Aim

To take orders exported from website, and choose menu items to minimise required cooking, whilst ensuring all orders are fulfilled.

Input

Data is exported in a csv format from website (see Fig 1), the csv file is named in a date-time format upon export (Fig 1 was FuelzOrders-2019-11-22.csv for example). In particular we are paying attention to columns G, H, and I.

	A	B	C	D	E	F	G	H	I	J
1	Order	Order Status	First Name	Last Name	Phone	Address	Item Name	Product Variation	Quantity	Order Total Amount
2	7495	Processing	Dan				Subscription Plan	Meals per day (Monday to Friday): 1 | Collection Method: Collection Monday 10am (At UQ St. Lucia)	1	32.5
3	7485	Processing	Scott				Fusion Meal Plan	Meals Per Day: 2 | Collection Method: Sunday Home Delivery	1	100
4	7485	Processing	Scott				Tarragon Chicken		1	100
5	7485	Processing	Scott				Bacon Carbonara		2	100
6	7485	Processing	Scott				Macaroni Cheese		1	100
7	7483	Processing	Dale				Fusion Meal Plan	Meals Per Day: 1 | Collection Method: Sunday Home Delivery	1	34.5
8	7481	Processing	Tao				Fusion Meal Plan	Meals Per Day: 2 | Collection Method: Sunday Home Delivery	1	70
9	7479	Processing	Dalton				Subscription Plan	Meals per day (Monday to Friday): 1 | Collection Method: Collection Monday 10am (At UQ St. Lucia)	1	32.5

Figure 1- Exported csv

Desired Output

Ideally, three documents/excel spreadsheets will be generated; one with the meals we will be making that week (see Fig 2), and another with our required ingredients (Fig 3), finally the actual order for each customer for when we pack their order. I have already written the program to handle the ingredients, it just needs to know what portions are needed.

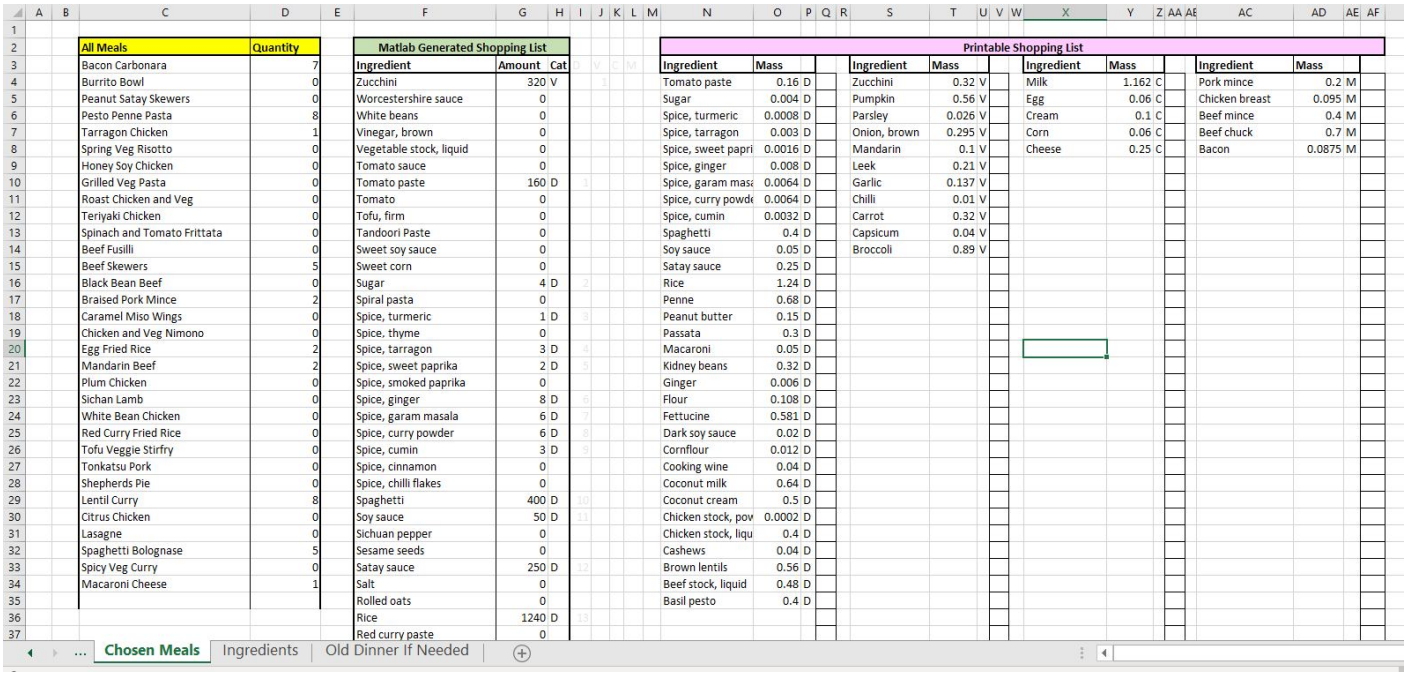
Meals from 24 June

F/T	Classic	Quantity	F/T	Asian Style	Quantity
F	Citrus	6	F	Tonkatsu	6
F	Shepherds Pie	6	F	Red Curry fried rice	4
F	Grilled veg	8	F	Mandarin beef	5
F	Macaroni Cheese	6	F	Plum chicken	4
F	Rissotto	8	F	Beef satay skewers	4
T	Tarragon chicken	3	T	Whitebean chicken	2
T	Carbonara	2	T	Tofu veggie stir fry	4
T	Pesto penne	5	T	Miso Wings	2
T	Fusilli	2	T	Blackbean beef	2
T	Burrito Bowl	3	T	Sichuan lamb	6

Custom:

Egg Fried Rice	2
Spicy Veg Curry	1
Chicken satay skewers	1
Honey soy chicken	2
Spag bol	1
Coconut lentil curry	1

Fig 2 – Desired format of meals we are cooking each week. F/T represents if the meal goes in the 5/10 meal per week plans. Custom is used to clean up any remaining custom ordered meals that don't overlap what we were cooking for the surprise plans.



All Meals	Quantity
Bacon Carbonara	7
Burrito Bowl	0
Peanut Satay Skewers	0
Pesto Penne Pasta	8
Tarragon Chicken	1
Spring Veg Risotto	0
Honey Soy Chicken	0
Grilled Veg Pasta	0
Roast Chicken and Veg	0
Teriyaki Chicken	0
Spinach and Tomato Frittata	0
Beef Fusilli	0
Beef Skewers	5
Black Bean Beef	0
Braised Pork Mince	2
Caramel Miso Wings	0
Chicken and Veg Nimono	0
Egg Fried Rice	2
Mandarin Beef	2
Plum Chicken	0
Sichan Lamb	0
White Bean Chicken	0
Red Curry Fried Rice	0
Tofu Veggie Stirfry	0
Tonkatsu Pork	0
Shepherds Pie	0
Lentil Curry	8
Citrus Chicken	0
Lasagne	0
Spaghetti Bolognase	5
Spicy Veg Curry	0
Macaroni Cheese	1

Matlab Generated Shopping List		
Ingredient	Amount	Cat
Zucchini	320	V
Worcestershire sauce	0	
White beans	0	
Vinegar, brown	0	
Vegetable stock, liquid	0	
Tomato sauce	0	
Tomato paste	160	D
Tomato	0	
Tofu, firm	0	
Tandoori Paste	0	
Sweet soy sauce	0	
Sweet corn	0	
Sugar	4	D
Spiral pasta	0	
Spice, turmeric	1	D
Spice, thyme	0	
Spice, tarragon	3	D
Spice, sweet paprika	2	D
Spice, smoked paprika	0	
Spice, ginger	8	D
Spice, garam masala	6	D
Spice, curry powder	6	D
Spice, cumin	3	D
Spice, cinnamon	0	
Spice, chilli flakes	0	
Spaghetti	400	D
Soy sauce	50	D
Sichuan pepper	0	
Sesame seeds	0	
Satay sauce	250	D
Salt	0	
Rolled oats	0	
Rice	1240	D
Red curry paste	0	

Printable Shopping List

Ingredient	Mass		Ingredient	Mass		Ingredient	Mass		Ingredient	Mass	
Tomato paste	0.16	D	Zucchini	0.32	V	Milk	1.162	C	Pork mince	0.2	M
Sugar	0.004	D	Pumpkin	0.56	V	Egg	0.06	C	Chicken breast	0.095	M
Spice, turmeric	0.0008	D	Parsley	0.026	V	Cream	0.1	C	Beef mince	0.4	M
Spice, tarragon	0.003	D	Onion, brown	0.295	V	Corn	0.06	C	Beef chuck	0.7	M
Spice, sweet paprika	0.0016	D	Mandarin	0.1	V	Cheese	0.25	C	Bacon	0.0875	M
Spice, ginger	0.008	D	Leek	0.21	V						
Spice, garam masala	0.0064	D	Garlic	0.137	V						
Spice, curry powder	0.0064	D	Chilli	0.01	V						
Spice, cumin	0.0032	D	Carrot	0.32	V						
Spaghetti	0.4	D	Capsicum	0.04	V						
Soy sauce	0.05	D	Broccoli	0.89	V						
Satay sauce	0.25	D									
Rice	1.24	D									
Penne	0.68	D									
Peanut butter	0.15	D									
Passata	0.3	D									
Macaroni	0.05	D									
Kidney beans	0.32	D									
Ginger	0.006	D									
Flour	0.108	D									
Fettucine	0.581	D									
Dark soy sauce	0.02	D									
Cornflour	0.012	D									
Cooking wine	0.04	D									
Coconut milk	0.64	D									
Coconut cream	0.5	D									
Chicken stock, pow	0.0002	D									
Chicken stock, liqu	0.4	D									
Cashews	0.04	D									
Brown lentils	0.56	D									
Beef stock, liquid	0.48	D									
Basil pesto	0.4	D									

Chosen Meals | Ingredients | Old Dinner If Needed

Fig 3 – Matlab shopping list generator takes meals in columns C/D and generates required ingredients, which are sorted and output into N-AE. I then print this list and use it to buy ingredients, but you don't need to touch this, we just need the meal quantities in column D.

Current Program

To start off with the program identifies the csv file relevant to the orders of that week, loading it into data-frame df (Fig 4).

```
1
2 import pandas as pd
3 from datetime import date
4 import numpy as np
5 import openpyxl as oxl
6
7 # ----------------------------------------------------------------------
8 # Creating data frame from csv of todays date
9
10 today=date.today()
11 OrderDate=today.strftime("%Y-%m-%d")
12 CSVOrderFile="FuelzOrders-2019-11-22.csv "
13 df=pd.read_csv(CSVOrderFile)
14 weeknum=date.isocalendar(today)[1]
15
16 # ----------------------------------------------------------------------
```

Index	er Nun	Order Status	Name (Bill	ame (	Phone	I&2 (S	Item Name	Product Variation	Quantity	Order Total Amount	Plan?	MPD
0	7495	Processing	Dan	[redacted]			Subscription Plan	Meals per day (Monday to Friday): 1 | Collection Method: Collection Monday 10am (At UQ St. Lucia)	1	32.5	1	1
1	7485	Processing	Scott				Fusion Meal Plan	Meals Per Day: 2 | Collection Method: Sunday Home Delivery (For St. Lucia, Toowong, Indooroopilly, So…	1	100	1	2
2	7485	Processing	Scott				Tarragon Chicken	nan	1	100	0	0
3	7485	Processing	Scott				Bacon Carbonara	nan	2	100	0	0
4	7485	Processing	Scott				Macaroni Cheese	nan	1	100	0	0
5	7483	Processing	Dale				Fusion Meal Plan	Meals Per Day: 1 | Collection Method: Sunday Home Delivery (For St. Lucia, Toowong, Indooroopilly, So…	1	34.5	1	1
6	7481	Processing	Tao				Fusion Meal Plan	Meals Per Day: 2 | Collection Method: Sunday Home Delivery (For St. Lucia, Toowong, Indooroopilly, So…	1	70	1	2
7	7479	Processing	Dalton				Subscription Plan	Meals per day (Monday to Friday): 1 | Collection Method: Collection Monday 10am (At UQ St. Lucia)	1	32.5	1	1

Fig 4 – Loading current csv, and df

Each plan order (that classic (subscription), fusion, or Asian) contains the word 'plan' in the 'Item Name' column which is searched for. Then in the product variation column for a row containing a plan, the first ": " string contains the number of meals per day and is extracted (Fig 5). Note, 1 meal/day = 5 meals/week, and 2 = 10 meals/week. The data is saved in the MPD column in df.

```
16 # ------------------------------------------------------------------
17 # Searching orders for meal plans
18 sub ='Plan'
19 df['Plan?']=df["Item Name"].str.find(sub)
20 c=0
21 for entry in df['Plan?']:
22     if entry <=0:
23         df.at[c,'Plan?']=0
24     else:
25         df.at[c,'Plan?']=1
26     c=c+1
27 
28 # Text searching for meals per day from plans (MPD)
29 Meals_per_day = df["Product Variation"].str.extract(pat = '(: .)')
30 for i in range(len(Meals_per_day)):
31     if str(df.at[i,'Product Variation'])[0]!='M':
32         Meals_per_day.iat[i,0]=int(0)
33     else:
34         Meals_per_day.iat[i,0]=int(str(Meals_per_day.iat[i,0])[2])
35 df['MPD']=Meals_per_day
36 
37 del Meals_per_day
38 
39 # ------------------------------------------------------------------
```

Fig 5 – Extracting meals per day for plans

Using MPD, the number of orders for classic, and Asian style dishes from plans are combined in POrds (Plan Orders). This is done by text searching for each plan in the 'Item Name' column, then adding 1 to either the first 5 or 10 values in POrds depending on whether the order is for 5 or 10 meals/week and which menu it is relating to (Fig 6). Fusion plans use dishes from both classic and Asian menus, so receiving 5 fusion meals/week means you will alternate in receiving 2:3 classic:Asian, and 3:2 classic:Asian. I do this by toggling between even and odd week numbers (line 61, Fig 7). For 10 fusion meals per week, 5 classic and 5 Asian meals are ordered.

Index	Classic	Asian
0	5	3
1	5	3
2	4	3
3	4	2
4	4	2
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

Fig 6 – Pords, showing that there are no current 10 meal/week plans, as shown in Fig 4. There are 2 x 5 meals/week classic plans + 2 x 10 meals/week fusion so 4 portion are required for the first 5 classic dishes we make, then there is 1 x 5 meals/week fusion order, meaning the first 2 classic dishes now have 5 required servings. The process is identical for the Asian column, however with the first three dishes are increased by the 5 m/w fusion plan.

```
# ------------------------------------------------------------------
# Creating classic/asian orders from MPD
POrds=pd.DataFrame(np.array([[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0],[0,0]]),columns=['Classic','Asian'])

for i in range(len(df['Order Number'])):
    if df.at[i,'Item Name']=='Subscription Plan' and df.at[i,'MPD']==1:
        for s in range(0,5):
            POrds.at[s,'Classic']=POrds.at[s,'Classic']+1

    if df.at[i,'Item Name']=='Subscription Plan' and df.at[i,'MPD']==2:
        for s in range(0,10):
            POrds.at[s,'Classic']=POrds.at[s,'Classic']+1

    if df.at[i,'Item Name']=='Asian Plan' and df.at[i,'MPD']==1:
        for s in range(0,5):
            POrds.at[s,'Asian']=POrds.at[s,'Asian']+1

    if df.at[i,'Item Name']=='Asian Plan' and df.at[i,'MPD']==2:
        for s in range(0,10):
            POrds.at[s,'Asian']=POrds.at[s,'Asian']+1

    if df.at[i,'Item Name']=='Fusion Meal Plan' and df.at[i,'MPD']==1:
        if (weeknum % 2) == 0:
            for s in range(0,2):
                POrds.at[s,'Asian']=POrds.at[s,'Asian']+1
            for s in range(0,3):
                POrds.at[s,'Classic']=POrds.at[s,'Classic']+1
        else:
            for s in range(0,3):
                POrds.at[s,'Asian']=POrds.at[s,'Asian']+1
            for s in range(0,2):
                POrds.at[s,'Classic']=POrds.at[s,'Classic']+1

    if df.at[i,'Item Name']=='Fusion Meal Plan' and df.at[i,'MPD']==2:
        for s in range(0,5):
            POrds.at[s,'Asian']=POrds.at[s,'Asian']+1
            POrds.at[s,'Classic']=POrds.at[s,'Classic']+1


# ------------------------------------------------------------------
```

Fig 7 – Code creating POrds

The next section is kind of garbage, but loosely speaking it looks at the last two weeks of dishes cooked, this is for our optimisation to ensure that the meals our surprise plan customers are receiving are not too repetitive. The dishes chosen for each week should be saved into the Meal Orders-Meals-Recipes excel spreadsheet, in the Order History tab (Fig 7).

```
# ------------------------------------------------
# Loading previous 2 weeks orders
sheet=OMR['Order History']

# 1 week back
if weeknum<46:
    step1=(6+weeknum)*11+2
else:
    step1=(weeknum-46)*11+2

Classics1=[0,0,0,0,0,0,0,0,0,0]
for i in range(10):
    Classics1[i]=sheet['D'+str(i+step1)].value

# 2 weeks back
if weeknum<46:
    step2=(5+weeknum)*11+2
else:
    step2=(weeknum-47)*11+2


Classics2=[0,0,0,0,0,0,0,0,0,0]
for i in range(10):
    Classics2[i]=sheet['D'+str(i+step2)].value
```

	A	B	C	D	E	F	G	H	I	J	K
1	yyww	Week Start		Classics	Num		Asian-Style	Num		Misc Custom	Num
2	1945			Baked Drumsticks with Potat	9		Sticky Asian Meatballs	6			
3				Veggie Lasagne	10		White Bean Chicken	5			
4				Roast Chicken and Veg	9		Braised Pork Mince	2			
5				Beef Burrito Bowl	6		Egg Fried Rice	2			
6				Macaroni Cheese	7		Tonkatsu Pork	3			
7				Bacon Carbonara	3		Nil	0			
8				Tarragon Chicken	2		Nil	0			
9				Pesto Penne Pasta	1		Nil	0			
10				Spicy Vegetable Curry	2		Nil	0			
11				Shepherd's Pie	2		Nil	0			
12											
13					5			3			
14					5			3			
15					4			3			
16					4			2			
17					4			2			
18					0			0			
19					0			0			
20					0			0			
21					0			0			
22					0			0			
23											

Fig 7 – Order History loading and spreadsheet

Optimising Dish Selection

In order to choose the dishes we make each week I was intending of having python optimise a score, where selecting certain dishes have consequences on the score. The number of portions we make of each dish is irrelevant, its just the number of different dishes we have to prepare, and the amount of overlap between weeks, that needs to be balanced. So:

Constraints:

- All custom ordered dishes must be prepared
- If there is a surprise plan ordered for that menu, then the appropriate number of different dishes must be made from that menu (e.g. if there is a classic order for 5 dishes, then 5 different classic dishes must be made at a minimum)

Scoring (values are set to change, just representing the proportional impact):

- +1 per new dish that is made
- +5 if a dish that was in the surprise plans last week is again in the surprise plans this week
- +2 (as above) but for two weeks previous (from Fig 7 stuff)

So the program will aim to minimise the number of separate dishes we have to prepare by creating overlap between the custom orders and the plans, however will need to balance any week-to-week overlaps. Again, once this list is known, the MPD data in addition with the custom orders can be added together to produce a list like Fig 2. And that data can be run through my ingredient list generator. The plan dishes can then be saved to the Order History spreadsheet for use next week and so on.