In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn import metrics
```

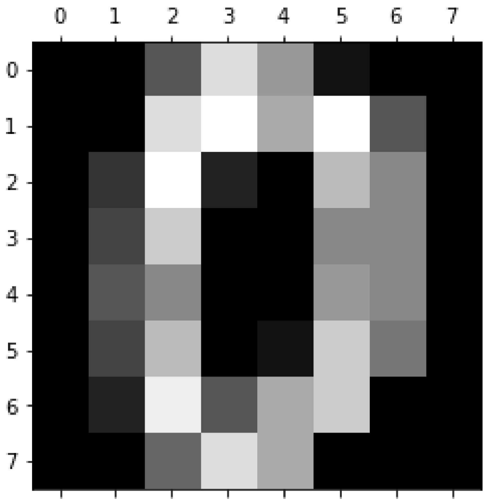
In [5]:

```
# 1.Scikit learn comes with pre-loaded dataset, load the digits dataset from t
hat collection
# and write a helper function to plot the image using matplotlib.
digits = load_digits()
plt.gray()
plt.matshow(digits.images[0])
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x13dc4431b88>
```

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



In [11]:

```
images = digits.images.reshape(digits.images.shape[0], -1)
labels = digits.target
```

In [22]:

```
# 2.Make a train -test split with 20% of the data set aside for testing.
# Fit a logistic regression model and observe the accuracy.
x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size=
0.20, random_state = 10)

log_model = LogisticRegression()
log_model.fit(x_train, y_train)

predicted_value = log_model.predict(x_test)
acc_score = metrics.accuracy_score(predicted_value, y_test)*100
print(acc_score)
```

```
95.0

C:\Users\hp\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: 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-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [18]:

```
# 3.Using scikit learn perform a PCA transformation such that the transformed
 dataset
# can explain 95% of the variance in the original dataset. Find out the number
of
# components in the projected subspace.

pca_model = PCA(n_components = 10)     # To get 95% of variance n_components s
hould be 10
pca_model.fit(images)

#transforming the data
pca_model.fit(x_train, y_train)
x_train = pca_model.transform(x_train)
x_test = pca_model.transform(x_test)

print("Variance Ratio\n")
print(pca_model.explained_variance_ratio_)
```

```
Variance Ratio

[0.14659533 0.13563025 0.11901049 0.08581575 0.05884243 0.04888265
 0.04350547 0.03691069 0.0333279  0.03062844]
```

In [21]:

```
# 4.Transform the dataset and fit a logistic regression and observe the accura
cy.
# Compare it with the previous model andcomment on the accuracy.
log_model_1 = LogisticRegression()
log_model.fit(x_train, y_train)

predicted_data = log_model.predict(x_test)

print("Accuracy Score\n")
print(metrics.accuracy_score(predicted_data, y_test)*100)
```

```
Accuracy Score

93.05555555555556

C:\Users\hp\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: 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-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [23]:

```
# 5.Compute the confusion matrix and count the number of instances that has go
ne wrong.
# For each of the wrong sample,plot the digit along with predicted and origina
l label.
conf_metrics = metrics.confusion_matrix(predicted_data, y_test)
print("Confusion Metrices\n")
print(conf_metrics)
```

```
Confusion Metrices

[[36  0  0  0  1  0  0  0  0  0]
 [ 0 31  0  0  0  0  1  0  1  2]
 [ 0  0 34  0  0  0  0  0  2  0]
 [ 0  0  0 38  0  1  0  0  0  0]
 [ 1  1  0  0 31  0  0  1  0  0]
 [ 0  2  0  0  0 29  0  0  0  2]
 [ 0  0  0  0  0  0 36  0  0  0]
 [ 0  0  0  0  0  0  0 37  0  0]
 [ 0  0  0  1  0  0  0  1 30  2]
 [ 0  0  0  1  2  2  0  1  0 33]]
```

In [25]:

```
classification_report = metrics.classification_report(predicted_data, y_test)
print("Classification Report\n")
print(classification_report)
```

```
Classification Report

              precision    recall  f1-score   support

           0       0.97      0.97      0.97        37
           1       0.91      0.89      0.90        35
           2       1.00      0.94      0.97        36
           3       0.95      0.97      0.96        39
           4       0.91      0.91      0.91        34
           5       0.91      0.88      0.89        33
           6       0.97      1.00      0.99        36
           7       0.93      1.00      0.96        37
           8       0.91      0.88      0.90        34
           9       0.85      0.85      0.85        39

    accuracy                           0.93       360
   macro avg       0.93      0.93      0.93       360
weighted avg       0.93      0.93      0.93       360
```

In [ ]:

```
```