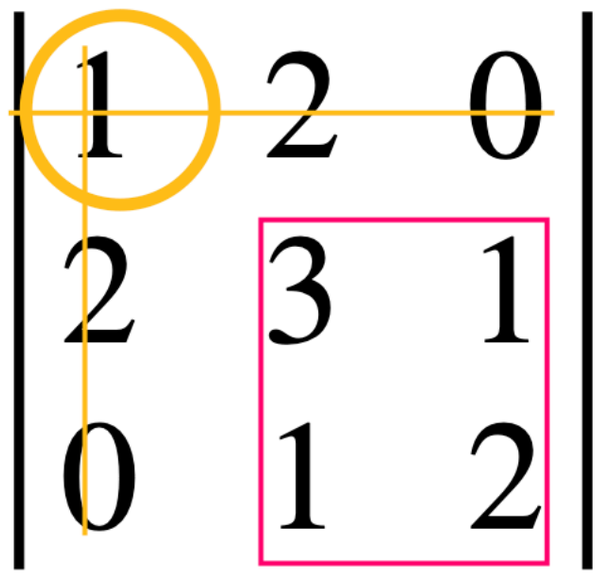
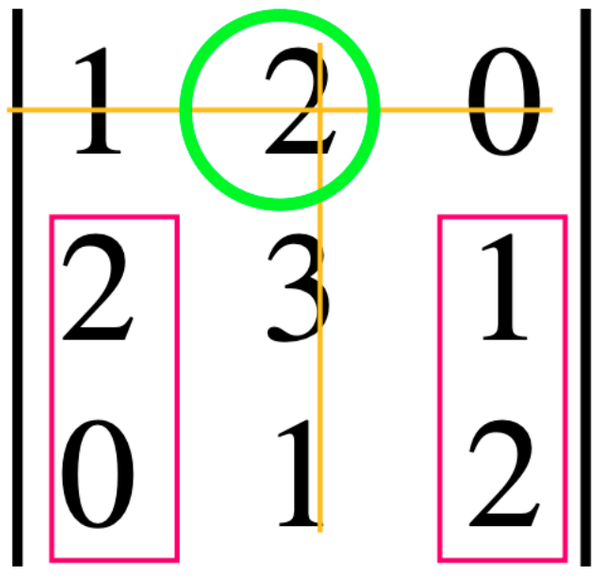
實驗19-20

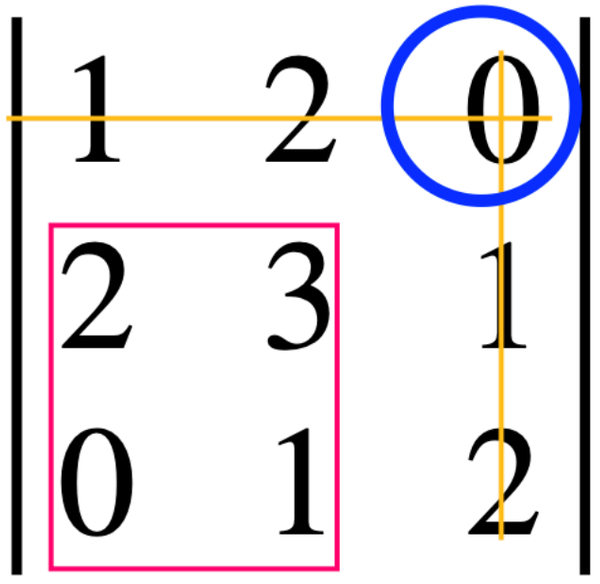
19. 遞迴程式設計：行列式運算

20.物件導向：河內塔遞迴程式設計

EX19: 遞迴程式設計：行列式運算







$$ans = \textcircled{1} \times \begin{vmatrix} 3 & 1 \\ 1 & 2 \end{vmatrix} - \textcircled{2} \times \begin{vmatrix} 2 & 1 \\ 0 & 2 \end{vmatrix} + \textcircled{0} \times \begin{vmatrix} 2 & 3 \\ 0 & 1 \end{vmatrix}$$

問題大小n不是初始問題，在等號右邊使用前一階問題的答案，合成問題大小n的答案表示式

參考連結

步驟一、撰寫mydet模組

```
def mydet(A):
    n,m = A.shape
    if n == 2 and m == 2:
        return
    else:
        ans = 0
        for i in range(0,n):

    return ans
```

B 為(n-1)x(n-1)矩陣

問題大小n不是初始問題，使用前一階問題的答案，合成問題大小n的答案表示式

步驟二、撰寫demo_det程式，測試3x3矩陣行列式運算

```
A = np.matrix([[1,2,0],[2,3,1],[0, 1,2]])
print(A)
print(          (A))
```

```
 [[1 2 0]
  [2 3 1]
  [0 1 2]]
-3
```

步驟三、撰寫demo_det2程式，測試7x7矩陣行列式運算

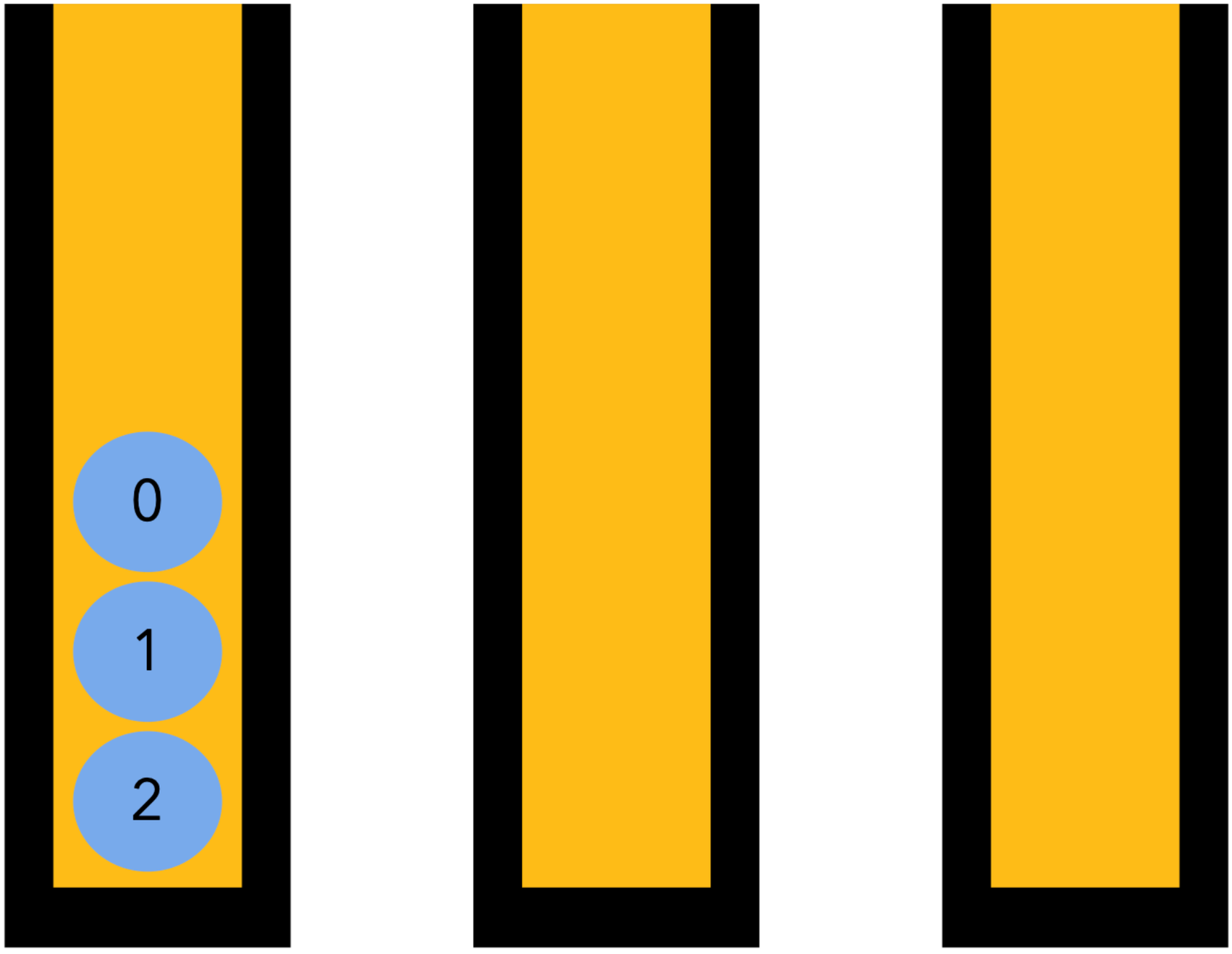
```
n =7
P = np.random.randint(0,7,(n,n))
print(P)
print(mydet(P))
```

Ex20. (50 points)物件導向：河內塔遞迴程式設計

n = 3

初始狀態

移動限制：
編號大的球
不能在編號
小的球上方



起始
堆疊

參考連結

步驟一

修改stack2

```
class Stack:
    def __init__(self,name):
        
    def is_empty(self):
        return self.items == []
    def push(self, data):
        self.items.append(data)
```

Name代表堆疊名稱

```
    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            print("stack is empty")
```

步驟二

設計hanoi_tower模組中的初始化方法

hanoi_tower.py

```
from stack2 import Stack
def init_hanoi_tower(n):
    a = Stack("a")
    b = Stack("b")
    c = Stack("c")

    for i in range(n-1,-1,-1):
        item = str(i)
        a.push(item)
    return a,b,c
```

從模組stack2匯入類別Stack

定義hanoi_tower的初始化方法

宣告變數a的類別為Stack，堆疊名稱為"a"

依序將代表編號n-1,n-2,...,1,0的字串推入堆疊a

回傳堆疊a,b,c

步驟三

設計hanoi_tower模組中的解決方法

hanoi_tower.py

```
def solve_hanoi_tower(n,a,b,c):
    if n == 1:
        [                                    ]

        print(inst1)
        print(inst2)
    else:
        [                                    ]
```

N代表問題大小，a代表啟始堆疊，c為目標堆疊

產生inst1與inst2

當問題size為1時，所需執行的搬移動作

當問題size大於1時，
以遞迴的概念
設計需執行的搬移動作
與子問題

步驟四：

設計demo_hanoi程式

demo_hanoi.py

```
from hanoi_tower import *

n = 5
a,b,c = init_hanoi_tower(n)
print(a.items,b.items,c.items)
a,b,c = solve_hanoi_tower(n,a,b,c)
print(a.items,b.items,c.items)
```

步驟五：

設計demo_hanoi2程式

請複製步驟四產生的指令，貼在demo_hanoi2的程式中，進行測試

要設計的河內塔模組

```
from hanoi_tower import *

n = 5
a,b,c = init_hanoi_tower(n)
print("a:",a.items)
print("b:",b.items)
print("c:",c.items)
print()
```

河內塔模組的初始化方法

n=4的手動解決方案

將問題size為4的河內塔從堆疊a搬移到堆疊b

將問題size為1的河內塔從堆疊a搬移到堆疊c

n=4的手動解決方案

將問題size為4的河內塔從堆疊b搬移到堆疊c

```
print("a:",a.items)
print("b:",b.items)
print("c:",c.items)
```