國立成功大學碩士學分班

動態規畫 第一組專題報告

動態規劃尋找有限預算下餐飲組合最佳解

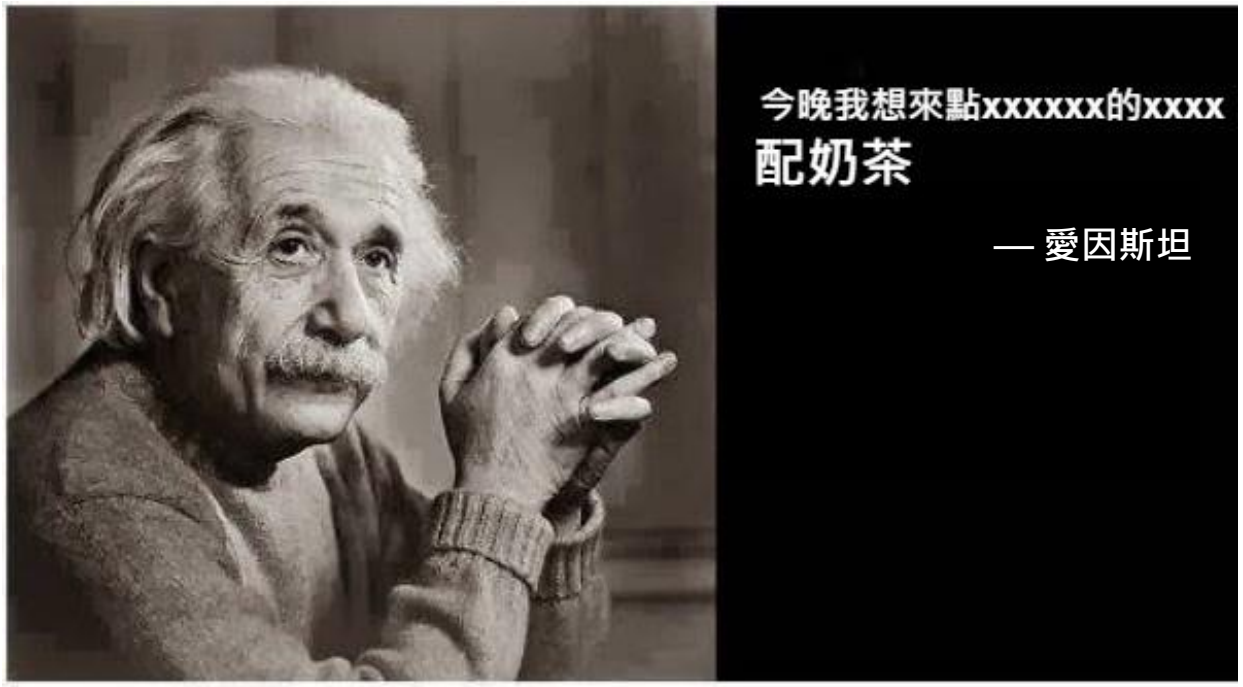
小組成員：
111961018 陳昕瑋
111961031 方明旋
111961023 蔡瑋哲

報告日期：2022年12月27日

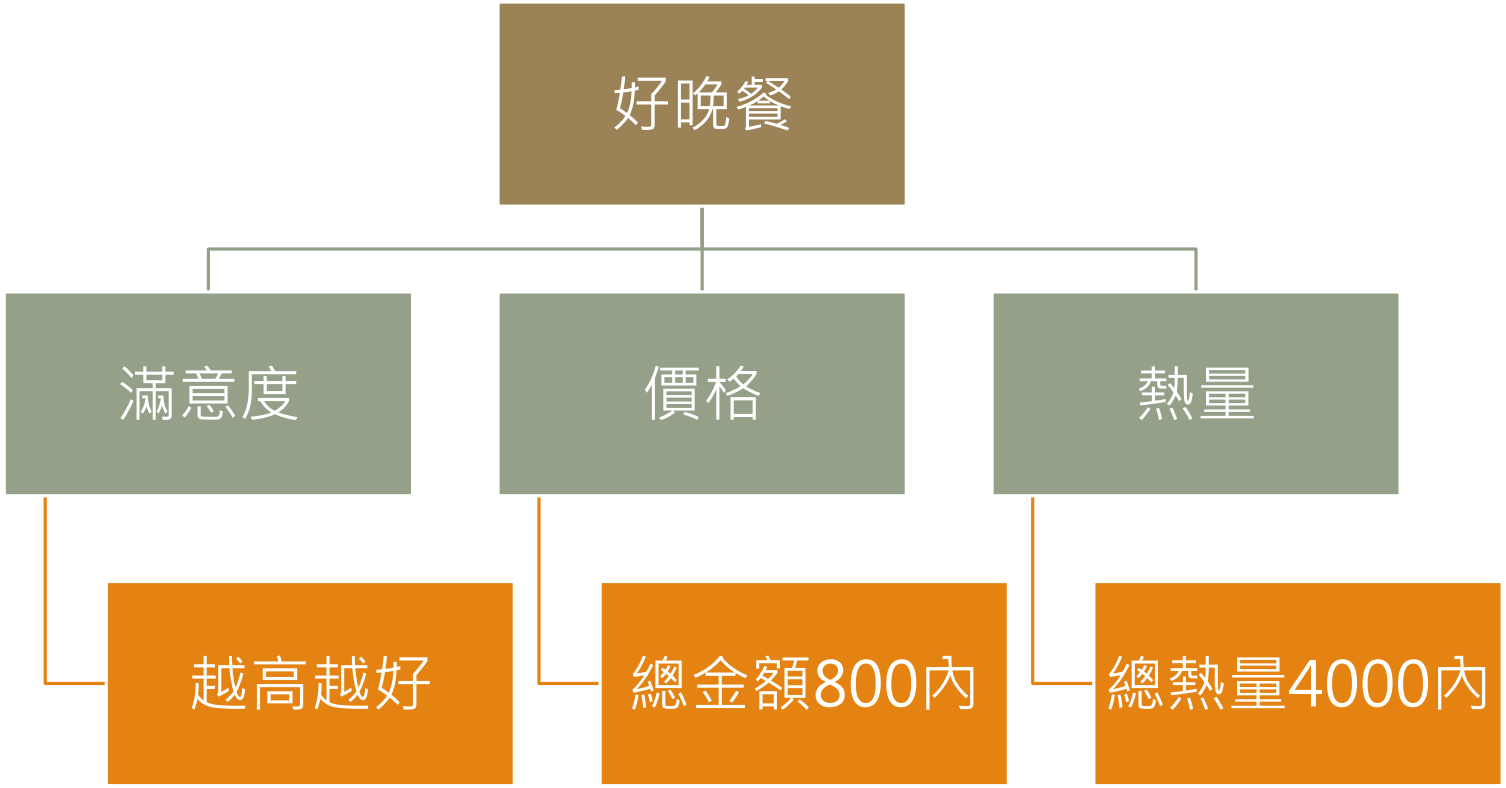
一、背景與動機

民以食為天，每天都需要填飽肚子，人們對的標準隨著食物種類增加越來越高，「晚餐吃什麼？」成了每日最大難題。

以好吃、低熱量、不重複、不傷荷包為目標，使用動態規劃分析，目前規劃一週的餐點組合，之後可以推展成以月為主的飲食計劃。



二、條件設定



二、條件設定

1. 選過不能再選
2. 整體滿意度=(-熱量(正規後)*10)+滿意度
3. 最少必須保留剩下餐點裡，剩餘天數個最低價格餐點的價格總和元
    剩餘天數: 2
    剩下餐點: ['叻沙牛肉麵', '石鍋拌飯', '花生雞腿堡', '原汁牛肉麵]
    剩下餐點價格: [160, 140, 129, 115]
    應該剩下: 115+129=244元
4. 最少必須保留剩下餐點裡，剩餘天數個最低熱量餐點的熱量總和卡
5. 有預算上限(800)
6.有熱量上限(4000)
7. 從右表選出符合以上條件整體滿意度最高組合

餐點	熱量	價格	滿意度	整體滿意度	熱量(正規化後)
叻沙牛肉麵	945	160	100	90	1
石鍋拌飯	875	140	88	79	0.89147
花生雞腿堡	716	129	76	70	0.64496
原汁牛肉麵	677	115	74	68	0.5845
炸豬排飯	570	110	71	67	0.4186
咖哩飯	550	120	69	65	0.3876
雞胸肉餐	450	135	68	66	0.23256
雙層牛肉吉士堡	440	105	65	63	0.21705
涼麵	380	60	62	61	0.12403
魷魚羹	370	70	59	58	0.10853
功夫麵	300	65	53	53	0
烤雞排	300	60	50	50	0

二、條件設定

共幾期 T = 7

第幾期 t = {1,2,…,7}

總餐費(預算上限) Sn = 800

總熱量(熱量上限) Sc = 4000

整體滿意度 Xt

三、實際規劃(示意圖)

	S3		X3*	
				洛
S3				
叻沙牛肉麵				0
石鍋拌飯				0
花生雞腿堡				9
原汁牛肉麵				5
魷魚羹				0
功夫麵				5

															X2*			
		新整體滿意度	價格	新整體滿意度	價格	新整體滿意度	價格	新整體滿意度	價格	新整體滿意度	價格	新整體滿意度	價格					
		叻沙牛肉麵	叻沙牛肉	石鍋拌	石鍋拌	花生雞腿	花生雞腿	原汁牛	原汁牛	魷魚羹	魷魚羹	功夫麵	功夫麵					
剩餘金額	S2	S2						X2*							新整體滿意度	價格		
183	叻沙牛肉麵												X		0	X		
203	石鍋拌飯												205		204	210	石鍋拌飯	魷魚羹
214	花生雞腿堡												194		204	199	花生雞腿堡	魷魚羹
228	原汁牛肉麵												180		219	275	原汁牛肉麵	叻沙牛肉麵
273	魷魚羹	199	230	180	210	161	199	158	185	X	X	129	135		199	230	魷魚羹	叻沙牛肉麵
278	功夫麵	190	225	171	205	153	194	149	180	129	135	X	X		190	225	石鍋拌飯	叻沙牛肉麵

三、實際規劃

```python
import pandas as pd


Dish = {'餐點': ['叻沙牛肉麵', '石鍋拌飯', '花生雞腿堡', '原汁牛肉麵', '炸豬排飯', '咖哩飯',
        '雞胸肉餐', '雙層牛肉吉士堡', '涼麵', '魷魚羹', '功夫麵', '烤雞排'],
'熱量': [945, 875, 716, 677, 570, 550, 450, 440, 380, 370, 300, 300],
'價格': [160, 140, 129, 115, 110, 120, 135, 105, 60, 70, 65, 60],
'滿意度': [100, 88, 76, 74, 71, 69, 68, 65, 62, 59, 53, 50],
'整體滿意度': [90, 79, 70, 68, 67, 65, 66, 63, 61, 58, 53, 50]}

T = 7 # 總期數
Sn = 800 # 總餐費(預算上限)
Sc = 4000 # 總熱量(熱量上限)
S = {} #裝各期各value
```

三、實際規劃

```
16  Dish = pd.DataFrame(Dish) # 丟進DF方便篩選
17
18  for t in range(T, 0, -1): # 建立字典中的字典，每個小字典裝當期值
19      S[t] = {}
20      for i in range(0 , len(Dish)):
21          S[t][i] = {}
22          for I in range(0 , len(Dish)):
23              S[t][i][I] = {}
24
```

剩下餐點裡，剩餘天數個最低價格餐點的價格總和元

St		Xt*

St-1	a	b	c	Xt-1*
a	X	ba		
b	ab	X		
c			X	

```
# {3期數: {0上一期: {0這一期:
for t in range(T, 0,
    if t == T:
        for i in rang
    else:
        DishList = []
        for I in range(len(S[t + 1])): # 上一期的餐
            for i in range(len(S[t + 1][I])):
                LastDish = sorted(S[t+1][I]['餐點'].split(','))
                Money = int(Dish[(~Dish['價格'].isin(LastDish))] sort_values(by = '價格')['價格'][:t-1].sum()) # 最低飯錢
                Car = int(Dish[(~Dish['熱量'].isin(LastDish))].sort_values(by = '熱量')['熱量'][:t-1].sum()) # 最低熱量
                Dish2 = Dish[(~Dish['餐點'].isin(LastDish)) &
                    (Dish['價格'] <= Sn - Money - S[t + 1][I]['價格']) &
                    (Dish['熱量'] <= Sc - Car - S[t + 1][I]['熱量'])].reset_index()# 濾掉重複、超出預算、超出熱量餐點
                order = 0
                for a in range(0, len(Dish2)): # 這一期的餐點組合數
                    ThisDish = sorted((S[t + 1][I]['餐點'] + "," + Dish2['餐點'][a]).split(',')) # 過濾重複餐點組合減少運算量
                    if ThisDish not in DishList:
                        S[t][I][order]['餐點'] =  S[t + 1][I]['餐點'] + "," + Dish2['餐點'][a]
                        S[t][I][order]['整體滿意度'] = Dish2['整體滿意度'][a] + S[t + 1][I]['整體滿意度']
                        S[t][I][order]['價格']  = Dish2['價格'][a] + S[t + 1][I]['價格']
                        S[t][I][order]['熱量']  = Dish2['熱量'][a] + S[t + 1][I]['熱量']
                        DishList += [sorted(S[t][I][order]['餐點'].split(','))]
                        order += 1

for
    {0: {'餐點': '叻沙牛肉麵,石鍋拌飯', '整體滿意度
    1: {'餐點': '石鍋拌飯,花生雞腿堡', '整體滿意度
    2: {'餐點': '花生雞腿堡,原汁牛肉麵', '整體滿意
    3: {'餐點': '原汁牛肉麵,炸豬排飯', '整體滿意度
    4: {'餐點': '炸豬排飯,雞胸肉餐', '整體滿意度':
>[t][i]['餐點'] = MaxS # 六個最大值組合
```

St		Xt*

找出各期最佳解

```
64  Opt = {}
65  for t in range(T, 0, -1):
66      MaxS = S[t][0]
67      Order = 0
68      for i in range(len(S[t])):
69          if len(S[t][i]):
70              if S[t][i]['整體滿意度'] >= MaxS['整體滿意度']:
71                  MaxS = S[t][i]
72                  Opt[t] = {Order: MaxS}
73                  if S[t][i]['整體滿意度'] == MaxS['整體滿意度']:
74                      Order += 1
75
76
77  print(Opt)
78
```

最佳解

```
{7: {0: {'餐點': '叻沙牛肉麵', '整體滿意度': 90, '價格': 160, '熱量': 945}},
6: {0: {'餐點': '叻沙牛肉麵,石鍋拌飯', '整體滿意度': 169, '價格': 300, '熱量': 1820}},
5: {0: {'餐點': '叻沙牛肉麵,石鍋拌飯,花生雞腿堡', '整體滿意度': 239, '價格': 429, '熱量': 2536}},
4: {0: {'餐點': '叻沙牛肉麵,石鍋拌飯,花生雞腿堡,雞胸肉餐', '整體滿意度': 305, '價格': 564, '熱量': 2986}},
3: {1: {'餐點': '炸豬排飯,雞胸肉餐,叻沙牛肉麵,石鍋拌飯,咖哩飯', '整體滿意度': 367, '價格': 665, '熱量': 3390}},
2: {2: {'餐點': '咖哩飯,雞胸肉餐,叻沙牛肉麵,石鍋拌飯,雙層牛肉吉士堡,涼麵', '整體滿意度': 424, '價格': 720, '熱量': 3640}},
1: {2: {'餐點': '咖哩飯,雞胸肉餐,叻沙牛肉麵,石鍋拌飯,雙層牛肉吉士堡,涼麵,功夫麵', '整體滿意度': 477, '價格': 785, '熱量': 3940}}}
```

四、小組總結

個人專屬飲食計畫

可隨時增減品項、條件

從自訂餐點清單生成，不怕選到不愛吃的

預算、熱量目標自訂，兩項指標一次控制

條件(健康程度、葷素等)輕鬆加，通通丟給Pandas處理

附錄一 – 加上葷素食判定

新資料表加入素食及葷素欄位

餐點	熱量	價格	喜愛程度	新整體滿意度	葷素
叻沙牛肉麵	945	160	100	90	葷
我是素食	875	140	88	79	素
神奇沙拉	716	129	76	70	素
原汁牛肉麵	677	115	74	68	葷
炸豬排飯	570	110	71	67	葷
咖哩飯	550	120	69	65	葷
雞胸肉餐	450	135	68	66	葷
素肉全餐	440	105	65	63	素
涼麵(素)	380	60	62	61	素
魷魚羹	370	70	59	58	葷
功夫麵	300	65	53	53	葷
烤雞排	300	60	50	50	葷

附錄一 – 加上葷素食判定

此例題食物順序不重要，先裝葷食，等t到吃素天數期再裝素食
設定3天吃素(VeDays = 3)

```
T = 7 # 總期數
Sn = 800 # 總餐費(預算上限)
Sc = 4000 # 總熱量(熱量上限)
S = {} #裝各期各x下的value
IsVe = "葷" # 先裝葷食
VeDays = 3 # 吃素天數
Dish = pd.DataFrame(Dish) # 丟進DF方便篩選
Opt = {}
for t in range(T, 0, -1): # 建立字典中的字典，每個小字典裝當期值
    S[t] = {}
    for i in range(0 , len(Dish)):
        S[t][i] = {}
        for I in range(0 , len(Dish)):
            S[t][i][I] = {}
```

附錄一 – 加上葷素食判定

因為先裝葷，為防止設定每天都吃素(VeDays = T)時因沒東西能放進最後一期而出錯(一樣用Pandas濾掉不符合的餐點)

```
# {3期數: {0上一期: {0這一期:
for t in range(T, 0, -1):
    if t == T:
        Dish2 = Dish[Dish['葷素'] == IsVe].reset_index()
        for i in range(0, len(Dish2)):
            S[t][i][0]['餐點'] = Dish2['餐點'][i]
            S[t][i][0]['整體滿意度'] = Dish2['整體滿意度'][i]
            S[t][i][0]['價格'] = Dish2['價格'][i]
            S[t][i][0]['熱量'] = Dish2['熱量'][i]
```

附錄一 – 加上葷素食判定

等t到吃素天數期(t<= VeDays)再裝素食

```
else:
    DishList = []
    if t <= VeDays: # 是否要開始裝素食
        IsVe = "素" # 後裝素食
    for I in range(len(S[t + 1])): # 上一期的餐點組合數
        for i in range(len(S[t + 1][I])):
            LastDish = sorted(S[t+1][I]['餐點'].split(','))
            Money = int(Dish[(~Dish['價格'].isin(LastDish))].sort_values(by = '價格')[['價格']][:t-1].sum()) # 最低飯錢
            Car = int(Dish[(~Dish['熱量'].isin(LastDish))].sort_values(by = '熱量')[['熱量']][:t-1].sum()) # 最低熱量
            Dish2 = Dish[(~Dish['餐點'].isin(LastDish)) &
                (Dish['葷素'] == IsVe) &
                (Dish['價格'] <= Sn - Money - S[t + 1][I]['價格']) &
                (Dish['熱量'] <= Sc - Car - S[t + 1][I]['熱量'])].reset_index()# 濾掉重複、超出預算、超出熱量餐點、葷或素
```

附錄一 – 加上葷素食判定

最佳解(三天吃素)

```
{7: {0: {'餐點': '叻沙牛肉麵', '整體滿意度': 90, '價格': 160, '熱量': 945}},
6: {0: {'餐點': '叻沙牛肉麵,原汁牛肉麵', '整體滿意度': 158, '價格': 275, '熱量': 1622}},
5: {0: {'餐點': '叻沙牛肉麵,原汁牛肉麵,炸豬排飯', '整體滿意度': 225, '價格': 385, '熱量': 2192}},
4: {0: {'餐點': '叻沙牛肉麵,原汁牛肉麵,炸豬排飯,雞胸肉餐', '整體滿意度': 291, '價格': 520, '熱量': 2642}},
3: {1: {'餐點': '炸豬排飯,雞胸肉餐,叻沙牛肉麵,咖哩飯,我是素食', '整體滿意度': 367, '價格': 665, '熱量': 3390}},
2: {1: {'餐點': '雞胸肉餐,魷魚羹,叻沙牛肉麵,原汁牛肉麵,我是素食,涼麵(素)', '整體滿意度': 422, '價格': 680, '熱量': 3697}},
1: {1: {'餐點': '魷魚羹,功夫麵,叻沙牛肉麵,原汁牛肉麵,我是素食,素肉全餐,涼麵(素)', '整體滿意度': 472, '價格': 715, '熱量': 3987}}}
```

附錄一 – 加上葷素食判定

完整程式碼

```
import pandas as pd

# 選過不能再選
# 整體滿意度=(-熱量(正規後)*10)+喜愛程度
# '熱量(正規後)': [1.0, 0.8914728682170543, 0.6449612403100775,
0.5844961240310077, 0.4186046511627907, 0.3875968992248062,
0.23255813953488372, 0.21705426356589147, 0.12403100775193798,
0.10852713178294573, 0.0, 0.0]
# 有預算上限(800)
# 有熱量上限(4000)
# 最少必須保留剩下餐點裡，剩餘天數個最低價格餐點的價格總和元
   # ex
      # 剩餘天數: 2
      # 剩下餐點: ['叻沙牛肉麵', '石鍋拌飯', '花生雞腿堡', '原汁牛肉麵']
      # 剩下餐點價格: [160, 140, 129, 115]
      # 應該剩下: 115+129元
# 最少必須保留剩下餐點裡，剩餘天數個最低熱量餐點的熱量總和卡


Dish = {'餐點': ['叻沙牛肉麵', '我是素食', '神奇沙拉', '原汁牛肉麵', '炸豬排飯
', '咖哩飯', '雞胸肉餐', '素肉全餐', '涼麵(素)', '魷魚羹', '功夫麵', '烤雞排'],
'熱量': [945, 875, 716, 677, 570, 550, 450, 440, 380, 370, 300, 300],
'價格': [160, 140, 129, 115, 110, 120, 135, 105, 60, 70, 65, 60],
'喜愛程度': [100, 88, 76, 74, 71, 69, 68, 65, 62, 59, 53, 50],
'整體滿意度': [90, 79, 70, 68, 67, 65, 66, 63, 61, 58, 53, 50],
'葷素': ['葷', '素', '素', '葷', '葷', '葷', '葷', '素', '素', '葷', '葷', '葷']}
```

附錄一 – 加上葷素食判定

完整程式碼

```
T = 7 # 總期數
Sn = 800 # 總餐費(預算上限)
Sc = 4000 # 總熱量(熱量上限)
S = {} #裝各期各x下的value
IsVe = "葷" # 先裝葷食
VeDays = 3 # 吃素天數
Dish = pd.DataFrame(Dish) # 丟進DF方便篩選
Opt = {}
for t in range(T, 0, -1): # 建立字典中的字典，每個小字典裝當期值
    S[t] = {}
    for i in range(0 , len(Dish)):
        S[t][i] = {}
        for I in range(0 , len(Dish)):
            S[t][i][I] = {}

# {3期數: {0上一期: {0這一期:
for t in range(T, 0, -1):
    if t == T:
        Dish2 = Dish[Dish['葷素'] == IsVe].reset_index()
        for i in range(0, len(Dish2)):
            S[t][i][0]['餐點'] = Dish2['餐點'][i]
            S[t][i][0]['整體滿意度'] = Dish2['整體滿意度'][i]
            S[t][i][0]['價格'] = Dish2['價格'][i]
            S[t][i][0]['熱量'] = Dish2['熱量'][i]
```

附錄一 – 加上葷素食判定

完整程式碼

```
else:
    DishList = []
    if t <= VeDays: # 是否要開始裝素食
        IsVe = "素" # 後裝素食
    for I in range(len(S[t + 1])): # 上一期的餐點組合數
        for i in range(len(S[t + 1][I])):
            LastDish = sorted(S[t+1][I]['餐點'].split(','))
            Money = int(Dish[(~Dish['價格'].isin(LastDish))].sort_values(by = '價格')[['價格']][:t-1].sum()) # 最低飯錢
            Car = int(Dish[(~Dish['熱量'].isin(LastDish))].sort_values(by = '熱量')[['熱量']][:t-1].sum()) # 最低熱量
            Dish2 = Dish[(~Dish['餐點'].isin(LastDish)) &
                (Dish['葷素'] == IsVe) &
                (Dish['價格'] <= Sn - Money - S[t + 1][I]['價格']) &
                (Dish['熱量'] <= Sc - Car - S[t + 1][I]['熱量'])].reset_index()# 濾掉重複、超出預算、超出熱量餐點、葷或素
            Max = []
            order = 0
            for a in range(0, len(Dish2)): # 這一期的餐點組合數
                ThisDish = sorted((S[t + 1][I]['餐點'] + "," + Dish2['餐點'][a]).split(',')) # 過濾重複餐點組合減少運算量
                if ThisDish not in DishList:
                    S[t][I][order]['餐點'] =  S[t + 1][I]['餐點'] + "," + Dish2['餐點'][a]
                    S[t][I][order]['整體滿意度'] = Dish2['整體滿意度'][a] + S[t + 1][I]['整體滿意度']
                    S[t][I][order]['價格']  = Dish2['價格'][a] + S[t + 1][I]['價格']
                    S[t][I][order]['熱量']  = Dish2['熱量'][a] + S[t + 1][I]['熱量']
                    Max += [S[t][I][order]['整體滿意度']]
                    DishList += [sorted(S[t][I][order]['餐點'].split(','))]
                    order += 1
```

附錄一 – 加上葷素食判定

完整程式碼

```
for l in range(len(S[t])):
    MaxS = S[t][l][0]
    for i in range(len(S[t][l])): # 找出當下最佳解
        if len(S[t][l][i]):
            if S[t][l][i]['整體滿意度'] > MaxS['整體滿意度']:
                MaxS = S[t][l][i]
        else:
            del S[t][l][i] # 清掉多餘dict
    S[t][l] = MaxS  # 只留最大值組合

for i in range(len(S[t])):
    if len(S[t][i]):
        MaxS = S[t][i]
        break
Order = 0
for i in range(len(S[t])):
    if len(S[t][i]):

        if S[t][i]['整體滿意度'] >= MaxS['整體滿意度']:
            MaxS = S[t][i]
            Opt[t] = {Order: MaxS}
            if S[t][i]['整體滿意度'] == MaxS['整體滿意度']:
                Order += 1


print(Opt)
```

附錄二 - Pandas—強大的資料分析處理模組

優點：速度快、上手快、使用靈活、可讀性佳、泛用性高(接受CSV、JSON、SQL、excel等格式)

以此報告中在第9頁第38行程式碼為例(篩選並加總資料)：

使用Pandas：

```
Money = int(Dish[(~Dish['價格'].isin(LastDish))].sort_values(by = '價格')[['價格']][:t-1].sum())
```

手刻：

```
Money = []
for i in range(len(Dish['餐點'])):
        if Dish['餐點'][i] not in LastDish:
                Money += [int(Dish['價格'][i])]
Money.sort()
Money = sum(Money[:t-1])
```

可讀性、可編輯性高

#剩下餐點裡，剩餘天數個最低價格餐點的價格總和元

附錄二- Pandas—強大的資料分析處理模組

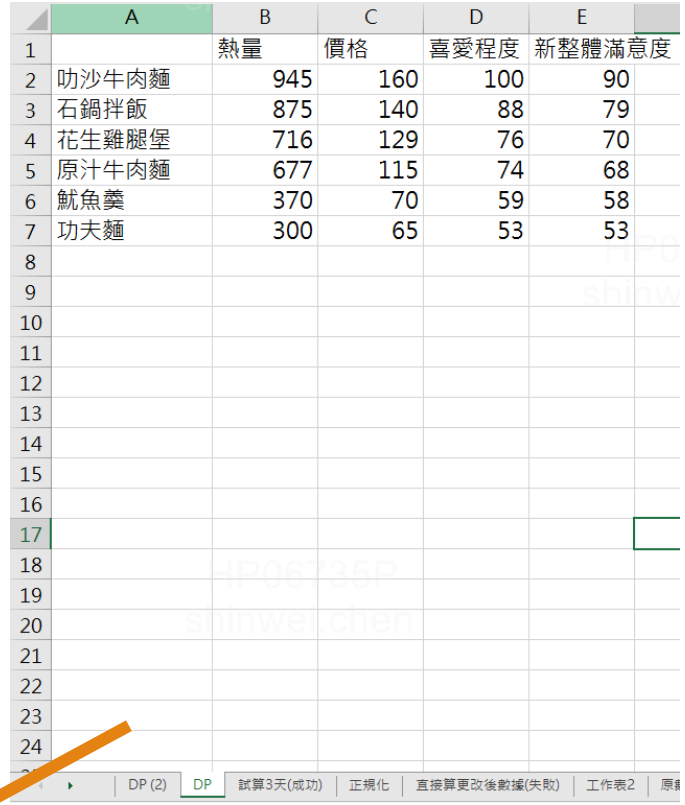
Pandas也能輕鬆將桌面檔案DP2.xlsx中工作表DP的資料轉成dict

```
import pandas as pd
import os.path

File_Path = "D:\\User\\Desktop\\DP2.xlsx"

if os.path.isfile(File_Path):
    df = pd.read_excel(File_Path, sheet_name = "DP").to_dict('list')
    print(df)

else:
    print("404 NOT FOUND")
```



	A	B	C	D	E
1		熱量	價格	喜愛程度	新整體滿意度
2	叻沙牛肉麵	945	160	100	90
3	石鍋拌飯	875	140	88	79
4	花生雞腿堡	716	129	76	70
5	原汁牛肉麵	677	115	74	68
6	魷魚羹	370	70	59	58
7	功夫麵	300	65	53	53

DP (2) | DP | 試算3天(成功) | 正規化 | 直接算更改後數據(失敗) | 工作表2

```
# 輸出結果
{'Unnamed: 0': ['叻沙牛肉麵', '石鍋拌飯', '花生雞腿堡', '原汁牛肉麵', '魷魚羹', '功夫麵'],
'熱量': [945, 875, 716, 677, 370, 300], '價格': [160, 140, 129, 115, 70, 65],
'喜愛程度': [100, 88, 76, 74, 59, 53],
'新整體滿意度': [90, 79, 70, 68, 58, 53]}
```

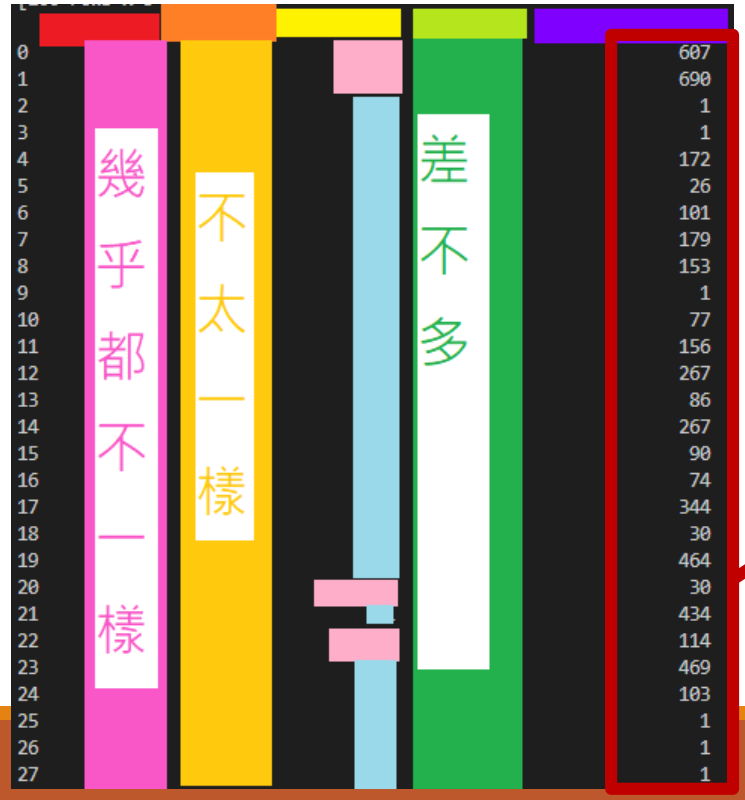
或是抓取DB中資料再做樞紐(將紅到綠欄值相同的紫欄資料相加)

```
from sqlalchemy import create_engine
import pandas as pd

 engine = create_engine(DB的位址和帳密)
 sql =  “一些SQL語法(select * from DB.Table;之類的)”
```

100筆各式各樣的資料，
第一橫排是欄位名稱
(紅色到紫色)

df = pd.read_sql_query(sql, engine) #從DB拿資料並放進Pandas轉成特有的DataFrame格式(如上圖)



df_sum = df.groupby(['紅欄','橘欄','黃欄','綠欄'], as_index=False)['紫欄'].sum()

完成！

五、分工表

題目	報告	程式	數據調整	公式	試算	提供餐點表
陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋
方明旋			方明旋	方明旋		方明旋
						蔡瑋哲

一、背景與動機	二、條件設定	三、實際規劃	四、小組總結	附錄一	附錄二
陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋	陳昕瑋
蔡瑋哲	蔡瑋哲				蔡瑋哲


THANKS FOR
YOUR ATTENTION
memegenerator.es