

國立高雄大學

課程：基礎程式設計

112學年 第1學期 期中議題專題報告

**題目：健康飲食生活與餐廳管理系統**

|  |  |  |  |
| --- | --- | --- | --- |
| 系所：  姓名：  學號： | 系所： 資工系  姓名： 林彧頎  學號：A1105505  系所： 生科系  姓名：黃榛柔  學號：A1106135 | 系所： 金管系  姓名：彭一珊  學號：A1103239  系所： 財法系  姓名：郭珈妤  學號：A1092301 | 系所：  姓名：  學號： |

指導教授：曾智義 教授

中　華　民　國 112 年 11月 14 日

目錄

壹、議題與問題說明……………………………………………………… 1

貳、課堂所學及課外自學運用說明……………………………………… 5

參、Python Coding………………………………………………………… 10

肆、執行結果………………………..…………………………………… 24

伍、參考資料……………………………………………………………… 33

陸、心　　得…………………………………..………………………… 34

**壹、議題與問題說明**

一、議題動機：

我們生活在一個食物供應鏈高度全球化的時代，飲食不僅影響我們的健康，也對環境和社會產生必然的影響。同時，餐廳管理也是需要精密的計劃和資源管理，以確保的長期成功。所以我們想要建立一個健康飲食生活與餐廳管理系統。

以下幾個SDGS發展目標，也與我們的專題息息相關。除了期望可以透過系統，促進各年齡層的健康飲食，以達到SDGs 第三項目標「確保及促進各年齡層健康生活與福祉」。

還有因為餐廳經營需要大量的人力、能源，我們的系統可以幫助餐廳促進節能、提高效率，實現SDG 7第七項目標為「確保所有的人都可取得負擔得起、可靠、永續及現代的能源」。

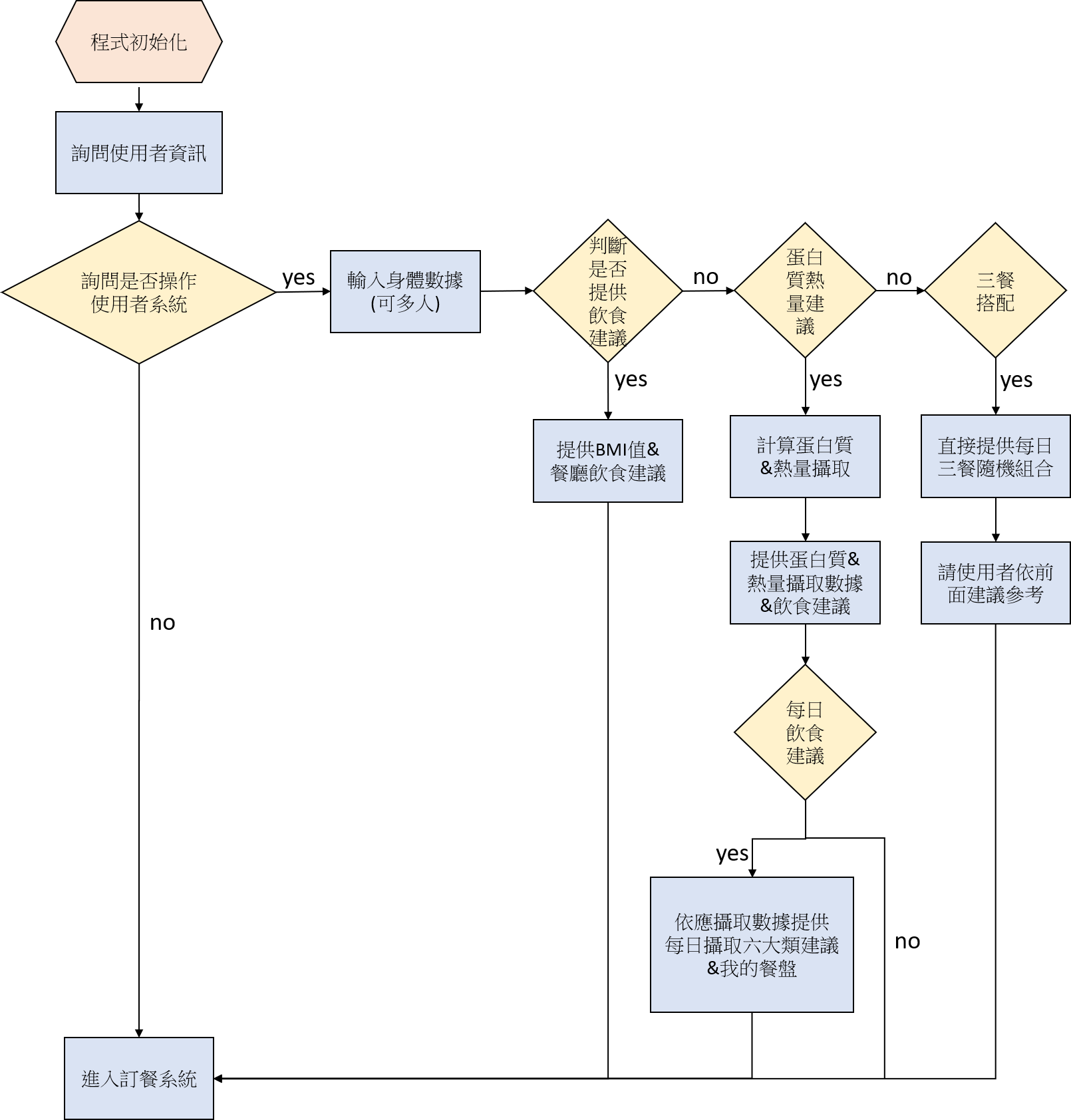
最後通過提供關於飲食需求的信息，有助於餐廳更好地滿足不同人群的需求，以促進SDGs 第十項目標為「減少國內及國家間的不平等」。

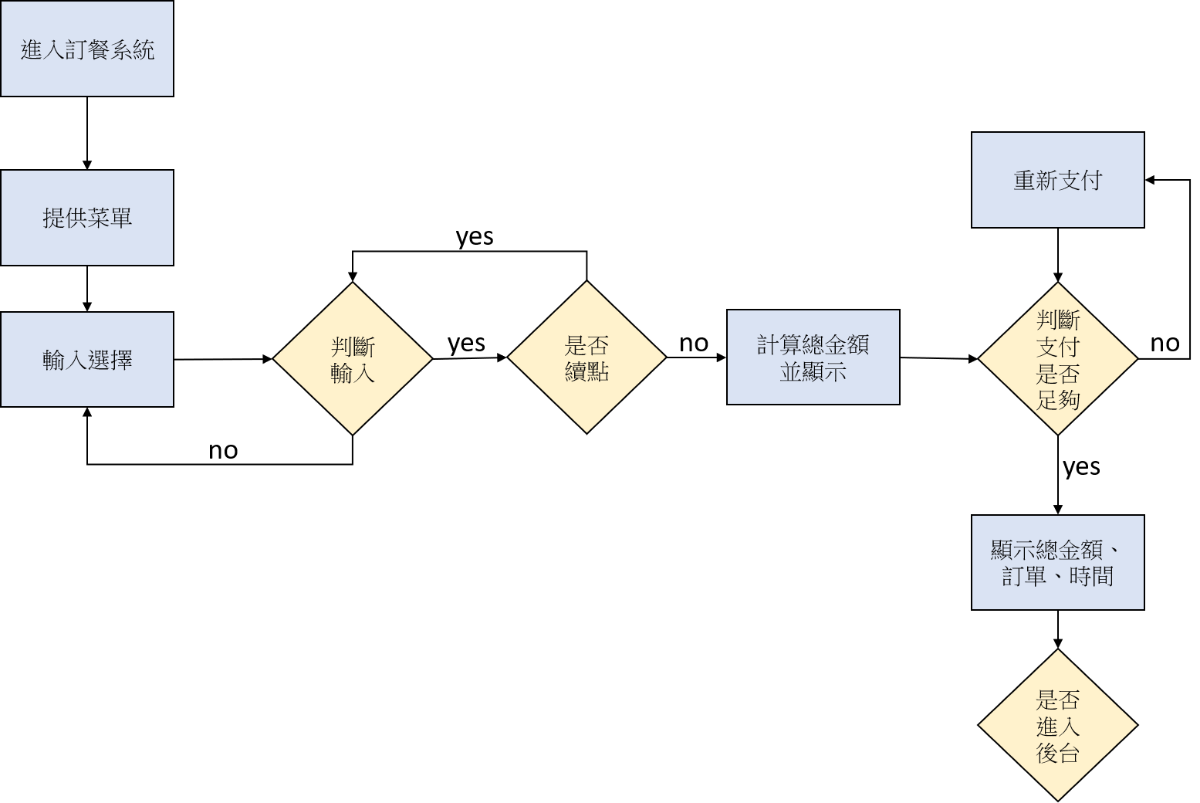
二、問題構想：

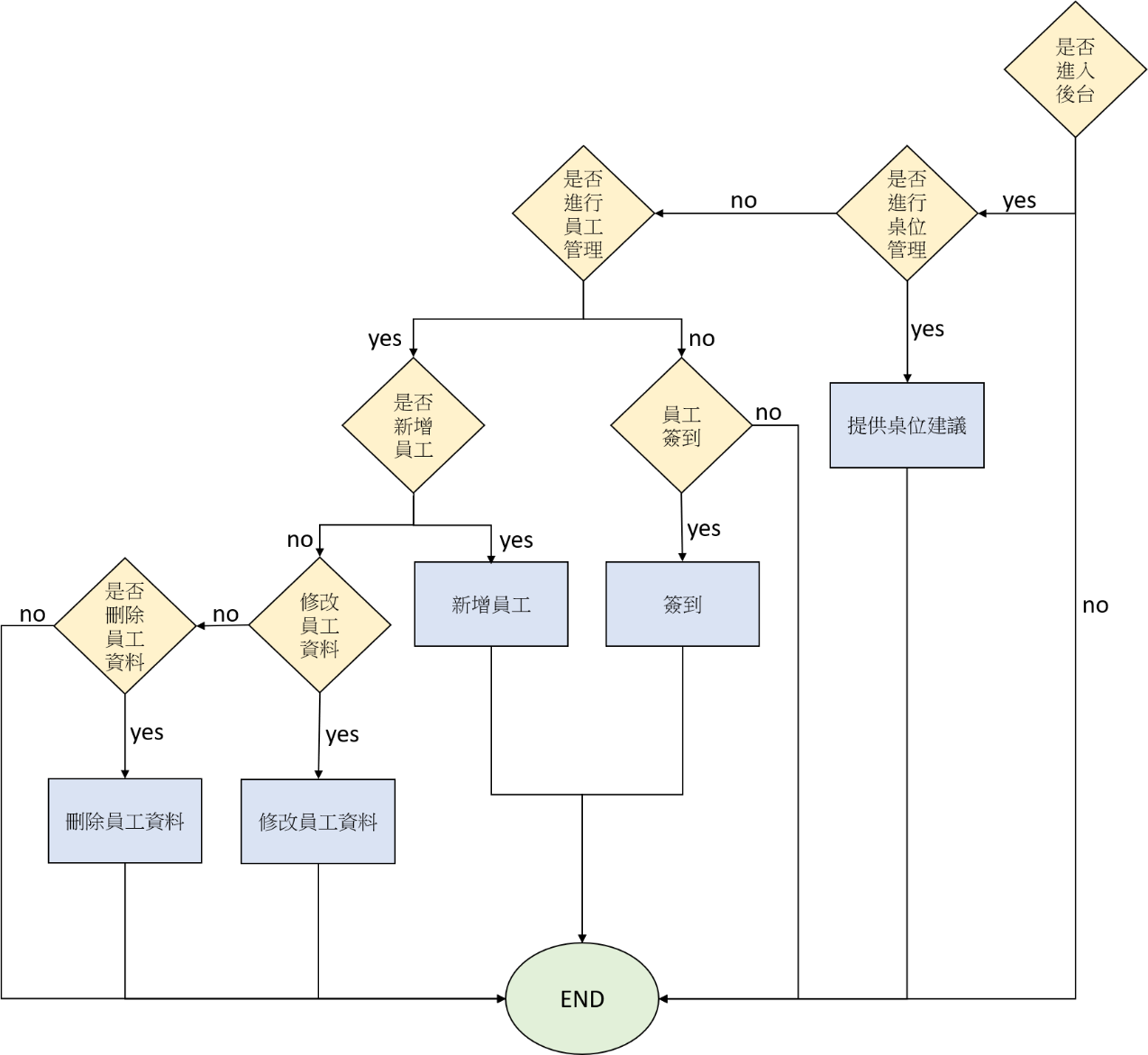
承上述的議題動機，我們設立了一個系統，對餐廳而言，可以進行員工管理、訂餐訂位安排、推銷適合客人的產品；對客人而言，可以藉由提供身體數據計算飲食建議與相關數據分析，並獲得推薦的健康飲食搭配或知道自己選取的餐點是否符合自己的營養需求

1. 使用dict結合list的方法,將不同工作量與體重所需的卡路里相對應，以利之後做相關運算
2. 運用算術運算子計算使用者每日所需熱量與每日飲食建議量
3. 運用dict與list提供使用者建議的食物組合及其熱量
4. 使用tuple與set 使三餐的飲食不僅僅是符合熱量需求的，並且可以隨每一次重新啟動程式，皆有不同的三餐飲食組合
5. 客人輸入訂位相關資訊與自身身體數值,餐廳以此計算客人的BMI並提供飲食建議,使餐廳可行銷自家產品之時,客人也可吃得健康
6. 使用dict與list讓客人仍可自行選擇要吃的餐點,而非僅可選擇餐廳提供的飲食建議,但仍會顯示客人所點餐點的熱量,並使用if幫助客人判斷是否超過其建議攝取熱量
7. 將所點的餐點內容計算出總金額，顯示結果給使用者，並模擬期付款
8. 對餐廳而言, 透過資料輸入訂餐與訂位系統中所需的相關資料，包含:訂餐者、人員數、菜品、菜品數量、用餐時間,使用if幫其自動化的進行座位分配
9. 針對服務人員端新增員工資料管理與簽到系統,以利餐廳進行員工管理

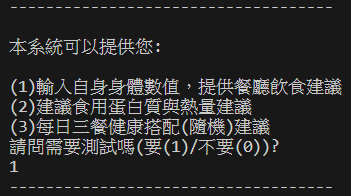
三、方案建立：





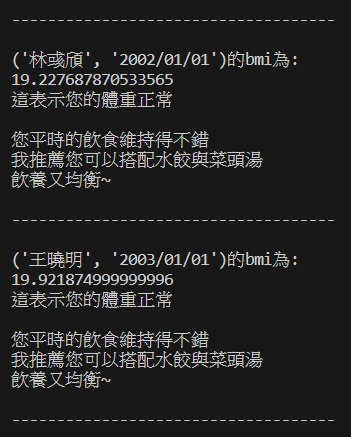


對於上述議題，我們於系統的架設中主要分成3個部分，希望透過這三個部分可以使消費者有達到生活飲食建議、數據分析並達到與人機互動，並可以於餐廳與門市管理系統中，達到服務與商業化的需求，結合人性與商業。其中我們分為使用者端(消費者)、餐廳系統與門市管理系統，我們的系統主要流程為，程式初始化以後，首先(1)會向使用者詢問個人資訊，像是:姓名、使用人數(訂餐人數)，等等…，第二點(2)，我們會向使用者詢問是否使用使用者系統，是的話會進行到使用者系統中，並在依據使用者人數，建一個for迴圈，讓每個使用者都可以輸入自身的身體數據以供系統分析，接著再進入使用者系統主要會進行if、else的判斷，依據判斷會對應到(2-1)是否提供飲食建議、(2-2)是否提供蛋白質與熱量建議與(2-3)是否提供一日三餐隨機搭配，如果皆不執行以上操作則進入訂餐系統(3)。



圖表 1 提供使用者健康生活照護

2-1:我們於飲食建議中，會依照前述(2)的身體數據，進行bmi與餐廳飲食搭配的運算，並將運算結果提供給消費者，進而進入訂餐系統(3)。



圖表 2 飲食建議

2-2:我們於蛋白質與熱量建議，會依照前述(2)的身體數據，進行蛋白質與熱量計算，並提供蛋白質與熱量計算後的數據於消費者，並且我們會近一步詢問使用者是否要提供每日飲食建議，如果為否則跳回訂餐系統(3)，如果為是的話則會依照前面計算後的熱量等等…資訊，提供六大類飲食攝取量建議與我的餐盤建議，結束後進而進入訂餐系統(3)。

2-3:我們於提供一日三餐隨機搭配中，會先提醒使用者可以先操作過(2) 蛋白質與熱量建議後再來執行可以達到更好的建議搭配，進入本系統後我們會直接隨機提供每日三餐組合給消費者，結束後跳回訂餐系統(3)。

進入(3)訂餐系統後，系統會先提供使用者菜單，並請使用者輸入選擇，如果輸入錯誤則會不斷進行while迴圈直到輸入正確，並在輸入正確後，會詢問是否點餐結束，如果未結束的話會重複執行上述步驟，如果結束則會計算總金額，並判斷是否尚未達到之金額，未達到則會不斷進行while迴圈要求重新支付，支付成功後則會輸出所有訂餐結果(包含總金額、時間等等…)。

執行為(3)後會詢問是否進入後台，否會執行系統結束，而如果為是，則會依照if、else if、else判斷要執行的操作為桌位管理、員工管理、簽到系統，期中如果選擇為桌位管理或簽到系統，則執行結束後會執行系統結束，而員工管理中，則會切分成新增、修改、刪除員工資料，這些部分會由if、else if、else判斷執行哪一個，執行結束後會執行系統結束。

**貳、課堂所學及課外自學運用說明**

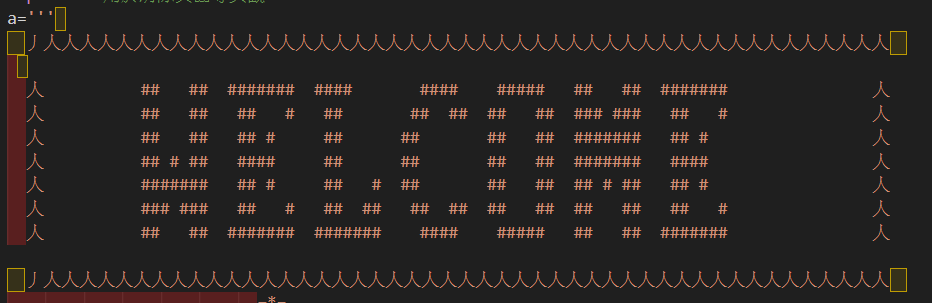
**一、第二章之運用**

1. **變數、運算、字串、比較**

我們在本專題中有許多地方都有使用到**變數、運算、字串**與比較的使用，我們的**變數**可能是存取**字串**也可能是存取值等等…，除此之外，於運算中我們也運用到許多的應用中，例如:bmi計算、價格、總量、身體數值等等...，而這些**變數、運算、字串**的運用，可以幫助我們更好的重複利用會不斷出現的**變數**值，或是可以幫助我們更好的設計介面(如下圖所示)，亦或者是可以透過**運算與比較**，幫助我們有更好的數值分析等等…。

(以下是其中的一個範例程式)

1. #詢問使用者各項身體數值
2. print('請輸入您的年紀、身高、體重、工作狀況')
3. age=int(input('您的年紀是:'))
4. height=float(input('您的身高(cm)是:'))
5. weight=float(input('您的體重(kg)是:'))
6. work=int(input('您的工作量是(輕度工作(0)/中度工作(1)/重度工作(2)):'))
7. bmi=(weight)/(((height)/100)\*\*2)



1.**串列型別(List Type)**

我們在程式中，使用**List**型別來儲存我們的多個關鍵字、數據集合等等，我們主要將List與Dictionary做一個統整的實用，例如於訂餐系統針對list、dictionary增加套餐功能，或是使用者輸入自身身體數值(此針對List、Dict、Tuple、Set)，提供餐廳飲食建議，過程中也有利用取list的第幾個數據進行字串比對，數值運算等等…，這些方法可以幫助我們更好的統整資料與數據，並可以於後續的條件式判斷將我們需要的值取出，更可以說是我們本次專題中不可或缺的一部份，因為如果沒有list，則我們的資料會很難處理，可能需要多個變數等等…。

(以下是其中一個程式範例)

1. age=int(input('您的年紀是:'))
2. height=float(input('您的身高(cm)是:'))
3. weight=float(input('您的體重(kg)是:'))
4. work=int(input('您的工作量是(輕度工作(0)/中度工作(1)/重度工作(2)):'))
5. bmi=(weight)/(((height)/100)\*\*2)
6. #將數據儲存到list
7. value=[age,height,weight,work,bmi]
8. #儲存到user{}
9. user[key\_t]=value

2.**字典形別(Dictionary Type)**

我們在程式中，使用Dict型別來儲存我們的多個關鍵字、數據集合等等…，相較於list他可以說主要負責keyword的部分，幫助我們可以快速找到對應的資料，我們主要將List與Dictionary做一個統整的實用，例如於訂餐系統針對list、dictionary增加套餐功能，或是使用者輸入自身身體數值(此針對List、Dict、Tuple、Set)，提供餐廳飲食建議，過程中也有利用先取Dictionay對應的關鍵字(key方法)，找對應的值(value方法)，並與list搭配，將dictionay中list的第幾個數據進行字串比對，數值運算等等…，這些方法可以幫助我們更好的統整資料與數據，並可以於後續的條件式判斷將我們需要的值取出，更可以說是我們本次專題中不可或缺的一部份，因為如果沒有dict，則我們的資料會很難處理，可能需要多個變數等等…。

(以下是其中一個程式範例)

1. print('請參考以下\n每日飲食建議量')
2. #使用變數，可幫助之後有其他大卡的數據可以直接修正
3. Kcal\_1200=[1.5,3,1.5,3,2,4]
4. Kcal\_1500=[2.5,4,1.5,3,2,4]
5. Kcal\_1800=[3,5,1.5,3,2,5]
6. Kcal\_2000=[3,6,1.5,4,3.5,6]
7. Kcal\_2200=[3.5,6,1.5,4,3.5,6]
8. Kcal\_2500=[4,7,1.5,5,4,7]
9. Kcal\_2700=[4,8,2,5,4,8]
10. diet\_tips={1200:Kcal\_1200,1500:Kcal\_1500,1800:Kcal\_1800
11. ,2000:Kcal\_2000,2200:Kcal\_2200,2500:Kcal\_2500,2700:Kcal\_2700}
12. if(kcal<=1200):
13. i=1200
14. elif(kcal>1200 and kcal<=1500):
15. i=1500
16. elif(kcal>1500 and kcal<=1800):
17. i=1800
18. elif(kcal>1800 and kcal<=2000):
19. i=2000
20. elif(kcal>2000 and kcal<=2200):
21. i=2200
22. elif(kcal>2200 and kcal<=2500):
23. i=2500
24. elif(kcal>2500 and kcal<=2700):
25. i=2700
26. print(f'由於我是{kcal}大卡，因此需要:')
27. print('全榖雜糧類:'+str(diet\_tips[i][0])+'份')
28. print('豆魚蛋肉類:'+str(diet\_tips[i][1])+'份')
29. print('乳品類 :'+str(diet\_tips[i][2])+'份')
30. print('蔬菜類 :'+str(diet\_tips[i][3])+'份')
31. print('水果類 :'+str(diet\_tips[i][4])+'份')
32. print('油脂類 :'+str(diet\_tips[i][5])+'份')

3.**序對形別(Tuple Type)**

我們在程式中，使用Tuple型別來儲存我們的dictionary中key的集合，因為不僅僅可以多個元素組成一起，更有無法一動的特性，十分適合拿來的dictionary的key，而相對地list就會是value，此方法幫助我們可以快速找到對應的資料，我們主要將List、Tuple與Dictionary做一個統整的實用，例如:於使用者輸入自身身體數值(此針對List、Dict、Tuple、Set)，提供餐廳飲食建議，過程中也有利用先取Dictionay對應的關鍵字(key方法)(取Tuple)，找對應的值(value方法)，並與list搭配，將dictionay中list的第幾個數據進行字串比對，數值運算等等…，這些方法可以幫助我們更好的統整資料與數據，並可以將我們的個資統一成一個關鍵字集合，對於抓取個人資料會十分便利，因此可以說是我們本次專題中不可或缺的一部份，因為如果沒有tuple，則我們的資料會很難處理，而且對於多重值組會很難處理等等…。

(以下是其中一個程式範例)

1. #建tuple關於個人資料建立(name,birthday,date)
2. print('請輸入第'+str(i+1)+'位的:姓名、生日')
3. n=input('姓名: ')
4. b=input('生日:yyyy/mm/dd(請依照此格式) ')
5. key\_t=(n,b)
6. #略過一些list存取方法
7. #將數據儲存到list
8. value=[age,height,weight,work,bmi]
9. #儲存到user{}
10. user[key\_t]=value

4.**集合形別(Set Type)**

我們在程式中使用**Set**型別來做隨機編排每日飲食，其中有用到update功能與List的結合，這可以幫助我們更好的對每日飲食進行隨機處理，也是妥善使用set的特性。

1. #list與set的使用(並同時做早餐的重新排序)
2. list\_rice=['鍋貼(餐廳)','水餃(餐廳)']
3. set\_rice=set(list\_rice)
4. #測試update
5. set\_rice.update(['白飯'])
6. set\_rice.update(['紫米飯'])
7. set\_rice.update(['五穀雜糧飯'])
8. list\_rice=list(set\_rice)

**二、第三章之運用**

1.**條件分支(if…elif…else…)**

我們在程式中運用條件分支來達成判斷的功能。例如，在程式結束前與使用者互動，或是讓使用者抉擇要執行的操作時：(以下為其中一個示範)

1. if(test==1):
2. print('------------------------------------\n')
3. print('(1)輸入自身身體數值，提供餐廳飲食建議')
4. print('(2)建議食用蛋白質與熱量建議')
5. print('(3)每日三餐健康搭配(隨機)建議')
6. diet=int(input('您想執行上述哪一個選項(1 or 2 or 3)'))
7. print('------------------------------------\n')
8. time.sleep(3)
9. os.system("cls")
10. if(diet==1):
11. diet\_recommand()
12. if(diet==2):
13. diet\_kcal\_protein()
14. if(diet==3):
15. print('操作此操作前，提醒您~\n熱量建議請操作2獲得每日熱量建議喔~')
16. food()
17. time.sleep(3)
18. os.system("cls")

2.**For迴圈**

當我們在進行比對資料時，利用到for迴圈來完成逐筆比對(可能適用於計算、比對等等…)的功能(以下為其中一個示範)。

1. for s in sum.values():
2. total=int(total)+int(s)
3. print(str(total)+'元')

3.**While迴圈**

當我們需要不斷執行某事件直到可以做出正確行位或是需要不斷執行某操作時，我們就會使用While: (以下為其中一個示範)

1. while(pay<total):
2. print('付款不足!\n')
3. pay=int(input('請付款，請重新輸入付款金額:\n'))
4. if(pay>total):
5. print('找您'+pay-total+'元，謝謝您!\n')
6. if(pay==total):
7. print('整付，謝謝您~\n')

4.**Def 函式**

當我們想將功能劃分開來我們就會使用Def函式，以下是定義的方法:

1. **def name():**
2. #執行的操作

**二、課外自學**

**1.圖片轉ASCLL碼**

利用ASCLL ART將圖片轉成ASCLL碼，可以直接在python呈現圖片

SCLL ART將圖片轉成ASCLL碼，可以直接在python呈現圖片

1. @@@@@@@@@@@@@@@@@@@@%\*+=====+\*#@@@@@@@@@@@@@@@@@@@@@
2. @@@@@@@@@@@@@@@@%+--=\*#%%@%%#\*=--+%@@@@@@@@@@@@@@@@@
3. @@@@@@@@@@@@@@#--\*@@@@@@@@@@@@@@@#--#@@@#-::\*@@@@@@@
4. @@@@@@@@@@@@%-=%@@@@@@@@@@@@@@@@@@@%\*@#- #@@@@@@
5. @@@@@@@@@@@+:%@@@@@@@@@@@@@@@@@@@@@@%= #@@@@@@
6. @@@@@@@@@@==@@@@@@@@@@@@@@@@@@@@@@@\*. :#@@@@@@@
7. @@@@@@@@@==@@@@@@@@@@@@@@@@@@@@@@%- .\*@@@@@@@@@
8. @@@@@@@@+-@@@@@@@@@@@@@@@@@@@@@@\*. +@@@@@@@@@@@
9. @@@@@@@%.@@@#=+#@@@@@@@@@@@@@@@= :%@@:#@@@@@@@@
10. @@@@@@@=+@@+ +@@@@@@@@@@@@%: +@@@@\*-@@@@@@@@
11. @@@@@@@.%@@- \*@@@@@@@@@@%. .#@@@@@@.@@@@@@@@
12. @@@@@@%:@@@%. #@@@@@@@@#. -@@@@@@@@-#@@@@@@@
13. @@@@@@\*-@@@@# .#@@@@@@#. =@@@@@@@@@=\*@@@@@@@
14. @@@@@@\*-@@@@@\* .%@@@@#. +@@@@@@@@@@=\*@@@@@@@
15. @@@@@@%:@@@@@@+ :@@@%. \*@@@@@@@@@@@:#@@@@@@@
16. @@@@@@@.%@@@@@@= :@%. +@@@@@@@@@@@@.@@@@@@@@
17. @@@@@@@=+@@@@@@@- -: +@@@@@@@@@@@@+-@@@@@@@@
18. @@@@@@@%.%@@@@@@@: -@@@@@@@@@@@@@.%@@@@@@@@
19. @@@@@@@@\*-@@@@@@@%. :@@@@@@@@@@@@@-+@@@@@@@@@
20. @@@@@@@@@==@@@@@@@%. .%@@@@@@@@@@@@=-@@@@@@@@@@
21. @@@@@@@@@@=-@@@@@@@#. \*@@@@@@@@@@@@==@@@@@@@@@@@
22. @@@@@@@@@@@\*:#@@@@@@%=::-#@@@@@@@@@@@#:\*@@@@@@@@@@@@
23. @@@@@@@@@@@@%=-#@@@@@@@@@@@@@@@@@@@%--%@@@@@@@@@@@@@
24. @@@@@@@@@@@@@@%=-+%@@@@@@@@@@@@@%\*-=#@@@@@@@@@@@@@@@
25. @@@@@@@@@@@@@@@@@\*=--+\*##%##\*+=-=\*@@@@@@@@@@@@@@@@@@
26. @@@@@@@@@@@@@@@@@@@@@#\*+++++\*#%@@@@@@@@@@@@@@@@@@@@@
27. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

**2.** **math、 datetime、time、os**

以上這幾種方法，分別對應math:計算數學方程式(可幫助我計算一些比較複雜的身體數據)、datetime:可以幫我抓取本機時間、time:可以幫我執行暫停操作、os:可以幫我執行清除頁面(cls)的操作，以下為範例程式碼:

1. time.sleep(1)#time(暫停時間1秒)
2. os.system("cls")#(clear 視窗)
3. now = datetime.datetime.now()#(顯示現在時間)
4. a.sqrt(b)#maath

**參、Python Coding**

一、函式庫匯入

我們 import math、 datetime、time、os以上這幾種方法，分別對應math:計算數學方

程式(可幫助我計算一些比較複雜的身體數據)、datetime:可以幫我抓取本機時間、time:可以幫我執行暫停操作、os:可以幫我執行清除頁面(cls)的操作。 \*參考資料[3~5]

1. import math#maath
2. import datetime#(顯示現在時間)
3. import time#time(暫停時間1秒)
4. import os#用於清除頁面等美觀

二、美觀介面變數宣告與對應之呼叫function

我們在在程式中有做到美觀介面的流程，為了使畫面更加美觀，以下為我們的程式範例，而我們總共有4組圖，分別對應的時機為:歡迎頁面、點餐完畢頁面、後臺管理系統頁面、員工簽到頁面。

1. #美觀介面變數宣告
2. a='''⠀
3. 丿人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人
4. ⠀
5. 人 ## ## ####### #### #### ##### ## ## ####### 人
6. 人 ## ## ## # ## ## ## ## ## ### ### ## # 人
7. 人 ## ## ## # ## ## ## ## ####### ## # 人
8. 人 ## # ## #### ## ## ## ## ####### #### 人
9. 人 ####### ## # ## # ## ## ## ## # ## ## # 人
10. 人 ### ### ## # ## ## ## ## ## ## ## ## ## # 人
11. 人 ## ## ####### ####### #### ##### ## ## ####### 人
12. 丿人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人人
13. @@@@@@@@@@@@@@@@@@@@##@@@@@@@@@@%#%@@@%#%@@@@%#%@#%@@@##@@@%\*: -\*#@@@@@@@@@@@@@@@@@@@@@@@@@@@
14. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@+. =@@@@@@@@@@@@@@@@@@@@@@@@@@@@
15. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%: -= .+@@@@@@@@@@@@@@@@@@@@@@@@@@@
16. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@+ \*%= .\*@@@@@@@@@@@@@@@@@@@@@@@@@@
17. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%: -%@%- .#@@@@@@@@@@@@@@@@@@@@@@@@@
18. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@+ .\*@@@%: .#@@@@@@@@@@@@@@@@@@@@@@@@
19. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#\*%@@@@@@@@@@@@@@@@@@@#. =%@@@@#: :%@@@@@@@@@@@@@@@@@@@@@@@
20. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\*. =@@@@@@@@@@@@@@@@@@@- .#@@@@@@#==@@@@@@@@@@@@@@@@@@@@@@@@
21. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#. :%@@@@@@@@@@@@@@@@@\*. =@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
22. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%: .\*@@@@@@@@@@@@@@@@%- .%@@@@@%#%@@@@@%%%@@@@@@@@@@@@@@@@@@
23. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%- :+. +@@@@@#=\*%@@@@@@@\*. +@@@@\*-...:+%\*:...:\*@@@@@@@@@@@@@@@@
24. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@= .+%: -%@@@#. #@@@@@@%- :%@@@+. . .+@@@@@@@@@@@@@@@
25. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@+. =@@- :%@@#: =@@@@@@\* \*@@@@: :%@@@@@@@@@@@@@@
26. @@@@@@@@@@@@@@#+++++++++++++++. -%@@+ .\*@#: :: :%@@@@%: :%@@@@- -%@@@@@@@@@@@@@@
27. @@@@@@@@@@@@@#:................:%@@@\*. =#: :#+ .\*@@@@+ .\*@@@@@%- -%@@@@@@@@@@@@@@@
28. @@@@@@@@@@@@@@%%%%%%%%%%%%%%%#%@@@@@%: :: :#@#. +@@@%: -%@@@@@@@\*: :\*@@@@@@@@@@@@@@@@@
29. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@- :#@@%: :%@@= .\*@@@@@@@@@%+: .+%@@@@@@@@@@@@@@@@@@
30. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\* :#@@@@= .#@#. -%@@@@@@@@@@@%+. .=%@@@@@@@@@@@@@@@@@@@@
31. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%==#@@@@@\*. +@= .#@@@@@@@@@@@@@@%+#@@@@@@@@@@@@@@@@@@@@@@
32. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%: -#. =@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
33. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@= .: .#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
34. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\*. =@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
35. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@%. :%@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
36. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@= +@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
37. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@#=+%@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
38. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
39. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
40. @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
41. (SDGS第三項：確保及促進各年齡層健康生活與福祉)
43. 你好，歡迎光臨日日香~ '''
44. #呼叫變數並印出
45. def start():
46. print(a)

三、系統進入開始判斷：

在系統的一開始會進行相關判斷是否要使用使用者系統，因此這裡會使用到邏輯判斷if else等等…的功能。

1. start()

2. time.sleep(1)

3. os.system("cls")

4. print('以下是一個訂餐訂位系統請依照指示執行!\n')

5. print('------------------------------------\n')

6. #使用者輸入相關資訊

7. print('welcome 日日香~\n')

8. name=input('請問您的大名:\n')

9. print('歡迎您~'+name+'\n')

10. people\_num=int(input('請問'+name+'會有幾位用餐呢?\n'))

11. test=True

12. print('-----------------------------------\n')

13. time.sleep(1)

14. os.system("cls")

15. print('-----------------------------------\n')

16. check=int(input('請問您需要進入使用者健康飲食生活測試嗎\n(要(1)/不要(0))?\n'))

17. print('------------------------------------\n')

四、健康生活飲食系統進入判斷之執行操作選擇：

我們使用if else判斷來判斷是否進入使用者介面，並進行讓使用者決定要執行甚麼操作，其相對操作會對應呼叫的函式庫，並在呼叫函式庫前，讓使用者輸入個人數據，以幫助後續數據分析，並可以輸入多個使用者資料。

1. if(check==1):#如果判斷為1，則讓使用者依照自身需求選擇是否進入操作

2.     time.sleep(1)#系統暫停1秒

3.     os.system("cls") #系統清除頁面

4.     print('------------------------------------\n')

5.     print('本系統可以提供您:\n')

6.     print('(1)輸入自身身體數值，提供餐廳飲食建議')

7.     print('(2)建議食用蛋白質與熱量建議')

8.     print('(3)每日三餐健康搭配(隨機)建議')

9.     test=int(input('請問需要測試嗎(要(1)/不要(0))?\n'))

10.     print('------------------------------------\n')

11.     time.sleep(1) #系統暫停1秒

12.     os.system("cls")#系統清除頁面

13.     p\_num=1

14.     for i in range(people\_num):#執行多人輸入個人數據

15.         personal\_data(i)

16.

17.if(test==1):#將對應的操作呼叫對應function

18.     print('------------------------------------\n')

19.     print('(1)輸入自身身體數值，提供餐廳飲食建議')

20.     print('(2)建議食用蛋白質與熱量建議')

21.     print('(3)每日三餐健康搭配(隨機)建議')

22.     diet=int(input('您想執行上述哪一個選項(1 or 2 or 3)'))

23.     print('------------------------------------\n')

24.     time.sleep(3)

25.     os.system("cls")

26.     if(diet==1):

27.         diet\_recommand()

28.     if(diet==2):

29.         diet\_kcal\_protein()

30.     if(diet==3):

31          print('操作此操作前，提醒您~\n熱量建議請操作2獲得每日熱量建議喔~')

32.         food()

33.     time.sleep(3)

34.     os.system("cls")

五、定義函式：**定義個人數據收集(包含其用到的功能)**

我們使用*user{}***Dictionary**來達成使用者於執行健康飲食生活錢的個人資料與數據輸入的字典建立、並以此定義函數收集使用者的個人資料與身體數據，並使用到**Tuple、List、Dictionary**存取資料的方法。

1. user={}

2.

3. #使用def定義個人數據

4. def personal\_data(i):

5. #建tuple關於個人資料建立(name,birthday,date)

6.

7. print('請輸入第'+str(i+1)+'位的:姓名、生日')

8. n=input('姓名: ')

9. b=input('生日:yyyy/mm/dd(請依照此格式) ')

10. key\_t=(n,b)

11.

12. #詢問使用者各項身體數值

13. print('請輸入您的年紀、身高、體重、工作狀況')

14. age=int(input('您的年紀是:'))

15. height=float(input('您的身高(cm)是:'))

16. weight=float(input('您的體重(kg)是:'))

17. work=int(input('您的工作量是(輕度工作(0)/中度工作(1)/重度工作(2)):'))

18. bmi=(weight)/(((height)/100)\*\*2)

19.

20. #將數據儲存到list

21. value=[age,height,weight,work,bmi]

22.

23. #儲存到user{}

24. user[key\_t]=value

六、定義函式：**定義飲食建議功能(包含其用到的功能)**

我們使用*user{}***Dictionary**的資料並使用對應的index呼叫其中的**List**方法做bmi判斷，並提供對應的建議，並且我們會給予他餐廳飲食建議，不僅使使用者身體更健康，也可以達到商業推廣的目的。

1. #使用def定義飲食建議功能

2. def diet\_recommand():

3. print('------------------------------------\n')

4. for i in user.keys():#會自動的依照使用者的bmi數據給予建議

5. if(user[i][4]<=18.5):

6. print(f'{i}的bmi為\n'+str(user[i][4])+'\n這表示您的體重過輕\n')

7. print('您可以任意點選我們餐廳的食物\n推薦您我們的招牌鍋貼喔~\n')

8. elif(user[i][4]>=18.5 and user[i][4]<24.0):

9. print(f'{i}的bmi為:\n'+str(user[i][4])+'\n這表示您的體重正常\n')

10. print('您平時的飲食維持得不錯\n我推薦您可以搭配水餃與菜頭湯\n飲養又均衡~\n')

11. elif(user[i][4]>=24.0):

12. print(f'{i}的bmi為:\n'+str(user[i][4])+'\n這表示您的體重過重\n')

13. print('系統不建議您喝紅茶，甜度太高，可以嘗試我們的無糖豆漿喔~\n')

14. print('------------------------------------\n')

七、定義函式：**定義建議食用蛋白質與熱量建議(包含其用到的功能)**

我們使用*user{}***Dictionary**的資料並使用對應的index呼叫其中的**List**方法做相對應的蛋白質攝取量判斷，與熱量建議，並會詢問是否提供每日飲食參考建議，如果有需要，就會輸出我的餐盤與六大類食物對應使用者的數據之攝取量建議。

1. #使用def建議食用蛋白質與熱量建議

2. def diet\_kcal\_protein():

3.         print('------------------------------------\n')

4.         for i in user.keys():

5.             protein=0

6.             kcal=0

7.             #計算蛋白攝取量

8.             if(int(user[i][0]) < 18):

9.                 protein=(user[i][2] \*1.3)

10.             elif(int(user[i][0]) >= 18 and int(user[i][0]) <= 70):

11.                 protein=(user[i][2] \*1.1)

12.             elif(int(user[i][0]) > 70):

13.                 protein=(user[i][2] \*1.2)

14.

15.             #計算熱量攝取量

16.             if(user[i][4] <= 18.5 and user[i][3] == 0):

17.                 kcal=(user[i][2] \*35)

18.             elif(user[i][4] <= 18.5 and user[i][3]==1):

19.                 kcal=(user[i][2] \*40)

20.             elif(user[i][4] <= 18.5 and user[i][3]==2):

21.                 kcal=(user[i][2] \*45)

22.             elif(user[i][4] > 18.5 and user[i][4] < 24.0 and user[i][3]==0):

23.                 kcal=(user[i][2] \*30)

24.             elif(user[i][4] >18.5 and user[i][4] <24.0 and user[i][3]==1):

25.                 kcal=(user[i][2] \*35)

26.             elif(user[i][4] > 18.5 and user[i][4] < 24.0 and user[i][3]==2):

27.                 kcal=(user[i][2] \*40)

28.             elif(user[i][4] >= 24.0 and user[i][3]==0):

29.                 kcal=(user[i][2] \*22.5)

30.             elif(user[i][4] >= 24.0 and user[i][3]==1):

31.                 kcal=(user[i][2] \*30)

32.             elif(user[i][4] >= 24.0 and user[i][3]==2):

33.                 kcal=(user[i][2] \*35)

34.             #顯示使用者的每日蛋白質和熱量攝取量

35.             user\_kcal[i]=kcal

36.             print(f'建議{i}每日應攝取'+str(protein)+'公克的蛋白質，以及'+str(kcal)+'大卡的熱量哦')

37.             print('------------------------------------\n')

38.             t=int(input('請問需要提供您每日飲食建議嗎?(1)要/(0)不要'))

39.             if(t==1):

40.                 print('------------------------------------\n')

41.                 #顯示我的每日飲食建議量

42.                 print('請參考以下\n每日飲食建議量')

43.                 #使用變數，可幫助之後有其他大卡的數據可以直接修正

44.                 Kcal\_1200=[1.5,3,1.5,3,2,4]

45.                 Kcal\_1500=[2.5,4,1.5,3,2,4]

46.                 Kcal\_1800=[3,5,1.5,3,2,5]

47.                 Kcal\_2000=[3,6,1.5,4,3.5,6]

48.                 Kcal\_2200=[3.5,6,1.5,4,3.5,6]

49.                 Kcal\_2500=[4,7,1.5,5,4,7]

50.                 Kcal\_2700=[4,8,2,5,4,8]

51.                 diet\_tips={1200:Kcal\_1200,1500:Kcal\_1500,1800:Kcal\_1800

52.                         ,2000:Kcal\_2000,2200:Kcal\_2200,2500:Kcal\_2500,2700:Kcal\_2700}

53.                 if(kcal<=1200):

54.                     i=1200

55.                 elif(kcal>1200 and kcal<=1500):

56.                     i=1500

57.                 elif(kcal>1500 and kcal<=1800):

58.                     i=1800

59.                 elif(kcal>1800 and kcal<=2000):

60.                     i=2000

61.                 elif(kcal>2000 and kcal<=2200):

62.                     i=2200

63.                 elif(kcal>2200 and kcal<=2500):

64.                     i=2500

65.                 elif(kcal>2500 and kcal<=2700):

66.                     i=2700

67.                 print(f'由於我是{kcal}大卡，因此需要:')#六大類食物攝取量建議

68.                 print('全榖雜糧類:'+str(diet\_tips[i][0])+'份')

69.                 print('豆魚蛋肉類:'+str(diet\_tips[i][1])+'份')

70.                 print('乳品類    :'+str(diet\_tips[i][2])+'份')

71.                 print('蔬菜類    :'+str(diet\_tips[i][3])+'份')

72.                 print('水果類    :'+str(diet\_tips[i][4])+'份')

73.                 print('油脂類    :'+str(diet\_tips[i][5])+'份')

74.

75.                 print('-----------------')#每日餐盤建議

76.                 print('以下為每日餐盤建議')

77.                 print('每天早晚一杯奶\n')

78.                 print('每餐水果拳頭大\n')

79.                 print('菜比水果多一點\n')

80.                 print('飯跟蔬菜一樣多\n')

81.                 print('豆魚蛋肉一掌心\n')

82.                 print('堅果種子一茶匙\n')

83.                 print('-----------------')

84.             print('------------------------------------\n')

八、定義函式：**隨機飲食搭配(包含其用到的功能)**

我們使用**list、set**的結合來做隨機編排每日飲食，其中有用到update功能與List的結合，這可以幫助我們更好的對每日飲食進行隨機處理，也是妥善使用set的特性

1. def food():

2.         print('------------------------------------\n')

3.         print('以下為我的食物組合:')

4.         print('請選擇最適合的~')

5.

6.         print('3餐食物隨機組合:')

7.         #list與set的使用(並同時做早餐的重新排序)

8.         list\_rice=['鍋貼(餐廳)','水餃(餐廳)']

9.         set\_rice=set(list\_rice)

10.         #測試update

11.         set\_rice.update(['白飯'])

12.         set\_rice.update(['紫米飯'])

13.         set\_rice.update(['五穀雜糧飯'])

14.         list\_rice=list(set\_rice)

15.         #print(list\_rice) #測試是否隨機產生(是)

16.

17.         list\_meat=['炒豬肉絲','煎魚','雞肉丁','丁香魚','肉丸','扁食','豆漿(餐廳)']

18.         set\_meat=set(list\_meat)

19.         list\_meat=list(set\_meat)

20.

21.         list\_milk=['牛奶','羊奶','起司','優格','乳酪蛋糕']

22.         set\_milk=set(list\_milk)

23.         list\_milk=list(set\_milk)

24.

25.         list\_veg=['炒高麗菜','燙地瓜葉','苦瓜鹹蛋','炒A菜','炒筍子','菜頭湯(餐廳)']

26.         set\_veg=set(list\_veg)

27.         list\_veg=list(set\_veg)

28.

29.         list\_fruit=['蘋果','西瓜','奇異果','香蕉','蓮霧','水蜜桃']

30.         set\_fruit=set(list\_fruit)

31.         list\_fruit=list(set\_fruit)

32.

33.         list\_oil=['酪梨','堅果','橄欖油','玄米油']

34.         set\_oil=set(list\_oil)

35.         list\_oil=list(set\_oil)

36.

37.         tuple\_breakfast=('早餐')

38.         tuple\_lunch=('中餐')

39.         tuple\_dinner=('晚餐')

40.

41.         dict\_meals={}

42.

43.         dict\_meals[tuple\_breakfast]=(list\_rice[0],list\_meat[1],list\_milk[2],list\_veg[0],list\_fruit[1],list\_oil[2])

44.         print('早餐:'+str(dict\_meals[tuple\_breakfast]))

45.         setb={list\_rice[0],list\_meat[1],list\_milk[2],list\_veg[0],list\_fruit[1],list\_oil[2]}

46.

47.         #再次重新排序(中餐)

48.         set\_rice=set(list\_rice)

49.         list\_rice=list(set\_rice)

50.

51.         set\_meat=set(list\_meat)

52.         list\_meat=list(set\_meat)

53.

54.         set\_milk=set(list\_milk)

55.         list\_milk=list(set\_milk)

56.

57.         set\_veg=set(list\_veg)

58.         list\_veg=list(set\_veg)

59.

60.         set\_fruit=set(list\_fruit)

61.         list\_fruit=list(set\_fruit)

62.

63.         set\_oil=set(list\_oil)

64.         list\_oil=list(set\_oil)

65.

66.         dict\_meals[tuple\_lunch]=(list\_rice[1],list\_meat[2],list\_milk[0],list\_veg[1],list\_fruit[2],list\_oil[0])

67.         print('中餐:'+str(dict\_meals[tuple\_lunch]))

68.         setl={list\_rice[1],list\_meat[2],list\_milk[0],list\_veg[1],list\_fruit[2],list\_oil[0]}

69.

70.         #再次重新排序(晚餐)

71.         set\_rice=set(list\_rice)

72.         list\_rice=list(set\_rice)

73.

74.         set\_meat=set(list\_meat)

75.         list\_meat=list(set\_meat)

76.

77.         set\_milk=set(list\_milk)

78.         list\_milk=list(set\_milk)

79.

80.         set\_veg=set(list\_veg)

81.         list\_veg=list(set\_veg)

82.

83.         set\_fruit=set(list\_fruit)

84.         list\_fruit=list(set\_fruit)

85.

86.         set\_oil=set(list\_oil)

87.         list\_oil=list(set\_oil)

88.

89.         dict\_meals[tuple\_dinner]=(list\_rice[2],list\_meat[1],list\_milk[0],list\_veg[2],list\_fruit[1],list\_oil[0])

90.         setd={list\_rice[2],list\_meat[1],list\_milk[0],list\_veg[2],list\_fruit[1],list\_oil[0]}

91.

92.         print('晚餐:'+str(dict\_meals[tuple\_dinner]))

93.         print('早餐與中餐有重複嗎?',setb>=setl)

94.         print('中餐與晚餐有重複嗎?',setd>=setl)

95.         print('早餐與晚餐有重複嗎?',setb>=setd)

96.         print('------------------------------------\n')

97.     if(test==1):

98.         print('------------------------------------\n')

99.         print('(1)輸入自身身體數值，提供餐廳飲食建議')

100.         print('(2)建議食用蛋白質與熱量建議')

101.         print('(3)每日三餐健康搭配(隨機)建議')

102.         diet=int(input('您想執行上述哪一個選項(1 or 2 or 3)'))

103.         print('------------------------------------\n')

104.         time.sleep(3)

105.         os.system("cls")

106.         if(diet==1):

107.             diet\_recommand()

108.         if(diet==2):

109.             diet\_kcal\_protein()

110.         if(diet==3):

111.             print('操作此操作前，提醒您~\n熱量建議請操作2獲得每日熱量建議喔~')

112.             food()

113.         time.sleep(3)

114.         os.system("cls")

115. print('------------------------------------\n')

116.

九、訂餐系統的邏輯判斷、數據分析(收集)與運算

我們使用**Dictionary**的方法， 對menu{}->菜單與金額的統合(結合使用**list**)、order{}->針對使用者所訂的餐點(計算相對應的數量)、sum{}->計算所有金額並記錄，過程中也會有許多true、false判斷，或是輸入錯時會不斷地while迴圈要求輸入直到正確，等…除錯判斷，也有許多接著要執行的操作的if else判斷，可以讓使用者於訂餐系統中，以最完整的方式執行所有流程，最後也有一個輸出的點餐完畢圖。

1. result=0

2. calories = {'鍋貼':630, '水餃':500,'菜頭湯':122,'紅茶':111,'豆漿':188,'套餐1':741,'套餐2':611}

3. #新增套餐(list+dict應用)

4. m\_set1=[70,'鍋貼','紅茶']

5. m\_set2=[80,'水餃','紅茶']

6. menu={'鍋貼':50,'水餃':60,'菜頭湯':35,'紅茶':25,'豆漿':25,'套餐1':m\_set1,'套餐2':m\_set2}

7. order={'鍋貼':0,'水餃':0,'菜頭湯':0,'紅茶':0,'豆漿':0,'套餐1':0,'套餐2':0}

8. #使用者訂餐(while迴圈測試)

9. while(test):

10.     print('好的~以下是我們的菜單\n')

11.

12.     #dict的for 迴圈

13.     for m in menu.items():

14.         print(m)

15.     choose=input('請問您想選擇?(請輸入菜名(或套餐名))')

16.     t=True

17.     #while進行偵錯是否輸入錯誤

18.     while(not(choose =='鍋貼' or  choose =='水餃' or choose =='菜頭湯' or choose =='紅茶' or choose =='豆漿'or choose =='套餐1'or choose =='套餐2')):

19.         print('輸入錯誤請重新輸入\n') #使用者輸入錯誤例外處理

20.         choose=input('請問您想選擇?(請輸入菜名(或套餐名))\n')

21.     q=int(input('請問要幾份?'))

22.     order[choose]=int(order[choose])+q

23.     result=result+(calories[choose]\*q)

24.     print(f'提示您目前的熱量為{result}\n如果您有於飲食健康生活中測試過熱量\n可以參考是否要繼續點餐')

25.     print('請問您還要繼續點嗎?\n')

26.     keep=int(input('要(1)/不要(2)'))

27.     #if判斷點餐是否繼續

28.     if(keep==2):

29.         test=False

30. print('------------------------------------\n')

31. time.sleep(3)

32. os.system("cls")

33. #使用者訂餐結果與金額

34. print('------------------------------------\n')

35. print('以下是您的訂餐內容\n')

36. #dict的for 迴圈

37. for o in order.keys():

38.     #if判斷

39.     if(order[o]>0):

40.         print(o+":"+str(order[o])+"份\n")

41. print('\n以下是您的訂餐總金額\n')

42. total=0

43. sum={'鍋貼':0,'水餃':0,'菜頭湯':0,'紅茶':0,'豆漿':0,'套餐1':0,'套餐2':0}

44. #dict的for 迴圈

45. for o in order.keys():

46.     #if判斷

47.     if(o!='套餐1' and o!='套餐2' and int(order[o])>0):

48.         sum[o]=int(order[o])\*int(menu[o])

49.     #新增套餐(list+dict應用) & elif判斷

50.     elif((o=='套餐1' or o=='套餐2') and int(order[o])>0):

51.         sum[o]=int(order[o])\*int(menu[o][0])

52. for s in sum.values():

53.     total=int(total)+int(s)

54. print(str(total)+'元')

55. #使用者支付頁面與判斷(有錯誤例外處理)

56. pay=int(input('請付款，請輸入付款金額:\n'))

57. while(pay<total):

58.     print('付款不足!\n')

59.     pay=int(input('請付款，請重新輸入付款金額:\n'))

60. if(pay>total):

61.     print('找您'+pay-total+'元，謝謝您!\n')

62. if(pay==total):

63.     print('整付，謝謝您~\n')

64.

65. now = datetime.datetime.now()

66. print('最後確認您的用餐時間為(現在時間):'+now.strftime("%Y-%m-%d %H:%M:%S")+'後的兩小時喔~\n')

67.

68. print('------------------------------------\n')

69. time.sleep(3)

70. os.system("cls")

71.

72. after\_order()#點餐完畢圖

73. time.sleep(3)

74. os.system("cls")

十、門市管理系統的邏輯判斷、數據分析(收集)與運算

我們一開始系統會展示後台的進入圖，並使用**Dictionary**的方法， 對employee{}->紀錄員工數據(搭配**List**)，過程中也會有許多true、false判斷，也有許多接著要執行的操作的if else判斷，可以讓使用者於門式管理中，以最完整的方式選擇要執行簽到、劃位、員工管理之操作，並執行所有流程，最後如果是簽到完畢也有一個簽到畫面。。

1. back\_pic()#進入後台圖

2. time.sleep(3)

3. os.system("cls")

4.

5. #模擬服務人員(劃位分析

6. print('------------------------------------\n')

7. print('模擬服務人員\n')

8. employee1=['店長','09123456789','女','到']

9. employee2=['店員','09987654321','男','未到']

10. employee={'1':employee1,'2':employee2}

11. employee\_num=2

12. store\_test=int(input('請輸入您要執行的操作:桌位管理(1)/員工管理(2)/員工簽到(3)'))

13. print('------------------------------------\n')

14. time.sleep(3)

15. os.system("cls")

16. print('------------------------------------\n')

17. #桌位管理

18. if(store\_test==1):

19.     print('訂位人數有:'+str(people\_num)+'\n')

20.     print('如果想要都是兩人桌的話會需要\n'+str(people\_num//2)+'但可能有'+str(people\_num%2)+'人沒位子~\n\n')

21.     print('如果想要都是三人桌的話會需要\n'+str(people\_num//3)+'但可能有'+str(people\_num%3)+'人沒位子~\n\n')

22.     table={'兩人桌':2,'三人桌':3}

23.     print('您可以參考目前剩餘桌位排桌位~\n')

24.     for t in table.keys():

25.         print('目前有:'+t+'剩餘:'+str(table[t])+'\n')

26.     print('謝謝您使用本系統~\n')

27. #員工管理

28. elif(store\_test==2):

29.     print('員工資料:\n')

30.     for e in employee.keys():

31.         print(e+'號')

32.         print(employee[e])

33.     e\_test=int(input('請輸入:無操作(0)/新增員工(1)/刪除員工(2)/修改員工資料(3)'))

34.     #員工新增

35.     if(e\_test==1):

36.         name=input('姓名:')

37.         tel=input('電話')

38.         gender=input('性別(男|女)')

39.         list\_add=[name,tel,gender,'未到']

40.         employee[str(employee\_num+1)]=list\_add

41.         print('已新增')

42.         print('員工資料:\n')

43.         for e in employee.keys():

44.             print(e+'號')

45.             print(employee[e])

46.     #員工刪除

47.     elif(e\_test==2):

48.         print('請問要刪除幾號的資料?\n')

49.         e\_num=int(input())

50.         del employee[str(e\_num)]

51.         print('已刪除')

52.         print('員工資料:\n')

53.         for e in employee.keys():

54.             print(e+'號')

55.             print(employee[e])

56.     #員工修改資料

57.     elif(e\_test==3):

58.         print('請問要修改幾號的資料?\n')

59.         e\_num=int(input())

60.         name=input('姓名:')

61.         tel=input('電話')

62.         gender=input('性別(男|女)')

63.         list\_change=[name,tel,gender,'未到']

64.         employee[str(e\_num)]=list\_change

65.         print('已修改')

66.         print('員工資料:\n')

67.         for e in employee.keys():

68.             print(e+'號')

69.             print(employee[e])

70. #員工簽到

71. elif(store\_test==3):

72.     print('員工資料:\n')

73.     for e in employee.keys():

74.         print(e+'號')

75.         print(employee[e])

76.     print('請問要簽到幾號的資料?\n')

77.     e\_num=int(input())

78.     employee[str(e\_num)][3]='到'

79.     print('已簽到')

80.     ontime()#簽到圖

81. print('------------------------------------\n')

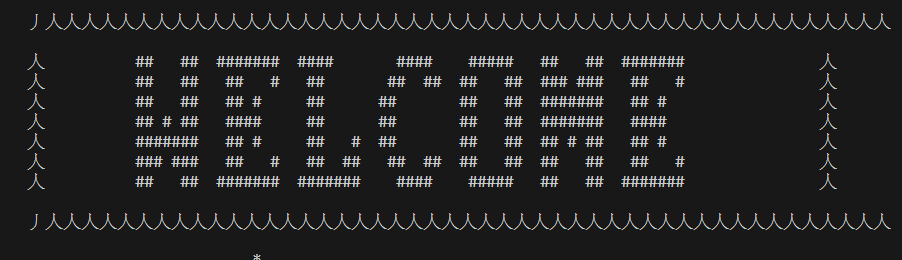
82. time.sleep(3)

83. os.system("cls")

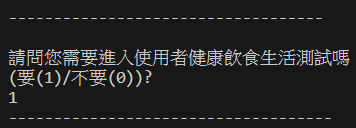
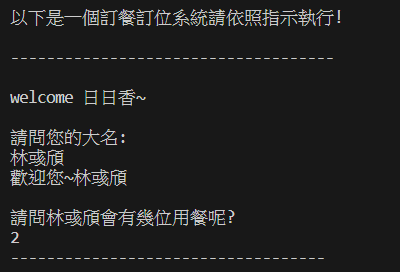
84.

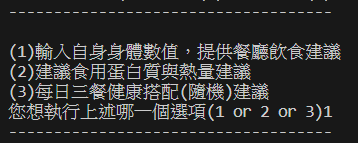
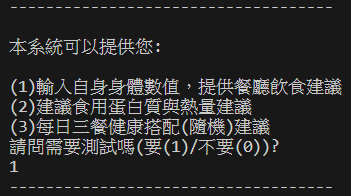
**肆、執行結果**

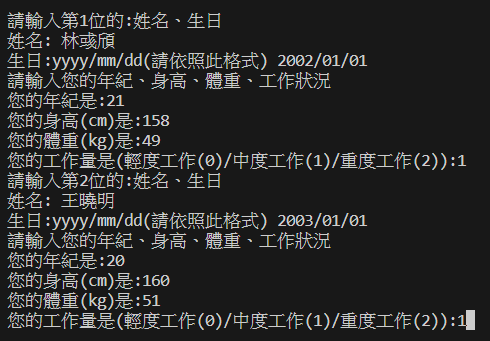
模擬一：林彧頎與王曉明使用者，想透過健康飲食生活與餐廳管理系統來計算自身的數據，並可以透過系統的運算結果知道自身的BMI與蛋白質建議量，得到關於日日香餐廳的一些餐食建議(此不僅達到關心使用者的健康飲食，更可以幫助日日香餐廳達到商業宣傳的廣告效果)。(下面圖是會含本系統的一開始流程，之後的模擬不會再次顯示)

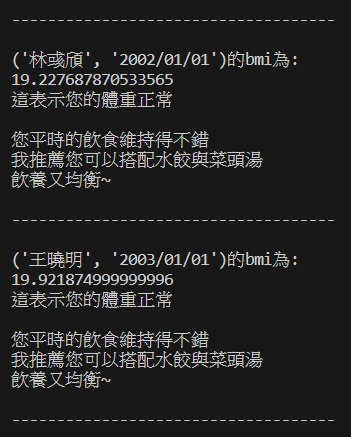




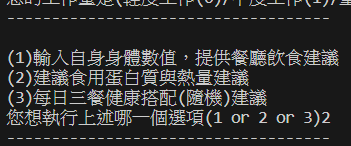


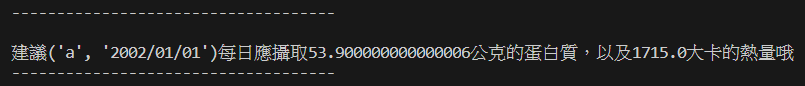


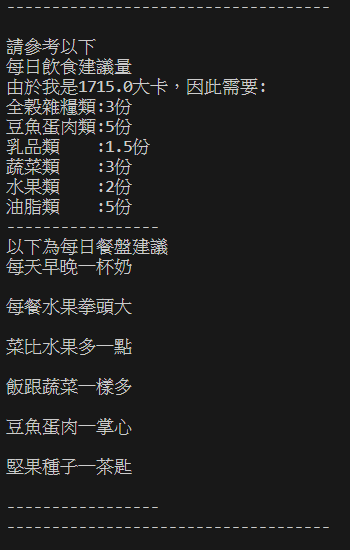




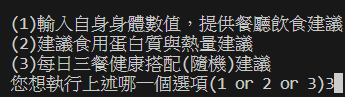
模擬二：今天來了一位十分注意個人蛋白質與bmi數據的健身者，正好健康飲食生活與餐廳管理系統有此項服務，幫助他可以針對自身數據做健身調整，而且也有對應的飲食建議十分好用，因此以下模擬此情境:

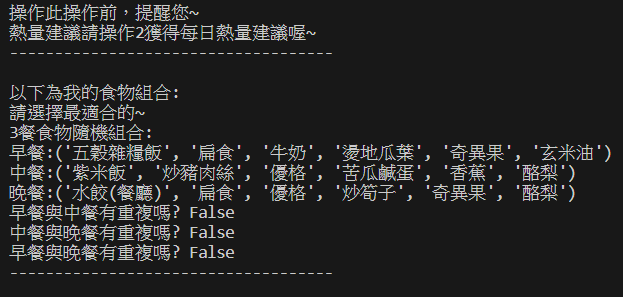




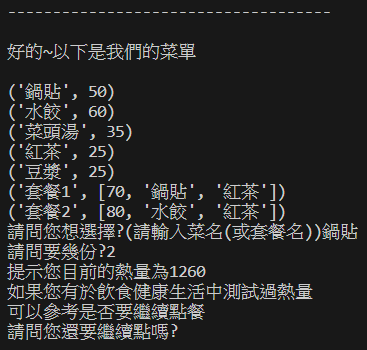


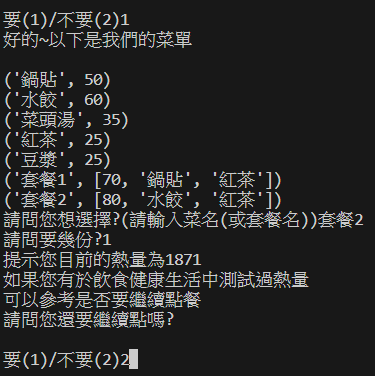
模擬三：今天有一位被三餐要怎麼吃，而且要吃得健康感到煩惱的大學生，正好健康飲食生活與餐廳管理系統有此項服務，可以幫助他，因此以下模擬此情境:

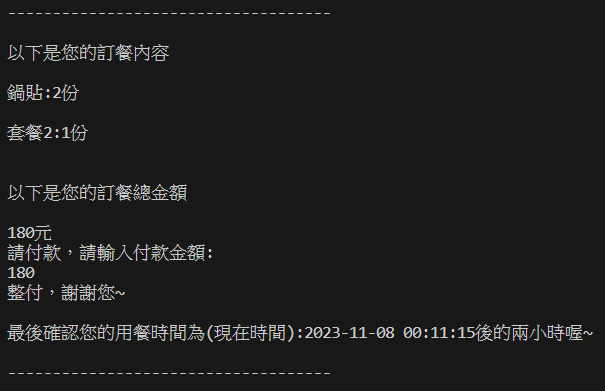


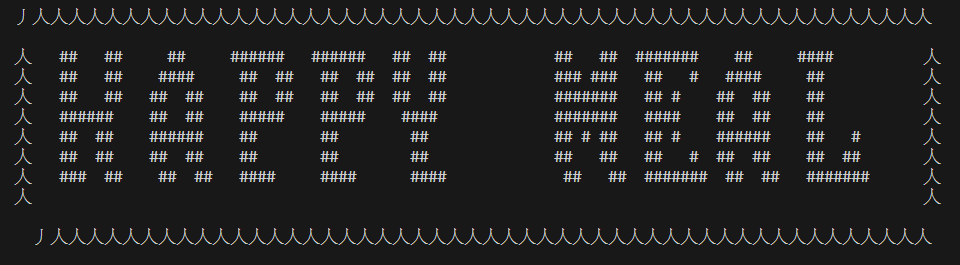


模擬四：今天有一名來到餐廳消費的使用者想透過訂餐APP訂餐，他需要於過成中決定自己想吃什麼，吃多少份，最終系統會將他所點的餐點、價格、熱量提供給他，並請他付款，結束訂餐系統。



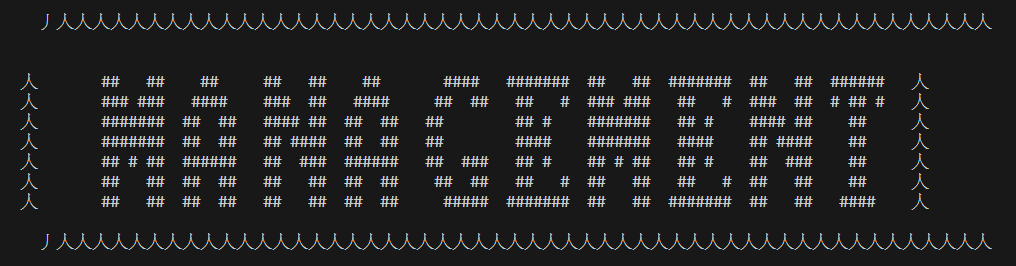




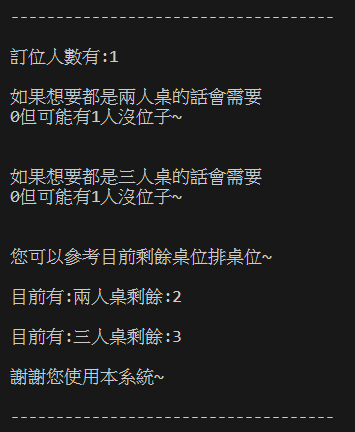




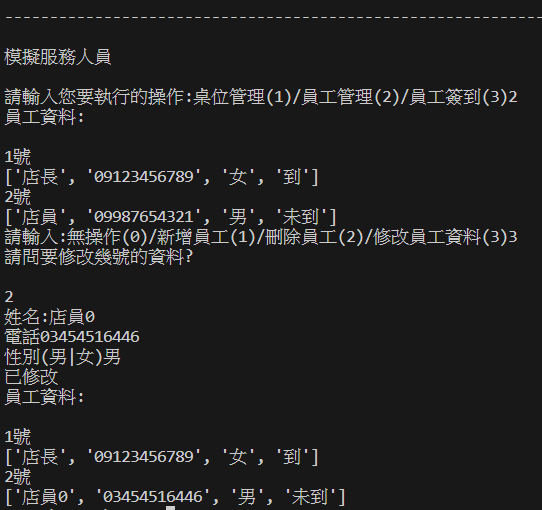
模擬五:門市管理系統中，身為門市相關人員可能需要幫忙顧客排桌位，或是需要進行個人簽到、或是管理者需要對員工之人力進行新增、修改、刪除的員工管理，因此以下將示範此系統:



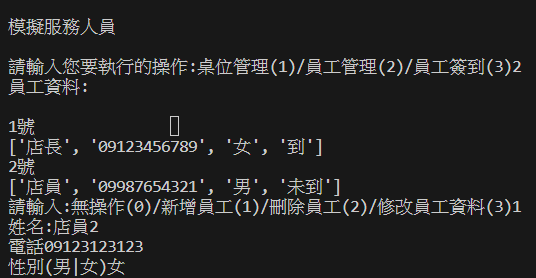




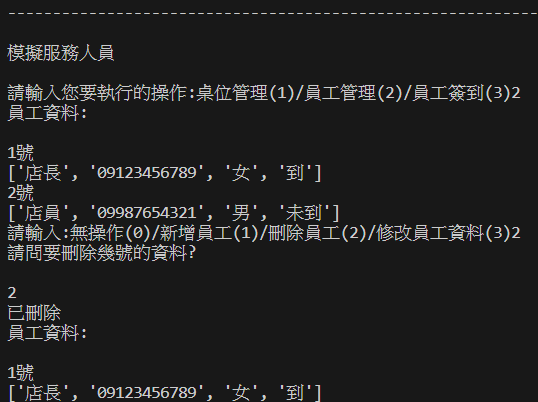
圖表 桌位管理



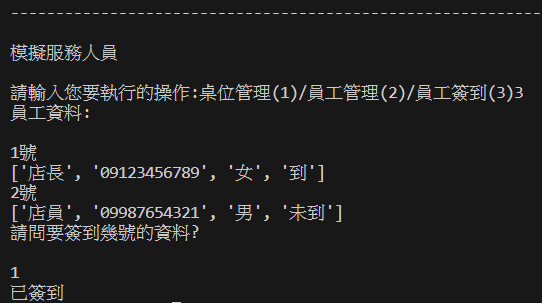
圖表 員工新增



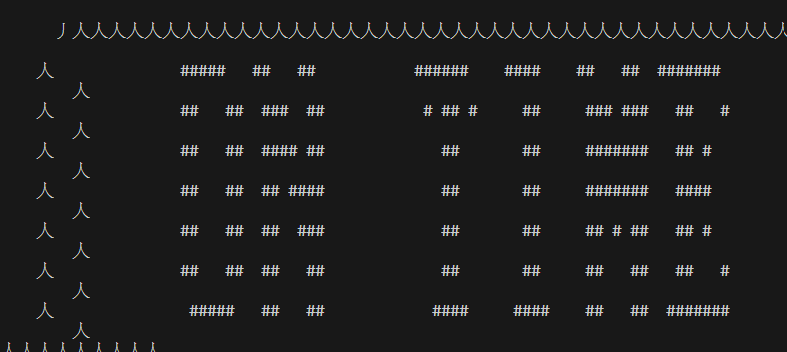
圖表 員工修改

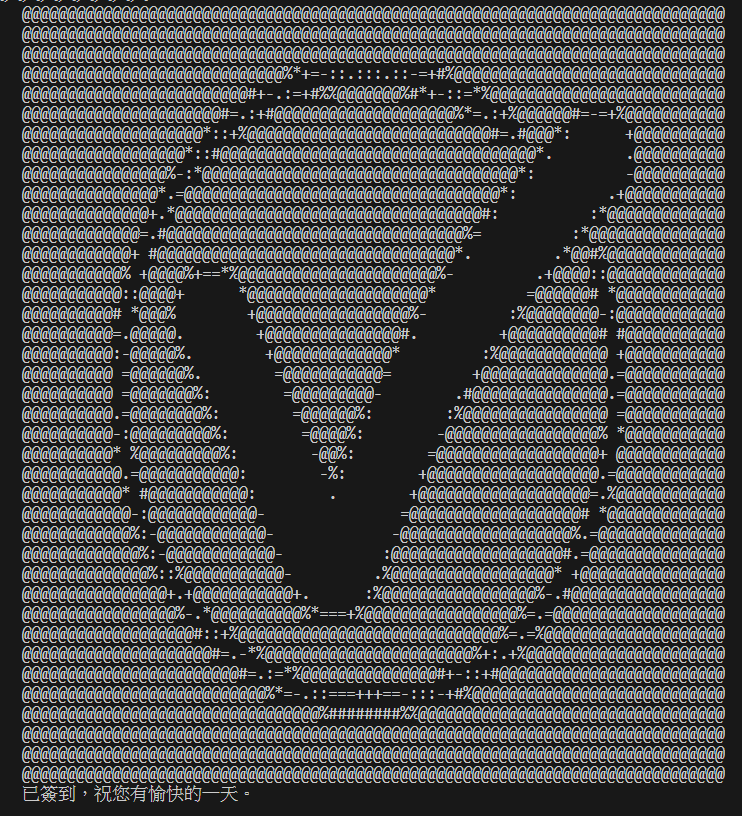


圖表 員工刪除



圖表 員工簽到





**伍、參考資料**

1. 每日餐盤-衛生福利部國民健康署

<https://www.hpa.gov.tw/Pages/Detail.aspx?nodeid=4687&pid=11680>

1. 每日飲食建議與熱量建議-衛生福利部國民健康署

<https://www.hpa.gov.tw/Pages/Detail.aspx?nodeid=544&pid=728>

1. datetime

<https://steam.oxxostudio.tw/category/python/library/datetime.html>

1. os

<https://adsl52410.pixnet.net/blog/post/188617714-python-idle%E4%B8%AD%E5%AF%A6%E7%8F%BE%E6%B8%85%E9%99%A4%E7%95%AB%E9%9D%A2%E6%8C%87%E4%BB%A4(%E9%81%A9%E7%94%A82.x%E3%80%813.x%E7%89%88%E6%9C%AC>

1. time

<https://officeguide.cc/python-time-tutorial-examples/>

1. math

<https://steam.oxxostudio.tw/category/python/library/math.html>

**陸、心得**

● 林彧頎

我在這上學期的課程中真的學習到了很多，由於我個人是資工系的，但自高中接觸arduino到大三上，我竟然都沒有學習從高中就耳熟能詳的python，直到在暑假期間，我開始著手畢業專題、競賽、論文，我發現無論是甚麼，都需要進行資料分析(語音、文本等等…)，而其中一個非常好用的工具就是Python，而我在做相關研究的過程中，常常會因為不太清楚Python的使用而十分痛苦，因此使我很想好好學習Python這項好用的程式語言，而基礎程式設計課程正好滿足我的需求，而在課程中我不僅僅學到我平常不會使用到的資料處理的方法，像是我平常並不會特別地去了解字典的內部可以運用到list、tuple等等...，而在本次的專題中，我從企畫構想、系統架構、問題、以及我主要負責的程式碼，我都有參與到，並且在報告中，關於程式相關的說明主要都是我撰寫的，還有架構的部分，透過本次的期中報告，我學會如何使用到商業結合生活的方法，並也學習到更多的python技巧，我也很感謝我的組員給我很多的發揮空間，並也如期繳交各自負責的部分，真的非常感謝組員與老師提供的學習機會，讓我學會與不同系所的同學合作，並可以從中學習到python這項不可或缺的技能，使程式不僅僅是coding，更是融入議題的實用工具。

● 彭一珊

我這次負責的是使用者介面的美化跟議題發想，本來以為美化介面應該是很簡單的事情啊，想辦法把讓圖片透過程式碼展現就好了，結果理想不如預期，碰到了超多困難，也換了三種以上的方式，才呈現出現在的結果，過程當中很感謝我們的小組長，一直陪我修修改改，還幫我程式碼除錯，想辦法讓它可以跑出結果。

● 黃榛柔

因為我剛學習到程式語言，對新接觸的領域有時會不知從何下手，我發現議題的發想其實是最難的，資訊量一大整個程式的架構也變得很重要。這次的專題我剛好遇到很厲害的組員，融合了我們的所有想法，先建構了一整個點餐系統，我們再陸續把一些功能加上去，也有給我一些寫程式的建議，我完成的很有成就感。這樣的議題對我來說很新鮮，完全是我平常不會碰到的，我也透過這次專題更認識程式語言的用途，非常有收穫。

● 郭珈妤

我是這學期才剛開始學習程式,同組的成員都很厲害,除了讓我覺得自己還有很多要學習的以外,也驚訝於程式的神奇,居然透過手指在鍵盤上敲打就可以創造出一個對自己或人們很有幫助的系統!令人很有成就感的是,我從開學時零基礎到能夠利用這門課所學完成這次的期中專題，我在這堂課的收穫很多!

自從開始學習程式,我越來越覺得在未來，科技勢必會蓬勃發展，每一個人都需要倚靠科技，不如就先學習起來，也許未來有一天會對於我的人生有不一樣的轉捩點!

