

第3章

學習OpenCV需要的Numpy知識

3-1：陣列ndarray

- ndarray陣列幾個特色如下：
 - 陣列大小是固定。
 - 陣列元素內容的資料型態是相同。

3-2 : Numpy的資料型態

- 下列是本書常用的Numpy所定義的資料型態：

- `int16`：16位元整數(-32768 ~ 32767)。

- `int32`：32位元整數(-2147483648 ~ 2147483647)。

- `uint8`：8位元無號整數(0 ~ 255)。

- `float32`：單精度浮點數，符號位，8位指數，23位尾數。

- `float64`：雙倍精度浮點數，符號位，11位指數，52位尾數。

3-3：建立一維或多維陣列

- 3-3-1：認識ndarray的屬性

- ndarray.dtype：陣列元素型態。
- ndarray.itemsize：陣列元素資料型態大小(或稱所佔空間)，單位是為位元組。
- ndarray.ndim：陣列的維度。
- ndarray.shape：陣列維度元素個數的元組，也可以用於調整陣列大小。
- ndarray.size：陣列元素個數。

3-3-2：使用array()建立一維陣列

- `numpy.array(object, dtype=None, copy=True, order='K', subok=False, ndmin)`
- **實例1**：建立一維陣列，陣列內容是1, 2, 3，同時列出陣列的資料型態。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> print(type(x))    ← 列印x資料類型
<class 'numpy.ndarray'>
>>> print(x)          ← 列印x陣列內容
[1 2 3]
```

x[0]

1

x[1]

2

x[2]

3

- 實例2：列出陣列元素內容。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> print(x[0])
1
>>> print(x[1])
2
>>> print(x[2])
3
```

- 實例3：設定陣列內容。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> x[1] = 10
>>> print(x)
[ 1 10  3]
```


- 實例4：認識ndarray的屬性。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> x.dtype          ← 列印x陣列元素型態
dtype('int32')
>>> x.itemsize       ← 列印x陣列元素大小
4
>>> x.ndim           ← 列印x陣列維度
1
>>> x.shape          ← 列印x陣列外形, 3是第1維元素個數
(3,)
>>> x.size           ← 列印x陣列元素個數
3
```

- **實例5**：`array()`函數也可以接受使用**dtype**參數設定元素的資料型態。

```
>>> import numpy as np
>>> x = np.array([2, 4, 6], dtype=np.int8)
>>> x.dtype
dtype('int8')
```

- 上述因為元素是8為元整數，所以執行**x.itemsize**，所得的結果是**1**。

```
>>> x.itemsize
1
```

- 實例6：浮點數陣列的建立與列印。

```
>>> import numpy as np
>>> y = np.array([1.1, 2.3, 3.6])
>>> y.dtype
dtype('float64')
>>> y
array([1.1, 2.3, 3.6])
>>> print(y)
[1.1 2.3 3.6]
```

x[0]

1.1

x[1]

2.3

x[2]

3.6

3-3-3：使用array()函數建立多維陣列

- 程式實例ch3_1.py：建立二維和三維陣列。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_1.py =====  
陣列維度 = 2  
陣列外型 = (1, 3)  
陣列大小 = 3  
陣列內容  
[[1 2 3]]  
-----  
陣列維度 = 2  
陣列外型 = (2, 3)  
陣列大小 = 6  
陣列內容  
[[1 2 3]  
 [4 5 6]]
```

- 程式實例ch3_2.py：另一種設定二維陣列的方式重新設計ch3_1.py。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_2.py =====  
陣列維度 = 2  
陣列外型 = (2, 3)  
陣列大小 = 6  
陣列內容  
[[1 2 3]  
 [4 5 6]]
```

1	2	3
4	5	6

二維陣列內容

x[0][0]	x[0][1]	x[0][2]
x[1][0]	x[1][1]	x[1][2]

二維陣列索引

- 程式實例ch3_3.py：認識引用二維陣列索引的方式。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_3.py =====  
3  
6  
3  
6
```

3-3-4：使用zeros()建立內容是0的多維陣列

- `np.zeros(shape, dtype=float)`

□ `shape`：陣列外型。

□ `dtype`：預設是浮點數資料類型，也可以用此設定資料類型。

- 程式實例`ch3_4.py`：分別建立1 x 3一維和2 x 3二維外型的陣列，一維陣列元素資料類型是浮點數(float)，二維陣列元素資料類型是8位元無號整數(unit8)。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_4.py =====  
[0. 0. 0.]  
-----  
[[0 0 0]  
 [0 0 0]]
```

3-3-5：使用ones()建立內容是1的多維陣列

- `np.ones(shape, dtype=None)`
 - `shape`：陣列外型。
 - `dtype`：預設是64浮點數資料類型(float64)，也可以用此設定資料類型。
- 程式實例ch3_5.py：分別建立1 x 3一維和2 x 3二維外型的陣列，一維陣列元素資料類型是浮點數(float)，二維陣列元素資料類型是8位元無號整數(unit8)。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_5.py =====  
[1. 1. 1.]  
-----  
[[1 1 1]  
 [1 1 1]]
```


3-3-6：使用empty()建立為初始化的多維陣列

- 函數empty()可以建立指定形狀與資料類型內容的陣列，陣列內容則未初始化，語法如下：
- `np.empty(shape, dtype=float)`
 - `shape`：陣列外型。
 - `dtype`：預設是浮點數資料類型(float)，也可以用此設定資料類型。
- 程式實例ch3_6.py：分別建立1 x 3一維和2 x 3二維外型的隨機數陣列，一維陣列元素資料類型是浮點數(float)，二維陣列元素資料類型是8位元無號整數(unit8)。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_6.py =====  
[9.13599696e+242 6.01334515e-154 1.03474869e-028]  
-----  
[[101  49 100]  
 [ 23 131   1]]
```

3-3-7：使用`random.randint()`建立隨機數內容的多維陣列

- `np.random.randint(low, high=None, size=None, dtype=int)`
 - `low`：隨機數的最小值(含此值)。
 - `high`：這是選項，如果有此參數代表隨機數的最大值(不含此值)。如果不含此參數，則隨機數是0 ~ `low`之間。
 - `size`：這是選項，陣列的維數。
 - `dtype`：預設是整數資料類型(`float`)，也可以用此設定資料類型。
- 程式實例`ch3_7.py`：分別建立單一隨機數、含10個元素陣列的隨機數、3 x 5的二維陣列的隨機數。

```
===== RESTART: D:\OpenCV_Python\ch3\ch3_7.py =====
```

回傳值是10(含)至20(不含)的單一隨機數

15

回傳一維陣列10個元素，值是1(含)至5(不含)的隨機數

[3 1 4 1 3 1 4 2 3 3]

回傳單3*5陣列，值是0(含)至10(不含)的隨機數

[[4 1 8 9 0]

[8 3 5 8 5]

[3 8 6 2 2]]

3-3-8：使用arange()函數建立陣列數據

- `np.arange(start, stop, step)` # `start`和`step`是可以省略
- `start`是起始值如果省略預設值是0，`stop`是結束值但是所產生的陣列不包含此值，`step`是陣列相鄰元素的間距如果省略預設值是1。
- 程式實例ch3_7_1.py：建立連續數值0 - 15的一維陣列。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_7_1.py =====  
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
```

3-3-9：使用`reshape()`函數更改陣列形式

- `np.reshape(a, newshape)`
- 上述`a`是要更改的陣列，`newshape`是新陣列的外形，`newshape`可以是整數或是元組。
- 程式實例`ch3_7_2.py`：將`1 x 16`陣列改為`4 x 4`陣列。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_7_2.py =====  
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]  
[[ 0  1  2  3  4  5  6  7]  
 [ 8  9 10 11 12 13 14 15]]
```

- 程式實例ch3_7_3.py：重新設計ch3_7_2.py，但是newshape元組的其中一個元素值是 -1，整個newshape內容是(4, -1)。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_7_3.py =====  
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]  
[[ 0  1  2  3]  
 [ 4  5  6  7]  
 [ 8  9 10 11]  
 [12 13 14 15]]
```

- 程式實例ch3_7_4.py：重新設計ch3_7_2.py，但是newshape元組的其中一個元素值是 -1，整個newshape內容是(-1, 8)。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_7_4.py =====  
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]  
[[ 0  1  2  3  4  5  6  7]  
 [ 8  9 10 11 12 13 14 15]]
```


3-4：一維陣列的運算與切片

- 3-4-1：一維陣列的四則運算
- 實例1：陣列與整數的加法運算。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> y = x + 5
>>> print(y)
[6 7 8]
```

- 實例2：陣列加法運算。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> y = np.array([10, 20, 30])
>>> z = x + y
>>> print(z)
[11 22 33]
```

- 實例3：陣列乘法運算。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> y = np.array([10, 20, 30])
>>> z = x * y
>>> print(z)
[10 40 90]
```

- 實例4：陣列除法運算。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> y = np.array([10, 20, 30])
>>> z = x / y
>>> print(z)
[0.1 0.1 0.1]
>>> z = y / x
>>> print(z)
[10. 10. 10.]
```

3-4-2：一維陣列的關係運算子運算

關係運算子	說明	實例	說明
>	大於	<code>a > b</code>	檢查是否 a 大於 b
>=	大於或等於	<code>a >= b</code>	檢查是否 a 大於或等於 b
<	小於	<code>a < b</code>	檢查是否 a 小於 b
<=	小於或等於	<code>a <= b</code>	檢查是否 a 小於或等於 b
==	等於	<code>a == b</code>	檢查是否 a 等於 b
!=	不等於	<code>a != b</code>	檢查是否 a 不等於 b

- 實例1：關係運算子應用在一維陣列的運算。

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> y = np.array([10, 20, 30])
>>> z = x > y
>>> print(z)
[False False False]
>>> z = x < y
>>> print(z)
[ True  True  True]
```

3-4-3：陣列切片

- Numpy陣列的切片與Python的串列切片相同，觀念如下：
- `[start : end : step]`

正值索引	0	1	2	3	4	5	6	7	8	9
陣列內容	0	1	2	3	4	5	6	7	8	9
負值索引	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- ❑ `start`：起始索引，如果省略表示從0開始的所有元素。
- ❑ `end`：終止索引，如果省略表示到末端的所有元素，如果有索引則是不含此索引的元素。
- ❑ `step`：用`step`作為每隔多少區間再讀取。

- 此切片語法的相關應用解說如下：
 - `arr[start:end]` # 讀取從索引|start到(end-1)索引的串列元素
 - `arr[:n]` # 取得串列前n名
 - `arr[:-n]` # 取得串列前面，不含最後n名
 - `arr[n:]` # 取得串列索引n到最後
 - `arr[-n:]` # 取得串列後n名
 - `arr[:]` # 取得所有元素

- 程式實例ch3_8.py：陣列切片的應用。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_8.py =====  
陣列元素如下： [0 1 2 3 4 5 6 7 8 9]  
x[2:]          = [2 3 4 5 6 7 8 9]  
x[:2]          = [0 1 2]  
x[0:3]         = [0 1 2]  
x[1:4]         = [1 2 3]  
x[0:9:2]       = [0 2 4 6 8]  
x[-1]          = 9  
x[::2]         = [0 2 4 6 8]  
x[2::3]        = [2 5 8]  
x[:]           = [0 1 2 3 4 5 6 7 8 9]  
x[::]          = [0 1 2 3 4 5 6 7 8 9]  
x[-3:-7:-1]    = [7 6 5 4]
```


3-4-4：使用參數`copy=True`複製數據

- `x2 = np.array(x1, copy=True)`
- 經過上述複製後，`x2`是`x1`的副本，當內容修改時彼此不會互相影響。
- 程式實例`ch3_9.py`：使用`np.array()`函數複製陣列數據的實例。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_9.py =====  
[0 1 2 3 4 5]  
[0 1 2 3 4 5]  
-----  
[0 1 2 3 4 5]  
[9 1 2 3 4 5]
```

3-4-5：使用`copy()`函數複製陣列

- `x2 = x1.copy()`
- 經過上述複製後，`x2`是`x1`的副本，當內容修改時彼此不會互相影響。
- 程式實例`ch3_10.py`：使用`copy()`函數重新設計`ch3_9.py`。
- 執行結果：與`ch3_9.py`相同。

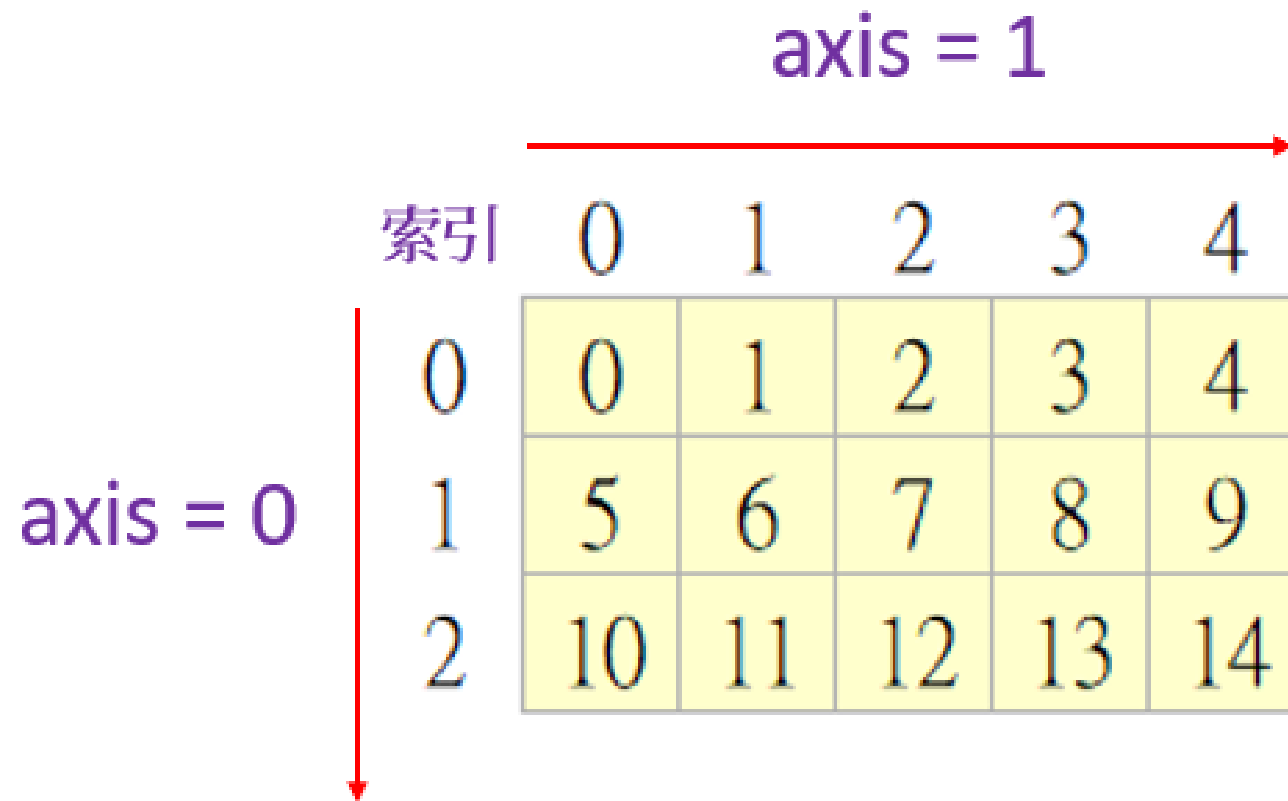
3-5：多維陣列的索引與切片

axis = 1

索引

axis = 0

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14

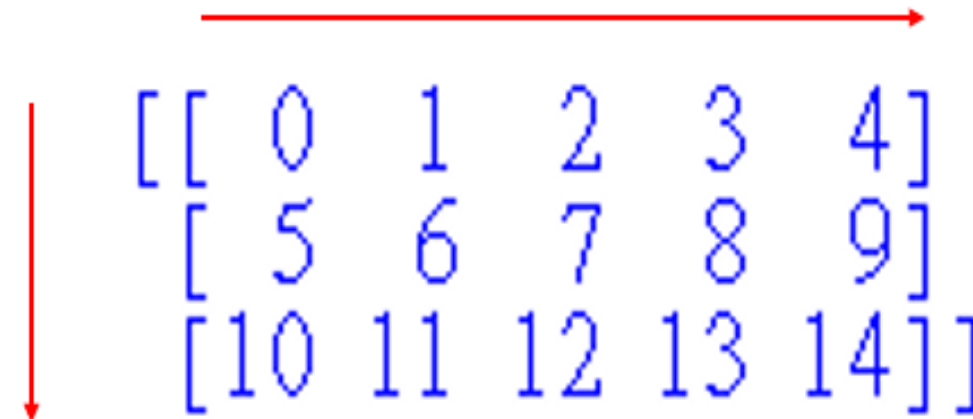


3-5-1：認識axis的定義

- 程式實例ch3_11.py：建立3 x 5的二維陣列同時列印結果。

axis = 1

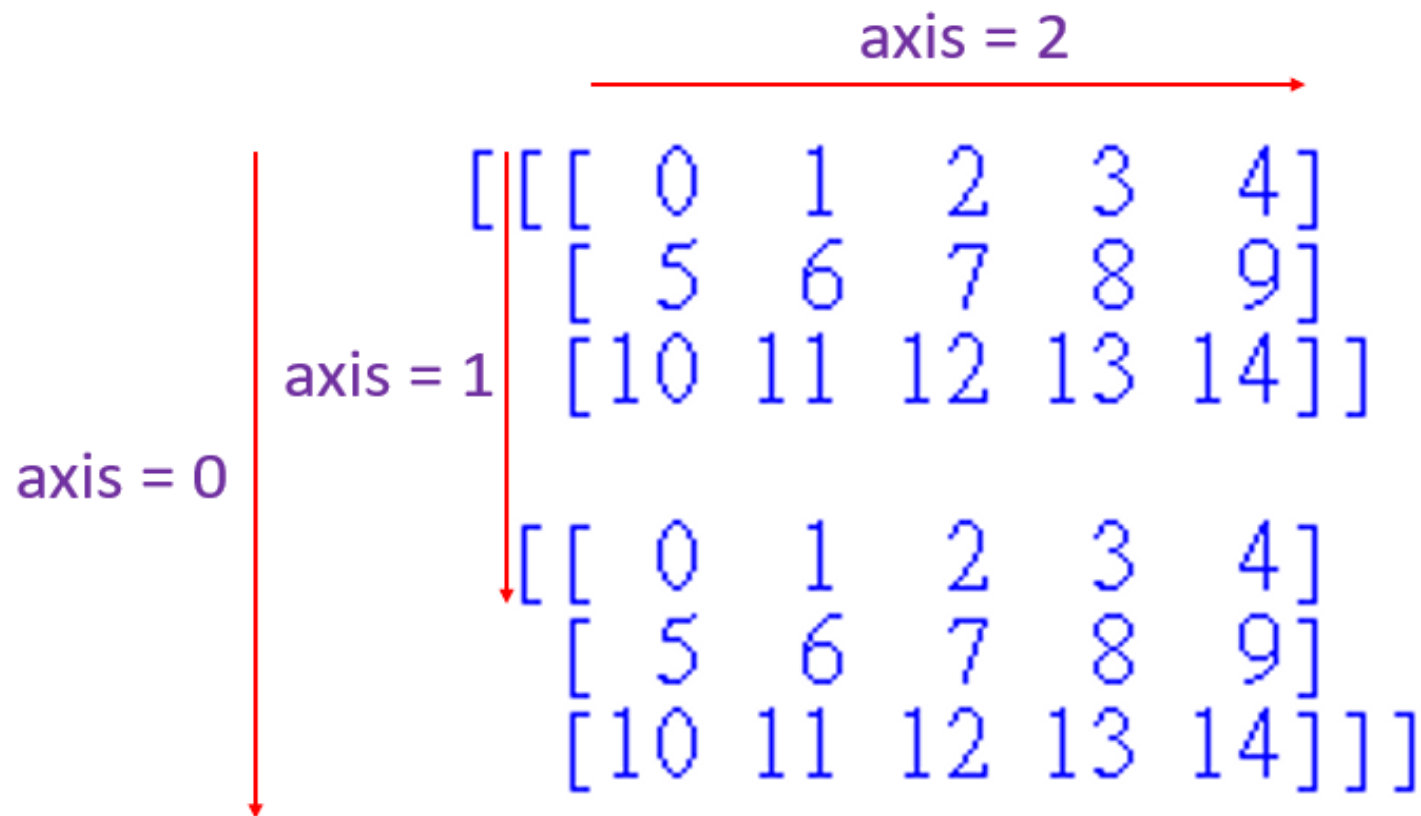
axis = 0



The diagram shows a 3x5 2D array represented as a list of lists. A horizontal red arrow points to the right, labeled 'axis = 1', indicating the column axis. A vertical red arrow points downwards, labeled 'axis = 0', indicating the row axis.

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
```

- 程式實例ch3_12.py：建立2 x 3 x 5的三維陣列同時列印結果。



3-5-2：多維陣列的索引

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

二維陣列內容

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(1,4)	(2,4)

二維陣列索引

- 程式實例ch3_13.py：列出二維陣列特定索引的陣列元素。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_13.py =====  
x4[2][1] = 11  
x4[1][3] = 8
```

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

(0,0,0)	(0,0,1)	(0,0,2)	(0,0,3)	(0,0,4)
(0,1,0)	(0,1,1)	(0,1,2)	(0,1,3)	(0,1,4)
(0,2,0)	(0,2,1)	(0,2,2)	(0,1,4)	(0,2,4)

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

(1,0,0)	(1,0,1)	(1,0,2)	(1,0,3)	(1,0,4)
(1,1,0)	(1,1,1)	(1,1,2)	(1,1,3)	(1,1,4)
(1,2,0)	(1,2,1)	(1,2,2)	(1,1,4)	(1,2,4)

三維陣列內容

三維陣列索引

- 程式實例ch3_14.py：列出三維陣列特定索引的陣列元素。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_14.py =====  
x5[0][2][1] = 11  
x5[0][1][3] = 8  
x5[1][0][1] = 1  
x5[1][1][4] = 9
```

3-5-3：多維陣列的切片

- 程式實例ch3_15.py：二維陣列切片的應用。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_15.py =====  
x[:,:] = 結果是二維陣列  
[[ 0  1  2  3  4]  
 [ 5  6  7  8  9]  
 [10 11 12 13 14]]  
-----  
x[2,:4] = 結果是一維陣列  
[10 11 12 13]  
-----  
x[:2,:1] = 結果是二維陣列  
[[0]  
 [5]]  
-----  
x[:,4:] = 結果是二維陣列  
[[ 4]  
 [ 9]  
 [14]]  
-----  
x[:,4] = 結果是一維陣列  
[ 4  9 14]
```

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

$x[2,:4]$

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

$x[:,4:]$

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

$x[:,1]$

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

$x[:,4]$

- 程式實例ch3_16.py：使用[][]切片造成錯誤的實例。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_16.py =====  
x[:2,4] = 結果是一維陣列  
[4 9]  
-----  
x[:2][4] = 結果是錯誤  
Traceback (most recent call last):  
  File "D:/OpenCV_Python/ch3/ch3_16.py", line 12, in <module>  
    print(x[:2][4])  
IndexError: index 4 is out of bounds for axis 0 with size 2
```

3-6：陣列水平與垂直合併

- 函數`vstack()`可以垂直合併陣列，此函數的語法如下：
- `x = np.vstack(tup)`
- 上述參數是元組，元組內容是要垂直合併的陣列。
- 程式實例`ch3_17.py`：垂直合併陣列。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_17.py =====
```

```
陣列 1
```

```
[[0 1]
```

```
 [2 3]]
```

```
陣列 2
```

```
[[4 5]
```

```
 [6 7]]
```

```
合併結果
```

```
[[0 1]
```

```
 [2 3]
```

```
 [4 5]
```

```
 [6 7]]
```

3-6-2：陣列水平合併hstack()

- 函數hstack()可以水平合併陣列，此函數的語法如下：
- `x = np.hstack(tup)`
- 上述參數是元組，元組內容是要水平合併的陣列。
- 程式實例ch3_18.py：水平合併陣列。

```
===== RESTART: D:/OpenCV_Python/ch3/ch3_18.py =====
```

```
陣列 1
```

```
[[0 1]
```

```
 [2 3]]
```

```
陣列 2
```

```
[[4 5]
```

```
 [6 7]]
```

```
合併結果
```

```
[[0 1 4 5]
```

```
 [2 3 6 7]]
```