

python 程式設計

第 四 講

串 列 (一)

串列：list

■ 由排成一系列的元素組成

```
a = [ 1 , 2 , 3 , 4 ]      # 四個元素
b = [ 8.3 ]                # 一個元素
c = [ 2 , "cat" , 3.4 ]    # 串列元素可以不同型別
d = [ [ 1 , 2 ] , 3+2j ]   # 串列元素也可以為串列
e = []                     # 空串列
```

■ 串列長度：len()

```
a = len([9,4])             # a 為 2

b = [ [8,5,6] , "dog" ]
c = len(b)                  # c 為 2
```

串列下標：取出單一元素

■ 串列元素可用下標取得

```
a = [ 9 , "中央" , "math" , 2.5 , [2,8] ]
```

a	9	"中央"	"math"	2.5	[2,8]
正向下標	0	1	2	3	4
正向元素	a[0]	a[1]	a[2]	a[3]	a[4]
逆向下標	-5	-4	-3	-2	-1
逆向元素	a[-5]	a[-4]	a[-3]	a[-2]	a[-1]

■ 二維串列：串列元素為串列

```
>>> b = [ [1,2] , [9,4,5] ]  
>>> c = b[0][1]           # c = 2  
>>> d = b[1][0]           # d = 9  
>>> e = len(b)             # e = 2  
>>> f = len(b[1])          # f = 3
```

下標範圍：取出多個元素

如果 `a` 為一串列

- `a[i:j]`: `a[i]` , `a[i+1]` , `a[i+2]` , ... , `a[j-1]`
- `a[i:]`: `a[i]` , `a[i+1]` , `a[i+2]` , ... , `a[-1]`
- `a[:j]`: `a[0]` , `a[1]` , `a[2]` , ... , `a[j-1]`
- `a[i:j:k]`: `a[i]` , `a[i+k]` , `a[i+2k]` , ... , 下標不超過 `j-1`
- `a[:]`: `a[0]` , `a[1]` , `a[2]` , ... , `a[-1]` 全部元素
- `a[::-1]`: `a[-1]` , `a[-2]` , `a[-3]` , ... , `a[0]` 逆向取全部元素

<code>a</code>	<code>[9,3,4,7,8]</code>	原串列
<code>a[2:]</code>	<code>[4,7,8]</code>	第三個元素之後
<code>a[:3]</code>	<code>[9,3,4]</code>	前三個元素
<code>a[1:-1:2]</code>	<code>[3,7]</code>	
<code>a[:]</code>	<code>[9,3,4,7,8]</code>	全部元素
<code>a[::-1]</code>	<code>[8,7,4,3,9]</code>	逆向取元素
<code>a[::-2]</code>	<code>[8,4,9]</code>	
<code>a[1:-1]</code>	<code>[3,4,7]</code>	中間元素
<code>a[-2:0:-1]</code>	<code>[7,4,3]</code>	逆向取中間元素

更動元素 (一)

```
a = [ 9 , '中央' , 'math' , 2.5 , [ 2 , 8 ] ]
```

■ 更改單一元素

動作	a
<code>a[2] = '數學'</code>	<code>[9, '中央', '數學', 2.5, [2, 8]]</code>
<code>a[-1] = []</code>	<code>[9, '中央', '數學', 2.5, []]</code>

更動元素 (二)

■ 更改部份元素

```
>>> a[:2] = [ 'ncu' ]  
>>> a  
['ncu', 'math', 2.5, [2, 8]]
```

將前兩個元素更改為 'ncu'

```
>>> a[-2:] = []  
>>> a  
['ncu', 'math']
```

去除末兩個元素

僅能更改正向緊鄰的元素

```
>>> a = [1,2,3,4,5]  
>>> a[::2] = [10,20]  
>>> a[::-1] = [10,20]  
>>> a[:,1] = 10  
>>> a[:,1] = [10,20]  
>>> a[:] = [10,20]
```

錯誤
錯誤
錯誤, 缺中括號
a 變成兩個元素
同上

更動元素 (三)

■ 末端加元素或串列

```
>>> a = [1,2]
>>> a += [3]                # a = [1,2,3]
>>> a = a + [4,5]           # a = [1,2,3,4,5]
>>> a[len(a):] = [6]        # a = [1,2,3,4,5,6]
>>> a[6:0] = [7]            # a = [1,2,3,4,5,6,7]
```

也可使用 `append` 函式

```
>>> a = [1,2]
>>> a.append(3)              # a = [1,2,3]
>>> a.append([4,5])          # a = [1,2,3,[4,5]]
```

更動元素 (四)

■ 前端加元素或串列

```
>>> a = [4,5]
>>> a = [2,3] + a           # a = [2,3,4,5]
>>> a[:0] = [0,1]          # a = [0,1,2,3,4,5]
```

■ 保留頭尾

```
>>> a = [1,2,3,4,5]
>>> a[1:-1] = []           # a = [1,5]
```


更動元素 (五)

■ 插入元素在某位置之前

```
>>> a = [7,3,9,8]
>>> a[2:0] = [1,2]          # a = [7,3,1,2,9,8]
```

也可使用 `insert` 函式

```
>>> b = [7,3,9]
>>> b.insert(2,6)           # b = [7,3,6,9]
>>> b.insert(2,[1,2])       # b = [7,3,[1,2],6,9]
```

更動元素 (六)

■ 刪除元素

```
>>> a = [1,2,3,4,5,6,7]
>>> del a[-1]                # a = [1,2,3,4,5,6]
>>> del a[:2]                # a = [3,4,5,6]
>>> del a[1:3]               # a = [3,6]
>>> del a                    # a 不存在 !!!
```

❖ `del a` 刪除串列，刪除後串列不存在。
若僅要清空串列，使用 `del a[:]`

```
>>> b = [1,2,3,4,5,6,7]
>>> del b[5:]                # b = [1,2,3,4,5]
>>> del b[:]                 # b = []
```

更動元素 (七)

■ 取出並移除末尾元素

```
>>> a = [1,2,3,4,5]
>>> a.pop()                # a = [1,2,3,4]
5
```

■ 複製串列

```
>>> a = [1,2,3]
>>> b = a[:]                # b 與 a 為相同內容的兩串列
>>> a[2] = 10               # 更改 a 不會影響 b
>>> b
[1, 2, 3]
```

■ 串列指定：串列多一個名稱，沒有複製元素

```
>>> a = [1,2,3]
>>> b = a                   # b 與 a 為同一串列
>>> a[2] = 10               # a = [1,2,10]
>>> b                       # 更改 a 也會影響 b
[1, 2, 10]
```

更動元素 (八)

➤ 串列指定常因疏忽而引起一些變數意外被改動：

```
>>> a = [1,2]
>>> b = [ a , a[:] ]           # b[0] = a , b[1] = a[:]
>>> a[1] = 10                  # a = [1,10]
>>> b                           # b[0][1] 也變更 !!!
[[1, 10], [1, 2]]
```

以下也是同樣：

```
>>> a = [1,2]
>>> b = a                      # b 與 a 為同一串列
>>> c = [ b , b[:] ]           # c[0] = b = a , c[1] = b[:]
>>> a[0] = 0                   # 更改 a[0]
>>> c                           # c[0][0] 也變更 !!!
[[0, 2], [1, 2]]
```

整數空間配置 (一)

- 整數為不可更動型別 (**immutable**) :

整數所在空間一旦存入某數後，即不能在原空間變更其值。
若需更動數值，要另找空間儲存新值，舊的空間隨即捨棄。

a 儲存 3

```
>>> a = 3
```

a
3

a 到新空間儲存 5

```
>>> a = 5
```

3
3

a
5

整數空間配置 (二)

- 整數指定代表兩個整數共享空間，當其中一變數改存其他數值時，兩整數才會分離

a 儲存 3

```
>>> a = 3
```

a
3

b 與 a 共享空間

```
>>> b = a
```

a	b
3	3

a 到新空間儲存 5

```
>>> a = 5
```

b
3

a
5

整數空間配置 (三)

- `id(x)` : 回傳 `x` 的所在記憶空間地址
- `x is y` 或 `x is not y` : 檢查 `x` 與 `y` 是否為(不)相同物件, 等同 `id(x) == id(y)` 與 `id(x) != id(y)`

```
>>> a = 3
>>> b = a
>>> a is b
True
>>> id(a)
140215574788704
>>> id(b)
140215574788704
```

```
>>> a = 5
>>> a is b
False
>>> a is not b
True
>>> id(a)
140215574788768
>>> id(b)
140215574788704
```

❖ 浮點數(`float`)與字串(`string`)也都是不可更動型別, 空間配置如同整數一般

串列空間配置

■ 串列指定：共享空間

a 串列有 1 與 2 兩個元素

```
>>> a = [1,2]
```

a	
1	2

b 與 a 共享空間

```
>>> b = a
```

a	b
1	2

b 增加一個元素，也影響 a

```
>>> b.append(3)
```

a		b	
1	2	3	

b 在新的儲存空間儲存 5

```
>>> b = [5]
```

a		
1	2	3

b
5

串列複製：配置新空間

a 串列有 1 與 2 兩個元素

```
>>> a = [1,2]
```

a	
1	2

b 複製 a 的所有元素到新空間

```
>>> b = a[:]
```

a	
1	2

b	
1	2

b 增加一個元素，不影響 a

```
>>> b.append(3)
```

a	
1	2

b		
1	2	3

b 與 c 共享空間

```
>>> c = b
```

a	
1	2

b	c
1	2
2	3

range 設定串列

■ 將 `range` 物件轉為 `list`

```
a = list( range(4) )           # a = [0,1,2,3]
b = list( range(5,-1,-1) )     # b = [5,4,3,2,1,0]
c = list( range(6) ) [::-1]    # c = [5,4,3,2,1,0] 同上
```

a in A 與 a not in A : a 是否(不)在 A 串列內

- `a in A` : 檢查 `a` 是否在 `A` 串列內, 回傳真假值
- `a not in A` : 檢查 `a` 是否不在 `A` 串列內, 回傳真假值
- 列印不重複數字的三位數, 以十個數字一列顯示 :

```
c = 0
for x in range(1,10):
    for y in range(0,10):
        if x == y : continue
        for z in range(0,10):
            if z in [x,y] : continue
            c += 1
            print( x*100+y*10+z , end=(" " if c%10 else "\n") )

print()
```

輸出:

```
102 103 104 105 106 107 108 109 120 123
124 125 126 127 128 129 130 132 134 135
136 137 138 139 140 142 143 145 146 147
...
```

使用 index 取得元素下標

■ 基本用法

```
>>> dirs = [ "east" , "north" , "west" , "south" ]  
>>> print( dirs.index("west") )
```

```
2
```

```
>>> for d in dirs : print( d , dirs.index(d) )
```

```
east 0  
north 1  
west 2  
south 3
```

串列進階設定 (一): list comprehension

■ 基本型式

```
>>> [ x for x in range(4) ]           # x 為參數，可為其他有效變數名稱  
[0, 1, 2, 3]
```

```
>>> [ t*t for t in [1,2,3] ]  
[1, 4, 9]
```

```
>>> [ t//2+1 for t in range(10) ]  
[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]
```

```
>>> [ [x,x*x] for x in range(3) ]    # 每個元素可為串列  
[[0, 0], [1, 1], [2, 4]]
```

串列進階設定 (二)

■ 有條件式的設定

```
>>> [ x for x in range(10) if x not in [3,4,5] ]  
[0, 1, 2, 6, 7, 8, 9]
```

```
>>> [ x for x in range(10) if x not in range(3,6) ]  
[0, 1, 2, 6, 7, 8, 9]
```

```
>>> [ [x,x*x] for x in range(4) if x%2 ]  
[[1, 1], [3, 9]]
```

■ 多重迴圈設定

```
>>> [ x+y for x in range(3) for y in range(2) ]  
[0, 1, 1, 2, 2, 3]
```

```
>>> [ [x,y] for x in range(3) for y in range(10,12) ]  
[[0, 10], [0, 11], [1, 10], [1, 11], [2, 10], [2, 11]]
```

串列進階設定 (三)

■ 二維陣列設定

```
>>> s , t = 2 , 3
>>> [ [ 0 for j in range(t) ] for i in range(s) ] # 2 x 3 二維陣列
[[0, 0, 0], [0, 0, 0]]
```

串列合成與複製 (一)

■ 合成串列：+

```
a = [1,2,3] + [4,5]
```

```
# a = [1,2,3,4,5]
```

■ 元素複製：*

```
a = [0] * 5
```

```
# a = [0,0,0,0,0]
```

```
b = [1,2] * 3
```

```
# b = [1,2,1,2,1,2]
```

■ 合併使用 + 與 *

```
c = [1]*2 + [2]*3
```

```
# c = [1,1,2,2,2]
```


串列合成與複製 (二)

■ 乘法複製一維串列的空間配置

a 兩個元素共享一個整數空間

```
>>> a = [3]*2
```

a	
3	3

a[1] 另找新空間儲存 5

```
>>> a[1] = 5
```

a	
3	5

兩個 4 與兩個 5 各自共享空間

```
>>> b = [4,5]*2
```

b			
4	5	4	5

b[1] 另找新空間儲存 3

```
>>> b[1] = 3
```

b			
4	3	4	5

乘法複製陷阱 (一)

- 陷阱：以乘法複製串列，元素都佔用同個空間，這種現象很容易造成二維串列元素處理上的疏忽

```
>>> a = [1,2]
>>> b = [a]*2                # b = [[1, 2], [1, 2]]
>>> a[1] = 3                  # 設定 a[1] = 3
>>> b                          # b 的對應位置也變成 3 !!!
[[1, 3], [1, 3]]
```

- 正確方式：使用 `[:]` 複製元素，元素各自獨立

```
>>> a = [1,2]
>>> c = [ a[:] for x in range(2) ]
>>> c[0][0] = 3
>>> c
[[3, 2], [1, 2]]
```

乘法複製陷阱 (二)

■ 乘法複製二維串列的空間配置

a 有兩個元素

```
>>> a = [3,4]
```

a	
3	4

b 的兩個串列與 a 共享空間

```
>>> b = [ a ] * 2
```

a		b	
3	4	[3,4]	[3,4]

更動 a[1] 影響 b[0][1] 與 b[1][1]

```
>>> a[1] = 8
```

a		b	
3	8	[3,8]	[3,8]

乘法複製陷阱 (三)

同樣的

c 的三個串列共享一個串列空間

```
>>> c = [[4,5]]*3
```

c		
[4,5]	[4,5]	[4,5]

c[0][1] c[1][1] c[2][1] 共享空間

```
>>> c[1][1] = 3
```

c		
[4,3]	[4,3]	[4,3]

c[0] 於新空間儲存 [6,8]

```
>>> c[0] = [6,8]
```

c		
[6,8]	[4,3]	[4,3]

複製陷阱 (四)

■ 元素複製二維串列的空間配置

兩個元素自佔不同空間

```
>>> a = [ 3 for i in range(2) ]
```

a	
3	3

兩個元素自佔不同空間

```
>>> b = [4,5]
```

b	
4	5

c 的所有元素各自獨立，互不影響

```
>>> c = [ b[:] for i range(3) ]
```

c		
[4,5]	[4,5]	[4,5]

更改 c[0][1] 不會影響其他元素

```
>>> c[0][1] = 3
```

c		
[4,3]	[4,5]	[4,5]

設定一到三維串列 (一)

■ 一維串列

```
a = [ 1 ] * 3          # a = [1, 1, 1]
b = [ None ] * 3       # b = [None, None, None]
```

❖ None 為 python 常數，代表數值尚未設定

設定一到三維串列 (二)

■ 二維串列

設定 3x2 二維串列

```
>>> c = [ [None]*2 for x in range(3) ]  
>>> c  
[[None, None], [None, None], [None, None]]
```

設定列元素數量遞增的二維串列

```
>>> d = [[None]*(i+1) for i in range(3) ]  
>>> d  
[[None], [None, None], [None, None, None]]
```

設定一到三維串列 (三)

■ 三維串列

設定 3x2x4 三維串列

```
>>> [ [ [0]*4 for j in range(2) ] for k in range(3) ]  
[[[0, 0, 0, 0], [0, 0, 0, 0]], [[0, 0, 0, 0], [0, 0, 0, 0]],  
[[0, 0, 0, 0], [0, 0, 0, 0]]]
```


for 迴圈與一維串列

■ **for** 迴圈內變數可直接取用串列元素

```
s = 0
for x in [1,2,3,4,5] : s += x           # 計算串列元素和

s = 0
for x in [ i for i range(1,6) ] :      # 同上
    s += x
```

for 迴圈與二維串列

- 各別串列元素分別設定到 **x** 與 **y**

```
for x , y in [ [1,2] , [4,5] ] :  
    print( x , '+' , y , '=' , x+y )
```

輸出 :

```
1 + 2 = 3  
4 + 5 = 9
```

- 各別串列元素設定為 **p** 串列

```
for p in [ [1,2] , [4,5] ] :  
    print( p[0] , '+' , p[1] , '=' , p[0]+p[1] )
```

輸出同上

字串分解

■ 使用 `list` 分解字串

```
>>> list( "abc" )  
['a', 'b', 'c']
```

```
>>> list( str(123) )  
['1', '2', '3']
```

■ 分解數字的各位數成串列

```
>>> a = [ int(n) for n in list(str(423)) ]      # a = [4, 2, 3]
```

❖ 以上 `list(str(423))` 的 `list` 可以省略

串列比大小

- 依次比較各元素直到比出大小為止

```
>>> [ 8 , 5 , 3 ] < [ 7 , 6 , 4 ]  
False
```

```
>>> [ 8 , 5 , 3 ] < [ 8 , 6 , 4 ]  
True
```

```
>>> [ 8 , 5 , 3 ] >= [ 8 , 5 , 3 ]  
True
```

```
>>> [ 8 , 5 , 3 ] > [ 8 , 5 ]  
True
```

tuple : 常串列

- **tuple** 為元素一旦設定後即不可更動的串列
- 使用小括號

```
>>> a = (1,2,3)          # 三個元素
>>> a[0] = 5             # 錯誤，不可更動

>>> b = (2,)*4           # 四個 2
>>> c = ()               # 空 tuple
```

- 一個元素的常串列須加逗點

```
>>> d = (5,)            # d 為常串列，僅有一個元素 5
>>> e = (5)             # e 為整數 5，不是常串列
```

tuple 與 list 型別互換

■ 使用 tuple 與 list 互換型別

```
>>> a = [1,2,3]
>>> b = tuple(a)           # b = (1,2,3)

>>> c = (5,4,3)
>>> d = list(c)           # d = [5,4,3]
```

tuple 元素設定

- 使用 `list comprehension` 產生串列再轉為 `tuple` 設定元素

```
>>> e = tuple( [ x for x in range(5) ] )  
>>> e  
(0, 1, 2, 3, 4)
```

- 使用 `+` 與 `*` 產生 `tuple`

```
>>> a = (1,2) + (5,)  
>>> a  
(2, 3, 5)  
  
>>> a = a + (7,8)*2  
>>> a  
(2, 3, 5, 7, 8, 7, 8)
```

使用 tuple 設定資料

■ 快速設定元素

```
>>> ( one , two , three ) = ( 1 , 2 , 3 )
>>> three
3

>>> a = ( 4 , 5 , 6 )
>>> ( four , five , six ) = a
>>> six
6
```

■ 小括號可省略

```
>>> one , two = 1 , 2
>>> two
2
```

■ 對調元素值

```
>>> x , y = 1 , 2
>>> x , y = y , x          # x = 2 , y = 1
```


random package (一)：隨機程式套件

■ 常用函式：

- `random()`：隨意產生一個介於 `[0,1)` 之間浮點
- `randint(a,b)`：隨意產生一個介於 `[a,b]` 之間整數
- `randrange(a,b,s)`：隨意產生一個在 `range(a,b,s)` 內的數字
- `choice(seq)`：隨意由 `seq` 序列 (序列、字串等) 找出一個元素
- `shuffle(seq)`：打亂 `seq` 序列元素

- `import random`：將 `random` 套件程式加入程式中使用，使用時需在函式名稱前加上套件名稱與句點 `random.`，例如 `random.randint(a,b)`

- `from random import *`：同上，但可直接使用函式名稱

random package (二)

■ 操作範例：

```
# 使用 random 套件
import random

# 三個骰子
a = [ random.randint(1,6) for x in range(3) ]

# 52 張撲克牌
b = list( range(52) )

# c 為由 b 隨意取出 10 張，但 b 不變，c 的元素可能重複
c = [ random.choice(b) for x in range(10) ]

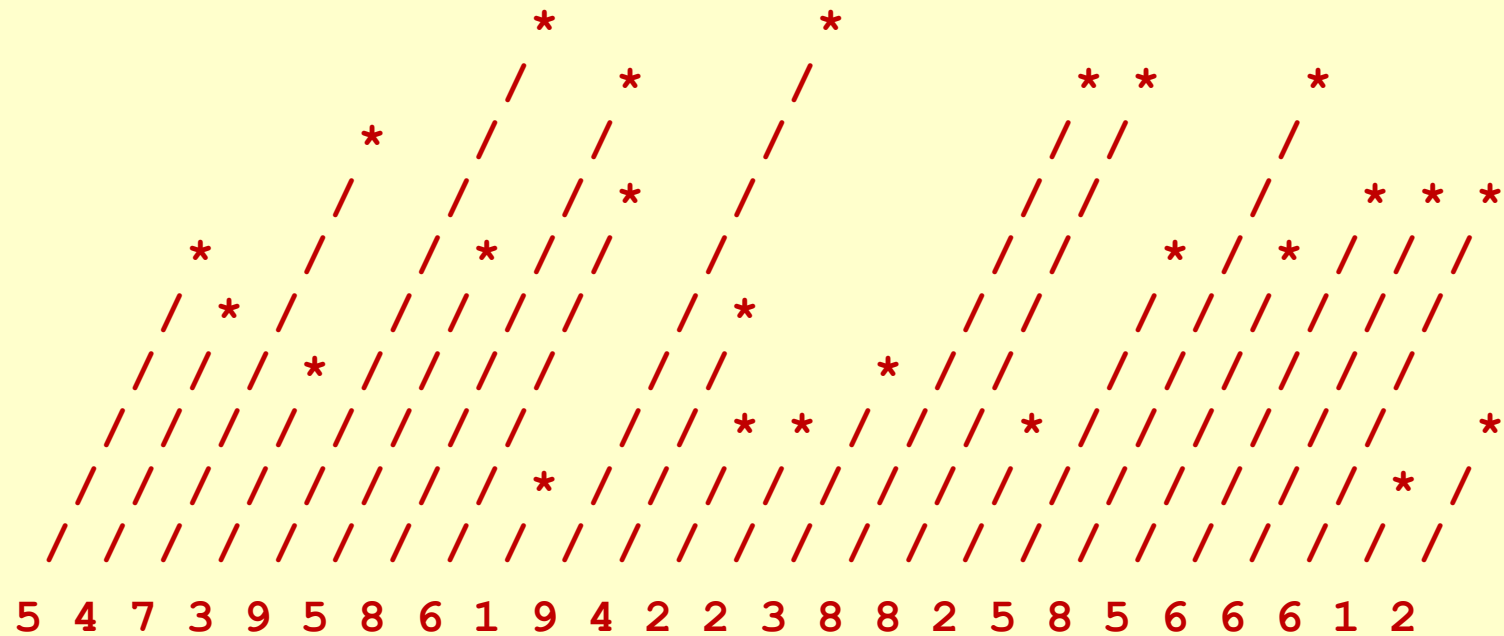
# 打亂 b 串列
random.shuffle( b )

# 分給四人，每人 3 張紙牌
n = 3
cards = [ b[i*n:i*n+n] for i in range(4) ]
print(cards)

輸出：
[[32, 31, 13], [27, 48, 21], [4, 42, 9], [49, 8, 7]]
```

斜條圖 (一)

> 25



斜條圖 (二)

```
import random

while True :

    # 斜條線數量
    n = int( input("> ") )

    # 最大的斜條線長度
    m = 9

    vals = [ random.randint(1,m) for i in range(1,n+1) ]

    # 畫各條斜線
    for h in range(m+1,0,-1) :

        print( " "*(h-1) , end=" " )

        for v in vals :

            if h > v+1 :
                print( " " , end=" " )
            elif h == v+1 :
                print( "*" , end=" " )
            else :
                print( "/" , end=" " )

        print()

    # 輸出長度
    for v in vals :
        print( "{:>2}".format(v) , end=" " )

    print()
```

螺旋數字 (一)

> 4

```
-----  
|  1  |  2  |  3  |  4  |  
-----  
| 12  | 13  | 14  |  5  |  
-----  
| 11  | 16  | 15  |  6  |  
-----  
| 10  |  9  |  8  |  7  |  
-----
```

> 5

```
-----  
|  1  |  2  |  3  |  4  |  5  |  
-----  
| 16  | 17  | 18  | 19  |  6  |  
-----  
| 15  | 24  | 25  | 20  |  7  |  
-----  
| 14  | 23  | 22  | 21  |  8  |  
-----  
| 13  | 12  | 11  | 10  |  9  |  
-----
```

螺旋數字 (二)

```
while True :
```

```
    n = int( input("> ") )
```

```
    # 二維串列儲存數字
```

```
    nums = [ [None]*n for i in range(n) ]
```

```
    # 起始位置
```

```
    s , t = 0 , 0
```

```
    # 四個方向的前進方式
```

```
    ds , dt = [0,1,0,-1] , [1,0,-1,0]
```

```
    # 中間數
```

```
    m = n//2 + ( 1 if n%2 else 0 )
```

```
    # 起始方向
```

```
    dir = 0
```

```
    for i in range(n*n) :
```

```
        nums[s][t] = i + 1
```

```
        # 判斷是否轉彎
```

```
        if s+t==n-1 or ( s >= m and s==t ) or ( s < m and s==t+1 ) :
```

```
            dir += 1
```

```
            if dir == 4 : dir = 0
```

```
        # 更新位置
```

```
        s += ds[dir]
```

```
        t += dt[dir]
```

```
    # 列印數字
```

```
    print( "-"*(5*n+1) )
```

```
    for i in range(n) :
```

```
        for j in range(n) :
```

```
            print( "|{:>3}".format(nums[i][j]) , end=" " )
```

```
        print( "|" )
```

```
    print( "-"*(5*n+1) )
```

```
    print()
```

點矩陣數字 (一)

> 9876543210

9999	8888	7777	6666	5555	4	4	3333	2222	1	0000			
9	9	8	8	7	6	5	4	4	3	2	1	0	0
9999	8888	7	6666	5555	4444	3333	2222	1	0	0			
	9	8	8	7	6	6	5	4	3	2	1	0	0
9999	8888	7	6666	5555	4	3333	2222	1	0000				

點矩陣數字 (二)

```
# 0 到 9 數字點矩陣
bmap = ( (15,9,9,9,15), (2,2,2,2,2), (15,1,15,8,15), (15,1,15,1,15),
         (9,9,15,1,1), (15,8,15,1,15), (15,8,15,9,15), (15,1,2,2,2),
         (15,9,15,9,15), (15,9,15,1,15) )

# 每個矩陣的橫列數與直行數
R , C = len(bmap[0]) , 4

while True :

    num = input("> " )

    # nos 為 num 的各個位數串列
    nos = [ int(s) for s in list(num) ]           # num 也可不用置於 list() 內

    print()

    # 數字的每一列
    for r in range(R) :

        # 每個數字
        for n in nos :

            # 數字的每一行
            for c in range(C-1,-1,-1) :

                s = bmap[n][r] & ( 1 << c )

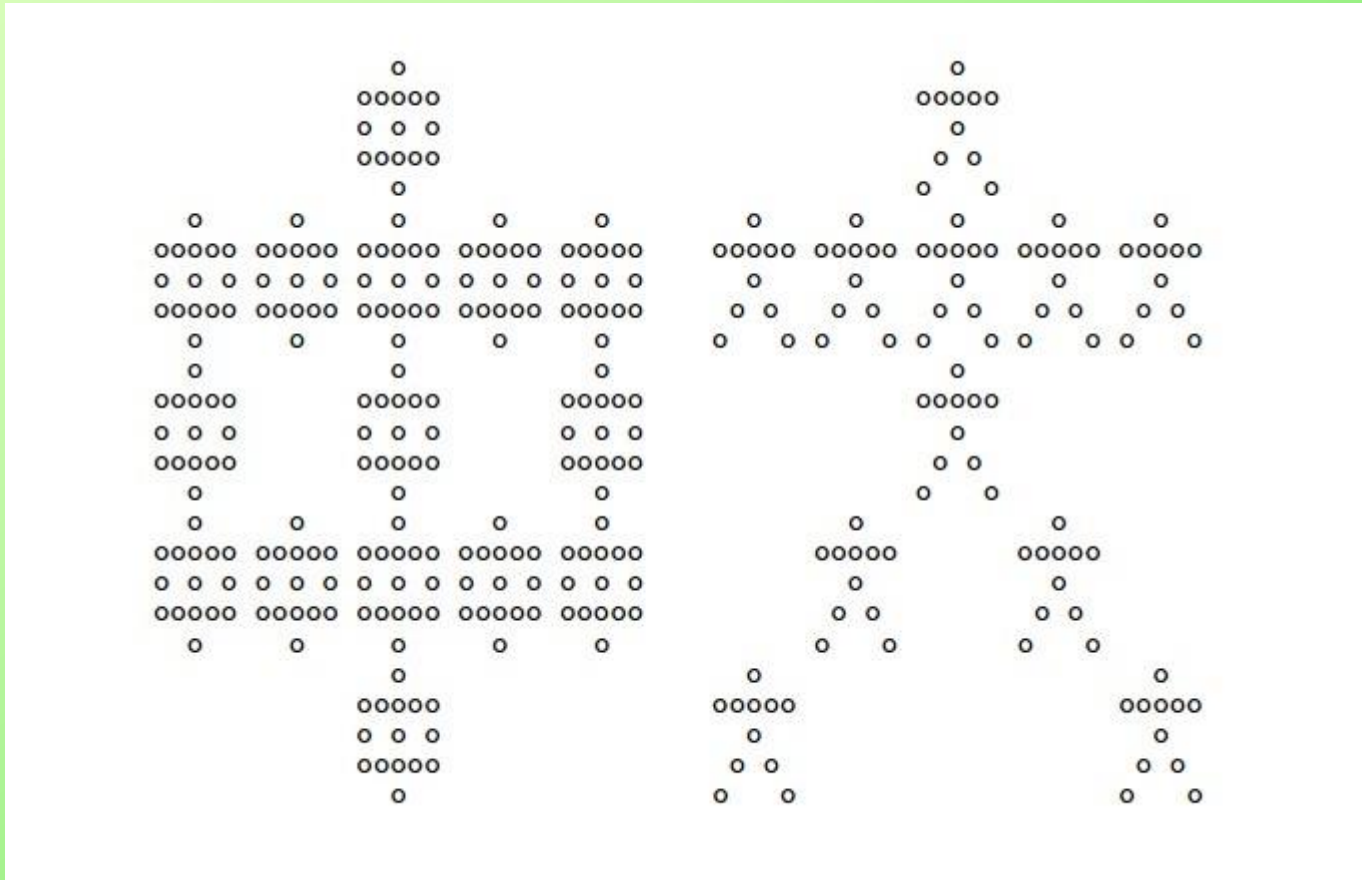
                print( n if s else " " , end="" )

            print( " " , end="" )

        print()

    print()
```


中大雙重點矩陣 (一)



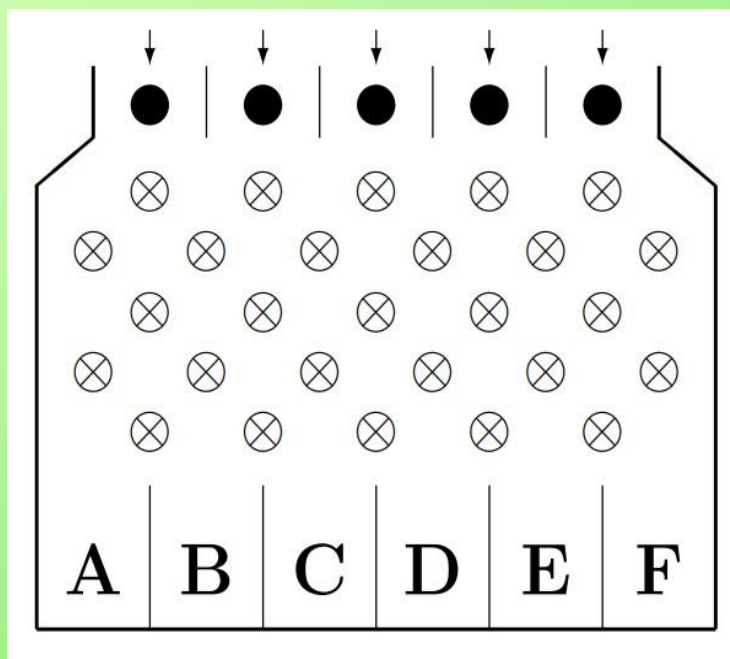
中大雙重點矩陣 (二)

```
# 「中大」兩字的點矩陣
ncu = ( (0x4,0x1f,0x15,0x1f,0x4) , (0x4,0x1f,0x4,0xa,0x11) )
R , C = len(ncu[0]) , 5

# 大橫向
for s in range(R):
    # 小橫向
    for r in range(R):
        print( " " , end="" )
        # 每個中文字
        for k in range(len(ncu)):
            # 大縱向
            for t in range(C-1,-1,-1):
                # 小縱向
                for c in range(C-1,-1,-1):
                    # 檢查列印條件是否滿足
                    if ( ncu[k][r] & ( 1 << c ) ) and ( ncu[k][s] & ( 1 << t ) ):
                        print( "o" , end="" )
                    else:
                        print( " " , end="" )
                print( " " , end="" )
            print( " " , end="" )
        print( " " , end="" )
    print(" ")
```

彈珠臺 (一)

■ 驗證彈珠台各位置的數學機率由左到右分別為： $\frac{24}{160}$ 、 $\frac{25}{160}$ 、 $\frac{31}{160}$ 、 $\frac{31}{160}$ 、 $\frac{25}{160}$ 、 $\frac{24}{160}$



彈珠臺(二)

```
from random import *
n , total = 5 , 50000
counts = [ 0 for x in range(n+1) ]
for k in range(total) :
    # 起始落下的位置
    p = 2*randint(1,n) - 1

    # 第一層釘子
    move = 2*randint(0,1) - 1
    p += move

    # 第二到第五層釘子
    for i in range(2) :
        move = 2*randint(0,1) - 1
        p += move

    # 碰到兩側，提前離開
    if p < 0 or p > 2*n : break
    move = 2*randint(0,1) - 1
    p += move
```

```
# 球數統計
if p < 0 :
    counts[0] += 1
elif p > 2*n :
    counts[-1] += 1
else :
    counts[p//2] += 1

# 列印
for no in counts :
    s = int(160*no/total+0.5)
    print(str(s)+"/160",end=" ")

print()
```

練習題 (一)

1. 撰寫程式，先儲存由中向外遞增的數字於方形矩陣，然後僅列印四個如扇葉的圖案：

> 5

```
5 5 5 5 5          5
  4 4 4 4          4 5
    3 3 3      3 4 5
      2 2 2 3 4 5
5 4 3 2 1 2 3 4 5
5 4 3 2 2 2
5 4 3      3 3 3
5 4      4 4 4 4
5          5 5 5 5 5
```

> 6

```
6 6 6 6 6 6          6
  5 5 5 5 5          5 6
    4 4 4 4      4 5 6
      3 3 3      3 4 5 6
        2 2 2 3 4 5 6
6 5 4 3 2 1 2 3 4 5 6
6 5 4 3 2 2 2
6 5 4 3      3 3 3
6 5 4      4 4 4 4
6 5      5 5 5 5 5
6          6 6 6 6 6 6
```

練習題 (二)

2. 撰寫程式，輸入 n ，產生一 $n \times n$ 二維串列，使得元素呈現以下的排列方式：

> 5

1	2	3	4	5
2	1	2	3	4
3	2	1	2	3
4	3	2	1	2
5	4	3	2	1

> 6

1	2	3	4	5	6
2	1	2	3	4	5
3	2	1	2	3	4
4	3	2	1	2	3
5	4	3	2	1	2
6	5	4	3	2	1

練習題 (三)

3. 撰寫程式，輸入 n ，產生 $2n \times 2n$ 的串列，分四塊存入 $[1, 9]$ 的亂數後印出。

> 3

```
2 2 2 9 9 9
2 2 2 9 9 9
2 2 2 9 9 9
2 2 2 1 1 1
2 2 2 1 1 1
2 2 2 1 1 1
```

> 4

```
6 6 6 6 9 9 9 9
6 6 6 6 9 9 9 9
6 6 6 6 9 9 9 9
6 6 6 6 9 9 9 9
7 7 7 7 3 3 3 3
7 7 7 7 3 3 3 3
7 7 7 7 3 3 3 3
7 7 7 7 3 3 3 3
```

練習題 (四)

4. 分別產生 a 、 b 、 c 三個 $n \times n$ 個串列，內存相同的 $[1, 9]$ 亂數，將之合併起來成為新的 $n \times 3n$ 串列後印出。為了便於分辨，新串列在列印時，各區塊以空白格開。

> 4

```
9 9 9 9 2 2 2 2 5 5 5 5
9 9 9 9 2 2 2 2 5 5 5 5
9 9 9 9 2 2 2 2 5 5 5 5
9 9 9 9 2 2 2 2 5 5 5 5
```

> 5

```
1 1 1 1 1 7 7 7 7 7 9 9 9 9 9
1 1 1 1 1 7 7 7 7 7 9 9 9 9 9
1 1 1 1 1 7 7 7 7 7 9 9 9 9 9
1 1 1 1 1 7 7 7 7 7 9 9 9 9 9
1 1 1 1 1 7 7 7 7 7 9 9 9 9 9
```


練習題 (五)

5. 輸入列數 n ，產生一個二維串列，第一列有一個元素，第二列兩個元素，依此類推，以先縱後橫方式設定遞增數字如下後印出：

> 4

1

2 5

3 6 8

4 7 9 10

> 5

1

2 6

3 7 10

4 8 11 13

5 9 12 14 15

練習題 (六)

6. 同上，但產生以下從上而下的螺旋數字：

> 4

```
1
3 2
4 5 6
10 9 8 7
```

> 5

```
1
3 2
4 5 6
10 9 8 7
11 12 13 14 15
```

練習題 (七)

7. 有兩個 $n \times n$ 的對稱矩陣，元素由 0 或 1 的亂數組成。由於矩陣為對稱關係，兩者都僅存下三角元素。請計算兩對稱矩陣的乘積，印出以下相乘過程。

> 3

```
0 1 1    1 1 1    2 1 2
1 0 0 x 1 0 1 = 1 1 1
1 0 1    1 1 1    2 2 2
```

> 4

```
1 0 0 1    0 1 1 1    1 2 2 1
0 1 0 1    1 1 0 1    2 2 1 1
0 0 1 0 x 1 0 0 1 = 1 0 0 1
1 1 0 1    1 1 1 0    2 3 2 2
```

練習題 (八)

8. 輸入數字，撰寫程式印出巴斯卡三角形。

> 5

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

> 6

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
```

練習題 (九)

9. 同上題，但印出上下對稱的圖案。

> 4

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 3 3 1
  1 2 1
   1 1
    1
```

> 5

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
 1 4 6 4 1
  1 3 3 1
   1 2 1
    1 1
     1
```

練習題 (十)

10. 撰寫程式，產生一個數字不重複的四位數 **foo**，然後再產生 **n** 個數字不重複的四位數 **bar**。比較 **foo** 與每個 **bar** 的數字，若數字相同位置相同的數量為 **A**，數字相同但位置不同的數量為 **B**，印出如以下的輸出。

> 5

6073

1 6170 : 2A1B

2 3247 : 0A2B

3 3742 : 0A2B

4 5724 : 0A1B

5 3062 : 1A2B

> 6

7045

1 5162 : 0A1B

2 1508 : 0A2B

3 8063 : 1A0B

4 1734 : 0A2B

5 6485 : 1A1B

6 5103 : 0A2B

練習題 (十一)

11. 撰寫程式，驗證 n 顆骰子的點數和機率，輸出如下。

> 2

2	3	4	5	6	7	8	9	10	11	12
==	==	==	==	==	==	==	==	==	==	==
1	2	3	4	5	6	5	4	3	2	1
--	--	--	--	--	--	--	--	--	--	--
36	36	36	36	36	36	36	36	36	36	36

> 3

3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
===	===	===	===	===	===	===	===	===	===	===	===	===	===	===	===
1	3	6	10	15	21	25	27	27	25	21	15	10	6	3	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---
216	216	216	216	216	216	216	216	216	216	216	216	216	216	216	216

提示：總擲骰子的次數設為 6^n 的倍數。

練習題 (十二)

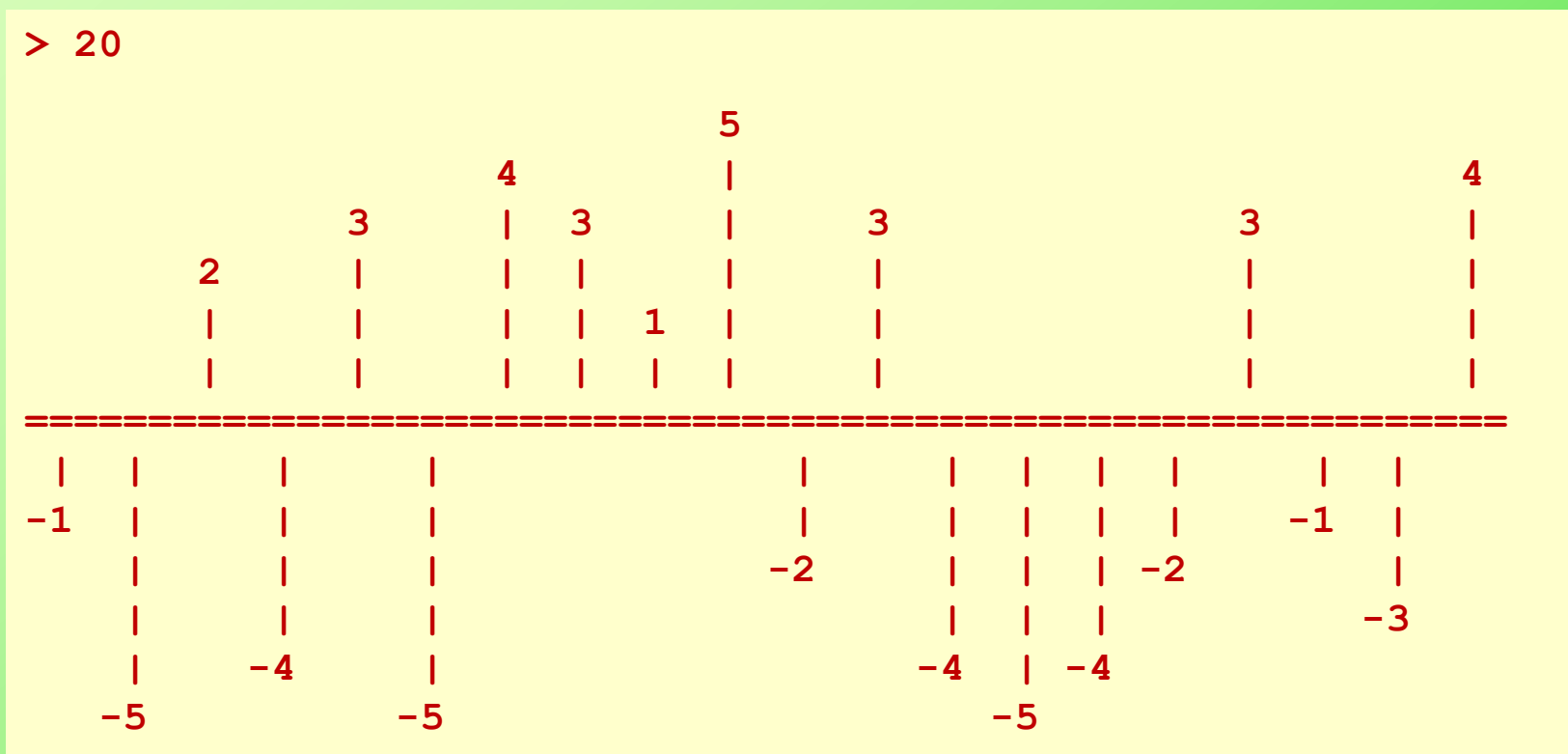
12. 撰寫程式驗證擲出 n 個骰子後, $2 \leq n \leq 7$, 僅有兩個

骰子的點數是一樣的機率 $p(n)$ 為 $\frac{c_1^6 c_1^5 c_1^4 \dots c_1^{8-n} c_2^n}{6^n}$

n	2	3	4	5	6	7
p	$\frac{6}{36}$	$\frac{90}{216}$	$\frac{720}{1296}$	$\frac{3600}{7776}$	$\frac{10800}{46656}$	$\frac{15120}{279936}$

練習題 (十三)

13. 撰寫程式使用亂數函式產生數據介於 $[-5, 5]$ ，但不包含零，然後列印成以下型式的直條圖。



練習題 (十四)

14. 撰寫程式，產生以下由長到短上下對稱斜線。

> 4

```
  *
 / *
/ / *
/ / / *
/ / / /
4 3 2 1
 \ \ \ \
  \ \ \ *
   \ \ *
    \ *
```

> 5

```
  *
 / *
/ / *
/ / / *
/ / / / *
5 4 3 2 1
 \ \ \ \ \
  \ \ \ \ *
   \ \ \ *
    \ \ *
     \ *
```

練習題 (十五)

15. 參考方塊螺旋數字範例，利用座標轉換將結果印為鑽石螺旋圖案。

> 4

```
      1
    12  2
  11 13  3
10 16 14  4
   9 15  5
    8  6
     7
```

> 5

```
          1
        16  2
      15 17  3
    14 24 18  4
  13 23 25 19  5
 12 22 20  6
 11 21  7
 10  8
   9
```

練習題 (十六)

16. 撰寫程式，輸入數字，印出三角螺旋數字圖案。

> 5

```
  1
12  2
11 13  3
10 15 14  4
 9  8  7  6  5
```

> 6

```
  1
15  2
14 16  3
13 21 17  4
12 20 19 18  5
11 10  9  8  7  6
```

練習題 (十七)

17. 撰寫程式，輸入數字 n ，產生 n 個小綠人點矩圖案：

> 6

```

  **      **      **      **      **      **
 ****     ****     ****     ****     ****     ****
  **      **      **      **      **      **
    **     **     **     **     **     **
    ****     ****     ****     ****     ****     ****
    **** *   **** *   **** *   **** *   **** *   **** *
 *  ** *   *  ** *   *  ** *   *  ** *   *  ** *   *  ** *
*   ** *   *   ** *   *   ** *   *   ** *   *   ** *   *
    ****     ****     ****     ****     ****     ****
    *  *     *  *     *  *     *  *     *  *     *  *
 *   ****   *   ****   *   ****   *   ****   *   ****   *
 *           *   *           *   *           *   *           *
 *           *   *           *   *           *   *           *
**          **          **          **          **          **

```

練習題 (十八)

18. 同上題，但在小綠人的身體以 $[1, n]$ 數字打亂編號。

> 6

```

  **      **      **      **      **      **
 ****     ****     ****     ****     ****     ****
  **      **      **      **      **      **
   **     **     **     **     **     **
   ****     ****     ****     ****     ****     ****
  *44 *    *11 *    *66 *    *22 *    *55 *    *33 *
 * 44 *    * 11 *    * 66 *    * 22 *    * 55 *    * 33 *
* 44 *    * 11 *    * 66 *    * 22 *    * 55 *    * 33 *
  ****     ****     ****     ****     ****     ****
  * *      * *      * *      * *      * *      * *
  *   ****  *   ****  *   ****  *   ****  *   ****  *   ****
  *       *  *       *  *       *  *       *  *       *  *
  *       *  *       *  *       *  *       *  *       *  *
**          **          **          **          **          **
```

練習題 (十九)

19. 撰寫程式，輸入數字，印出點矩陣數字與其倒影。

```
> 87231069
```

```
8888  7777  2222  3333  1  0000  6666  9999
8  8      7      2      3  1  0  0  6      9  9
8888      7  2222  3333  1  0  0  6666  9999
8  8      7  2      3  1  0  0  6  6      9
8888      7  2222  3333  1  0000  6666  9999
```

```
-----
*****  *  *****  *****  *  *****  *****  *****
*  *      *  *              *  *  *  *  *  *      *
*****  *  *****  *****  *  *  *  *****  *****
*  *      *      *      *  *  *  *  *  *      *  *
*****  *****  *****  *****  *  *****  *****  *****
```

練習題 (二十)

20. 撰寫程式，輸入數字，印出縱寬各放大兩倍的點矩陣數字。

> 2390764

```
22222222  33333333  99999999  00000000  77777777  66666666  44      44
22222222  33333333  99999999  00000000  77777777  66666666  44      44
      22      33    99      99    00      00      77    66      44      44
      22      33    99      99    00      00      77    66      44      44
22222222  33333333  99999999  00      00      77      66666666  44444444
22222222  33333333  99999999  00      00      77      66666666  44444444
22      33      99      99    00      00      77      66      66      44
22      33      99      99    00      00      77      66      66      44
22222222  33333333  99999999  00000000      77      66666666      44
22222222  33333333  99999999  00000000      77      66666666      44
```


練習題 (二十一)

21. 撰寫程式，輸入數字，印出傾斜的點矩陣數字。

> 98765

```
          99999999  88888888  77777777  66666666  55555555
          99999999  88888888  77777777  66666666  55555555
        99      99  88      88          77  66          55
        99      99  88      88          77  66          55
      99999999  88888888          77  66666666  55555555
      99999999  88888888          77  66666666  55555555
            99  88      88          77  66      66          55
            99  88      88          77  66      66          55
      99999999  88888888          77  66666666  55555555
      99999999  88888888          77  66666666  55555555
```

練習題 (二十二)

22. 撰寫程式，輸入數字，將此數字的點矩陣上下隨意調整位置後列印出來，點陣背景輸出橫線用以模擬數字訂於木板上的效果。

```
> 8273649018
```

```
-----2222-7777-----4--4-----  
-----2----7-3333-----4--4-----1--8888-  
-----2222---7-----3-----4444-----1--8--8-  
-----2-----7--3333-6666----4-----1--8888-  
-8888-2222---7-----3-6-----4-9999-0000---1--8--8-  
-8--8-----3333-6666-----9--9-0--0---1--8888-  
-8888-----6--6-----9999-0--0-----  
-8--8-----6666-----9-0--0-----  
-8888-----9999-0000-----
```

提示：將整個輸出範圍設定為二維字元串列，每個數字的點陣字元存於二維串列的對應位置。

練習題 (二十三)


23. 某猜獎節目中，來賓可由三扇門中選擇一扇門打開取得門後的獎品，假設三扇門中有一扇門後有一輛車子，其餘的則各有一隻羊。遊戲開始時，來賓先選定一扇門，然後主持人將剩餘兩扇門中有羊的門打開，接下來問來賓是否要變換心意，改選最後未打開的門。請撰寫程式驗證當來賓改選門後，得到車子的機率將會由原來不更換門的 $\frac{1}{3}$ 增加到 $\frac{2}{3}$ 。

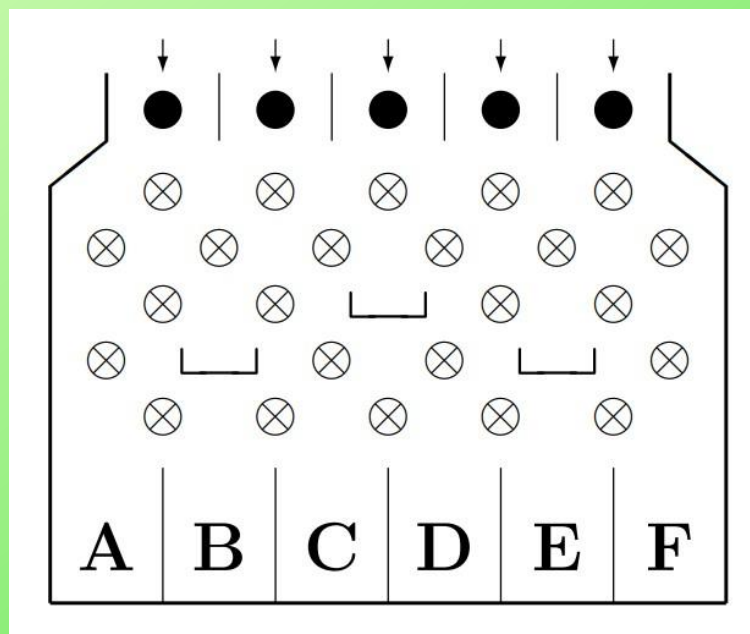
練習題 (二十四)

24. 如上題，假設節目中有 n 扇門，但僅有一扇門後有車子，猜獎的規則同上，撰寫程式驗證當來賓改選門時，獲得車子的機率會從原來的 $\frac{1}{n}$ 增加到 $\frac{n-1}{n(n-2)}$ 。以下為程式執行的結果，每一列輸出資料包含門的數量、模擬得到車子的機率、理論值與原始未換門的機率值。

```
3 : 0.668[0.667] up from 0.333
4 : 0.377[0.375] up from 0.250
5 : 0.258[0.267] up from 0.200
6 : 0.211[0.208] up from 0.167
7 : 0.172[0.171] up from 0.143
8 : 0.149[0.146] up from 0.125
9 : 0.117[0.127] up from 0.111
10 : 0.114[0.113] up from 0.100
```

練習題 (二十五)

25. 有一新式彈珠臺在檯面上增加了三個  平臺，彈珠若滾到平臺上會直接由隱藏在平臺下方的洞口掉離彈珠臺，不會繼續往下滾。撰寫程式驗證彈珠滾到 **A** 到 **F** 各個位置的機率分別為： $\frac{17}{160}$ 、 $\frac{7}{160}$ 、 $\frac{12}{160}$ 、 $\frac{12}{160}$ 、 $\frac{7}{160}$ 、 $\frac{17}{160}$



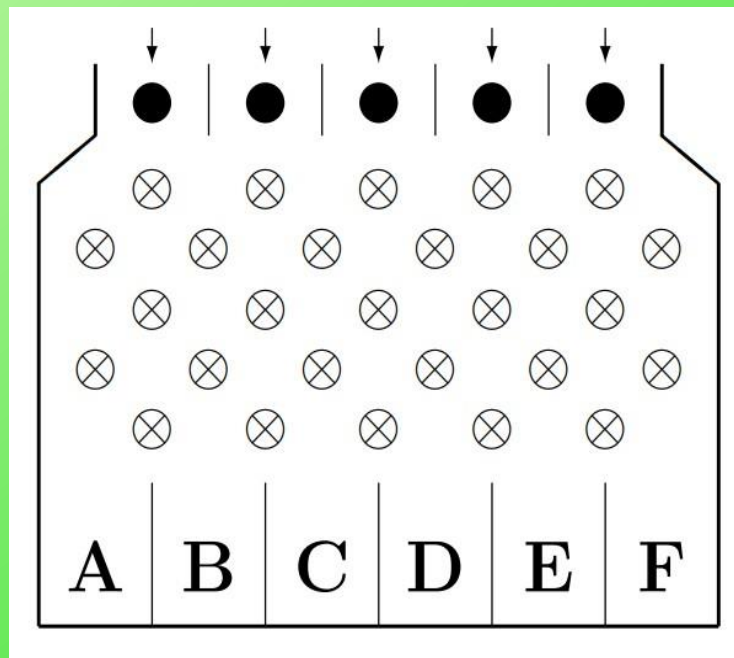
練習題 (二十六)

26. 參考彈珠臺範例，撰寫程式驗證彈珠滾到 **A** 到 **F** 各個位置的機率，同時也要將落在此位置的彈珠球是來自上端入口的機率由左到右一併列印，以下為輸出的內容：

```
24/160 => 17 + 6 + 1 + 0 + 0
25/160 => 9 + 10 + 5 + 1 + 0
31/160 => 5 + 10 + 10 + 5 + 1
31/160 => 1 + 5 + 10 + 10 + 5
25/160 => 0 + 1 + 5 + 10 + 9
24/160 => 0 + 0 + 1 + 6 + 17
```

可定義一個二維串列記錄 **A** 到 **F** 六個位置的球是來由上端五個入口球的數量：

```
froms = [ [0]*5 for x in range(6) ]
```



練習題 (二十七)

27. 修改第 ?? 頁的行道樹程式，使得行道樹可以隨意排列。

> 2

```

      *
    ***
  *****
*****          *           *
*****          ***         ***
*****          *****       *****
      |              |             |
      |              |             |
=====

```

> 3

[illegible]