作業研究

1. USD2NTD.py

```python
def usdtontd(x):
    x = x*30
    return x

usd = int(input("Please enter currency in USD:"))

print ("The equivalent NTD is :"+''+str(usdtontd(usd)))
```

第一次寫的時候就是寫得很醜的那種，直接得到 input*30 後 print 出來，後來改成函數的寫法，只有看起來好看一點，實際上是一樣的，遇到錯誤的 input 等等的時候沒有辦法做出回應，也沒有做好擴充的可能，不過題目沒有要求，也就讓它這樣啦。

```
In [4]: runfile('C:/Users/Liao/Desktop/OR/homework/homework2/USD2NTD.py', wdir='C:/Users/Liao/Desktop/OR/homework/homework2')

Please enter currency in USD:100
The equivalent NTD is :3000

In [5]: runfile('C:/Users/Liao/Desktop/OR/homework/homework2/USD2NTD.py', wdir='C:/Users/Liao/Desktop/OR/homework/homework2')

Please enter currency in USD:500
The equivalent NTD is :15000
```

2. Graphical.py

```python
import matplotlib.pyplot as plt
import numpy as np

#####################################################
plt.axis([0, 5, 0, 5]) # x , y axis
plt.xlabel('x1',fontsize=14, color='maroon')
h=plt.ylabel('x2',fontsize=14, color='maroon')
h.set_rotation(0)
plt.title('Operations Research Homework2-2')


#####################################################
```

1~2 行在 import 基本的 module

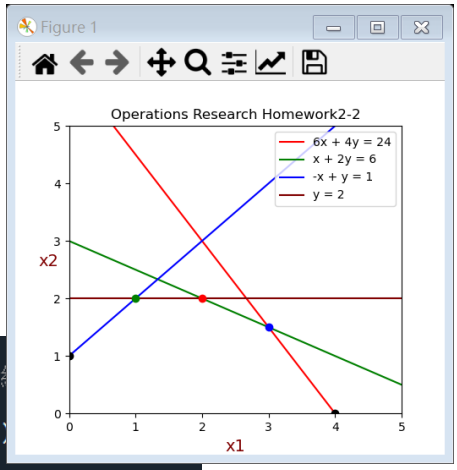
5~9 行在設定圖表的基本參數

13~15 行在圖上畫出紅線

17~19 行在圖上畫出綠線

21~23 行在圖上畫出藍線

26 在圖上畫出棕線




```python
#####################################################
line1x = np.arange(0, 5, 1) # x1座標從0到8，間隔是1，不含結
line1y = (24 - 6*line1x)/4.0 # 6x + 4y = 24
plt.plot(line1x,line1y,color='r',label = '6x + 4y = 24')
#####################################################
line2x = np.arange(0, 8, 1) # x1座標從0到8，間隔是1，不含結束點8
line2y = (6 - line2x)/2     # x + 2y = 6
plt.plot(line2x,line2y,color='g',label = 'x + 2y = 6')
#####################################################
line3x = np.arange(0, 8, 1) # x1座標從0到8，間隔是1，不含結束點8
line3y = 1 + line3x         #-x + y = 1
plt.plot(line3x,line3y,color='b',label = '-x + y = 1')
#####################################################

plt.axhline(y=2,color='maroon',label = 'y = 2 ') # y = 2
#####################################################
```

```
28
29    line12left = np.matrix([[6,4],[1,2]])
30    line12right = np.matrix([[24],[6]])
31    point12 = np.linalg.solve(line12left,line12right)
32
33    line24left = np.matrix([[1,2],[0,1]])
34    line24right = np.matrix([[6],[2]])
35    point24 = np.linalg.solve(line24left,line24right)
36
37
38    line34left = np.matrix([[-1,1],[0,1]])
39    line34right = np.matrix([[1],[2]])
40    point34 = np.linalg.solve(line34left,line34right)
41
42
43    linex1left = np.matrix([[0,1],[6,4]])
44    linex1right = np.matrix([[0],[24]])
45    pointx1 = np.linalg.solve(linex1left,linex1right)
46
47    liney3left = np.matrix([[1,0],[-1,1]])
48    liney3right = np.matrix([[0],[1]])
49    pointy3 = np.linalg.solve(liney3left,liney3right)
50
51    ######################################################
52    plt.plot(point12[0],point12[1],'o',color='b')
53    plt.plot(point24[0],point24[1],'o',color='r')
54    plt.plot(point34[0],point34[1],'o',color='g')
55    plt.plot(pointx1[0],pointx1[1],'o',color='black')
56    plt.plot(pointy3[0],pointy3[1],'o',color='black')
57
58    plt.legend()
59    plt.show()
60
61    ######################################################
```

29~49 行在圖上看出相交的可行解區域的點，再分別用矩陣運算把點解出後，52~59 行把五個點標示在圖上

```
61    ################################################
62    z=[0]*5
63    z[0] = 5*3 +4*1.5
64    z[1] = 5*1 +4*2
65    z[2] = 5*1 +4*2
66    z[3] = 5*4 +4*0
67    z[4] = 5*0 +4*1
68    maxelement =np.amax(z)
69    print(point12[0],point12[1],z[0])
70    print(point24[0],point24[1],z[1])
71    print(point34[0],point34[1],z[2])
72    print(pointx1[0],pointx1[1],z[3])
73    print(pointy3[0],pointy3[1],z[4])
74    print("the max z is :"+ str(maxelement)+" at point "+str(point12[0])+str(point12[1]))
```

```
In [2]: runfile('C:/Users/Liao/Desktop/OR/
homework/homework2/graphical.py', wdir='C:/
Users/Liao/Desktop/OR/homework/homework2')
[[3.]] [[1.5]] 21.0
[[2.]] [[2.]] 13
[[1.]] [[2.]] 13
[[4.]] [[0.]] 20
[[0.]] [[1.]] 4
the max z is :21.0 at point [[3.]][[1.5]]
```

最後把五個點帶入方程式得到解，並取最大值後得到最佳解在(3,1.5)點為

21。

先承認這個程式寫得非常醜，寫得像是把手寫計算過程寫出來而已，完全沒有用到程式的自動化或是幫我們判斷的優點，不過也是因為題目沒有要求要能夠通用在各種情況、不是接收 input 之後再自動判斷自動計算，所以我也沒有寫函式、判斷式等等較複雜一點的程式，而是像流水帳一樣紀錄運算過程而已。

3. ShadowPrice.py

```
1  import numpy as np
2  a = np.matrix( [[6,5],[10,20]] )
3  b = np.matrix( [[60],[150]] )
4  x = np.linalg.solve(a,b)
5  z = (500*x[0]+450*x[1])
6
7  b_adjust = np.matrix( [[62],[150]] )
8  x_adjust = np.linalg.solve(a,b_adjust)
9  z_adjust = (500*x_adjust[0]+450*x_adjust[1])
10
11 print('The final x1 and x2 and z is :'+str(x_adjust[0])+str(x_adjust[1])+str(z_adjust))
12 print('The Shadow Price is :'+str(z_adjust-z))
13
```

2~5 行在計算未更改的原點的 z，7~9 行計算更改後的 z，之後印出答案

```
In [3]: runfile('C:/Users/Liao/Desktop/OR/
homework/homework2/ShadowPrice.py', wdir='C:/
Users/Liao/Desktop/OR/homework/homework2')
The final x1 and x2 and z is :[[7.]][[4.]]
[[5300.]]
The Shadow Price is :[[157.14285714]]
```

得出的答案也符合理論，即更改為 2 倍，其 shadow price 也更改為兩倍，即(78+4/7)的兩倍。

這裡比較像是在驗證理論，比較沒什麼特別的地方。

整個作業算是非常佛心，很簡單，不過我沒有學過 python，只有 C++的一些基礎，在寫作業前花了很多時間自學一下 python，好險大部分程式的邏輯都相同很好上手，只有寫法有些不同，像是 array ,matrix 的寫法用法就不太一樣，花了一些時間熟悉。