Purchase Management with Python

Daniel Sharp

May 2020

1 Introduction

I developed this program for two reasons. Firstly, to significantly reduce the time and number of errors in the wine ordering process. Secondly, to create a bespoke purchase management system to facilitate the tracking, reconciliation and payment of multiple orders from multiple suppliers in a time efficient manner.

A description of the development process and implementation follows. Please get in touch at *dan@dan-sharp.com* if you have any feedback.

2 Ordering Process and Potential for Automation.

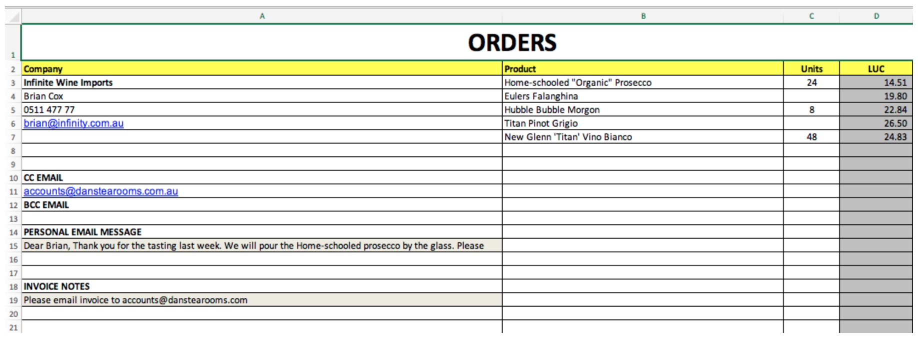
The wine ordering process can be broken down into smaller steps.

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

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



	A	B	C	D
1	ORDERS			
2	Company	Product	Units	LUC
3	Infinite Wine Imports	Home-schooled "Organic" Prosecco	24	14.51
4	Brian Cox	Eulers Falanghina		19.80
5	0511 477 77	Hubble Bubble Morgon	8	22.84
6	brian@infinity.com.au	Titan Pinot Grigio		26.50
7		New Glenn 'Titan' Vino Bianco	48	24.83
8				
9				
10	CC EMAIL			
11	accounts@danstearooms.com.au			
12	BCC EMAIL			
13				
14	PERSONAL EMAIL MESSAGE			
15	Dear Brian, Thank you for the tasting last week. We will pour the Home-schooled prosecco by the glass. Please			
16				
17				
18	INVOICE NOTES			
19	Please email invoice to accounts@danstearooms.com			
20				
21				

Figure 1: Wine ordering spreadsheet example.

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 the 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 `order.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. The PO reference can be absolutely anything simply by changing the function.

This module also saves a copy of the pdf in a series of folders called 'Draft Orders' with sub folders labelled with the current date and company.

A Pandas dataframe is created with the details of each draft order is created and printed in the console as pictured below. This allows you to check the purchase orders before they are sent and verify the amount spent to make sure no spending caps are being exceeded. This function can also be altered to make sure order are large enough for free freight and throw up a warning if not.

The `send_drafts` function takes three inputs. 'Y' calls `send_message` function and send all orders in the dataframe. 'N' deletes the drafts folder and any other input prints `Invalid input` and recalls the function.

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.

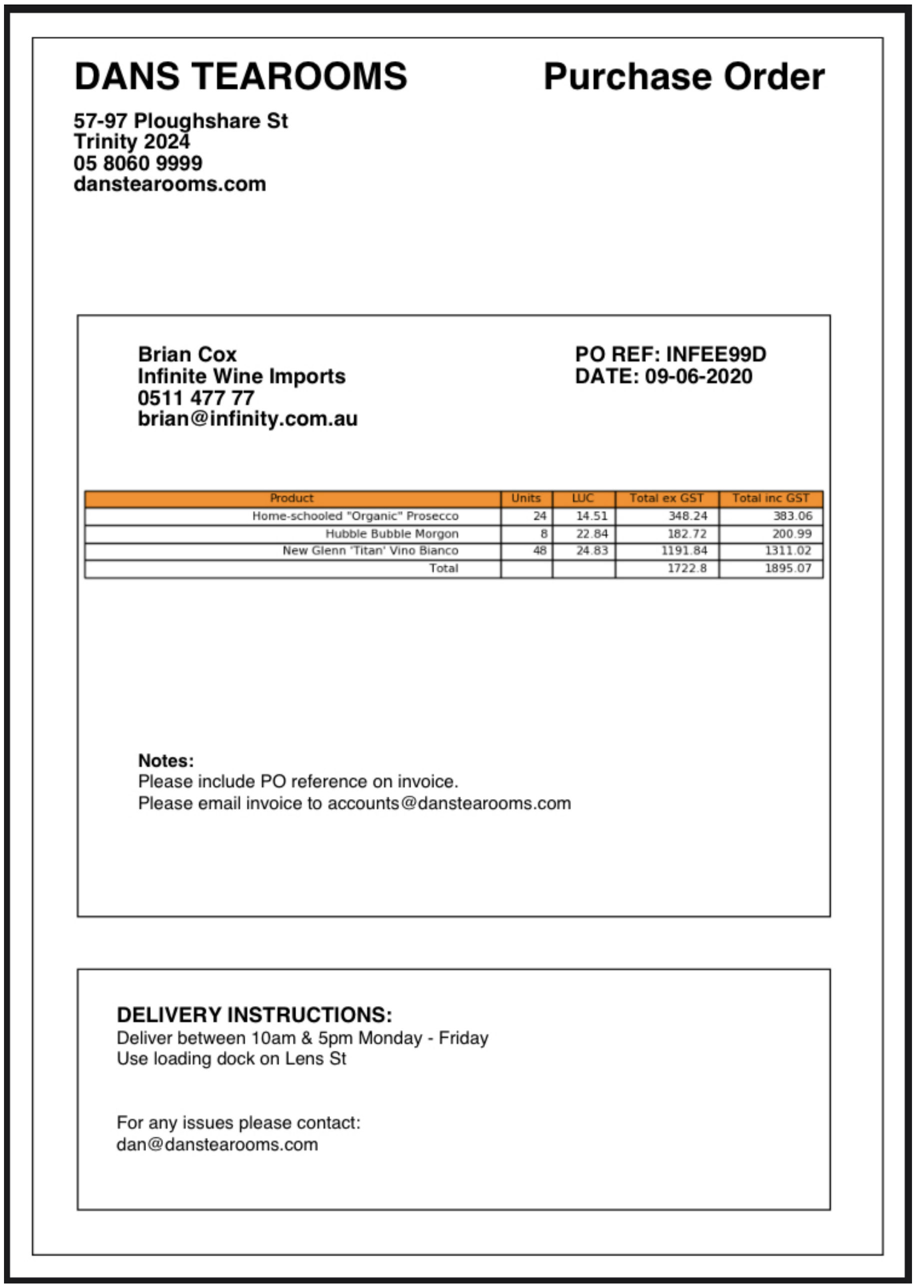
After all order blocks in the active sheet have been iterated over the workbook is saved and the orders in the 'Draft Orders' folder are moved into the 'Sent Orders' folder.

The 'Sent Orders' folder contains dated folders with all companies that orders were sent to on those dates. This allows any PO to be found quickly by date or company.

5 Conclusion

I have been using this program for the past few months. It has reduced the ordering time required per week from 90 to 20 minutes. The larger the number of suppliers the more time is saved. The greatest benefit has been in stock control and accounting. 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 the invoice total can easily be checked against the order sheet.

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



DANS TEAROOMS **Purchase Order**

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

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

PO REF: INFEE99D
DATE: 09-06-2020

Product	Units	LUC	Total ex GST	Total inc GST
Home-schooled "Organic" Prosecco	24	14.51	348.24	383.06
Hubble Bubble Morgon	8	22.84	182.72	200.99
New Glenn 'Titan' Vino Bianco	48	24.83	1191.84	1311.02
Total			1722.8	1895.07

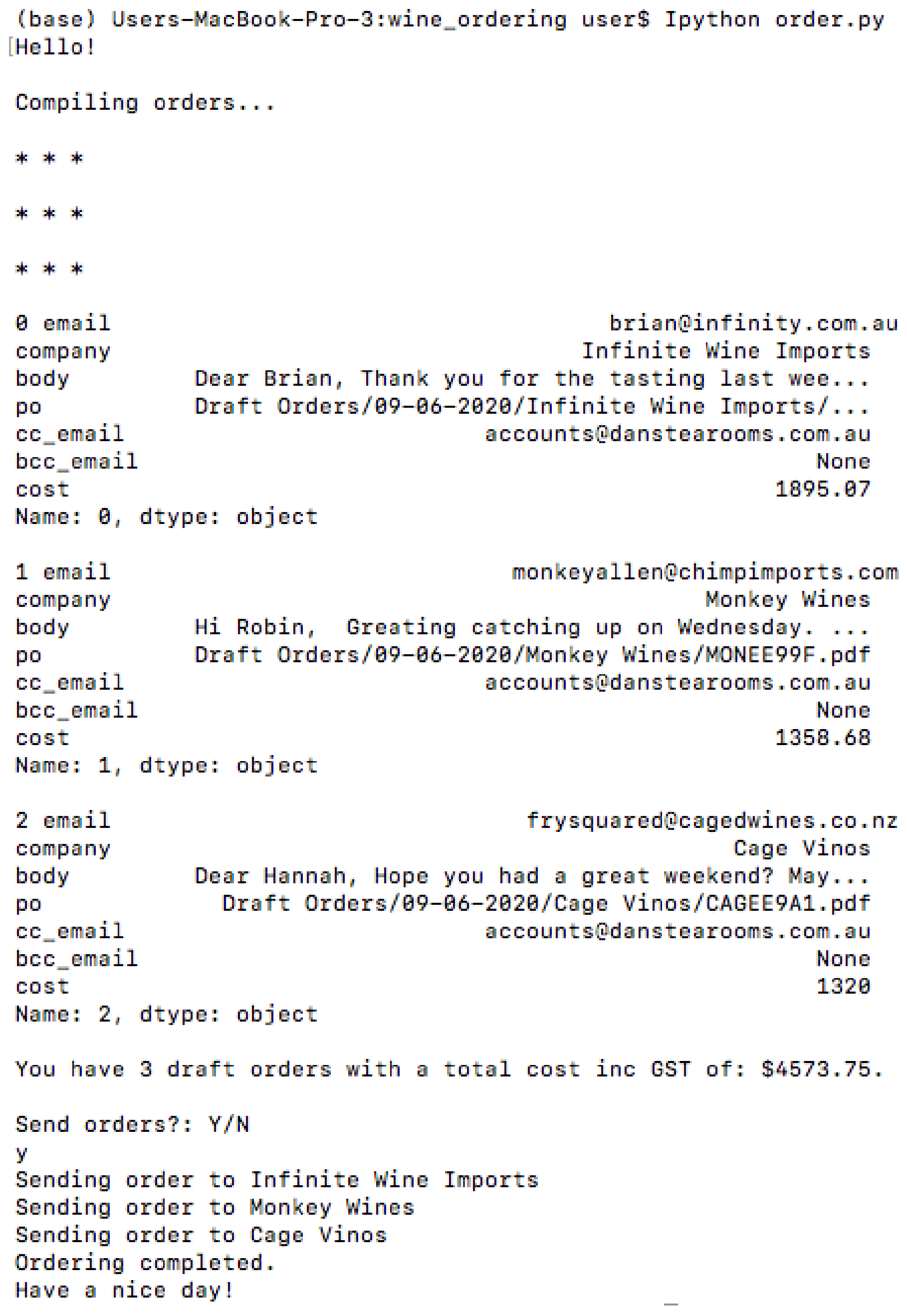
Notes:
Please include PO reference on invoice.
Please email invoice to accounts@danstearooms.com

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

For any issues please contact:
dan@danstearooms.com

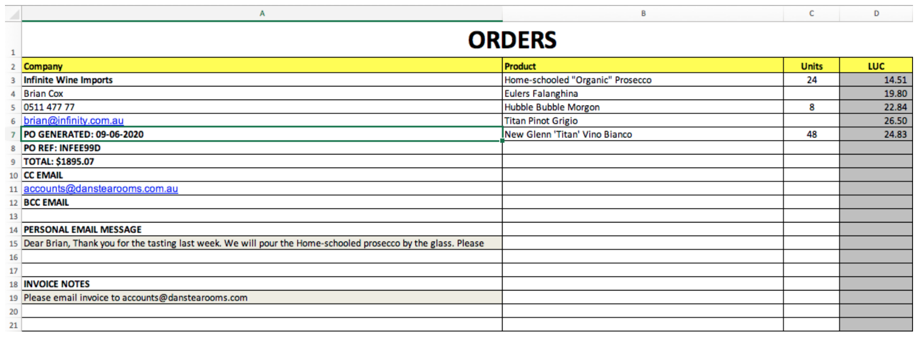
Figure 2: Purchase order pdf created from first order block in figure 1.

for the month to the accounting team, or rejects orders if they are over a certain budget cap.



```
(base) Users-MacBook-Pro-3:wine_ordering user$ Ipython order.py
Hello!

Compiling orders...

* * *

* * *

* * *

0 email                                    brian@infinity.com.au
company                                  Infinite Wine Imports
body         Dear Brian, Thank you for the tasting last wee...
po           Draft Orders/09-06-2020/Infinite Wine Imports/...
cc_email                          accounts@danstearooms.com.au
bcc_email                                                 None
cost                                                   1895.07
Name: 0, dtype: object

1 email                             monkeyallen@chimpimports.com
company                                           Monkey Wines
body         Hi Robin,  Greating catching up on Wednesday. ...
po           Draft Orders/09-06-2020/Monkey Wines/MONEE99F.pdf
cc_email                          accounts@danstearooms.com.au
bcc_email                                                 None
cost                                                   1358.68
Name: 1, dtype: object

2 email                              frysquared@cagedwines.co.nz
company                                             Cage Vinos
body         Dear Hannah, Hope you had a great weekend? May...
po             Draft Orders/09-06-2020/Cage Vinos/CAGEE9A1.pdf
cc_email                          accounts@danstearooms.com.au
bcc_email                                                 None
cost                                                      1320
Name: 2, dtype: object

You have 3 draft orders with a total cost inc GST of: $4573.75.

Send orders?: Y/N
y
Sending order to Infinite Wine Imports
Sending order to Monkey Wines
Sending order to Cage Vinos
Ordering completed.
Have a nice day!
```

Figure 3: Command line screen printout for three orders.



ORDERS

Company	Product	Units	LUC
Infinite Wine Imports	Home-schooled "Organic" Prosecco	24	14.51
Brian Cox	Eulers Falanghina		19.80
0511 477 77	Hubble Bubble Morgon	8	22.84
brian@infinity.com.au	Titan Pinot Grigio		26.50
PO GENERATED: 09-06-2020	New Glenn 'Titan' Vino Bianco	48	24.83
PO REF: INFEE99D			
TOTAL: $1895.07			
CC EMAIL			
accounts@danstearooms.com.au			
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			

Figure 4: Order spreadsheet from figure 1 after order has been sent.

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

6 Appendix

```python
"""
Created on Mon May 11 17:21:19 2020

Purpose: To compile, generate purchase order and email orders from
    an Excel spreadsheet.

Author: Daniel Sharp
"""

import os
import openpyxl
import generate_po_ref as pr
import generate_date as gd
import generate_table as gt
import generate_pdf as gp
import is_empty as ie
import send_drafts as sd
import pandas as pd
import time
import shutil

INCREMENT = 20 # Define order block length. Change to worksheet
    specifications.

print('Hello!\n \nCompiling orders...\n')

wb = openpyxl.load_workbook('Order Sheet/order_sheet_2020.xlsx') #
    Open order workbook.
sheet = wb.active # Select active sheet
row_count = sheet.max_row # Get length of sheet

start = 3 # Start value of first order block.
end = 21 # End value of first order block.

cost_list = [] # Create empty cost list.
email_list = []
company_list = []
body_list = []
po_list = []
cc_list = []
bcc_list = []

drafts = {}

for order_blocks in range(0,row_count): # Select worksheet rows to
    iterate over.

    order = {} # Create empty order dictionary.
    units_list = [] # Create empty number of units list.
    product_list = [] # Create empty product list
    luc_list = [] # Create empty price list.

    for cell_value in range(start, end): # Iterate over cells in
    order block.
        date = gd.generate_date() # Call date function.
        po_ref = pr.generate_po_ref() # Call generate po refernce function.

        company = sheet.cell(start,1).value # Get company name from spreadsheet
        if ie.is_empty(company) == True: # If no company in cell, break out of loop.
            break

        full_name = sheet.cell(start + 1, 1).value # Get email recipient.
        mobile = sheet.cell(start + 2, 1).value # Get mobile number.
        email = sheet.cell(start + 3, 1).value # Get company email.
        cc_email = sheet.cell(start + 8, 1).value # Get cc email.
        bcc_email = sheet.cell(start + 10, 1).value # Get bcc email.
        filename = company[0:3].upper() + po_ref # Create unique PO reference.
        notes = sheet.cell(start + 16, 1).value # Get invoice notes.
        body = sheet.cell(start + 12, 1).value # Get email body.
        suffix = '.pdf'
        po = os.path.join('Draft Orders',date, company, filename + suffix) # Defines PO attachment name.

        quantity = sheet.cell(cell_value,3).value # Get quantity ordered.

        if ie.is_empty(quantity) == False: # Create order if a number of units is specified.

            product = sheet.cell(cell_value,2).value # Create list of products ordered.
            product_list.append(product)

            units = sheet.cell(cell_value,3).value # Create list of units ordered.
            units_list.append(units)

            luc = sheet.cell(cell_value,4).value # Create list of prices.
            luc_list.append(luc)

        order['Units'] = units_list # Create order dictionary of lists.
        order['Product'] = product_list
        order['LUC'] = luc_list

    if ie.is_empty(units_list) == False: # If units list is not empty create order pdf and email.
        gt.generate_table(order) # Create order table and PO pdf.
        gp.generate_pdf(company, filename, full_name, mobile, email, date, notes)

        email_list.append(email)
        company_list.append(company)
        body_list.append(body)
        po_list.append(po)
        cc_list.append(cc_email)
        bcc_list.append(bcc_email)

        cost = gt.generate_table(order)
        cost_list.append(cost)

        sheet.cell(start + 4, 1).value = 'PO GENERATED: ' + date # Save date, PO ref and cost to spreadsheet.
        sheet.cell(start + 5, 1).value = 'PO REF: ' + filename
        sheet.cell(start + 6, 1).value = 'TOTAL: $' + cost

        cost = float(cost)

        drafts['email'] = email_list
        drafts['company'] = company_list
        drafts['body'] = body_list
        drafts['po'] = po_list
        drafts['cc_email'] = cc_list
        drafts['bcc_email'] = bcc_list
        drafts['cost'] = cost_list
        print('* * *\n')
        time.sleep(1)

    start += INCREMENT # Increment start and end values to next block.
    end += INCREMENT

wb.save('Order Sheet/order_sheet_2020.xlsx') # Save workbook

if ie.is_empty(cost_list) == False:
    df = pd.DataFrame(drafts)
    df['cost'] = df['cost'].astype(float)
    num_orders = len(df.index)
    total_cost = round(df['cost'].sum(),2)

    for i,j in df.iterrows():
        print(i,j)
        print()

    if num_orders == 1:
        print(f'You have {num_orders} draft order with a total cost inc GST of: ${total_cost}.\n')
    else:
        print(f'You have {num_orders} draft orders with a total cost inc GST of: ${total_cost}.\n')

    sd.send_drafts(df)

    shutil.rmtree('Draft Orders/') # Delete drafts.

else:
    print("You haven't placed any orders dumb dumb! Ask your Mum if you can have another go.")

print('Ordering completed.\nHave a nice day!')
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat May  9 09:49:40 2020

@author: Daniel Sharp
"""

# Import libaries
import pandas as pd
import matplotlib.pyplot as plt

def generate_table(order):
    """
    Generate table from order dictionary. Calculate tax and insert
    total columns and rows.

    Parameters
    ----------
    order : TYPE Dictionary of lists.
        DESCRIPTION. Dict with order columns as keys, values are
    order "Units", "Product", "LUC".

    Returns
    -------
    total_cost : TYPE float.
        DESCRIPTION. Total cost of order inc tax.

    """

    df = pd.DataFrame.from_dict(order) # Create dataframe.

    df[['Units', 'LUC']] = df[['Units', 'LUC']].apply(pd.to_numeric
    ) # Convert units & LUC to numeric type.
    df['Total ex GST'] = df['LUC'] * df['Units'] # Calculate
    product total ex GST.
    df[['Units', 'LUC', 'Total ex GST']] = df[['Units', 'LUC', '
    Total ex GST']].apply(pd.to_numeric)
    df['Total inc GST'] = df['LUC'] * df['Units'] * 1.1 # Calculate
     GST.

    df = df.round({'LUC': 2, 'Units': 0, 'Total ex GST': 2, 'Total
    inc GST':2}) # Round to two decimal places.
    df = df.set_index('Product')

    df.loc['Total'] = df[['Total ex GST', 'Total inc GST']].sum().
    reindex(df.columns, fill_value='') # Sum totals.
    df = df.round({'LUC': 2, 'Units': 0, 'Total ex GST': 2, 'Total
    inc GST':2}) #Round totals.
    df.reset_index(level=0, inplace=True)
    df = df.set_index('Product')
    df.reset_index(level=0, inplace=True)
    total_cost = str(df.loc[df.index[-1], 'Total inc GST']) #
    Create total cost string.

    table = pd.DataFrame(df) # Plot dataframe as table.
    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])
    table.auto_set_font_size(False)
    table.set_fontsize(8) # Set font size to 8.
    plt.axis('off')
    plt.savefig('order.png', bbox_inches='tight') #Save order table
     as .png.
    plt.clf()

    return total_cost
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat May  9 09:49:40 2020

@author: Daniel Sharp
"""

from reportlab.pdfgen import canvas
from reportlab.lib.units import mm
import os.path
import pathlib
import os
import coord as cd


def generate_pdf(company, filename, full_name, mobile, email, date,
     notes):
    """
    Draws pdf document. Inserts details from spreadsheet.
    Self commenting.

    Returns
    -------
    None. Saves PO in Draft Orders folder.

    """

    pdf = os.path.join('Draft Orders', date, company)

    pathlib.Path(pdf).mkdir(parents=True, exist_ok=True)

    pdf = os.path.join('Draft Orders', date, company, filename + '.
    pdf')

    c = canvas.Canvas(pdf, bottomup=1)

    c.drawImage('order.png',42.5, 300, width=850,
    preserveAspectRatio=True, anchor='c')

    c.rect(15, 15, 565, 810, stroke=1, fill=0)
    c.rect(45, 240, 500, 400, stroke=1, fill=0)
    c.rect(45, 45, 500, 160, stroke=1, fill=0)

    c.setFont('Helvetica-Bold', 25)

    c.drawString(*cd.coord(125,279, mm), text='Purchase Order')
```

```python

    c.drawString(*cd.coord(15,279, mm), text='DANS TEAROOMS')

    c.setFont('Helvetica-Bold', 14)

    # c.drawImage('Black logo - no background.png',40, 740, width
    =50, preserveAspectRatio=True, mask='auto', anchor='c')

    c.drawString(*cd.coord(15, 270, mm), text='57-97 Ploughshare St
    ')
    c.drawString(*cd.coord(15, 265, mm), text='Trinity 2024')
    c.drawString(*cd.coord(15, 260, mm), text='05 8060 9999')
    c.drawString(*cd.coord(15, 255, mm), text='danstearooms.com')

    c.drawString(*cd.coord(30, 215, mm), text=full_name)
    c.drawString(*cd.coord(30, 210, mm), text=company)
    c.drawString(*cd.coord(30, 205, mm), text=mobile)
    c.drawString(*cd.coord(30, 200, mm), text=email)

    c.drawString(*cd.coord(132.5,215, mm), text='PO REF: ' +
    filename)
    c.drawString(*cd.coord(132.5,210, mm), text='DATE: ' + date)

    c.drawString(*cd.coord(25, 60, mm), text='DELIVERY INSTRUCTIONS
    :')

    c.setFont('Helvetica-Bold', 12)

    c.drawString(*cd.coord(30 ,120, mm), text='Notes:')

    c.setFont('Helvetica', 12)

    c.drawString(*cd.coord(30 ,115, mm), text='Please include PO
    reference on invoice.')
    c.drawString(*cd.coord(30 ,110, mm), text=notes)

    c.drawString(*cd.coord(25, 55, mm), text='Deliver between 10am
    & 5pm Monday - Friday')
    c.drawString(*cd.coord(25, 50, mm), text='Use loading dock on
    Lens St')
    c.drawString(*cd.coord(25, 35, mm), text='For any issues please
     contact:')
    c.drawString(*cd.coord(25, 30, mm), text='dan@danstearooms.com'
    )

    c.showPage()
    c.save()
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat May  9 16:25:59 2020

@author: Daniel Sharp
"""

def generate_po_ref():
    """
    Convert date and time to hexidecimal.

    Returns
    -------
    hexNum : TYPE Hexidecial Number
        DESCRIPTION. Last five characters of the current date and time in hexidecimal.

    """
    from datetime import datetime

    now = datetime.now()
    date_time = now.strftime("%d%m%Y%H%M%S")
    int_date_time = int(date_time)
    intNum = int_date_time
    po_ref = hex(intNum).upper()[-5:]

    return po_ref
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon May 11 22:06:45 2020

@author: Daniel Sharp
"""

from datetime import datetime

def generate_date():
    """


    Returns
    -------
    date : TYPE String
        DESCRIPTION. Current date.

    """

    now = datetime.now() # current date and time
    date = now.strftime("%d-%m-%Y")
    date = str(date)
    return date
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 24 22:27:50 2020

@author: user
"""

import generate_date as gd
import send_loop as sl
import move_files as ml

```

```python
date = gd.generate_date()

def send_drafts(df):

    df.to_string(index=False)
    value = input("Send orders?: Y/N\n")
    value = value.upper()

    if value == 'Y':

        for i in range(len(df)):
            email = df.iloc[i, 0]
            company = df.iloc[i, 1]
            body = df.iloc[i, 2]
            po = df.iloc[i, 3]
            cc_email = df.iloc[i, 4]
            bcc_email = df.iloc[i, 5]

            print(f'Sending order to {company}')
            sl.send_loop(email, company, body,po,cc_email,bcc_email
    )

            ml.move_files(company, po)

    elif value == 'N':
        print('Orders not sent.')

    else:
        print('Invalid input. Please try again.')
        send_drafts(df)
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May 26 22:34:27 2020

@author: user
"""
import generate_date as gd
import os.path
import pathlib
import os
import shutil

date = gd.generate_date()

def move_files(company,po):

    filename = po[-12:]

    destination = os.path.join('Sent Orders', date, company)

    pathlib.Path(destination).mkdir(parents=True, exist_ok=True)

    destination = os.path.join('Sent Orders', date, company,
    filename)

    source = os.path.join('Draft Orders', date, company, filename)

    for item in source:
        shutil.copy(source, destination)
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat May  9 09:49:40 2020

@author: Daniel Sharp
"""

def is_empty(any_structure):
    """
    Check if any container is empty.

    Parameters
    ----------
    any_structure : TYPE Any data container.
        DESCRIPTION.

    Returns
    -------
    bool
        DESCRIPTION. True if container is empty. False if contains
    any data.

    """
    if any_structure:
        return False
    else:
        return True
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat May  9 10:02:48 2020

@author: Daniel Sharp
"""

def coord(x, y, unit=1):
    """
    Converts pdf spacing co-ordinates to metric.

    """
    x, y = x * unit, y * unit
    return x, y
```