, 3, 3, 9, 9, 6, 8, 5, 3, 1, 7, 0,
       7, 1, 0, 9, 6, 6, 7, 9, 2, 4, 3, 8, 6, 3, 9, 3, 6, 4, 7, 7, 8, 9,
       6, 6, 1, 7, 0, 6, 3, 0, 0, 0, 4, 8, 6, 8, 3, 4, 5, 8, 2, 3, 1, 6,
       0, 6, 1, 8, 8, 6, 0, 5, 2, 8, 8, 4, 7, 4, 3, 8, 5, 5, 6, 5, 1, 9,
       4, 9, 0, 9, 7, 0, 5, 3, 5, 3, 9, 1, 5, 1, 9, 5, 6, 1, 3, 7, 8, 7,
       9, 5, 4, 1, 1, 2, 9, 4, 8, 0, 3, 9, 1, 6, 9, 7, 0, 0, 5, 5, 9, 3,
       1, 2, 5, 1, 0, 3, 8, 2, 3, 6, 8, 2, 4, 6, 4, 5, 1, 5, 1, 2, 0, 2,
       6, 2, 7, 4, 5, 4, 9, 7, 5, 2, 4, 2, 7, 6, 2, 0, 2, 9, 0, 6, 6, 4,
       0, 3, 8, 5, 2, 2, 8, 6, 9, 3, 9, 3, 1, 9, 9, 9, 2, 8, 5, 9, 0, 5,
       9, 6, 1, 4, 2, 8, 4, 2, 6, 7, 7, 5, 4, 0, 4, 0, 0, 5, 9, 4, 1, 6,
       4, 1, 8, 2, 0, 1, 5, 6, 0, 2, 7, 8, 5, 2, 9, 4, 9, 1, 1, 7, 4, 7,
       8, 8, 3, 4, 4, 1, 5, 2])
```

In [24]:
```
reg.score(x_test,y_test)
```

Out[24]:
```
0.9444444444444444
```

In [26]:
```
plt.matshow(ram.images[67])   # this is 67th image
```

Out[26]:
```
<matplotlib.image.AxesImage at 0x26dc7b59690>
```



In [27]:
```
ram.target[67]
```

Out[27]:
```
6
```

In [28]:
```
# now we need to predict from our model
```

In [30]:
```
reg.predict([ram.data[67]]) # here we predict from our model
```

Out[30]:
```
array([6])
```

In [33]:
```
reg.predict(ram.data[0:5]) # predict the first five images
```

Out[33]:
```
array([0, 1, 2, 3, 4])
```

In [36]:
```
from sklearn.metrics import confusion_matrix
```

In [38]:
```
ypredicted=reg.predict(x_test) # this is the predict value by model
#y_test # this is the predefine value
```

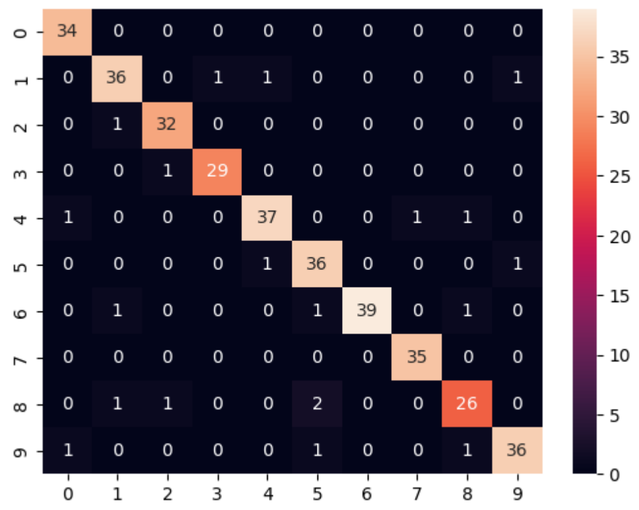
In [39]:
```
cm=confusion_matrix(y_test,ypredicted)
```

In [40]:
```
cm
```

Out[40]:
```
array([[34,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 36,  0,  1,  1,  0,  0,  0,  0,  1],
       [ 0,  1, 32,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, 29,  0,  0,  0,  0,  0,  0],
       [ 1,  0,  0,  0, 37,  0,  0,  1,  1,  0],
       [ 0,  0,  0,  0,  1, 36,  0,  0,  0,  1],
       [ 0,  1,  0,  0,  0,  1, 39,  0,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 35,  0,  0],
       [ 0,  1,  1,  0,  0,  2,  0,  0, 26,  0],
       [ 1,  0,  0,  0,  0,  1,  0,  0,  1, 36]], dtype=int64)
```

In [42]:
```
sns.heatmap(cm,annot=True)  # annot act as datalabel
# 34 times the the actual value was 0 but the model predicted as 0
# if we see not zero in two black side then there is error.
```

Out[42]:
```
<AxesSubplot: >
```


0 1 2 3 4 5 6 7 8 9
34 0 0 0 0 0 0 0 0 0
0 36 0 1 1 0 0 0 0 1
0 1 32 0 0 0 0 0 0 0
0 0 1 29 0 0 0 0 0 0
1 0 0 0 37 0 0 1 1 0
0 0 0 0 1 36 0 0 0 1
0 1 0 0 0 1 39 0 1 0
0 0 0 0 0 0 0 35 0 0
0 1 1 0 0 2 0 0 26 0
1 0 0 0 0 1 0 0 1 36
35 30 25 20 15 10 5 0

In [ ]: