In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_digits #we load digits datasets
```

In [2]:

```
digits= load_digits() # load digits function
```

In [3]:

```
dir(digits)
```

Out[3]:

```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_name
s']
```

In [4]:

```
digits.data[0]
```

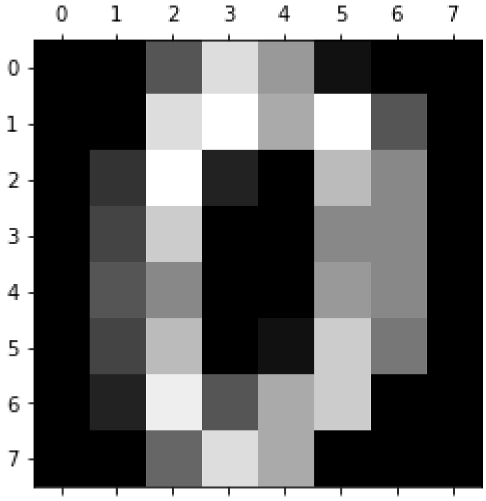
Out[4]:

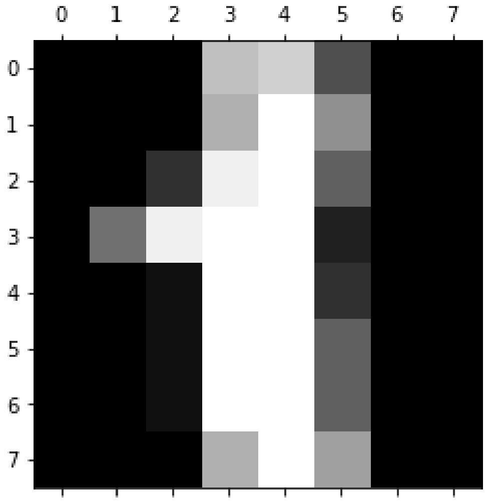
```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10.,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

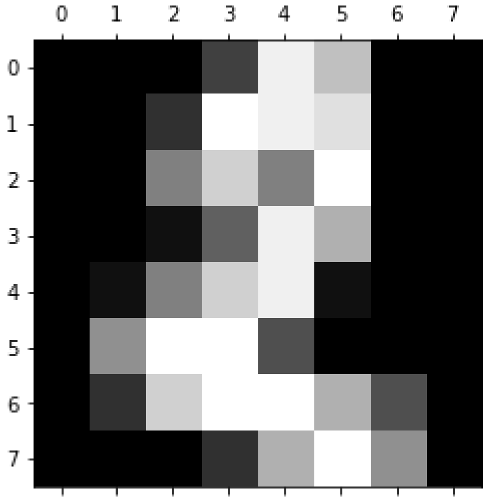
In [5]:

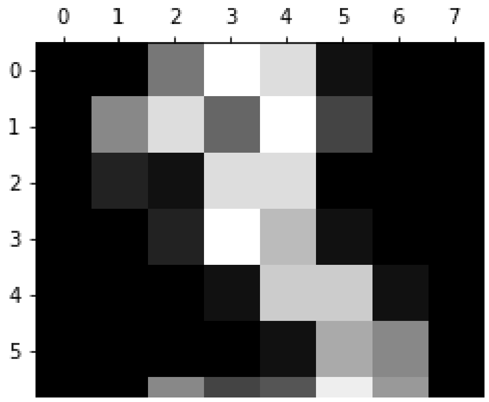
```
plt.gray()
for i in range(5):
    plt.matshow(digits.images[i])
```

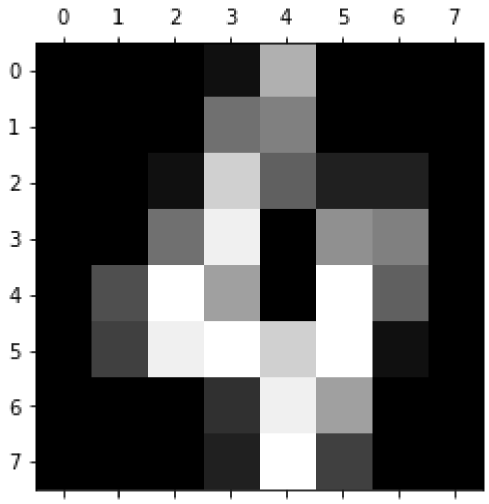
```
<Figure size 432x288 with 0 Axes>
```











In [6]:

```
digits.target[0:5]
```

Out[6]:

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

In [7]:

```
from sklearn.model_selection import train_test_split
```

In [8]:

```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.2)
```

In [9]:

```
len(x_train)
```

Out[9]:

```
1437
```

In [10]:

```
len(x_test)
```

Out[10]:

```
360
```

In [11]:

```
from sklearn.linear_model import LogisticRegression
```

In [12]:

```
model = LogisticRegression()
```

In [13]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:763: 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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[13]:

```
LogisticRegression()
```

In [14]:

```
model.score(x_test, y_test)
```

Out[14]:

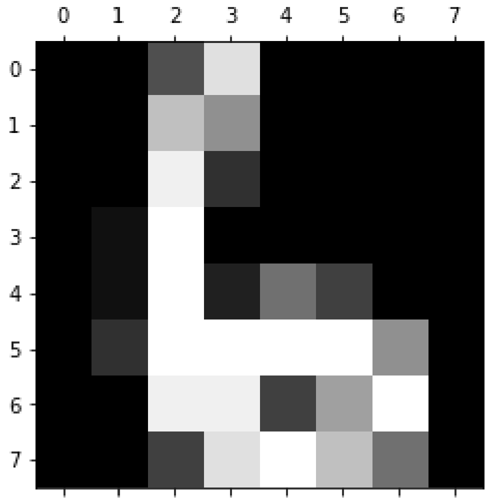
```
0.9722222222222222
```

In [15]:

```
plt.matshow(digits.images[67])
```

Out[15]:

```
<matplotlib.image.AxesImage at 0x1c3aaee57f0>
```



In [16]:

```
digits.target[67]
```

Out[16]:

```
6
```

In [17]:

```
model.predict([digits.data[67]])
```

Out[17]:

```
array([6])
```

In [18]:

```
model.predict(digits.data[0:5])
```

Out[18]:

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

In [19]:

```
#confusion matrix
# it is way visualising how well our model is working or evaluating
```

In [21]:

```
y_predicted = model.predict(x_test)
```

In [23]:

```
from sklearn.metrics import confusion_matrix
```

In [25]:

```
cm= confusion_matrix(y_test,y_predicted)
print(cm)
```

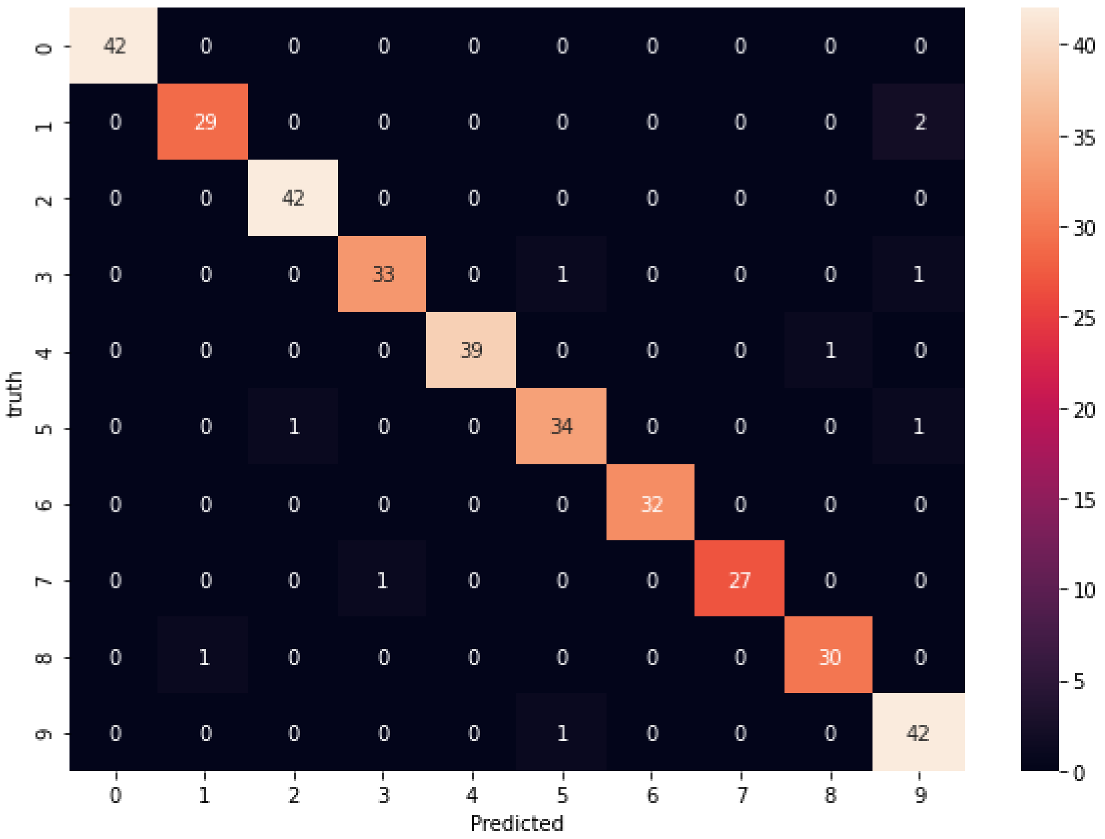
```
[[42  0  0  0  0  0  0  0  0  0]
 [ 0 29  0  0  0  0  0  0  0  2]
 [ 0  0 42  0  0  0  0  0  0  0]
 [ 0  0  0 33  0  1  0  0  0  1]
 [ 0  0  0  0 39  0  0  0  1  0]
 [ 0  0  1  0  0 34  0  0  0  1]
 [ 0  0  0  0  0  0 32  0  0  0]
 [ 0  0  0  1  0  0  0 27  0  0]
 [ 0  1  0  0  0  0  0  0 30  0]
 [ 0  0  0  0  0  1  0  0  0 42]]
```

In [28]:

```
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm,annot =True)#it forms a heatmap table
plt.xlabel('Predicted')
plt.ylabel('truth')
```

Out[28]:

```
Text(69.0, 0.5, 'truth')
```



	0	1	2	3	4	5	6	7	8	9
0	42	0	0	0	0	0	0	0	0	0
1	0	29	0	0	0	0	0	0	0	2
2	0	0	42	0	0	0	0	0	0	0
3	0	0	0	33	0	1	0	0	0	1
4	0	0	0	0	39	0	0	0	1	0
5	0	0	1	0	0	34	0	0	0	1
6	0	0	0	0	0	0	32	0	0	0
7	0	0	0	1	0	0	0	27	0	0
8	0	1	0	0	0	0	0	0	30	0
9	0	0	0	0	0	1	0	0	0	42