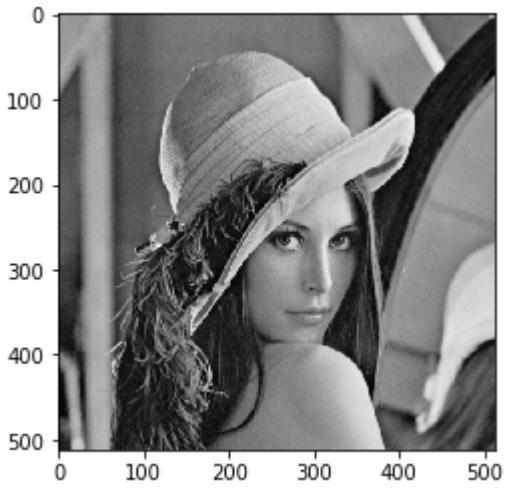
In [1]:
```
import sys

import numpy as np
import cv2
import math
import cv2
import matplotlib.pyplot as plt
import numpy as np
import sys
img = cv2.imread('lena.bmp', 0)
```

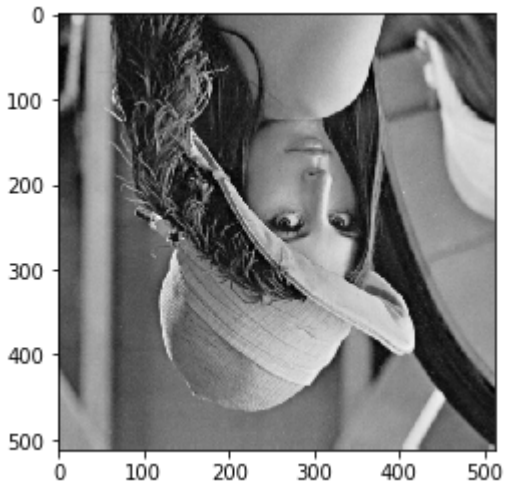
原圖

In [2]:
```
%matplotlib inline
plt.imshow(img,cmap='gray')
plt.show()
```



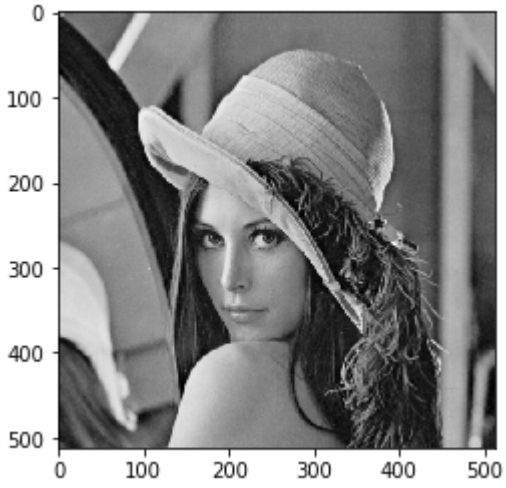
(a)利用 numpy slicing的特性對第一維度reverse 即可 upside_down

In [3]:
```
def upside_down(img):
    return img[::-1,:]
plt.imshow(upside_down(img),cmap='gray')
plt.show()
```



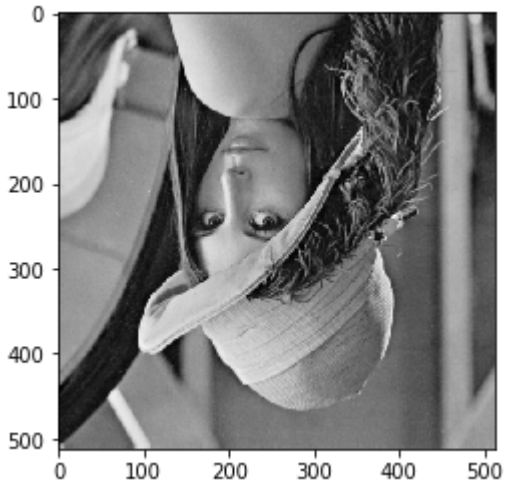
(b)利用 numpy slicing的特性對第二維度reverse 即可 right_side_left

In [4]:
```
def right_side_left (img):
    return img[:,::-1]
plt.imshow(right_side_left (img),cmap='gray')
plt.show()
```



(c)利用 numpy slicing的特性同時對第 一 二 維度reverse 即可 diagonal_mirror

In [5]:
```
def diagonal_mirror(img):
    return right_side_left(upside_down(img))
plt.imshow(diagonal_mirror (img),cmap='gray')
plt.show()
```



(d)rotate 前需要預留新的長高，使旋轉後不會被切到

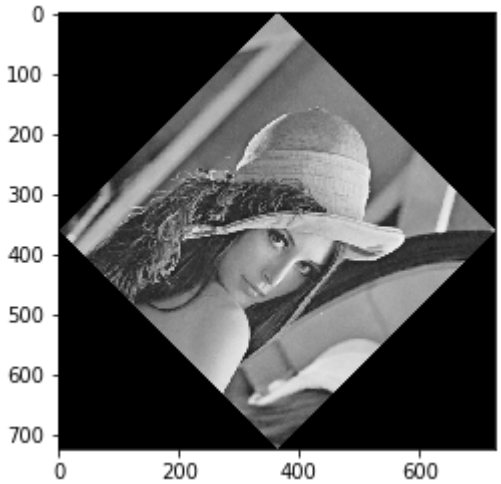
rotate 需要從原始座標warping到目標座標，我們可以將旋轉分成兩部分

1)旋轉 根據角度跟新的長高做出rotate_matrix

2)平移 根據新的長高做平移

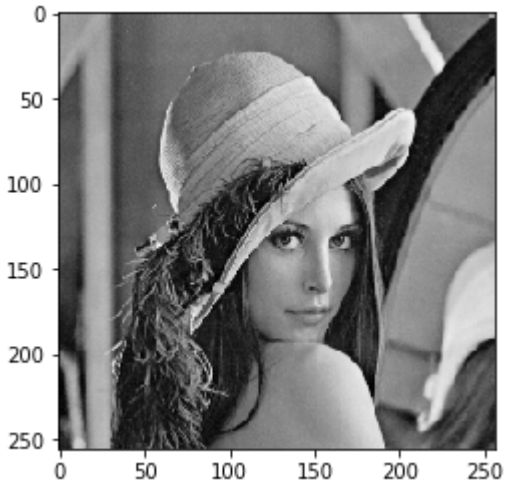
實際 做的時候為了避免rounding問題 所以需要逆著從目標座標warping到原始座標

In [6]:
```
def rotate_matrix(cx, cy):
    theta = np.radians(-45)
    c, s = np.cos(theta), np.sin(theta)
    return np.array([[ c, s,  (1-c)*cx-s*cy],
                     [-s, c,  s*cx+(1-c)*cy]
                     ,[0,0,1]])


def warping(image, M, nW, nH,cX, cY):
    dst_image=np.zeros((nW, nH))
    for i in range(nW):
        for j in range(nH):
            A=np.array([[M[0,0],M[0,1]],
                        [M[1,0],M[1,1]]])
            B=np.array([i-M[0,2],j-M[1,2]]).reshape(2, 1)
            ans = A.dot(B)
            x=int(ans[0])
            y=int(ans[1])
            if x<0 and x>-image.shape[0]  and y>0 and y<image.shape[1
]:
                    dst_image[i][j]=image[x][y]
    return dst_image

def rotate(image):
    (h, w) = image.shape
    (cX, cY) = (w // 2, h // 2)
    M=rotate_matrix(cX,cY)
    cos = np.abs(M[0, 0])
    sin = np.abs(M[0, 1])
    nW = int((h * sin) + (w * cos))
    nH = int((h * cos) + (w * sin))
    M[0, 2] += (nW / 2) - cX
    M[1, 2] += (nH / 2) - cY
    return warping(image, M, nW, nH,cX, cY)
plt.imshow(rotate (img),cmap='gray')
plt.show()
```



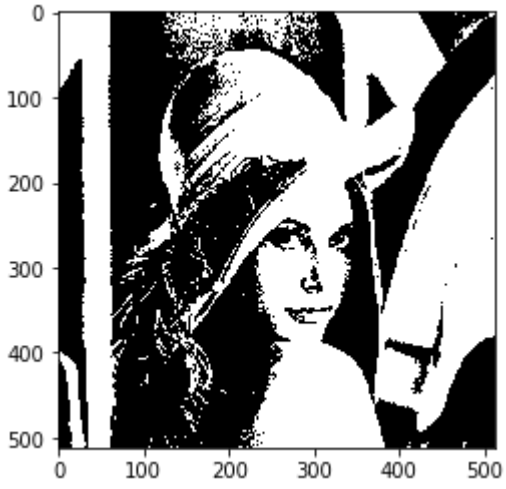
(e)利用 numpy slicing的特性同時對第 一 二 維度跳一格取 即可 shrink

In [7]:
```
def shrink(image):
    return image[::2,::2]
plt.imshow(shrink (img),cmap='gray')
plt.show()
```



(e)利用 判斷式判斷是否>128 變0 1在同乘255變成grayscale binarize

In [8]:
```
def binarize(image):
    return (img>128)*255
plt.imshow(binarize (img),cmap='gray')
plt.show()
```



In [9]:
```
cv2.imwrite('a_upside_down.bmp', upside_down(img))
cv2.imwrite('b_right_side_left.bmp', right_side_left(img))
cv2.imwrite('c_diagonal_mirror.bmp', diagonal_mirror(img))
cv2.imwrite('d_rotate.bmp', rotate(img))
cv2.imwrite('e_shrink.bmp', shrink(img))
cv2.imwrite('f_binarize.bmp', binarize(img))
```

Out[9]: True