

# 第15章

## 輪廓的檢測與匹配

## 15.1：影像內圖形的輪廓

- 15.1.1：找尋圖形輪廓**findContours()**
- `contours, hierarchy = cv2.findContours(image, mode, method)`

具名常數	值	說明
<b>RETR_EXTERNAL</b>	0	只檢測外部輪廓
<b>RETR_LIST</b>	1	檢測所有輪廓，但是不建立層次關係
<b>RETR_CCOMP</b>	2	檢測所有輪廓，同時建立兩個層級關係，如果內部還有輪廓則此輪廓與最外層的輪廓同級
<b>RETR_TREE</b>	3	檢測所有輪廓，同時建立一個樹狀層級關係

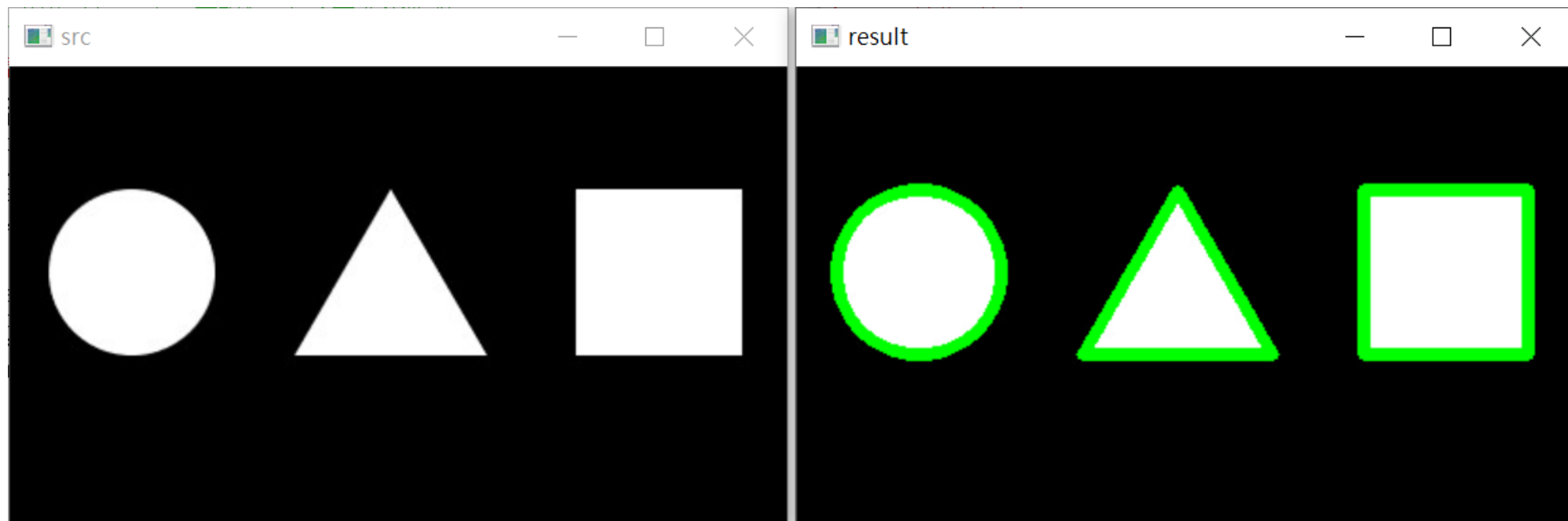
具名常數	值	說明
<b>CHAIN_APPROX_NONE</b>	1	儲存所有輪廓點
<b>CHAIN_APPROX_SIMPLE</b>	2	只有保存輪廓頂點的座標
<b>CHAIN_APPROX_TC89L1</b>	3	使用 Teh-Chin1 chain 近似的演算法
<b>CHAIN_APPROX_TC89KCOS</b>	4	使用 Teh-Chin1 chain 近似的演算法

## 15.1.2：繪製圖形的輪廓

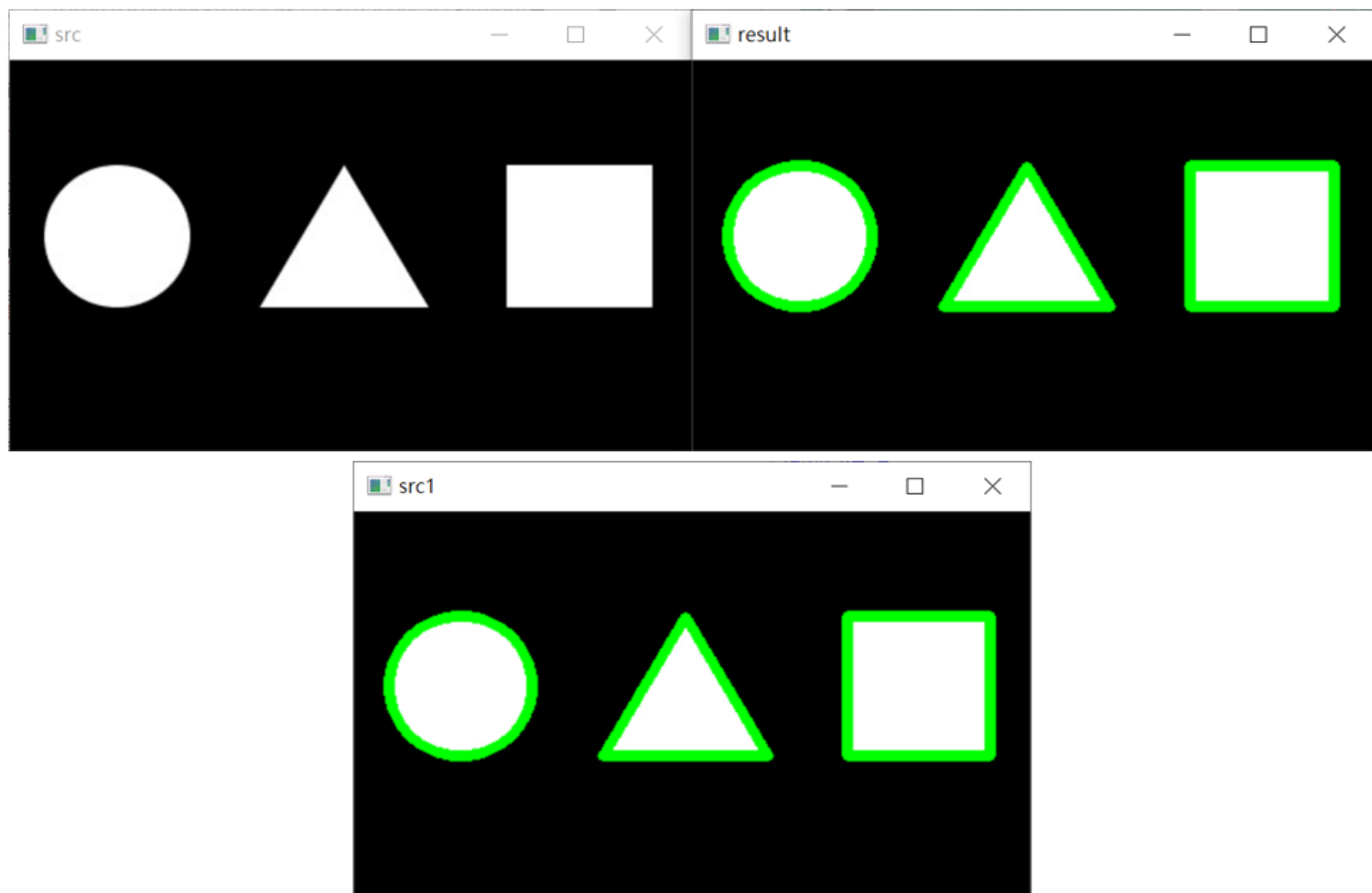
- `image = cv2.drawContours(src_image, contours, contourIdx, color, thickness, lineType, hierarchy, maxLevel, offset)`

## 15.2：繪製影像內圖形輪廓的系列實例

- **15.2.1：找尋與繪製影像內圖形輪廓的基本應用**
- 1：使用`cv2.cvtColor( )`函數，將影像轉成灰階，讀者可以參考下列程式第7列。
- 2：使用`threshold( )`函數將灰階影像二值化，如果沒有特別考量，可以將中間值127當作閾值，讀者可以參考下列程式第9列。
- 程式實例`ch15_1.py`：在影像`easy.jpg`內，輪廓檢測模式使用`RETR_EXTERNAL`，找尋圖形輪廓的應用，最後使用綠色繪製此輪廓。



- 程式實例ch15\_1\_1.py：擴充設計ch15\_1.py，再輸出原始影像一次，觀察是否影響原始影像，讀者可以參考第16列。



## 15.2.2：認識findContours( )函數的回傳值 contours

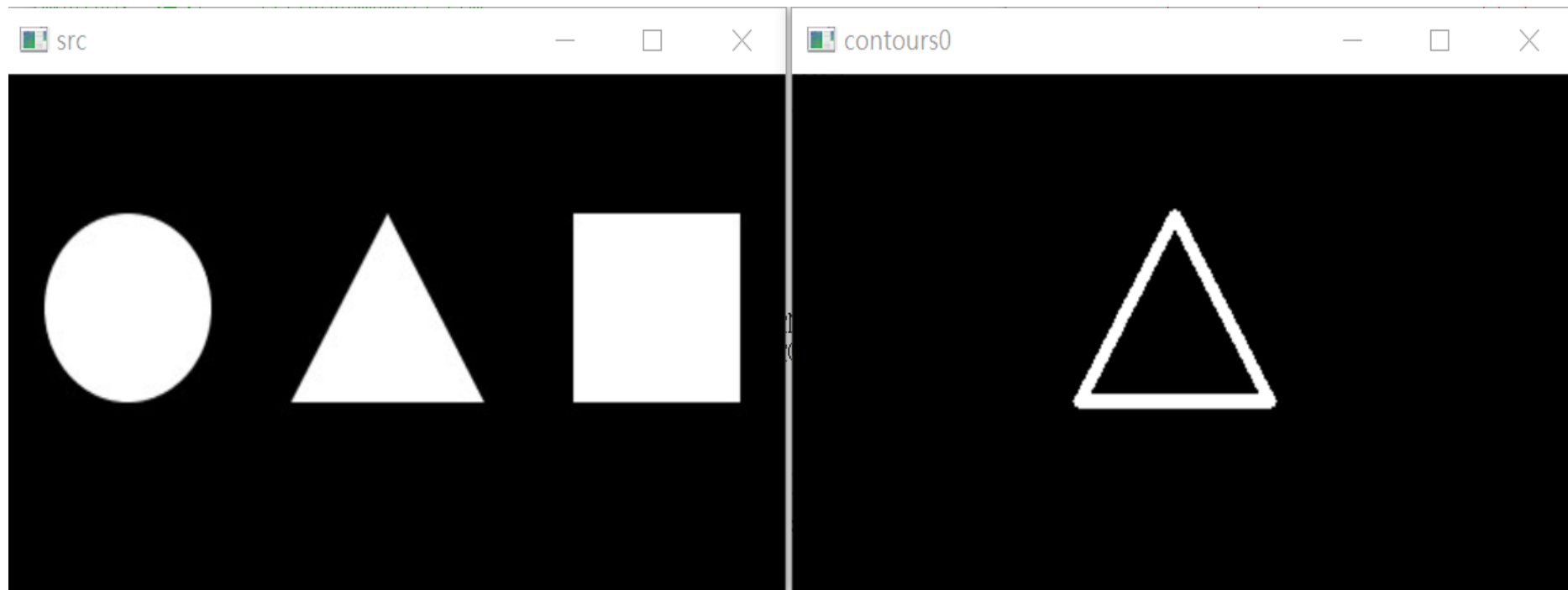
- 程式實例ch15\_2.py：使用easy.jpg列出所回傳contours的資料類型和長度。

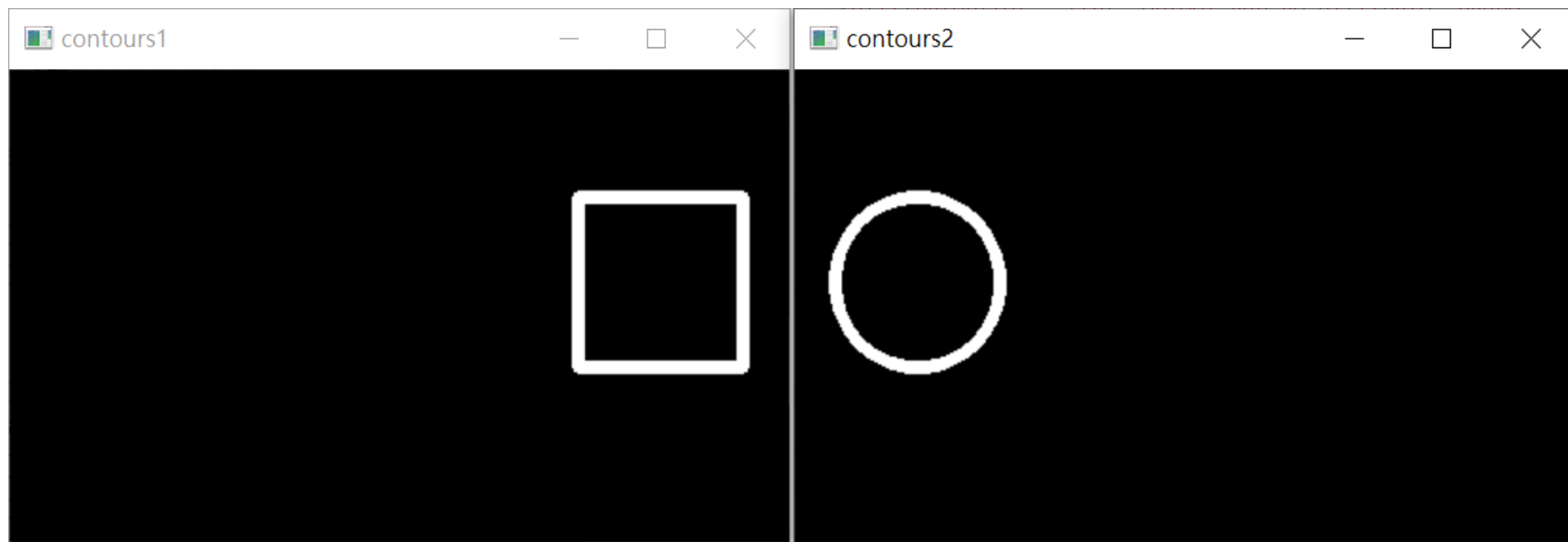
```
===== RESTART: D:/OpenCV_Python/ch15/ch15_2.py =====  
資料類型      : <class 'list'>  
輪廓數量      : 3
```



## 15.2.3：輪廓索引|contoursIdx

- 程式實例ch15\_3.py：重新設計ch15\_1.py，但是分別繪製影像內的圖形輪廓。





## 15.24：認識輪廓的屬性

- 程式實例ch15\_4.py：列出影像內圖片輪廓的屬性。

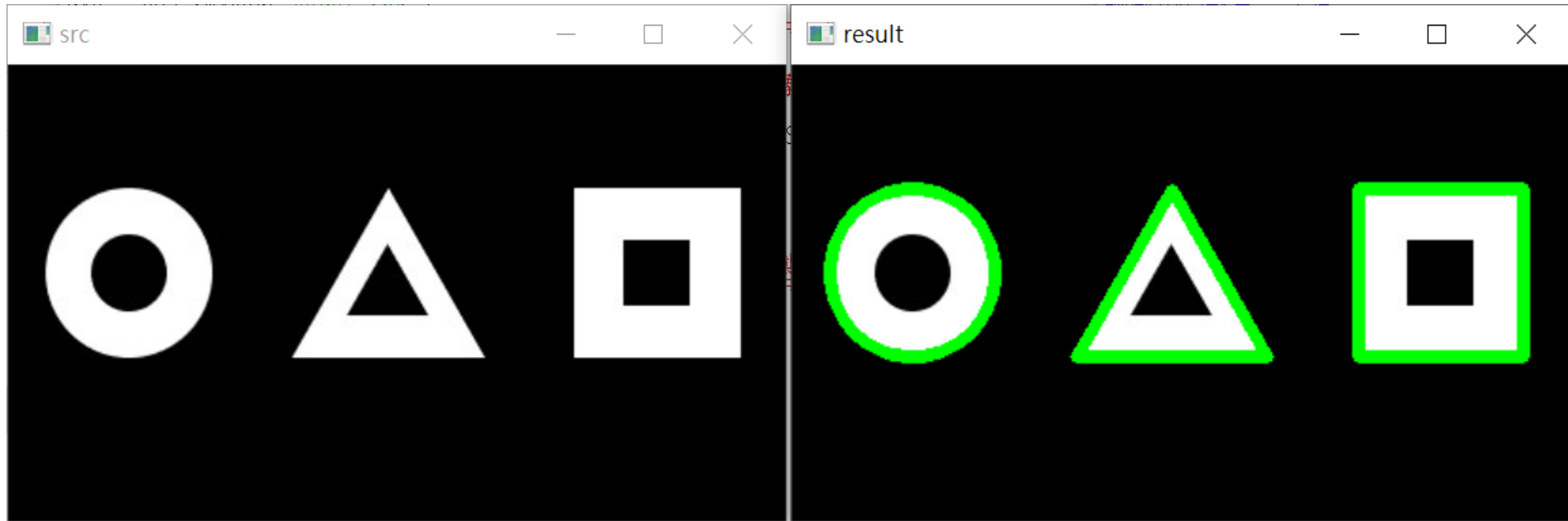
```
===== RESTART: D:/OpenCV_Python/ch15/ch15_4.py =====  
編號 = 0  
輪廓點的數量 = 146  
輪廓點的外形 = (146, 1, 2)  
編號 = 1  
輪廓點的數量 = 4  
輪廓點的外形 = (4, 1, 2)  
編號 = 2  
輪廓點的數量 = 134  
輪廓點的外形 = (134, 1, 2)
```

- 實例ch15\_5.py：列出編號1的輪廓點內容。

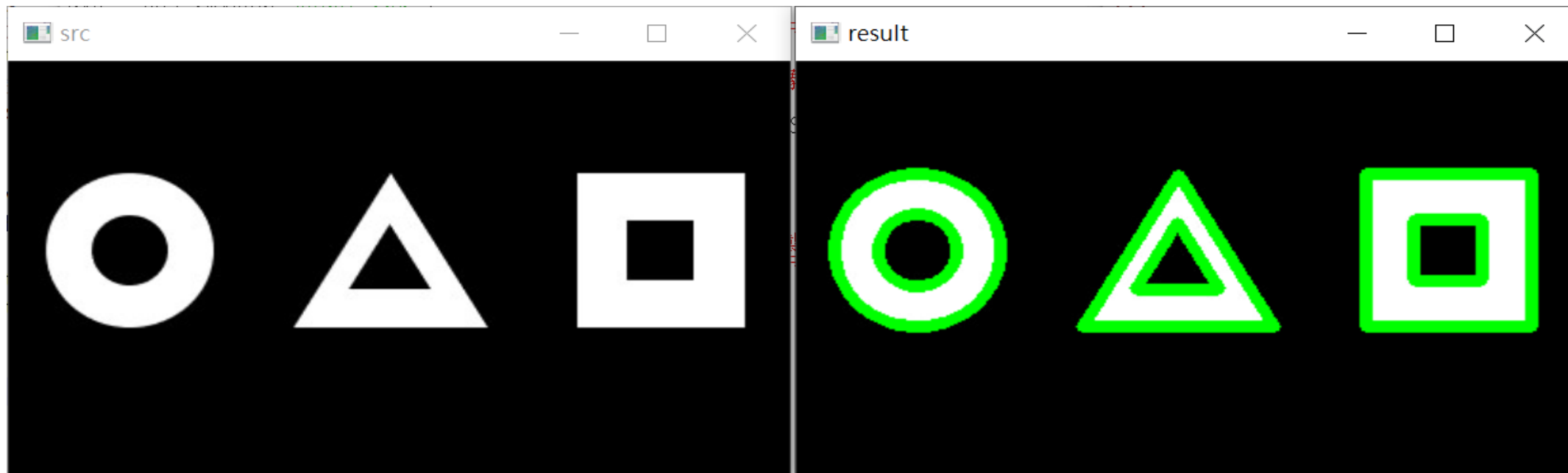
```
===== RESTART: D:/OpenCV_Python/ch15/ch15_5.py =====  
[[[300  65]]  
 [[300 152]]  
 [[387 152]]  
 [[387  65]]]
```

## 15.25：輪廓內有輪廓

- 程式實例ch15\_6.py：使用easy1.jpg影像，重新設計ch15\_1.py。

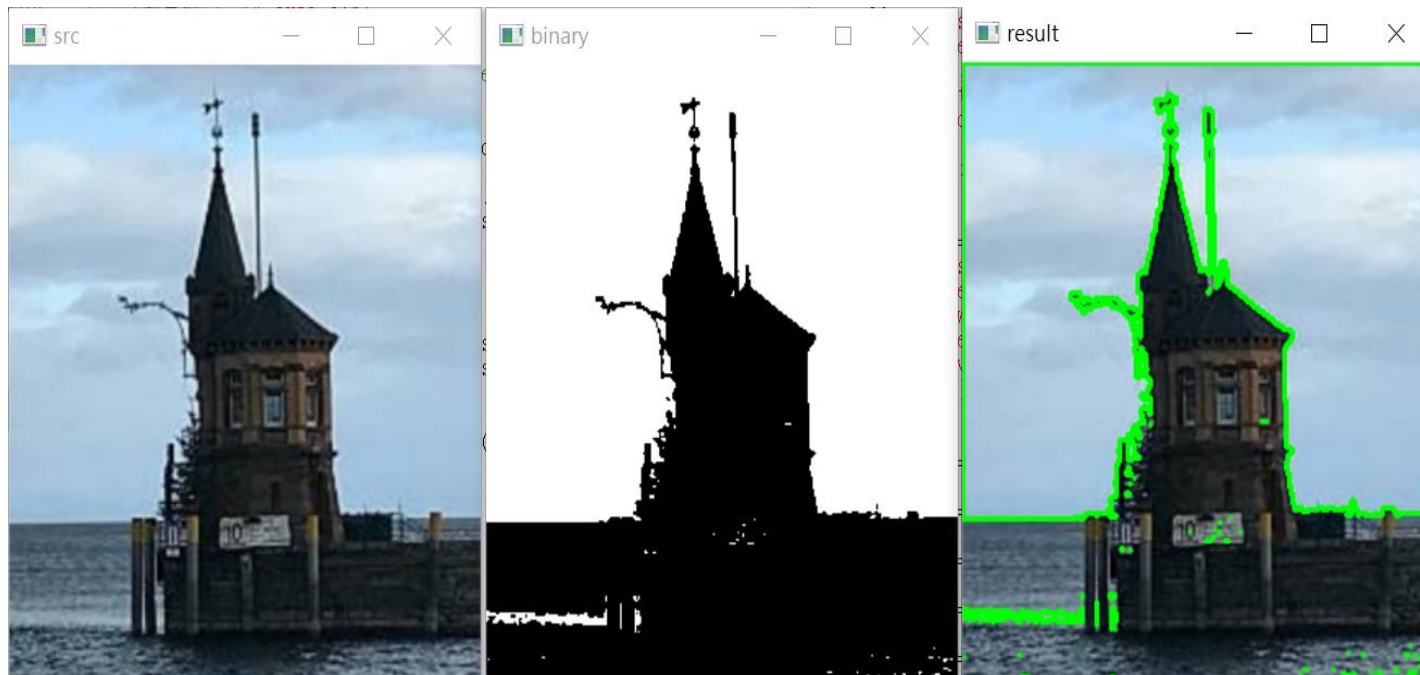


- 程式實例 `ch15_7.py`：重新設計 `ch15_6.py`，使用 `RETR_EXTERNAL` 檢測所有輪廓。

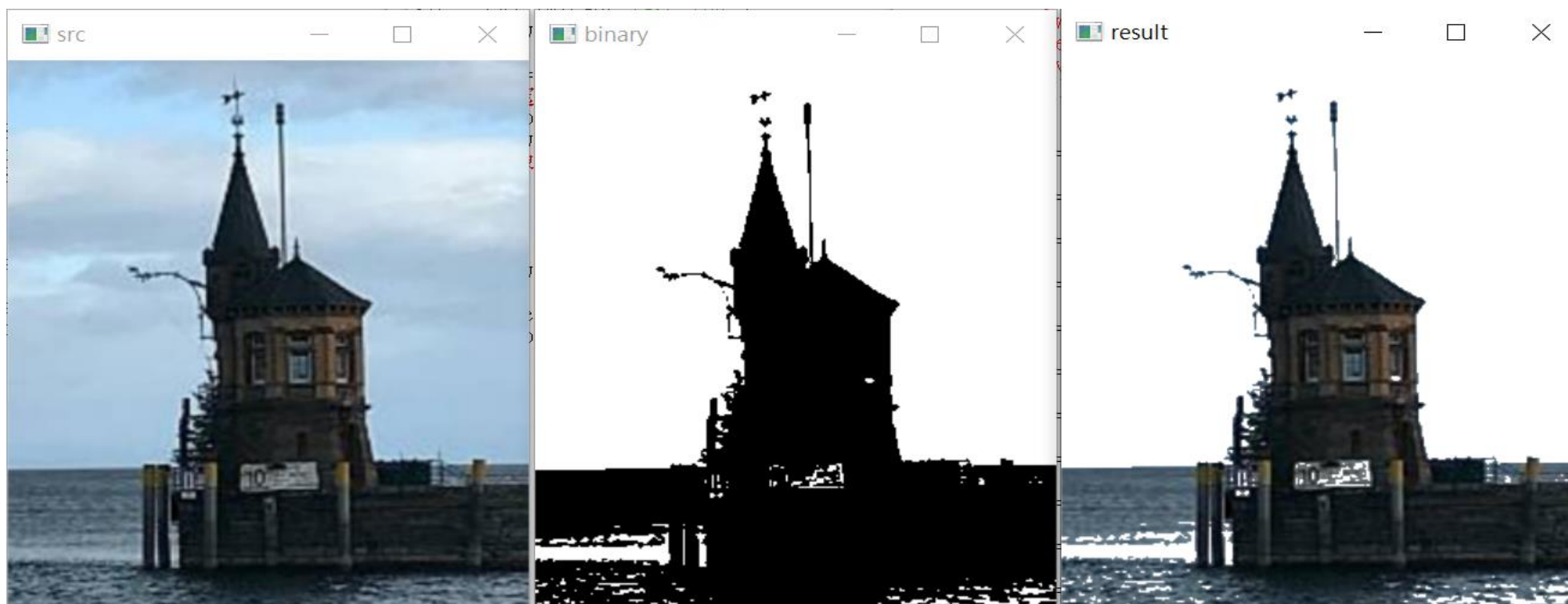


## 15.26：繪製一般影像的圖形輪廓

- 程式實例ch15\_8.py：繪製一般影像的輪廓，這個程式顯示原始影像、二元影像和繪製輪廓。

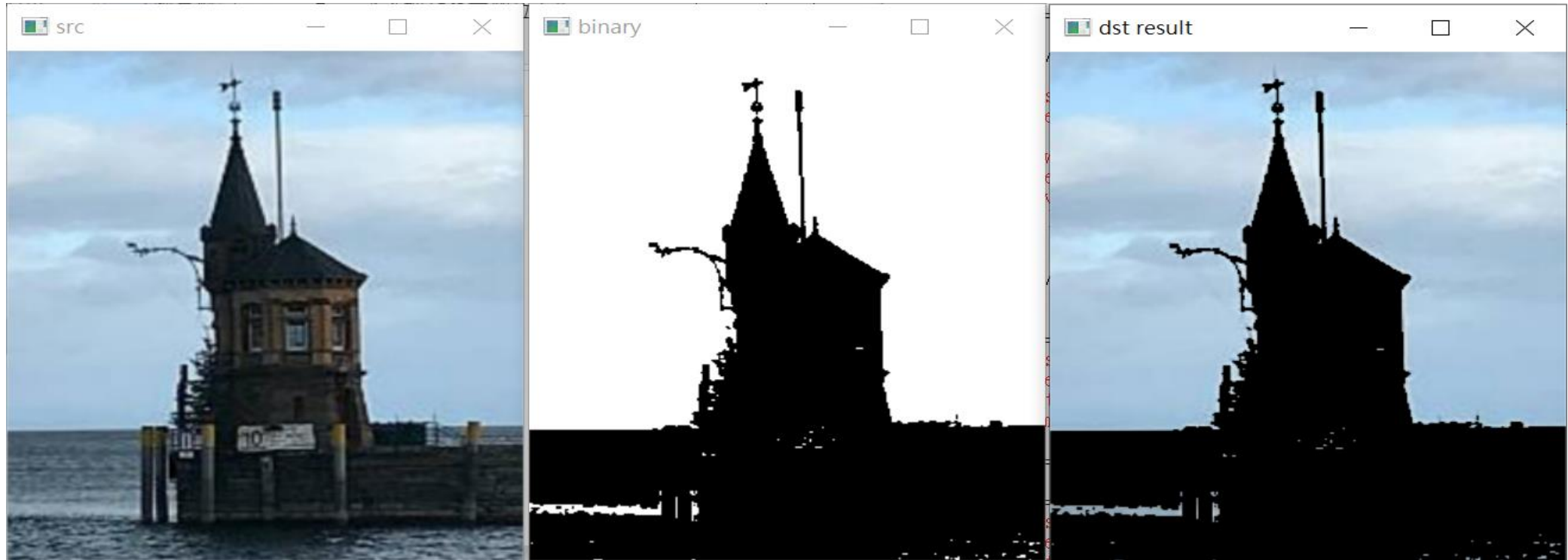


- 程式實例ch15\_9.py：重新設計ch15\_8.py，清除天空背景，也可以視為將圖案取出。



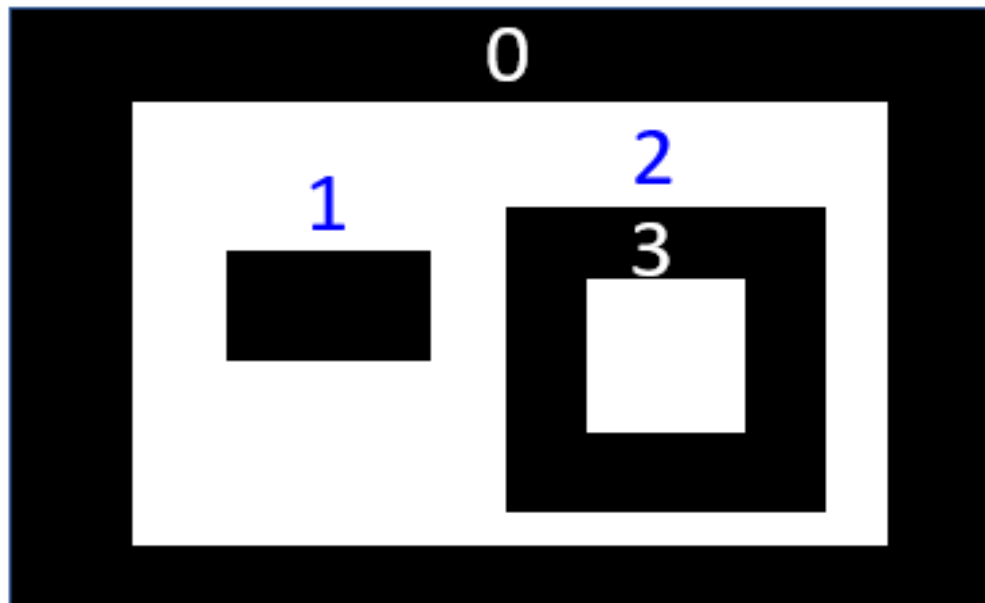


- 程式實例ch15\_10.py：重新設計ch15\_9.py，天空背景保留，但是將燈塔以黑色顯示。



## 15.3：認識輪廓層級hierarchy

- `contours, hierarchy = cv2.findContours(image, mode, method)`
- 這一小節將解說回傳的多維陣列資料型態 `hierarchy`，我們可以翻譯為層級。



- **父輪廓**：如果一個輪廓內部有一個輪廓，則稱外部的輪廓為父輪廓，**例如**：輪廓0是輪廓1和輪廓2的父輪廓。
- **子輪廓**：如果一個輪廓內部有一個輪廓，則稱內部的輪廓為子輪廓，**例如**：輪廓1和輪廓2是輪廓0的子輪廓。
- **同級輪廓**：如果2個輪廓有相同的父輪廓，則這2個輪廓稱同級輪廓，**例如**：輪廓1和輪廓2是有相同的父輪廓0，所以是同級輪廓。
- **註**：有關輪廓0、2、3之間的關係會依檢測模式不同，而有不同的定義。

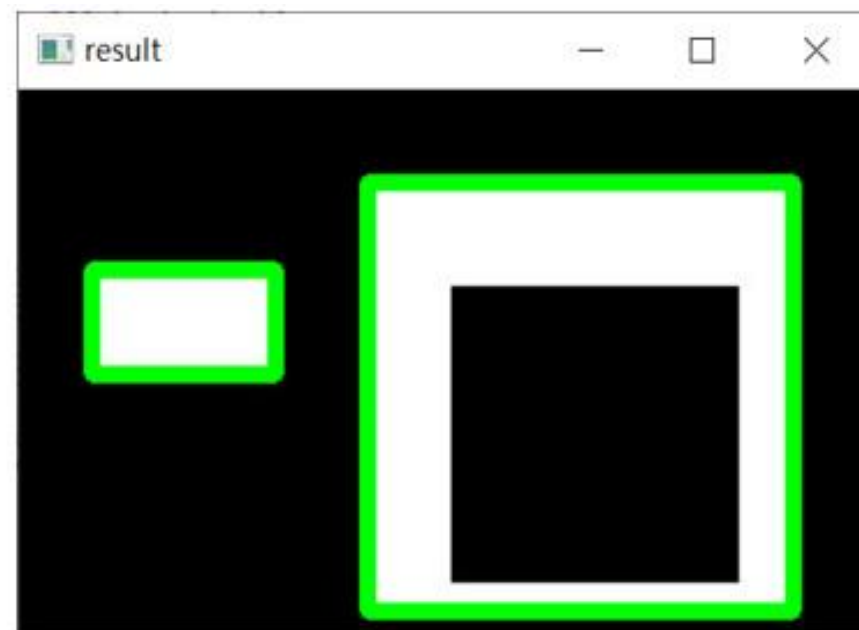
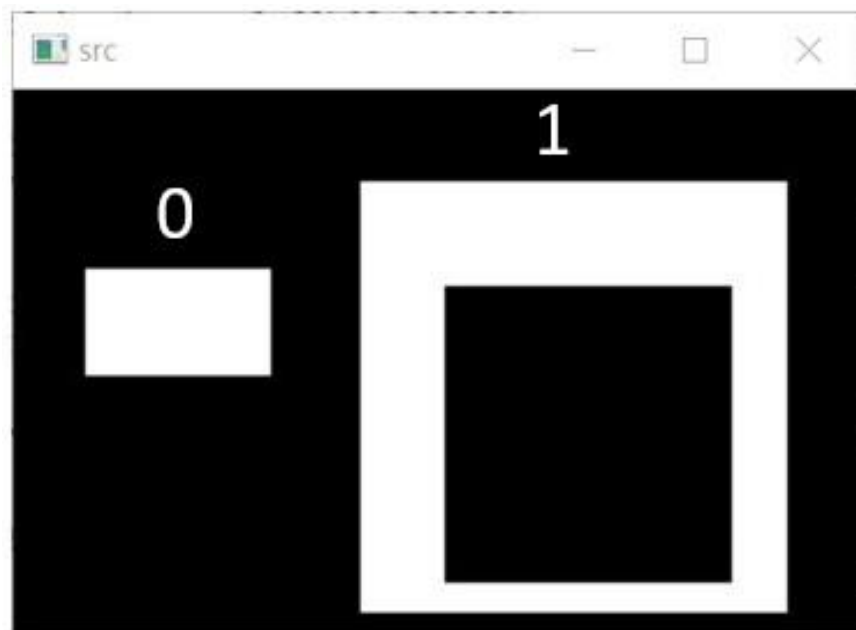
- `findContours()` 函數所回傳的 `hierarchy`，就是定義這個層級關係，這個層級關係所回傳的是一個多維陣列，每個輪廓皆對應一個陣列元素，此多維陣列內有4個元素，所代表的意義如下：
- `[Next Previous First_Child Parent]`
- 上述所代表的意義如下：
- `Next`：下一個同級輪廓的索引編號，如果沒有下一個輪廓則回傳-1。
- `Previous`：前一個同級輪廓的索引編號，如果沒有前一個輪廓則回傳-1。
- `First_Child`：第一個子輪廓的索引編號，如果沒有第一個輪廓則回傳-1。
- `Parent`：父輪廓的索引編號，如果沒有父輪廓則回傳-1。

- 使用 `findContours( )` 函數時，不同的檢測模式(`mode`)將有不同 `hierarchy` 回傳結果。例如：`RETR_EXTERNAL` 不檢測內部輪廓、`RETR_LIST` 不建立層級，所以回傳的 `hierarchy` 比較沒有意義，不過下列還是用實例解說。

## 15.3.1：檢測模式RETR\_EXTERNAL

- 程式實例 `ch15_11.py`：使用 `easy2.jpg` 檔案，檢測模式採用 `RETR_EXTERNAL`，列印 `hierarchy`。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_11.py =====  
hierarchy 資料類型：<class 'numpy.ndarray'>  
列印層級  
[[[ 1 -1 -1 -1]  
  [-1  0 -1 -1]]]
```



- hierarchy所回傳的陣列資料意義如下：

- [1 -1 -1 -1]：這是第0個輪廓。

- 1代表下一個同級輪廓的索引編號是1。

- -1代表前一個同級輪廓不存在。

- -1代表第一個子輪廓不存在。

- -1代表父輪廓不存在。

- [-1 0 -1 -1]：這是第1個輪廓。

- -1代表下一個同級輪廓不存在。

- 0代表前一個同級輪廓的索引編號是0。

- -1代表第一個子輪廓不存在。

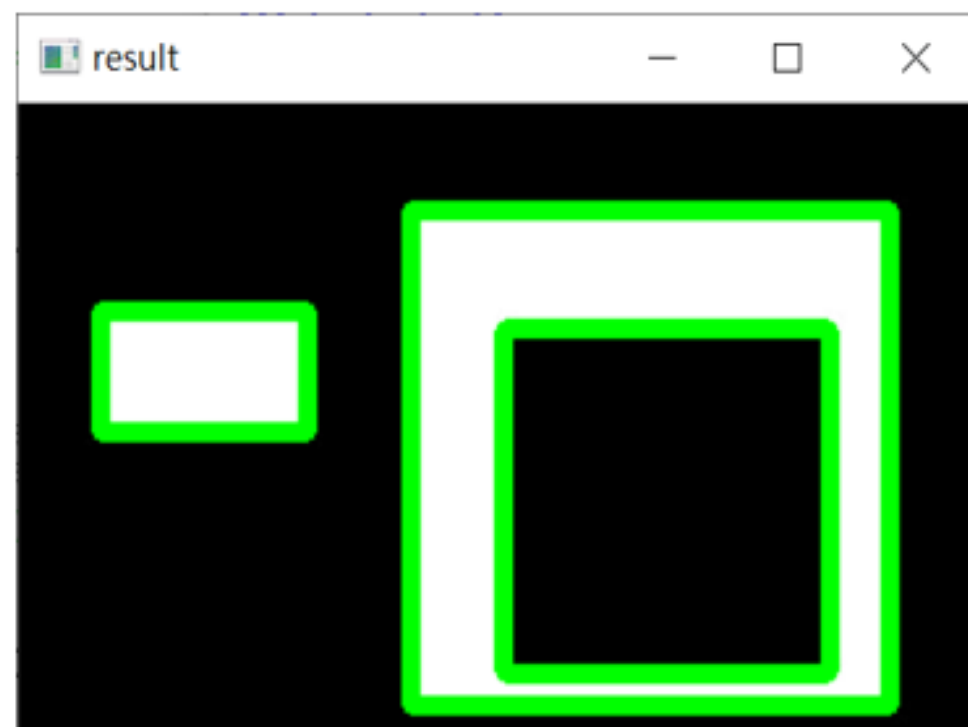
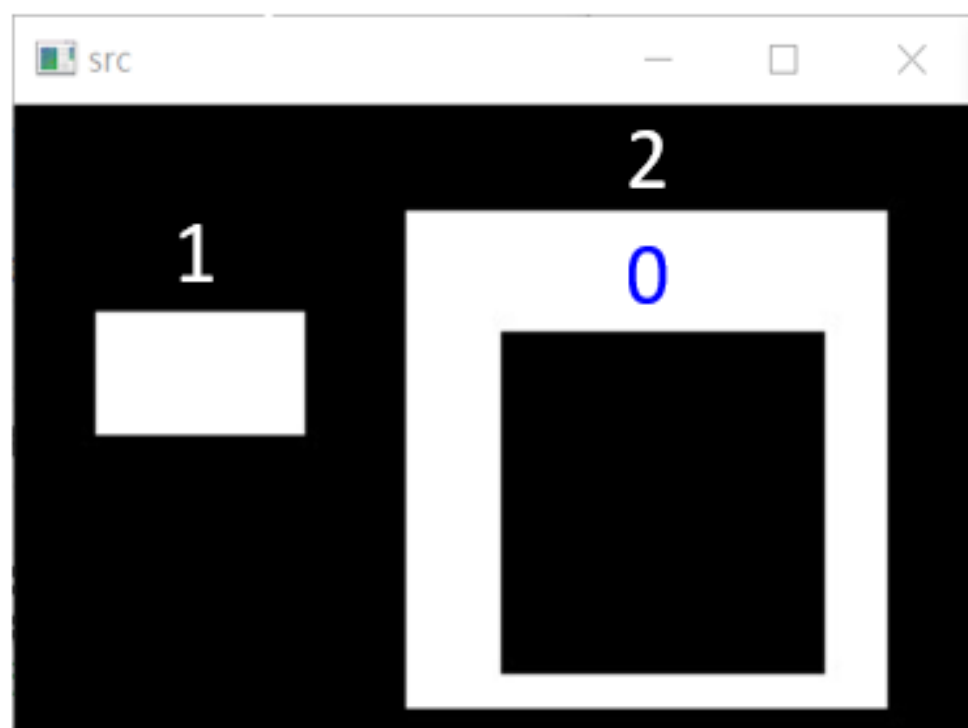
- -1代表父輪廓不存在。



## 15.3.1：檢測模式RETR\_LIST

- 程式實例ch15\_12.py：使用easy2.jpg檔案重新設計ch15\_11.py，檢測模式採用RETR\_LIST，列印hierarchy。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_12.py =====  
hierarchy 資料類型：<class 'numpy.ndarray'>  
列印層級  
[[[ 1 -1 -1 -1]  
 [ 2  0 -1 -1]  
 [-1  1 -1 -1]]]
```



- hierarchy所回傳的陣列資料意義如下：

- $[1 \ -1 \ -1 \ -1]$ ：這是第0個輪廓。

- 1代表下一個同級輪廓的索引編號是1。

- 1代表前一個同級輪廓不存在。

- 1代表第一個子輪廓不存在。

- 1代表父輪廓不存在。

- $[2 \ 0 \ -1 \ -1]$ ：這是第1個輪廓。

- 2代表下一個同級輪廓的索引編號是0。

- 0代表前一個同級輪廓的索引編號是0。

- 1代表第一個子輪廓不存在。

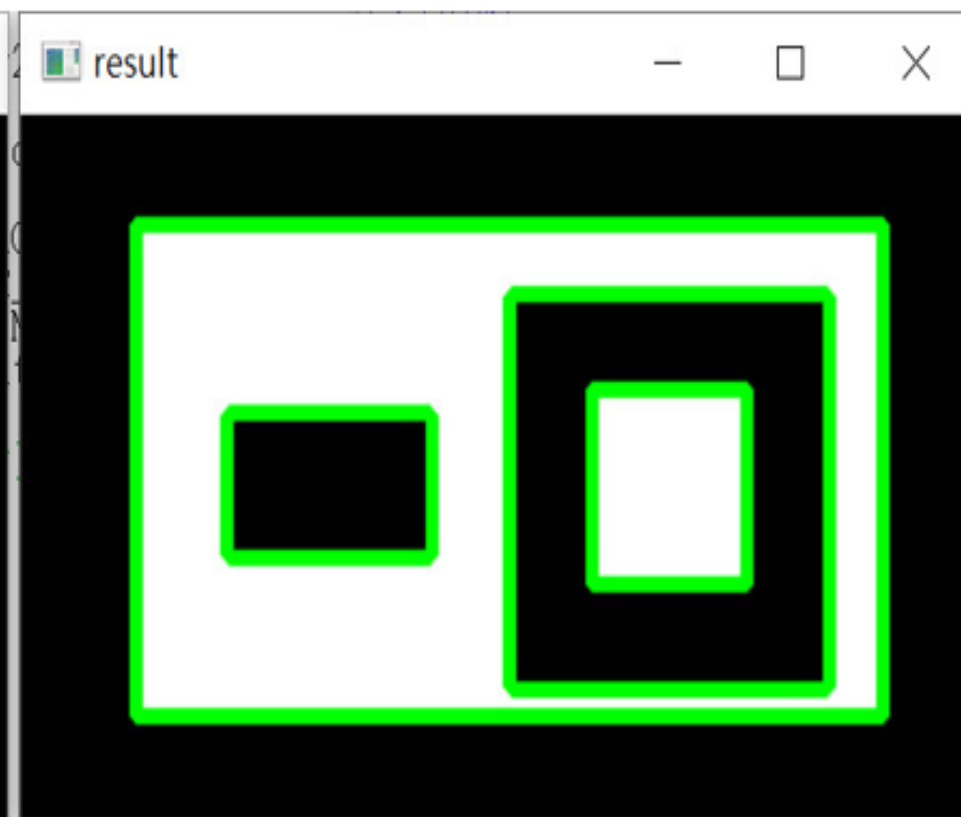
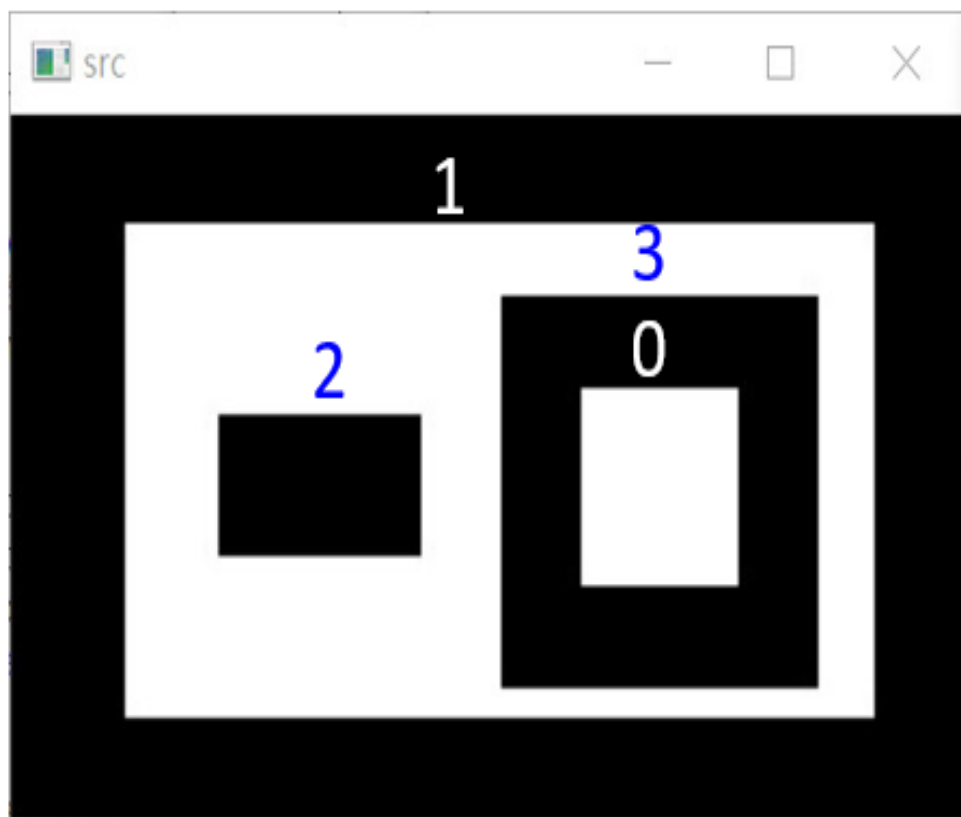
- 1代表父輪廓不存在。

- $[-1\ 1\ -1\ -1]$ ：這是第2個輪廓。
- $-1$ 代表下一個同級輪廓不存在。
- $1$ 代表前一個同級輪廓的索引編號是1。
- $-1$ 代表第一個子輪廓不存在。
- $-1$ 代表父輪廓不存在。

## 15.3.3：檢測模式RETR\_CCOMP

- 當檢測模式採用RETR\_CCOMP時，會檢測所有輪廓，同時建立兩個層級關係，下列將以實例解說。
- 程式實例ch15\_13.py：使用easy3.jpg檔案，檢測模式採用RETR\_CCOMP，列印輪廓與層次關係。

```
===== RESTART: D:\OpenCV_Python\ch15\ch15_13.py =====  
hierarchy 資料類型：<class 'numpy.ndarray'>  
列印層級  
[[[ 1 -1 -1 -1]  
  [-1  0  2 -1]  
  [ 3 -1 -1  1]  
  [-1  2 -1  1]]]
```



- hierarchy所回傳的陣列資料意義如下：
- [1 -1 -1 -1]：這是第0個輪廓。
  - 1代表下一個同級輪廓的索引編號是1，因為這是內部增加的輪廓，此輪廓與最外部的輪廓同級。
  - -1代表前一個同級輪廓不存在。
  - -1代表第一個子輪廓不存在。
  - -1代表父輪廓不存在，因為在兩個層級觀念中，這相當於與最外圍輪廓同級。
- [-1 0 2 -1]：這是第1個輪廓。
  - -1代表下一個同級輪廓不存在。
  - 0代表前一個同級輪廓的索引編號是0。
  - 2代表第一個子輪廓的索引編號是2。
  - -1代表父輪廓不存在。

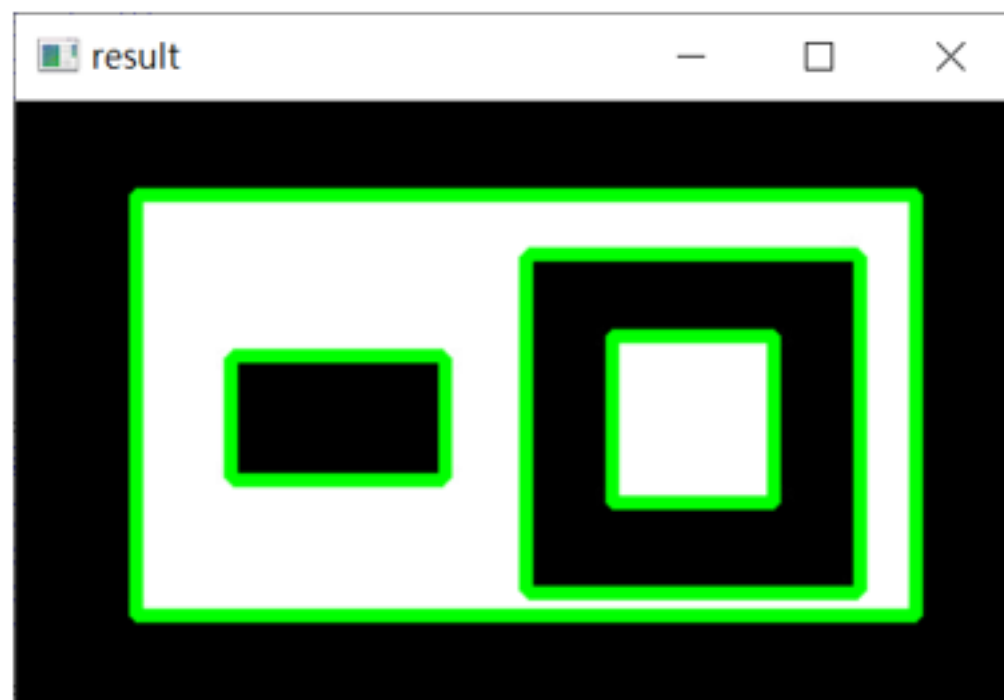
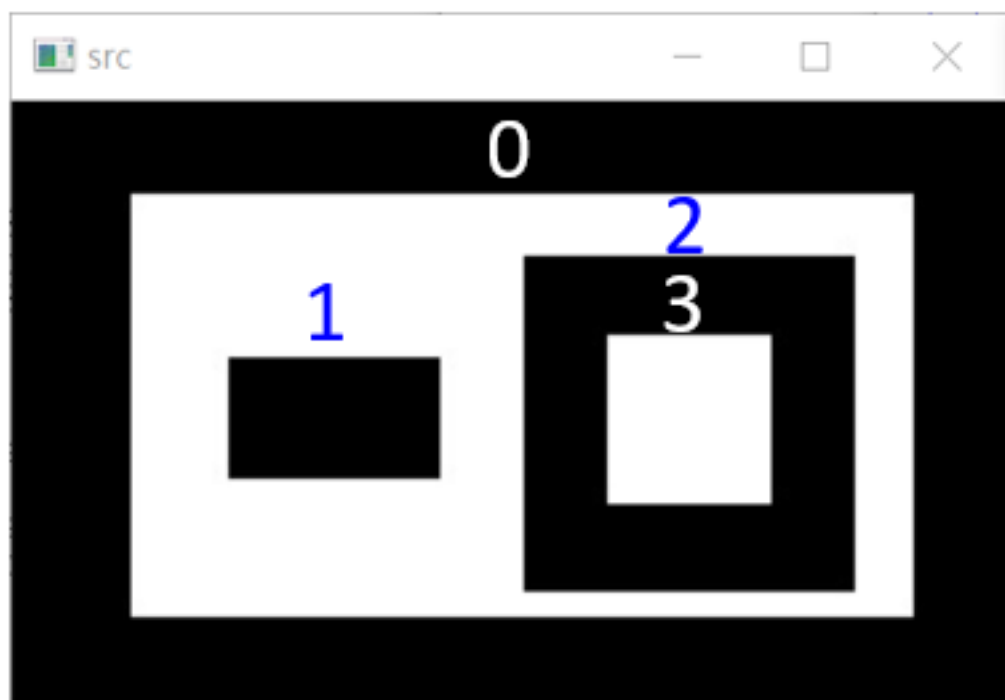
- $[3 -1 -1 1]$ ：這是第2個輪廓。
  - 3代表下一個同級輪廓的索引編號是3。
  - 1代表前一個同級輪廓不存在。
  - 1代表第一個子輪廓不存在。
  - 1代表父輪廓的索引編號是1。
- $[-1 2 -1 1]$ ：這是第3個輪廓。
  - 1代表下一個同級輪廓不存在。
  - 2代表前一個同級輪廓索引編號是2。
  - 1代表第一個子輪廓不存在，因為只有2層。
  - 1代表父輪廓的索引編號是1。



## 15.3.4：檢測模式RETR\_TREE

- 程式實例ch15\_14.py：修訂程式實例ch15\_13.py第12列，檢測模式採用RETR\_TREE，列印輪廓與樹狀層次關係。

```
===== RESTART: D:\OpenCV_Python\ch15\ch15_14.py =====  
列印層級  
[[[-1 -1 1 -1]  
 [ 2 -1 -1 0]  
 [-1 1 3 0]  
 [-1 -1 -1 2]]]
```



- $[-1 \ -1 \ 1 \ -1]$ ：這是第0個輪廓。
  - $-1$ 代表下一個同級輪廓不存在。
  - $-1$ 代表前一個同級輪廓不存在。
  - $1$ 代表第一個子輪廓的索引編號是1。
  - $-1$ 代表父輪廓不存在。
- $[2 \ -1 \ -1 \ 0]$ ：這是第1個輪廓。
  - $2$ 代表下一個同級輪廓的索引編號是2。
  - $-1$ 代表前一個同級輪廓不存在。
  - $-1$ 代表第一個子輪廓不存在。
  - $0$ 代表父輪廓的索引編號是0。

- $[-1\ 1\ 3\ 0]$ ：這是第2個輪廓。
  - $-1$ 代表下一個同級輪廓不存在。
  - $1$ 代表前一個同級輪廓的索引編號是 $1$ 。
  - $3$ 代表第一個子輪廓的索引編號是 $3$ 。
  - $0$ 代表父輪廓的索引編號是 $0$ 。
- $[-1\ -1\ -1\ 2]$ ：這是第3個輪廓。
  - $-1$ 代表下一個同級輪廓不存在。
  - $-1$ 代表前一個同級輪廓不存在。
  - $-1$ 代表第一個子輪廓不存在。
  - $2$ 代表父輪廓的索引編號是 $2$ 。

## 15.4：輪廓的特徵 – 影像矩(Image moments)

- 輪廓的特徵是指質心、面積、周長、邊界框 ... 等，這些觀念常被使用在模式識別、影像識別

## 15-4-1：矩特徵moments( )函數

- `m = cv2.moments(array, binaryImage)`

- 上述所回傳的內容是字典(dict)類型的影像矩，這個影像矩包含下列資料：

## □ 空間矩

- 零階矩： $m_{00}$
- 一階矩： $m_{10}, m_{01}$
- 二階矩： $m_{20}, m_{11}, m_{02}$
- 三階矩： $m_{30}, m_{21}, m_{12}, m_{03}$

## □ 中心矩

- 二階中心矩： $\mu_{20}, \mu_{11}, \mu_{02}$
- 三階中心矩： $\mu_{30}, \mu_{21}, \mu_{12}, \mu_{03}$

## □ 歸一化中心矩

- 二階Hu矩： $\nu_{20}, \nu_{11}, \nu_{02}$
- 三階Hu矩： $\nu_{30}, \nu_{21}, \nu_{12}, \nu_{03}$

## 15.4.2：基礎影像矩推導 – 輪廓質心

- 簡單的說影像矩就是一組統計參數，這個參數紀錄像素所在位置與強度分佈

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$



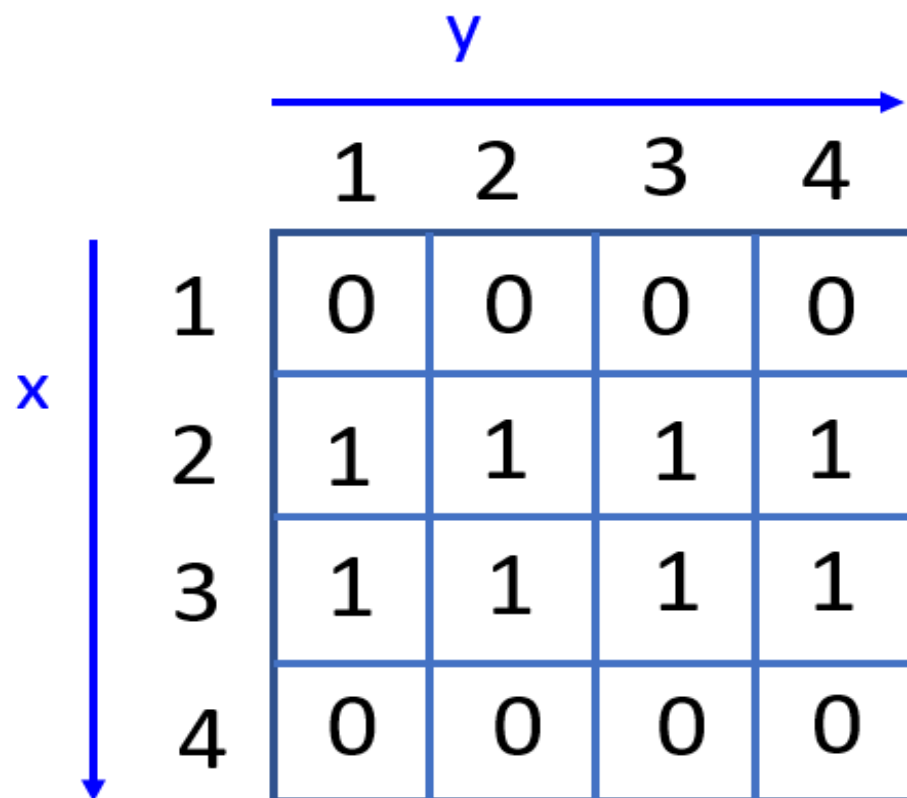
## □面積計算

$$M_{00} = \sum_x \sum_y I(x, y)$$

## □質心計算

$$(\bar{x}, \bar{y}) = \left( \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right)$$

- 假設有一個4 x 4的二元值影像，假設內容與座標如下：



A 4x4 binary image matrix is shown. The horizontal axis is labeled 'y' with a blue arrow pointing right, and the vertical axis is labeled 'x' with a blue arrow pointing down. The matrix contains binary values (0 and 1) arranged in a grid.

		1	2	3	4
1	0	0	0	0	
2	1	1	1	1	
3	1	1	1	1	
4	0	0	0	0	

$$M_{00} = \sum_x \sum_y I(x, y)$$

$$M_{10} = \sum_x \sum_y x I(x, y)$$

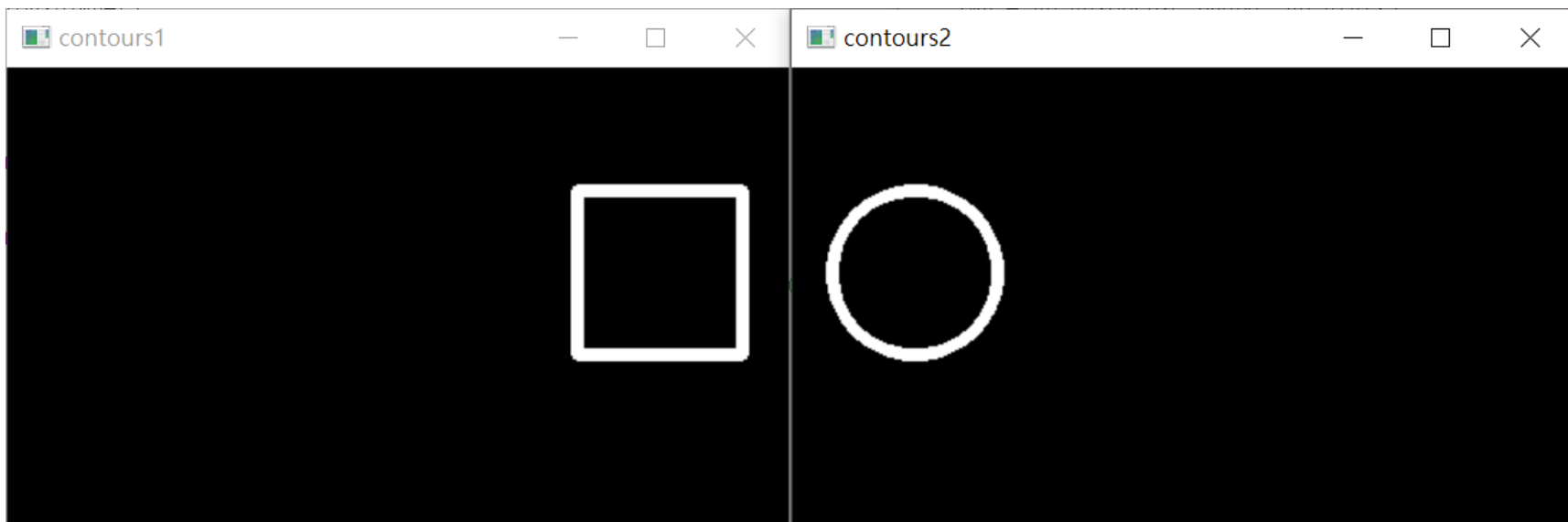
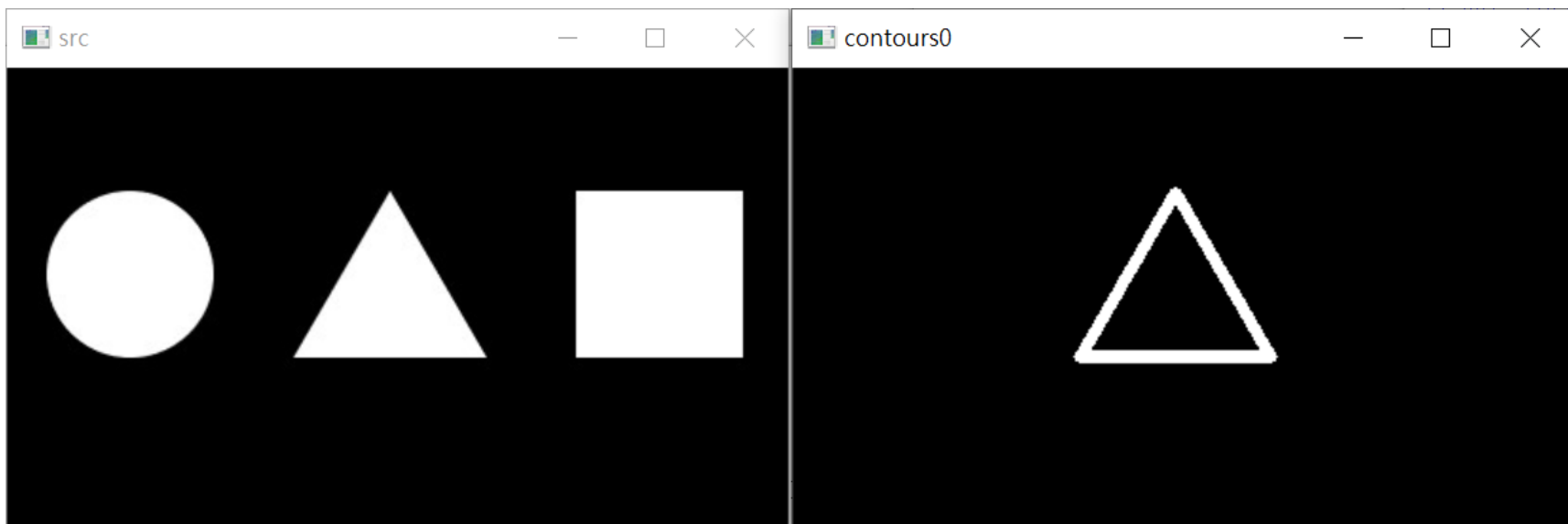
$$M_{01} = \sum_x \sum_y y I(x, y)$$

$$(\bar{x}, \bar{y}) = \left( \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) = \left( \frac{20}{8}, \frac{20}{8} \right) = (2.5, 2.5)$$

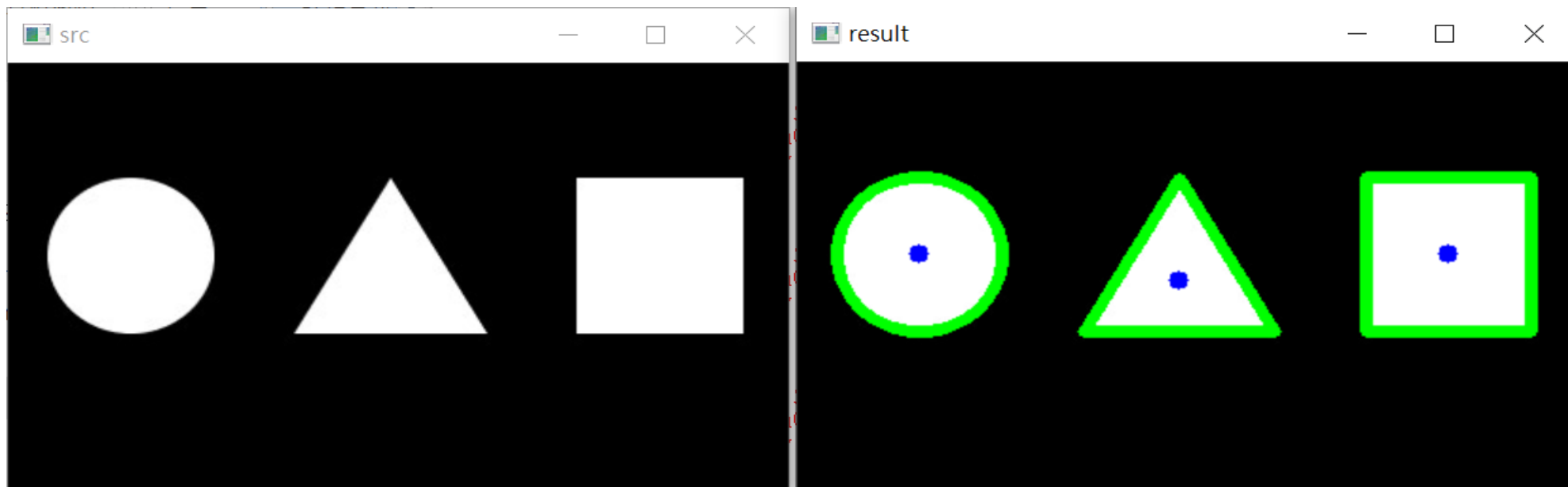
## 15-4-3：影像矩實例

- 程式實例ch15\_15.py：擴充ch15\_3.py列印輪廓面積與每個輪廓的影像矩。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_15.py =====
輪廓面積 str(i) = 4344.0
輪廓面積 str(i) = 7569.0
輪廓面積 str(i) = 5964.0
列印影像矩 0
{'m00': 4344.0, 'm10': 875551.0, 'm01': 534872.3333333333, 'm20': 178291996.0,
'm11': 107806662.83333333, 'm02': 67667946.66666666, 'm30': 36669616857.5, 'm21':
: 21974314231.166668, 'm12': 13638973452.333334, 'm03': 8756666917.5, 'mu20': 18
21104.2870626152, 'mu11': 952.3539748340845, 'mu02': 1809656.3891702518, 'mu30':
32430.916221618652, 'mu21': 21016905.17049837, 'mu12': -12999.67679822445, 'mu0
3': -20861350.522485733, 'nu20': 0.09650619295080995, 'nu11': 5.04683104123893e-
05, 'nu02': 0.09589953189865046, 'nu30': 2.6075622006486343e-05, 'nu21': 0.01689
8346973212075, 'nu12': -1.0452207272856162e-05, 'nu03': -0.01677327544654871}
列印影像矩 1
{'m00': 7569.0, 'm10': 2599951.5, 'm01': 821236.5, 'm20': 897857487.0, 'm11': 2
82094737.75, 'm02': 93878307.0, 'm30': 311693885601.75, 'm21': 97417537339.5, 'm
12': 32247198454.5, 'm03': 11221786154.25, 'mu20': 4774146.75, 'mu11': 0.0, 'mu0
2': 4774146.75, 'mu30': 0.0, 'mu21': 0.0, 'mu12': 0.0, 'mu03': 0.0, 'nu20': 0.08
33333333333333334, 'nu11': 0.0, 'nu02': 0.083333333333333334, 'nu30': 0.0, 'nu21':
0.0, 'nu12': 0.0, 'nu03': 0.0}
列印影像矩 2
{'m00': 5964.0, 'm10': 384986.5, 'm01': 647295.0, 'm20': 27681568.833333332, 'm
11': 41781606.5, 'm02': 73084880.0, 'm30': 2152224582.25, 'm21': 3004127667.9166
665, 'm12': 4717267592.083333, 'm03': 8546756317.5, 'mu20': 2830025.3755449355,
'mu11': -2403.6471327990294, 'mu02': 2831557.225855127, 'mu30': -34543.863102436
066, 'mu21': 54838.510442614555, 'mu12': 34121.83210071921, 'mu03': -55062.21492
099762, 'nu20': 0.07956371628904141, 'nu11': -6.75764606867403e-05, 'nu02': 0.07
960678293590986, 'nu30': -1.2575545037745031e-05, 'nu21': 1.996372426062902e-05,
'nu12': 1.2421906463689873e-05, 'nu03': -2.004516291544042e-05}
```



- 程式實例ch15\_16.py：重新設計ch15\_1.py，為每個輪廓繪製中心點。

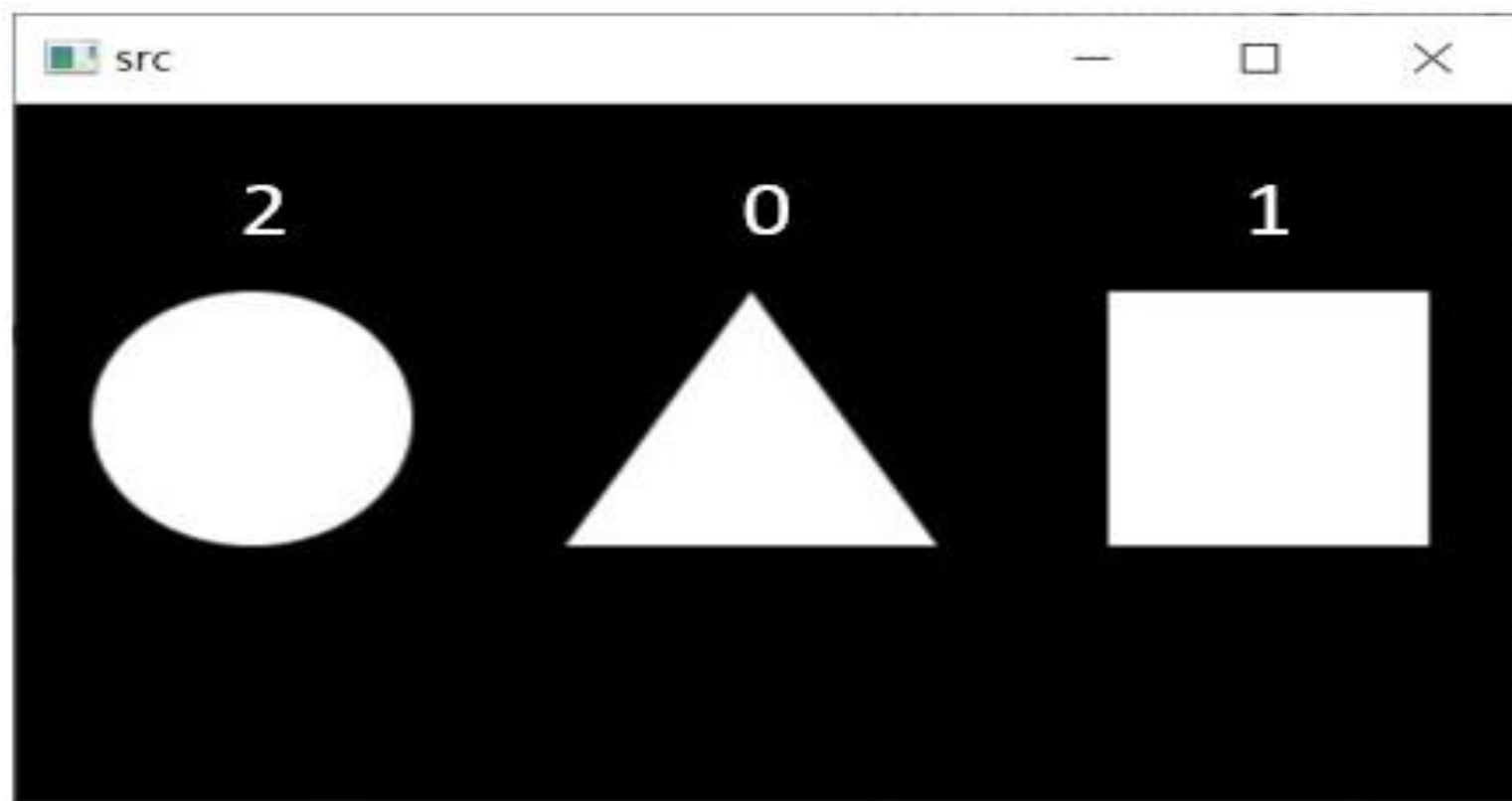


## 15-4-4：計算輪廓面積

- `area = cv2.contourArea(contour, oriented)`
- 程式實例`ch15_17.py`：使用`contourArea()`函數計算輪廓面積。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_17.py =====  
輪廓 0 面積 = 4344.0  
輪廓 1 面積 = 7569.0  
輪廓 2 面積 = 5964.0
```

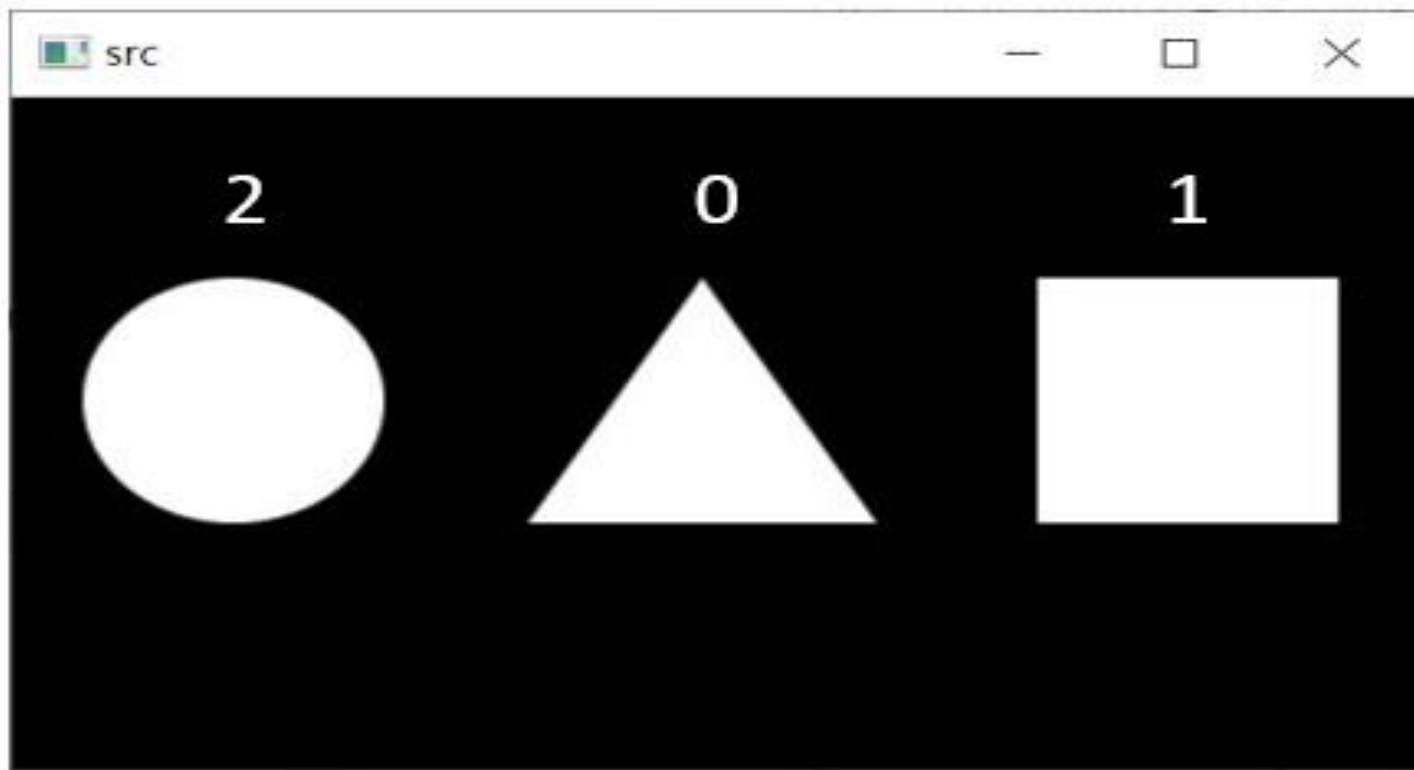




## 15-4-5：計算輪廓周長

- `area = cv2.arcLength(contour, closed)`
- 程式實例`ch15_18.py`：使用`arcLength()`函數計算輪廓周長。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_18.py =====  
輪廓 0 周長 = 315.4213538169861  
輪廓 1 周長 = 348.0  
輪廓 2 周長 = 289.42135322093964
```



## 15.5：輪廓外形的匹配 - Hu矩

- 15.5.1：OpenCV計算Hu矩的函數
- `hu = cv2.HuMoments(m)`
- 上述參數意義如下：
  - `hu`：這是Hu矩的回傳結果。
  - `m`：這是`moments( )`函數回傳的影像矩。

$$h_0 = \eta_{20} + \eta_{02}$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$h_3 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ + (3\eta_{21} - \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$h_5 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]$$

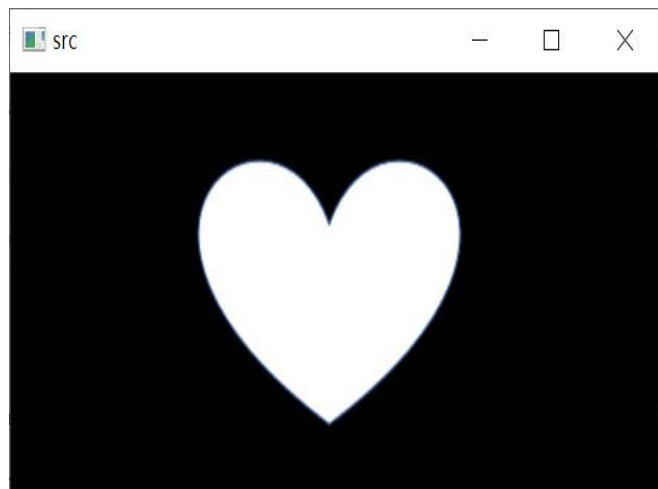
$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

## 15-5-2：第0個Hu矩的公式驗證

$$h_0 = \eta_{20} + \eta_{02}$$

$$h_0 - (\eta_{20} + \eta_{02}) = 0$$

- 程式實例ch15\_19.py：使用heart.jpg，驗證上述公式。



```
===== RESTART: D:/OpenCV_Python/ch15/ch15_19.py =====  
歸一化中心矩 nu20 = 0.00047926293904721644  
歸一化中心矩 nu02 = 0.00023866024885601947  
Hu  
[[ 7.17923188e-04]  
 [ 5.78896885e-08]  
 [ 6.81194047e-11]  
 [ 2.42937256e-12]  
 [-3.12515353e-23]  
 [-5.84507960e-16]  
 [ 1.53613252e-25]]  
驗證結果 h0 - nu20 - nu02 = 0.0
```

- 程式實例ch15\_20.py：有一幅影像3heart.jpg，請列出內部3個輪廓的Hu矩。在這3個輪廓中，使用findContours( )函數所找的順序可以參考執行結果圖。

```
===== RESTART: D:\OpenCV_Python\ch15\ch15_20.py =====  
h0 = [0.18400164] [0.18362647] [0.18362647]  
h1 = [0.00390046] [0.00388969] [0.00388969]  
h2 = [0.00118707] [0.00115877] [0.00115877]  
h3 = [4.41012234e-05] [4.27240647e-05] [4.27240647e-05]  
h4 = [-1.00904583e-08] [-9.50599826e-09] [-9.50599826e-09]  
h5 = [-2.75427701e-06] [-2.66455047e-06] [-2.66455047e-06]  
h6 = [4.40391468e-11] [6.22527623e-11] [6.22527623e-11]
```



2

1

0



- 程式實例ch15\_21.py：有一幅影像3shapes.jpg，請列出內部3個輪廓的Hu矩。在這3個輪廓中，使用findContours( )函數所找的順序可以參考執行結果圖。



0

2

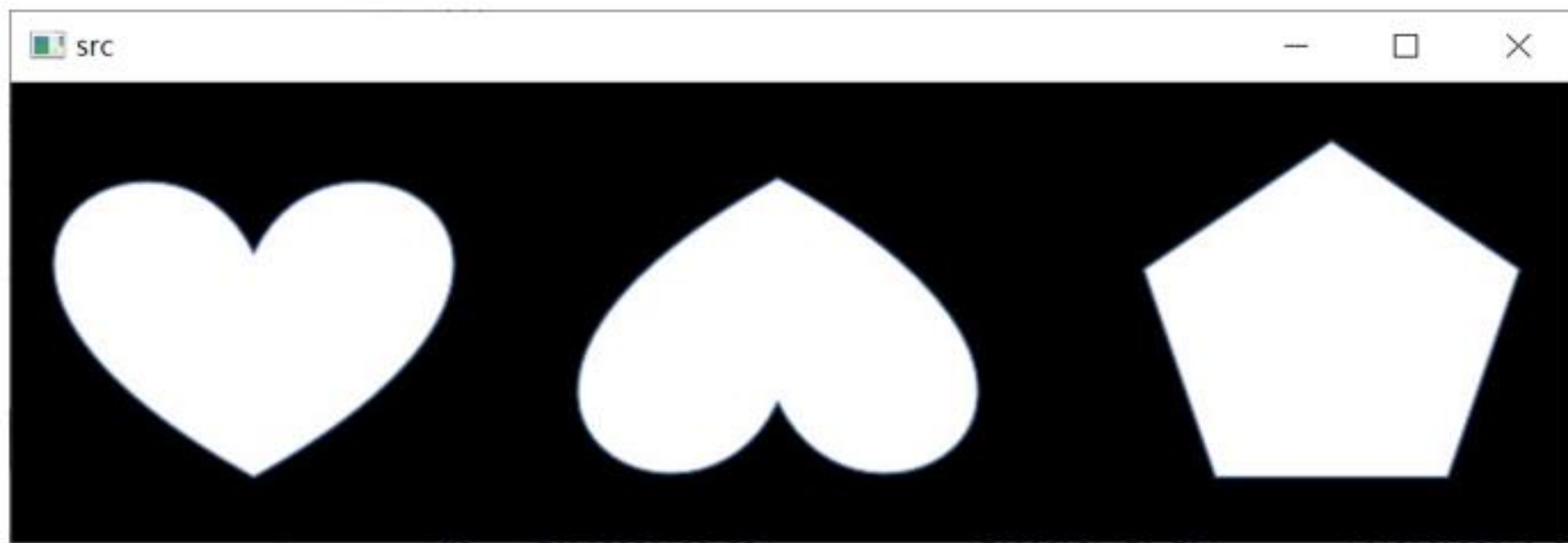
1

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_21.py =====  
h0 = [0.18362125] [0.15949742] [0.18383141]  
h1 = [0.00388853] [6.2205636e-05] [0.00388999]  
h2 = [0.00115914] [1.09030421e-09] [0.00117116]  
h3 = [4.25428614e-05] [8.53260565e-13] [4.34904474e-05]  
h4 = [-9.44727464e-09] [1.09940012e-23] [-9.81486955e-09]  
h5 = [-2.6528849e-06] [1.82308256e-16] [-2.7123853e-06]  
h6 = [2.90666076e-11] [2.35891942e-23] [8.19693167e-11]
```

## 15.5.3：輪廓匹配

- `retval = cv2.matchShapes(contour1, contour2, method, parameter)`
- 程式實例`ch15_22.py`：使用`myheart.jpg`列出各影像的比較結果。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_22.py =====  
輪廓0和0比較 = 0.0  
輪廓0和1比較 = 0.00046299603915214704  
輪廓0和2比較 = 0.29307592277155203
```



0

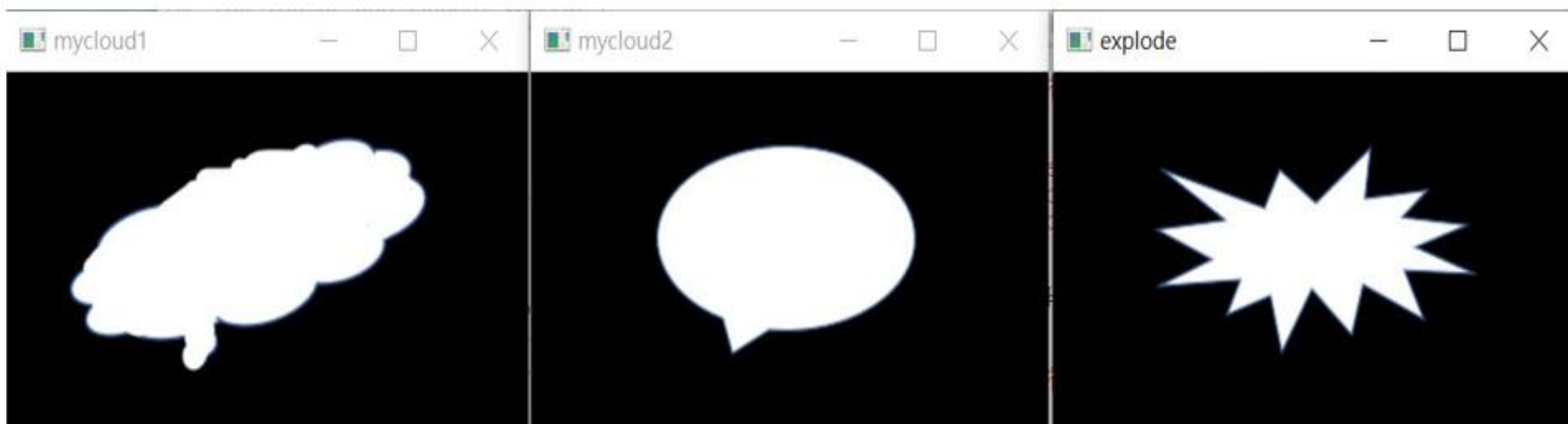
1

2

## 15.6：再談輪廓外形匹配

- `sd = cv2.createShapeContextDistanceExtractor( )`
- `retval = sd.computeDistance(contour1, contour2)`
- 程式實例`ch15_23.py`：使用形狀場景距離重新設計`ch15_22.py`。

```
===== RESTART: D:\OpenCV_Python\ch15\ch15_23.py =====  
影像1和1比較 = 0.0  
影像1和2比較 = 0.22617380321025848  
影像1和3比較 = 0.8256418704986572
```



影像 1

影像 2

影像 3

## 15-6-2 : Hausdorff距離

$$h(A, B) = \max_{a \in A} \left\{ \min_{b \in B} \{d(a, b)\} \right\}$$

- `hd = cv2.createHausdorffDistanceExtractor( )`
- `retval = hd.computeDistance(contour1, contour2)`
- 程式實例 `ch15_24.py`：使用 Hausdorff 距離觀念重新設計 `ch15_23.py`。

```
===== RESTART: D:/OpenCV_Python/ch15/ch15_24.py =====  
影像1和1比較 = 0.0  
影像0和1比較 = 12.206555366516113  
影像0和2比較 = 8.5440034866333
```

