

Wine Ordering with Python 2.0

Daniel Sharp

May 2020

1 Introduction

I am often frustrated by the lack of efficiency in wine ordering. Most businesses have no system at all, or use a combination of Excel and email. I have previously spent hours copying and pasting orders into emails. This method is time consuming and error prone.

After reading 'Automate the Boring Stuff with Python'[1] and doing some elementary programming I decided to attempt a Python solution. My first attempt was to be honest, pretty horrible. It worked, but was not in any way "Zen" and was mostly a Stack Overflow copy and paste affair.

After a few more months of practise I've produced a second iteration. Whilst not perfect, it is significantly more functional, concise, and has a much higher level of abstraction.

A description of the development process and implementation follows. Please get in touch at daniel.sharp@student.unsw.edu.au if you have any feedback.

2 Ordering Process and Potential for Automation.

The wine ordering process can be broken down into smaller steps.

1. Compile the orders.
2. Entering orders into a spreadsheet.
3. Copy and paste orders into email and send.

The third step is the time consuming process. This step also doesn't require any specialised knowledge or intuition. It's just a repetitive process and thus ripe for automation. Although it is possible to automate the order compiling process through analysis of past sales etc, this gets very complicated, very quickly. The greatest cost:benefit ratio comes from automating the emailing process.

3 The Interface

ORDERS				
Company	Product	Units	LUC	
Jefferys Wine Imports	Home-schooled "Organic" Prosecco	35	14.51	
Brian Cox	Eulers Falanghina		19.80	
0511 477 77	Hubble Bubble Morgon	35	22.84	
brian@jeffreyswine.com	Titan Pinot Grigio		26.50	
	New Glenn "Titan" Vino Bianco	12	24.83	
CC EMAIL				
BCC EMAIL				
PERSONAL EMAIL MESSAGE				
Dear Brian, Thank you for the tasting last week. We will pour the Home-schooled prosecco by the glass. Please				
INVOICE NOTES				
Please email invoice to accounts@danstearooms.com				
Company	Product	Units	LUC	
Monkey Fun Wines	Marmas Vino Bianco	12	23.75	
Robin Ince	Dione White		33.30	
0504 962 03	Hyperion Catarratto		31.68	
brian@jeffreyswine.com	Rhea Rosso del Veronese		20.90	
	Enocladus Langhe Nebbiolo	24	29.00	
CC EMAIL				
BCC EMAIL				
PERSONAL EMAIL MESSAGE				
Hi Robin, Great! catching up on Wednesday. May we please order as per the attached PO. Cheers, Dan				
INVOICE NOTES				
Please email invoice to accounts@danstearooms.com				
Company	Product	Units	LUC	
Cage Vinos	Deep Mind Bianco	24	12.00	
Hannah Fry	AlphaStar '06' Rosato		15.42	
0500 185 81	Orinus Catarratto	6	45.00	
hannah@cagevinos.com.au	Hestia Rosso	6	12.00	
CC EMAIL				
BCC EMAIL				

Figure 1: Wine ordering spreadsheet example.

To keep the development time short I decided to use Excel as the interface. This meant I didn't have to spend time learning to build a GUI. Excel is also pretty universal, almost everyone has at least an elementary knowledge of spreadsheets. Python also interacts very nicely with Excel thanks to the `openpyxl` library.

While the spreadsheet can take any form, each 'order block' needs to have exactly the same layout to be read correctly. The information in each order block (typically arranged by supplier) must be spaced at multiples of the same integer. In the example below, each block is separated by 20 cells. The order block length is set by changing the value of the `INCREMENT` constant. Arranged like this new blocks can be added indefinitely (at least until the end of the spreadsheet).

In each block there is the company name, sales representative and email address. I've also included a cell for CC and BCC emails. The personal message field is for the main body of the email. There is also a cell for notes such as delivery instructions.

4 How It Works

The program is broken down into several modules. Each module uses one or two functions to complete a specific step in the algorithm. The complete code for each module is included in the appendix.

The main program is `wine_ordering2.0.py`. When run, it opens the order spreadsheet and selects the active sheet. A for loop iterates over each order block, and a nested for loop iterates over each cell in each block. It calls the `is_empty` function to check if any cell in the order quantity column contains data. If the cell is not empty the product, quantity and price cells are added to their respective lists.

When the end of the order block is reached the three lists are compiled into a dictionary called `order`.

The `generate_table` module converts the dictionary into a pandas dataframe. The tax and total columns are created in pandas. I have included a total with and without GST, but any tax or price calculations can be done such as adding WET tax.

The dataframe is plotted as a table and saved as `order.png`. Whilst this is probably not the most elegant or efficient way to handle data it was the easiest way to insert order tables of different sizes into a pdf file. In the future I would like to write the order dictionary directly into the pdf.

The module `generate_pdf` creates the purchase order pdf using a package called report labs. Depending on how proficient you are at document design, this can be as detailed as you like. The company details from the order block are written into the pdf and the order table is inserted.

The unique PO reference is created by the `generate_po_ref` function. It is the first three letters of the supplier name, and the last five characters of the current date and time in hexadecimal. It could be absolutely anything you like simply by changing the function.

Purchase Order

57-97 Ploughshare St
Trinity 2024
05 8060 9999
danstearooms.com

Brian Cox
Infinite Wine Imports
0511 477 77
brian@infiniteimports.com

INF0E745
18/05/2020

Product	Units	U/C	Total ex GST	Total inc GST
Home-schooled "Organic" Prosecco	36	14.51	522.36	574.6
Hubble Bubble Morgon	36	22.84	822.24	904.46
New Glenn 'Titan' Vino Bianco	12	24.83	297.96	327.76
Total			1642.56	1806.82

Notes:

Please include PO reference on invoice.
Please email invoice to accounts@danstearooms.com

DELIVERY INSTRUCTIONS:

Deliver between 10am & 5pm Monday - Friday
Use loading dock on Lens St

For any issues please contact:
dan@danstearooms.com

Figure 2: Purchase order pdf created from first order block in figure 1.

This module also saves a copy of the pdf in a folder named after the company to create an automatic reference record of all orders sent.

After the pdf is created, the `send_message` function emails the order as an attachment to the email address specified in the spreadsheet. The personal message field is the main body of the email. I have not included `send_message` code for security reasons. For most email providers this function can be implemented very easily.

Lastly the date, PO reference, and total cost of the order are saved into the order spreadsheet. This is very useful as it allows the person receiving the order to easily check if there are any pricing discrepancies.

After all order blocks in the active sheet have been iterated over the workbook is saved.

ORDERS			
Company	Product	Units	LUC
Infinite Wine Imports	Home-schooled "Organic" Prosecco	36	14.51
Brian Cox	Eulers Falanghina		19.80
0511 477 77	Hubble Bubble Morgan	36	22.84
brian@infiniteimports.com	Titan Pinot Grigio		26.50
SENT: 17/05/2020	New Glenn "Titan" Vino Bianco	12	24.83
PO REF: INFABBB			
TOTAL: \$1806.82			
CC EMAIL			
BCC EMAIL			
dian@danstearooms.com			
PERSONAL EMAIL MESSAGE			
Dear Brian, Thank you for the tasting last week. We will pour the Home-schooled prosecco by the glass. Please			
INVOICE NOTES			
Please email invoice to accounts@danstearooms.com			

Figure 3: Order spreadsheet from figure 1 after order has been sent.

5 Conclusion

I have been using this program for the past few months (well at least until this whole COVID thing started). It has reduced the ordering time required per week from 90 to 20 minutes. However, the largest benefit has been in stock control. Because of the unique PO reference number each order can be pinned to an invoice payment and reconciliation is much simpler. Cost control is also much easier because it is immediately obvious if pricing is incorrect.

The program is also very easy to adapt to different email systems and companies. For example, it is easy to implement a module that emails all purchase orders for the month to the accounting team, or rejects orders if they are over a certain budget cap.

Thank you for reading if you got this far. If you would like to utilize this code it is available on my GitHub.

References

- [1] Al Sweigart. *Automate the Boring Stuff with Python*. San Francisco, CA: Penguin Random House, 2019.

6 Appendix

```
1 """
2 Created on Mon May 11 17:21:19 2020
3
4 Purpose: To compile, generate purchase order and email orders from
5         an Excel spreadsheet.
6
7 Author: Daniel Sharp
8 """
9 import os
10 import openpyxl
11 import generate_po_ref as pr
12 import generate_date as gd
13 import generate_table as gt
14 import generate_pdf as gp
15 import is_empty as ie
16 import send_message as sm
17
18 INCREMENT = 20 # Length of order blocks. Change to worksheet
19               # specifications.
20
21 print('Hello!')
22
23 wb = openpyxl.load_workbook('Order Sheet/order_sheet_2020.xlsx') #
24     Open order workbook.
25 sheet = wb.active # Select active sheet
26 row_count = sheet.max_row # Get length of sheet
27
28 start = 3 # Start value of first order block.
29 end = 21 # End value of first order block.
30
31 for order_blocks in range(0,row_count): # Select worksheet rows to
32     iterate over.
33
34     order = {} # Create empty order dictionary.
35     units_list = [] # Create empty number of units list.
36     product_list = [] # Create empty product list
37     luc_list = [] # Create empty price list.
38
39     for cell_value in range(start, end): # Iterate over cells in
40         order block.
41             date = gd.generate_date() # Call date function.
42             po_ref = pr.generate_po_ref() # Call generate po refernce
43             function.
44
45             company = sheet.cell(start,1).value # Get company name from
46             spreadsheet
```

```

41     if ie.is_empty(company) == True: # If no company in cell,
break out of loop.
42         break
43
44         full_name = sheet.cell(start + 1, 1).value # Get email
recipient.
45         mobile = sheet.cell(start + 2, 1).value # Get mobile number
.
46         email = sheet.cell(start + 3, 1).value # Get company email.
47         cc_email = sheet.cell(start + 8, 1).value # Get cc email.
48         bcc_email = sheet.cell(start + 10, 1).value # Get bcc email
.
49         filename = company[0:3].upper() + po_ref # Create unique PO
reference.
50         notes = sheet.cell(start + 16, 1).value # Get invoice notes
.
51         body = sheet.cell(start + 12, 1).value # Get email body.
52         suffix = '.pdf'
53         po = os.path.join('Sent Orders',company, filename + suffix)
# Defines PO attachment name.
54
55         quantity = sheet.cell(cell_value,3).value # Get quantity
ordered.
56
57         if ie.is_empty(quantity) == False: # Create order if a
number of units is specified.
58
59             product = sheet.cell(cell_value,2).value # Create list
of products ordered.
60             product_list.append(product)
61
62             units = sheet.cell(cell_value,3).value # Create list of
units ordered.
63             units_list.append(units)
64
65             luc = sheet.cell(cell_value,4).value # Create list of
prices.
66             luc_list.append(luc)
67
68             order['Units'] = units_list # Create order dictionary of
lists.
69             order['Product'] = product_list
70             order['LUC'] = luc_list
71
72         if ie.is_empty(units_list) == False: # If units list is not
empty create order pdf and email.
73             gt.generate_table(order) # Create order table and PO pdf.
74             gp.generate_pdf(company, filename, full_name, mobile, email
, date, notes)
75
76             print(f'Sending order to {company}...') # Call email
function.
77             sm.send_message(email,order, company, body,po,cc_email,
bcc_email)
78             sheet.cell(start + 4, 1).value = 'SENT: ' + date # Save
date, PO ref and cost to spreadsheet.
79             sheet.cell(start + 5, 1).value = 'PO REF: ' + filename

```

```

80     sheet.cell(start + 6, 1).value = 'TOTAL: $' + gt.
      generate_table(order)
81
82     start += INCREMENT # Increment start and end values to next
      block.
83     end += INCREMENT
84
85 wb.save('Order Sheet/order_sheet_2020.xlsx') # Save workbook
86 print('Ordering completed. Have a nice day.')

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Sat May 9 09:49:40 2020
5
6  @author: Daniel Sharp
7  """
8
9  # Import libraries
10 import pandas as pd
11 import matplotlib.pyplot as plt
12
13 def generate_table(order):
14     """
15     Generate table from order dictionary. Calculate tax and insert
      total columns and rows.
16
17     Parameters
18     -----
19     order : TYPE Dictionary of lists.
20             DESCRIPTION. Dict with order columns as keys, values are
      order "Units", "Product", "LUC".
21
22     Returns
23     -----
24     total_cost : TYPE float.
25                 DESCRIPTION. Total cost of order inc tax.
26
27     """
28
29     df = pd.DataFrame.from_dict(order) # Create dataframe.
30
31     df[['Units', 'LUC']] = df[['Units', 'LUC']].apply(pd.to_numeric
      ) # Convert units & LUC to numeric type.
32     df['Total ex GST'] = df['LUC'] * df['Units'] # Calculate
      product total ex GST.
33     df[['Units', 'LUC', 'Total ex GST']] = df[['Units', 'LUC', '
      Total ex GST']].apply(pd.to_numeric)
34     df['Total inc GST'] = df['LUC'] * df['Units'] * 1.1 # Calculate
      GST.
35
36     df = df.round({'LUC': 2, 'Units': 0, 'Total ex GST': 2, 'Total
      inc GST': 2}) # Round to two decimal places.
37     df = df.set_index('Product')
38
39     df.loc['Total'] = df[['Total ex GST', 'Total inc GST']].sum().
      reindex(df.columns, fill_value='') # Sum totals.

```



```

40 df = df.round({'LUC': 2, 'Units': 0, 'Total ex GST': 2, 'Total
inc GST':2}) #Round totals.
41 df.reset_index(level=0, inplace=True)
42 df = df.set_index('Product')
43 df.reset_index(level=0, inplace=True)
44 total_cost = str(df.loc[df.index[-1], 'Total inc GST']) #
Create total cost string.
45
46 table = pd.DataFrame(df) # Plot dataframe as table.
47 table = plt.table(cellText=df.values, colLabels=table.columns,
loc='left',colColours=['darkorange']*df.shape[1], colWidths
=[0.8,0.1,0.12,0.2,0.2])
48 table.auto_set_font_size(False)
49 table.set_fontsize(8) # Set font size to 8.
50 plt.axis('off')
51 plt.savefig('order.png', bbox_inches='tight') #Save order table
as .png.
52 plt.clf()
53
54 return total_cost

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat May 9 09:49:40 2020
5
6 @author: Daniel Sharp
7 """
8
9 from reportlab.pdfgen import canvas
10 from reportlab.lib.units import mm
11 import os.path
12 import pathlib
13 import os
14 import coord as cd
15
16
17 def generate_pdf(company, filename, full_name, mobile, email, date,
notes):
18     """
19     Draws pdf document. Inserts details from spreadsheet.
20     Self commenting.
21
22     Returns
23     -----
24     None. Saves PO in Sent Orders folder.
25
26     """
27
28     pdf = os.path.join('Sent Orders',company)
29
30     pathlib.Path(pdf).mkdir(parents=True, exist_ok=True)
31
32     pdf = os.path.join('Sent Orders',company, filename + '.pdf')
33
34     c = canvas.Canvas(pdf, bottomup=1)
35

```

```

36 c.drawImage('order.png',20, 320, width=890, preserveAspectRatio
    =True, anchor='c')
37
38 c.rect(15, 15, 565, 810, stroke=1, fill=0)
39 c.rect(45, 240, 500, 400, stroke=1, fill=0)
40 c.rect(45, 45, 500, 160, stroke=1, fill=0)
41
42 c.setFont('Helvetica-Bold', 25)
43
44 c.drawString(*cd.coord(125,279, mm), text='Purchase Order')
45
46 c.setFont('Helvetica-Bold', 14)
47
48 # c.drawImage('insert_logo_here.png',40, 740, width=162.5,
    preserveAspectRatio=True, mask='auto', anchor='c')
49
50 c.drawString(*cd.coord(15, 270, mm), text='57-97 Ploughshare St
    ')
51 c.drawString(*cd.coord(15, 265, mm), text='Trinity 2024')
52 c.drawString(*cd.coord(15, 260, mm), text='05 8060 9999')
53 c.drawString(*cd.coord(15, 255, mm), text='danstearooms.com')
54
55 c.drawString(*cd.coord(30, 215, mm), text=full_name)
56 c.drawString(*cd.coord(30, 210, mm), text=company)
57 c.drawString(*cd.coord(30, 205, mm), text=mobile)
58 c.drawString(*cd.coord(30, 200, mm), text=email)
59
60 c.drawString(*cd.coord(112.5,215, mm), text=filename)
61 c.drawString(*cd.coord(112.5,210, mm), text=date)
62
63 c.drawString(*cd.coord(25, 60, mm), text='DELIVERY INSTRUCTIONS
    :')
64
65 c.setFont('Helvetica-Bold', 12)
66
67 c.drawString(*cd.coord(30 ,120, mm), text='Notes:')
68
69 c.setFont('Helvetica', 12)
70
71 c.drawString(*cd.coord(30 ,115, mm), text='Please include PO
    reference on invoice.')
72 c.drawString(*cd.coord(30 ,110, mm), text=notes)
73
74 c.drawString(*cd.coord(25, 55, mm), text='Deliver between 10am
    & 5pm Monday - Friday')
75 c.drawString(*cd.coord(25, 50, mm), text='Use loading dock on
    Lens St')
76 c.drawString(*cd.coord(25, 35, mm), text='For any issues please
    contact:')
77 c.drawString(*cd.coord(25, 30, mm), text='dan@danstearooms.com'
    )
78
79 c.showPage()
80 c.save()

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """

```

```

4 Created on Sat May 9 16:25:59 2020
5
6 @author: Daniel Sharp
7 """
8
9 def generate_po_ref():
10     """
11     Convert date and time to hexadecimal.
12
13     Returns
14     -----
15     hexNum : TYPE Hexidecial Number
16             DESCRIPTION. Last five characters of the current date and
17             time in hexadecimal.
18     """
19     from datetime import datetime
20
21     now = datetime.now()
22     date_time = now.strftime("%d%m%Y%H%M%S")
23     int_date_time = int(date_time)
24     intNum = int_date_time
25     po_ref = hex(intNum).upper()[-5:]
26
27     return po_ref

```

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Mon May 11 22:06:45 2020
5
6 @author: Daniel Sharp
7 """
8
9 from datetime import datetime
10
11 def generate_date():
12     """
13
14     Returns
15     -----
16     date : TYPE String
17             DESCRIPTION. Current date.
18     """
19
20
21
22     now = datetime.now() # current date and time
23     date = now.strftime("%d/%m/%Y")
24     date = str(date)
25     return date

```

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat May 9 09:49:40 2020
5

```

```

6 @author: Daniel Sharp
7 """
8
9 def is_empty(any_structure):
10     """
11     Check if any container is empty.
12
13     Parameters
14     -----
15     any_structure : TYPE Any data container.
16                     DESCRIPTION.
17
18     Returns
19     -----
20     bool
21     DESCRIPTION. True if container is empty. False if contains
22     any data.
23     """
24     if any_structure:
25         return False
26     else:
27         return True

```

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat May 9 10:02:48 2020
5
6 @author: Daniel Sharp
7 """
8
9 def coord(x, y, unit=1):
10     """
11     Converts pdf spacing co-ordinates to metric.
12
13     """
14     x, y = x * unit, y * unit
15     return x, y

```