Contents

In the bible there is a story as old as Jesus Christ himself; when Mary and Joseph came to Bethlehem and walked up to the inn and asked for a room the innkeeper was all sold out. As he had sold his rooms to cheaply thus he was fully booked. If he had used dynamic pricing, then he probably would have had room and have made more money in the process.

The problem:

In hotels, the Revenue Manager's job is to maximise the Hotels yield for each and every day. The problem is that the revenue manager needs to trawl through a massive database and retrieve data, then make an educated guess as to how they might want to change the room price to maximise profits. The aim of my program is to help them to do that more efficiently and effectively so that they can make informed decisions and maximise the hotels daily yield.

The goal of my Program is to improve forecasting accuracy. This is about predicting how many rooms will be sold and at what price. This is essential as it enables the hotel to plan and budget staffing efficiently.

Room yield is about maximising the amount of money made by selling hotel rooms. For example, if your Average Daily Rate (ADR) is £200 a room, but you only sell 2 rooms then that's £400. Whereas if you sell 100 rooms for £50 each then your revenue is £5000. Although your ADR is only £50 you have made a lot more money. So, in conclusion, it's about selling all your rooms, whilst giving you the highest ADR to maximise profits.

It is suitable to solve this problem by computational methods. Whereas currently for humans to do the same thing, it takes them over an hour. Computers can do multiple things at once and very quickly. They can crawl through a massive database, making large calculations very easily and accurately. Thus, computers would be far faster and more accurate than a human undertaking the same task.

My program is a proof of concept, as it would be deployed on a server (Windows Server 2012), thus a python program would not be suitable. Once the program is designed and developed it will be sent off to a company to write in a suitable programming language. Suitable for a corporate Microsoft based client-server environment, such as C++ or Visual Basic so that it can be compiled and then run as an .EXE on the clients' computer.

Stakeholders:

1 - Managing Director of Anjuna Hotels (Yousif Al-Wagga)

Yousif Al-Wagga, the Manager of Anjuna Hotels would rely on my program to provide an efficient and effective solution. A successful program would aid his team in performing to the best of their ability.

2 - Anjuna Hotels group

Anjuna Hotels is the recent forming of 2 small hotel groups to create a group of 6 hotels. It is the partnership of Gerry Witton - entrepreneur, who owns 3 hotels under the Chocolate Hotel brand and Quantum group - who own 3 in East Cliff (Bournemouth). The partnership was created to minimalize costs, improve growth rate and stabilise finances. This can be done by streamlining processes and improving the business model. It will be overseen by Yousif Al-Wagga who has been hired for his experience to ensure the group's success.

Currently, they are considering buying a new hotel in Henley. To do this they need to ensure that their database will work with the additional hotels being added, as they intend to expand. They will need to ensure a smooth incorporation into the group and its subsequent database.

Hotels/ Revenue Manager (Any Nemeth)

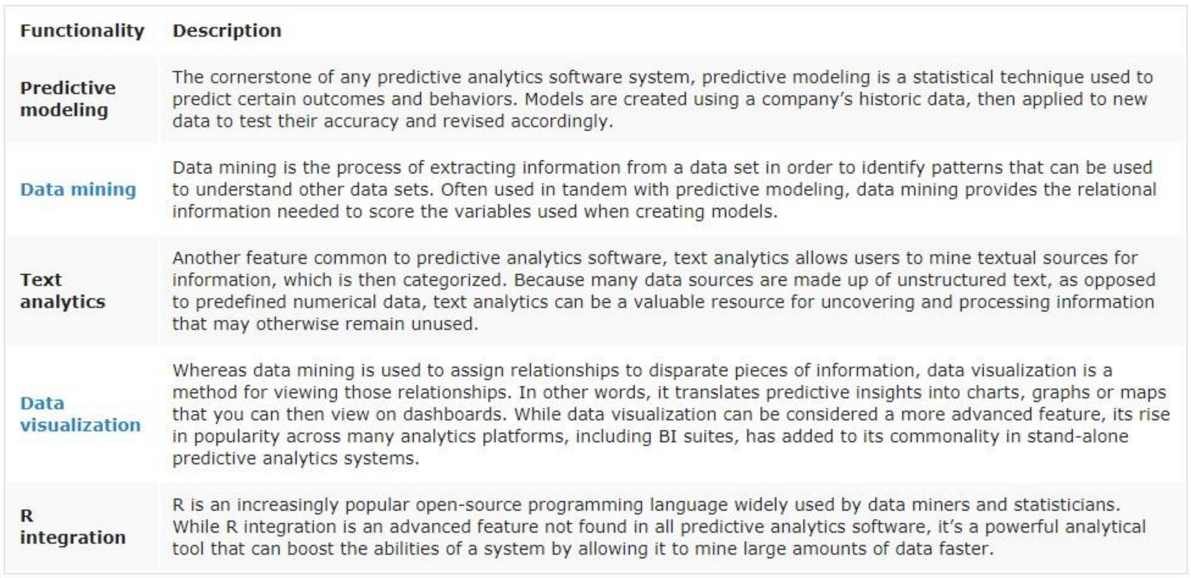
The Revenue Managers job is to maximise profits, this can be done by dynamic pricing ensuring that it is calculated faster and accurately every time. This will then inform the Revenue Manager to either increase the room’s price, leave it the same or to lower the price and subsequently allowing her to put out an advert campaign, for example, Facebook advertising. The program is intended to allow the Revenue Manager to spend more time using her experience and expertise. Rather than her making guesses and simple calculations for multiple hours which could have easily been done easily by a computer.

Project Consultant for Hotels System Solutions (my client)

My software will ideally be sold to the Financial Consultant, whose job is to manage and oversee the finances of hotels. Thus, the idea would be that he would recommend and implement the software. He would charge a subscription for its use this would include updates and other features added on at later stages.

Research into other systems/ topic areas:

Common Features of Predictive Analytics Software



Functionality	Description
Predictive modeling	The cornerstone of any predictive analytics software system, predictive modeling is a statistical technique used to predict certain outcomes and behaviors. Models are created using a company's historic data, then applied to new data to test their accuracy and revised accordingly.
Data mining	Data mining is the process of extracting information from a data set in order to identify patterns that can be used to understand other data sets. Often used in tandem with predictive modeling, data mining provides the relational information needed to score the variables used when creating models.
Text analytics	Another feature common to predictive analytics software, text analytics allows users to mine textual sources for information, which is then categorized. Because many data sources are made up of unstructured text, as opposed to predefined numerical data, text analytics can be a valuable resource for uncovering and processing information that may otherwise remain unused.
Data visualization	Whereas data mining is used to assign relationships to disparate pieces of information, data visualization is a method for viewing those relationships. In other words, it translates predictive insights into charts, graphs or maps that you can then view on dashboards. While data visualization can be considered a more advanced feature, its rise in popularity across many analytics platforms, including BI suites, has added to its commonality in stand-alone predictive analytics systems.
R integration	R is an increasingly popular open-source programming language widely used by data miners and statisticians. While R integration is an advanced feature not found in all predictive analytics software, it's a powerful analytical tool that can boost the abilities of a system by allowing it to mine large amounts of data faster.

IDeas

IDeaS Forecast Management System

Offers the ideal solution for hotels that want to profit from demand-driven forecasting insight and reporting without adopting a fully automated revenue management system.

I will be analysing the IDeas Forecast Management System, as it's also the only one I found that had any actual forecasting to it, most were budgeting systems. This program can set a hotel group back about £100,000 a year. It uses a huge amount of predictive formulas and does dynamic pricing which is what my program also intends to do, just cheaper. The idea of my program is to be a cheaper alternative to the few core fundamental things that IDeas does. To be used by small to medium Hotel groups, subsequently allowing me to add on extra features and bug fixes.

Guest Line

Guest Line deals with the billing of customers and room management, it is a fundamental core program to this group's hotels operation. It includes channel distribution, which is about sending the price of rooms and uploading it to sites such as Trivago and Late Bookings etc.

Jedox

Jedox is cube based business intelligence software accessed by the user through a Microsoft Excel Add-In. In this instance, it will process a data feed via an Import Process. My program would not interact with the data through the Jedox cube databases, but will instead act upon the excel reports that are populated by Jedox queries embedded in the spreadsheet.

Despite this, I still believe it to be beneficial to have a good understanding of Jedox (my client also recommends it). Thus, as I do research I will be talking about it:

Jedox data is stored in Cubes and the values stored in the cube are generally Double Integer values, although strings can also be stored. Throughout the project, Cubes will be referenced as interchangeable with database.

Dimensions

In the database my client has set up he uses a nine dimension cube that has an array of elements in each dimension. If you imagine an 18 sided shape and each path leads to one particular side that then makes up an axis, this is done 9 times to create the values representing the nine demotions that then point to a single location with one or no value.

For example: to find a value in a location for a 3D database, you have 3 axis X (Sales), Y (Month) and Z (Product) thus if you were to reference a location then:

DatabaseID (“Bike Sales”, “April”, “Cannondale”) = 54

These three coordinates correlate to a location in the database which is then read and a value would be returned. The only difference between this and a 9D database is that there are more axis.

DatabaseID (1, 3, 7, 2, 4, 9, 2, 3, 1)

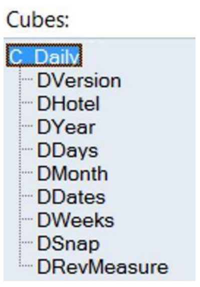
These nine coordinates also correlate to a location in the database which is then read and returned. It only starts getting complicated when you start talking about elements because they can be interpreted as their own axis but at the same time they aren't.

Elements

Let’s take our current example

DatabaseID (1, 3, 7, 2, 4, 9, 2, 3, 1)

This is who it would be written into the excel sheet with the Jedox plug-in:

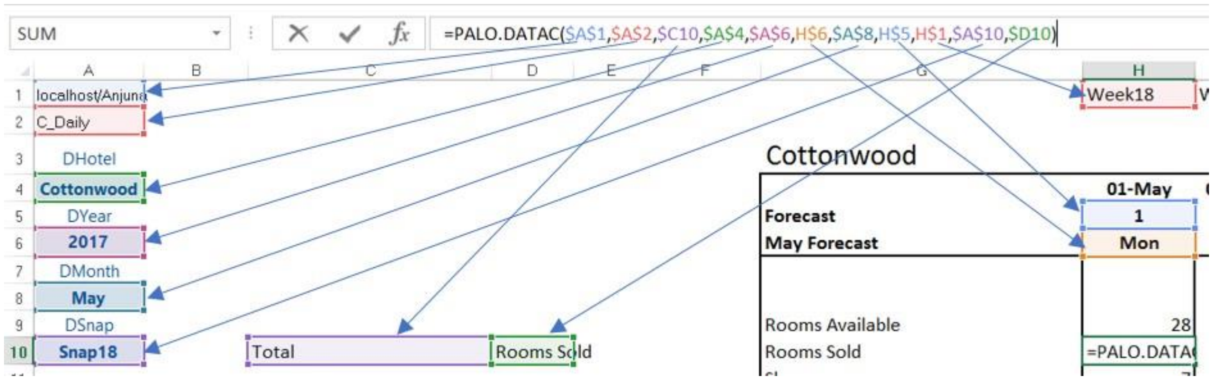


Cubes:

C_Daily
- DVersion
- DHotel
- DYear
- DDays
- DMonth
- DDates
- DWeeks
- DSnap
- DRevMeasure

This shows us all of the axes that make up the location of the value that we want to access

=PALO.DATAC(Database,C_daily,DVersion,DHotel,DYear,DDays,DMonth,DDates,DWeeks,DSnap,DRevMeasure) all of these colours refer to the formulae bar at the top of these screens. The ones that represent dimensions are from the 3rd to the last value referenced.



```
SUM   =PALO.DATAC($A$1,$A$2,$C10,$A$4,$A$6,H$6,$A$8,H$5,H$1,$A$10,$D10)
```

	A	B	C	D	E	F	G	H
1	localhost/Anjuna							Week18
2	C_Daily							
3	DHotel					Cottonwood		
4	Cottonwood							01-May
5	DYear					Forecast		1
6	2017					May Forecast		Mon
7	DMonth							
8	May							
9	DSnap					Rooms Available		28
10	Snap18		Total	Rooms Sold		Rooms Sold		=PALO.DATA(

Each axis that is referenced has a set of Elements, looking at the example above we can see:

DVersion: MY client is wanting to know about the element inside of DVersion that is called Total

DHotel: Specifies that it is for the hotel called Cottonwood

DYear: Specifies that it is for the year 2017

DDays: Specifies that it is for day Monday

DMonth: Specifies that it is for the month of May

DDates: Specifies that it is for the date number that represents the day of the month of 1

So from the previous 4 axes we know that it looking for something on Monday the 1st of May 2017

DWeeks: Specifies that it is for the week no. 18 (Week18)

DSnap: Specifies that it is for the snap no. 18 (Snap18)

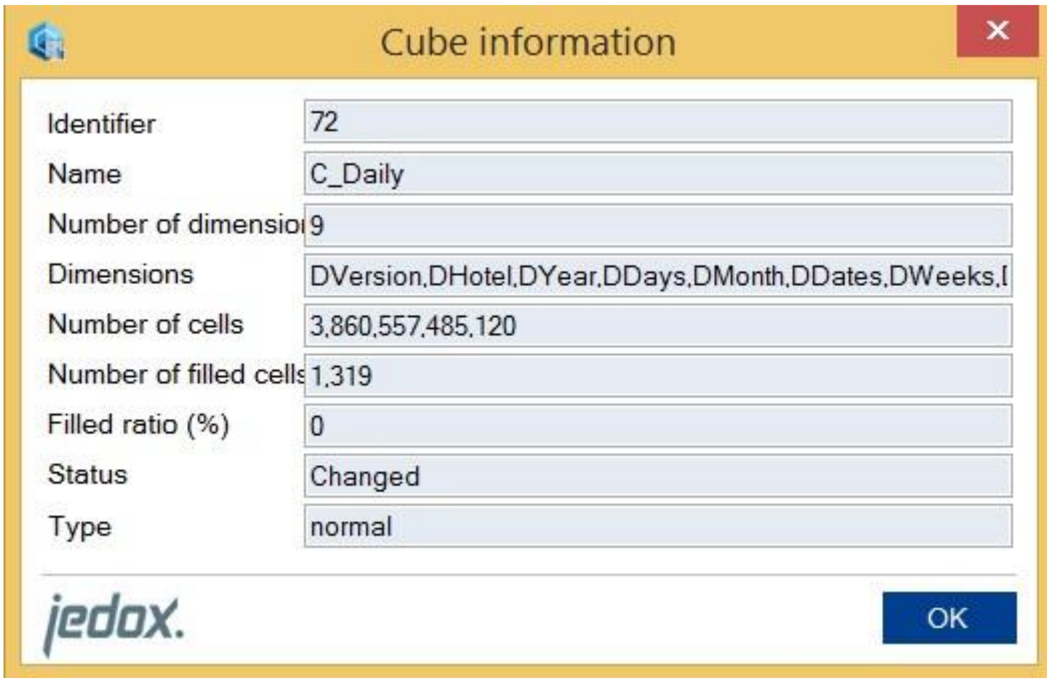
DRevMeasure: Specifies that it is for the request of what it wants to know which is rooms sold

So, in conclusion, we can say that this is a request for the data:

The total for the hotel Cottonwood in the year 2017, day Monday in the month of May (1st of May) Week 18 of the year, snap 18 of the year, and I want to know how many rooms have been sold

Storage size (possible combinations of elements)

This is a screenshot showing how many cells/data points can be stored inside of it.



Cube information

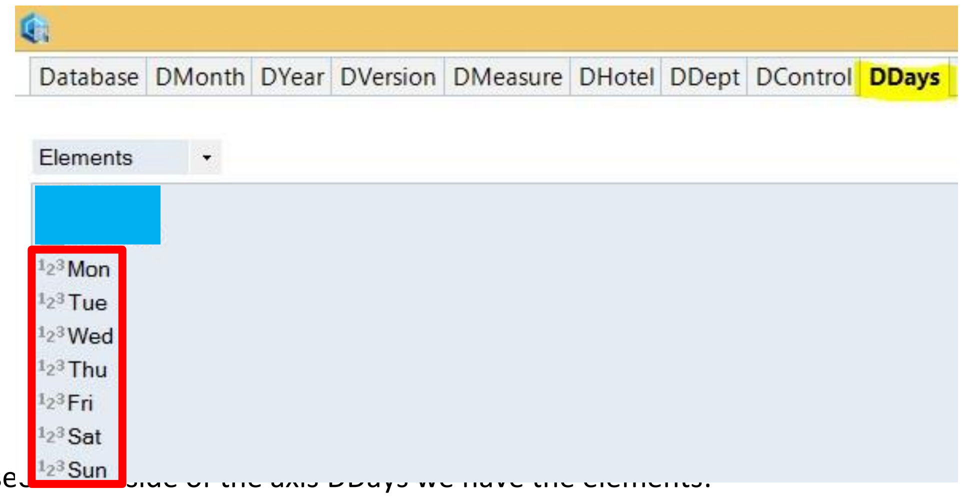
Identifier	72
Name	C_Daily
Number of dimensio	9
Dimensions	DVersion,DHotel,DYear,DDays,DMonth,DDates,DWeeks,I
Number of cells	3,860,557,485,120
Number of filled cells	1,319
Filled ratio (%)	0
Status	Changed
Type	normal

Number of cells: 3,860,557,485,120 this is an insanely huge amount of data points how is that calculated?

Axis name that is removed from the calculation	Total no. of usable cells	How the value of the Previous column is calculated. The 1* is a placeholder	Element count removed

ALL	3860557485120	12*9*9*23*42*94*54*54*15	N/A
DVersion	321713123760	1*9*9*23*42*94*54*54*15	12
DHotel	35745902640	1*1*9*23*42*94*54*54*15	9
DYear	3971766960	1*1*1*23*42*94*54*54*15	9
DDays	172685520	1*1*1*1*42*94*54*54*15	32
DMonth	4111560	1*1*1*1*1*94*54*54*15	43
DDates	43740	1*1*1*1*1*1*54*54*15	94
DWeeks	810	1*1*1*1*1*1*1*54*15	54
DSnap	15	1*1*1*1*1*1*1*1*15	54
DRevMeasure	1	1*1*1*1*1*1*1*1*1	12

What this table outlines are how many possible combinations there are for this table to show how truly large and complex it is. I also want to show you that because I want to demonstrate that for each Axis it contains many elements. Their elements can be seen as search terms, for example, looking at a simplified version of the Axis DDays:



We can se side of the axis DDays we have the elements:
Mon, Tue, Wed, Thu, Fri, Sat, Sun

These are all things that you can place in the axis called DDays which then are part of the search term.

This is where this comes in:

The total for the hotel Cottonwood in the year 2017, day Monday in the month of May (1st of May) Week 18 of the year, snap 18 of the year, and I want to know how many rooms have been sold

If you change DDays to Wed then

The total for the hotel Cottonwood in the year 2017, day Wednesday in the month of May (1st of May) Week 18 of the year, snap 18 of the year, and I want to know how many rooms have been sold

You have a problem because the 1st of May is not a Wednesday day the 3rd is. The good thing is that this isn't considered an error. The program will go to the location which it has been pointed to and since the date doesn't exist no data will have been entered into it thus it will return a value of 0.

Dynamic pricing

What is dynamic pricing?

Dynamic pricing is the continual adjustment of prices offered to guests depending upon the value these guests attribute to your products or services. Phrases such as flexible or open pricing are often used to denote dynamic pricing. We will use the term dynamic pricing in a broad sense and discuss why it is important for hoteliers in executing their revenue management strategy.

Why Should Hotels Price Dynamically?

Specifically, in the hospitality industry, dynamic pricing refers to the continual adjustment of prices based on the value of each type of demand for the remaining capacity available to sell. Determining the right prices to charge a guest for a product or a service is not as straightforward as it might appear on the surface. This task requires that a hotel company know not only its own operating costs and supply, i.e., the availability of rooms and function space etc., but also how much the guest values the product offering and what the future demand is. A hotel, therefore, needs a wealth of information about its guests and also be able to adjust its prices at minimal cost. Advances in big data and distribution technologies have dramatically increased the amount of information that hotels can gather about guests, and have provided universal connectivity to guests making it easy to change the prices.

Daily Forecast Sheet:

The Daily Forecast Sheet looks something like this with the blue arrows showing where it ends the loops around and down to the next one:

		DDays	
Date			
24th - 31st	AllWeeks	AllDays	AllDates
24/10/2017	Week43	Tue	24
25/10/2017	Week43	Wed	25
26/10/2017	Week43	Thu	26
27/10/2017	Week43	Fri	27
28/10/2017	Week44	Sat	28
29/10/2017	Week44	Sun	29
30/10/2017	Week44	Mon	30
31/10/2017	Week44	Tue	31

Rooms Available	Bob	BoB	PickUp
Available	Occupancy	Occupancy	Individuals
320	72	22.5%	40
40	7	17.5%	4
40	1	2.5%	4
40	1	2.5%	4
40	6	15.0%	8
40	28	70.0%	8
40	2	5.0%	4
40	1	2.5%	4
40	3	7.5%	4

Total	Total	Total	Business on Books
Unsold	Rooms Sold	Occupancy	average daily rate
208	112	35.0%	82.41
29	11	27.5%	75.00
35	5	12.5%	95.83
35	5	12.5%	137.49

26	14	35.0%	88.20
4	36	90.0%	83.08
34	6	15.0%	52.06
35	5	12.5%	81.66
33	7	17.5%	45.96
PickUp	PickUp	Total	Total
average daily rate	room revenue	average daily rate	room revenue
37.71	1,633	52.64	5,571
32.50	130	59.55	655
32.50	130	45.17	226
32.50	130	53.50	267
49.17	393	65.89	923
57.50	460	77.39	2,786
32.50	130	39.02	234
32.50	130	42.33	212
32.50	130	38.27	268

Date – this just tells the user what the date is that is being referenced

DDays – this is the Axis called DDays it shows us the date in a more detailed version

Rooms available – this is the quantity of rooms that the hotel has on any day

BoB occupancy – this is the current occupancy (how many rooms have been booked)

PickUp individuals – how many more rooms are predicted to be booked

Total unsold – is Rooms available – (Bob occupancy + PickUp individuals) this shows how many rooms are predicted to be empty

Total rooms sold – this is Bob occupancy + PickUp individuals and it shows the total number of room to be predicted to be sold including BoB and PickUp

Total occupancy – same as Total rooms sold just expressed as a percentage

BoB Average Daily Rate – is the average price for the rooms that have been sold

Pickup Average daily rate - is the average price for the rooms that are predicted to be sold

Pickup room revenue – this is calculated by Pickup Average daily rate * PickUp individuals

Total Average Daily Rate – is the averages from both PickUp and BoB in one

Total room revenues – is that multiplied by Total rooms sold

Why this table is important:

The reason why this table is important is that it is where my program will get all of its inputs and return value to the table. The columns headers highlighted in gold are where I will output my result in the rows below them.

Proposed Designs:

My project will be a python program run once a day that reads a database, then does maths on the dataset provided. The final output being 1 of 3 things.

1. Increase the price:
    a. This would be done if you are selling rooms fast for example 28-14 days before the day your 50% booked you would increase the price to reduce bookings and also to improve ADR.
2. Keep it the same your on track:
    a. Looking at what happened the same day last year which gave us a good result. If BoB seems to be following the same track then there is no need to change the pricing.
3. Lower the price:
    a. You are 10 days out and only at 25% capacity, you would lower the price and do an advertising campaign to encourage room bookings. Your ADR may be low but you then have more people in the hotel for breakfast lunch and dinner which increases revenue. Rooms may have been sold at the lowest price, but you have ensured that there are people in your hotel who are now spending money at the bar and for dinner etc.

After that, it will return a value which will indicate one of these 3 things within a range showing how much to drop the room by or increase the price by.

Key Features:

There will be a GUI after the program runs, it will produce a pop up stating recommendations for each respective day, giving a value to be returned. The user does not need to look at a database to find her required value. This is useful because it makes the program more user-friendly which is key to any good program.

Date	Returned Value A	Recommendation	Return Value B	Recommendation
3/8/17	0.4	Increase the value within an agreed pricing band.	60	Increase your price to this.

4/8/17	0.01	Increase the value within an agreed pricing band by a small amount, you might not even change the price.	50	Increase your price to this.
5/8/17	-0.5	Decrease the value within an agreed pricing band by an agreed amount.	40	Decrease your price to this.
6/8/17	0.9	Increase the value within an agreed pricing band within the highest bracket.	85	Increase your price to this.
7/8/17	-0.9	Decrease the value within an agreed pricing band by an agreed amount. You might go to the lowest band and then also employ an advertising campaign.	35	Decrease your price to this.
Etc.	Etc.	Etc.	Etc.	Etc.

The green cells would not be shown in the final programs output sheet. They merely represent what the value would mean in terms of the action to take for the corresponding value returned.

The yellow column holds the date referring to the date that is being related to

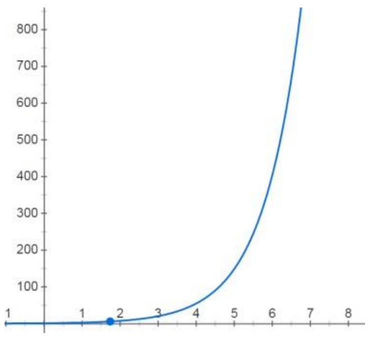
The blue columns hold the value that showing what the program recommends

1 = price to max

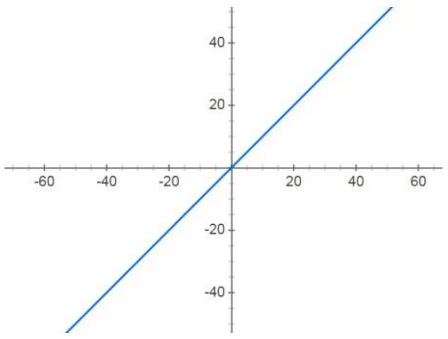
0 = keep the price the same

-1 = lower the price as much as possible

These values would sit on this range from -1 to 1 and would be linear. Thus 0.2 is as far away from 0 as 0.4 as is from 0.2. This is in comparison to 0.2 being as far away from 0 as 0.8 being as far away from 0.2 and thus 1 is as further from 0 as 0.2 and 0.8 added together. This would be an exponential representation of the value. Although this method of using an exponential method of representing my output may be more accurate, as a value it can be misleading.



This is an exponential graph; the value of 2 is barely above 0 whereas if you look 8 is off the chart with a value near 2,000 thus. This would ideally be used in conjunction with a graph to accurately map the value and display the information efficiently.



This a linear graph; the value of 1 is 1 whereas the value of 2 corresponds to 2 etc. this is more user-friendly as you don't need a graph to show what the outputted value represents

Functional requirements:

Python 3.3

I will be using python 3.3. That is what the school uses and thus I have experience of this. I was considering using python 3.6 and realised that there is very little difference with inbuilt functions. I will stick to python 3.3 as I already know how it works and it is already on the schools' computers.

https://www.python.org/download/releases/3.3.0/

TKinter

TKinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts. TKinter relies on the Tk library, the GUI library used by Tcl/Tk and Perl, which is in turn implemented in C. Therefore, TKinter can be said to be implemented using multiple layers. I will be using TKinter as it's built into python thus I don't need to download and attach a new library for its use. It is the defacto for python and it's more commonly used and thus more support/ tutorials are available.

Jedox

Jedox is a cube based business intelligence software getting a data feed in from Excel. Ultimately my program is not going to interact with the data through Jedox but will instead act upon the excel database that then Jedox will act upon that.

Jedox is how the hotel stores its data in multidimensional arrays (9D) and I will access the database through Excel, where each dimension is a column header that represents the name of that dimension referencing the location holding the data.

https://www.jedox.com/en/

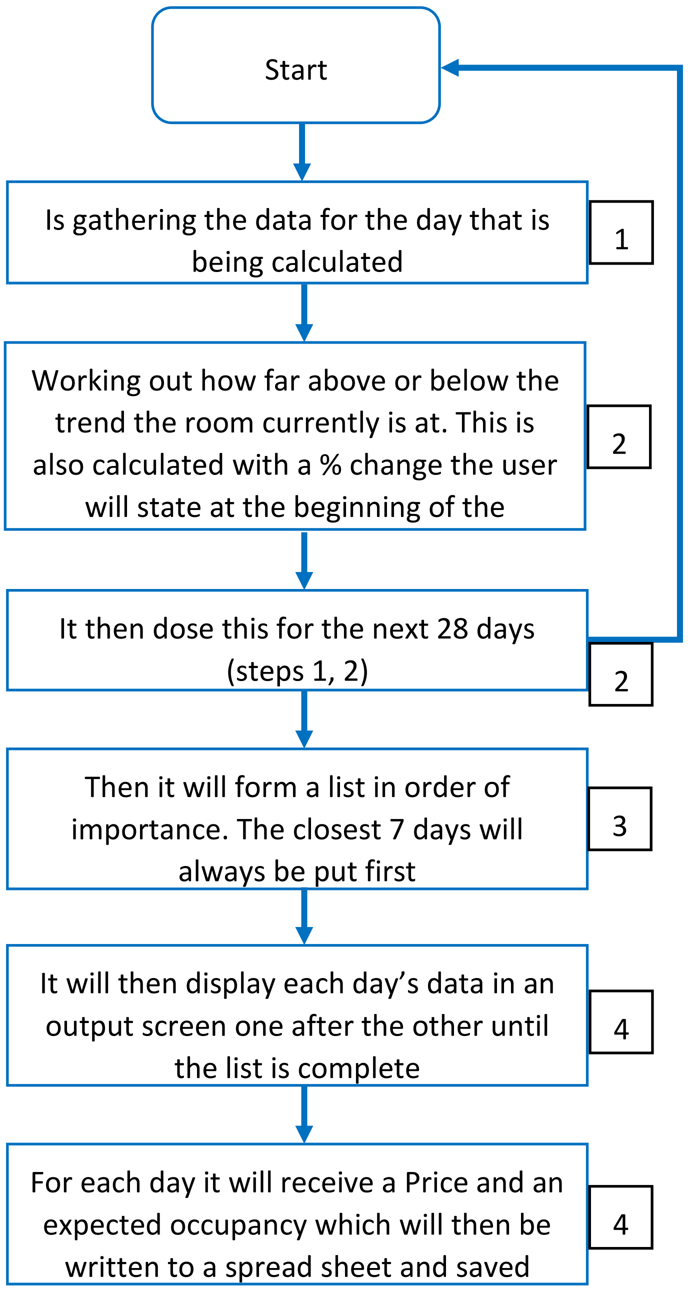
Success Criteria:

- My program will analyse large databases from hotels business on their books (BoB). It will use complex algorithms to spit out certain values that can be used by the user to set the price for the hotels rooms. They could also be used to provide business intelligence in decision making.
- My program will have a minimalistic user interface this is so that it makes using the program easier and thus faster, as my program is designed to speed up the time it takes for the finance manager to do this.
- The program must output its pricing in a desirable way such that it can be used by price parity programs that can update the hotels prices across many sites; travel agents, websites, local systems. If it wasn't outputted in a useable format then the revenue manager would have to input all of the prices separately which takes time

Design

Deconstruction of the program

This is a flowchart of my programs basic running structure of the program:

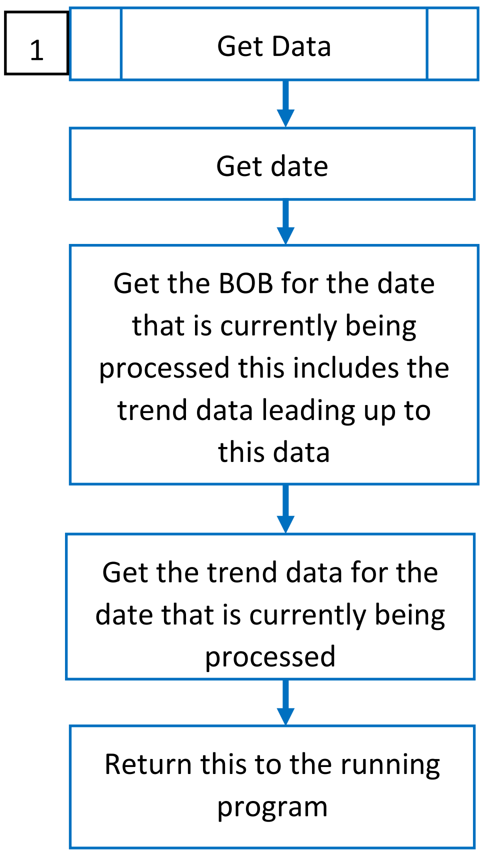

Start
Is gathering the data for the day that is being calculated
1
Working out how far above or below the trend the room currently is at. This is also calculated with a % change the user will state at the beginning of the
2
It then dose this for the next 28 days (steps 1, 2)
2
Then it will form a list in order of importance. The closest 7 days will always be put first
3
It will then display each day's data in an output screen one after the other until the list is complete
4
For each day it will receive a Price and an expected occupancy which will then be written to a spread sheet and saved
4

This Flow chart shows the initial breakdown into a set of clear functions that run sequentially. This idea of this is to help me break down the program into its functions and then further still into the individual procedures which will then be made into pseudocode. This will be useful as when I come to design my program I have its structure at multiple different levels allowing me to plan effectively.

Through doing this I have discovered that I need to finalise some of the input data and how it is formatted. This is so that when I come to write the program I'm still not figuring things out and having questions about how I'm going to solve certain things. This is important because it enables me to ensure that the development stage goes smoothly with minimal problems, which would hold up the development of the program.

The following pages are of the functions with the sub functions listed as flow charts. And for each sub function I will do a more detailed and in depth analysis of it.

Breaking the problem down into its main functions


1
Get Data
Get date
Get the BOB for the date that is currently being processed this includes the trend data leading up to this data
Get the trend data for the date that is currently being processed
Return this to the running program

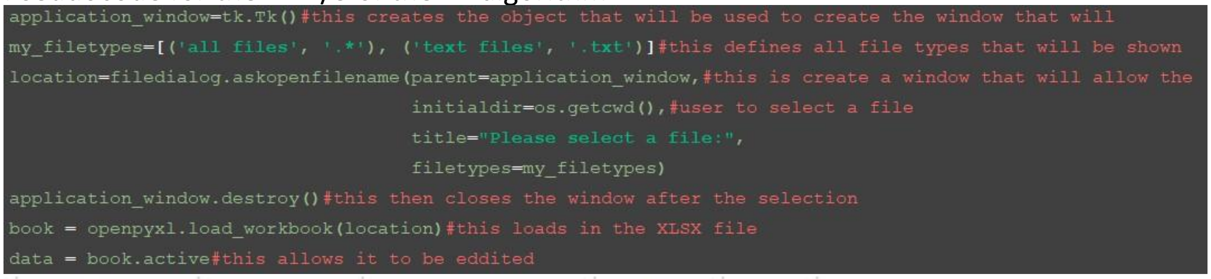
This function gets the necessary data for each of the days that it is calculated for. It after this it will then assemble an array within the program. This will be done for convenience purposes and speed. Instead of having to open up a file every time I need a value; thus making the programs running far faster.

"Get date" – this sub-function will out put the date. This reason why this is done is so that I know what day is being processed and at what time so that when I go to search for the data I have a clear label to use that I can use to ensure that I am getting the right day.

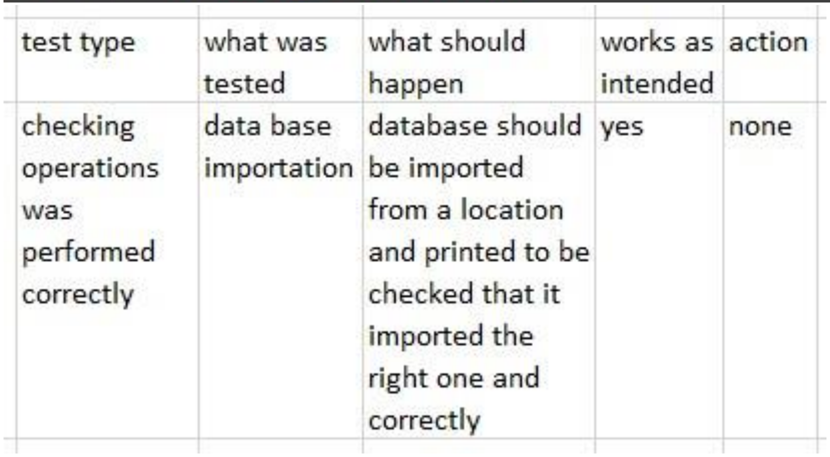
"Get the BOB…" – this sub-function will collect the business on books for the days that correspond to the date that they are holding the data for the run up to it. The reason why this is done is so that I can compare it with the trends data that I will be collecting. This data will be stored in an array during the running of the program.

"Get the trend data…" – this sub-function will get the trend data to do comparisons and math on it to do with the data that was just collected (this be later on). Once the data has been collected it will be put into an array and stored inside of the running program. The reason why I will do this because it improves the running of the program and its speed. This is because it won't have to be constantly opening a file to read values they will be there already within the program thus far easier to access.

Pseudocode for the 1st 1/3 of the 1st algorithm

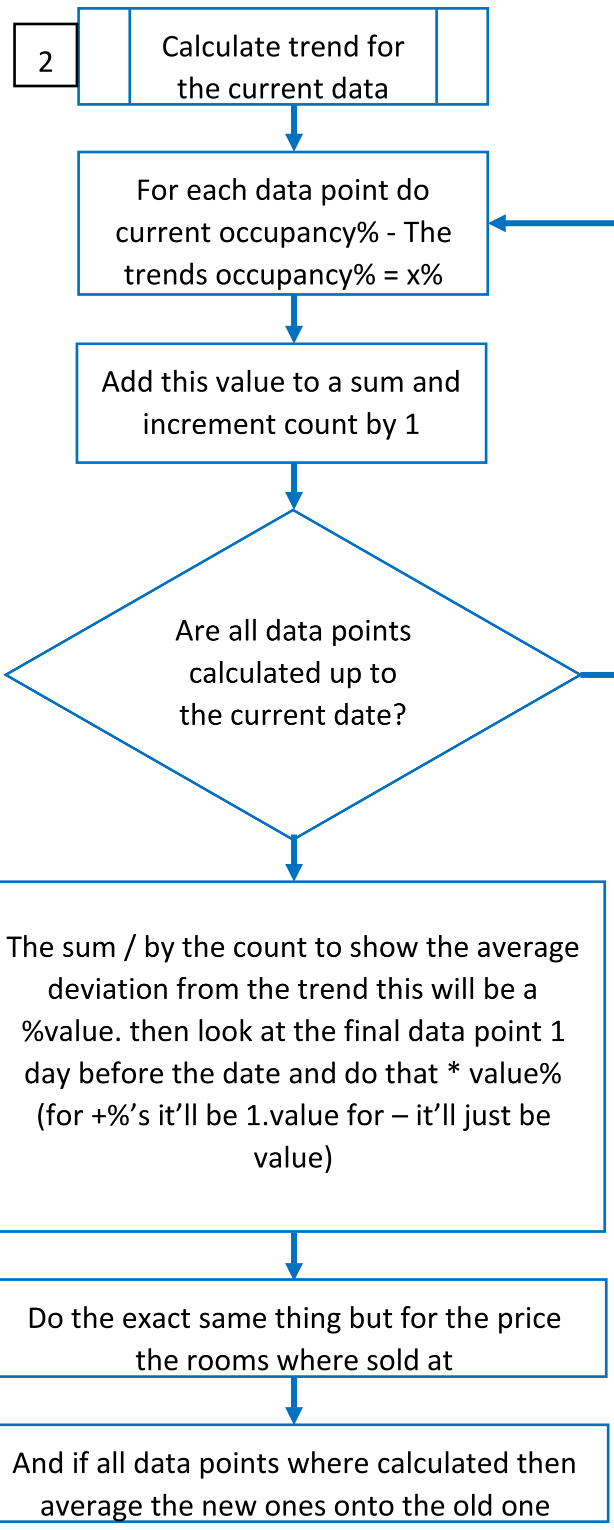


```
application_window=tk.Tk()#this creates the object that will be used to create the window that will
my_filetypes=[('all files', '.*'), ('text files', '.txt')]#this defines all file types that will be shown
location=filedialog.askopenfilename(parent=application_window,#this is create a window that will allow the
                                    initialdir=os.getcwd(),#user to select a file
                                    title="Please select a file:",
                                    filetypes=my_filetypes)
application_window.destroy()#this then closes the window after the selection
book = openpyxl.load_workbook(location)#this loads in the XLSX file
data = book.active#this allows it to be eddited
```



test type	what was tested	what should happen	works as intended	action
checking operations was performed correctly	data base importation	database should be imported from a location and printed to be checked that it imported the right one and correctly	yes	none

Test Passed


2
Calculate trend for the current data
For each data point do current occupancy% - The trends occupancy% = x%
Add this value to a sum and increment count by 1
Are all data points calculated up to the current date?
The sum / by the count to show the average deviation from the trend this will be a %value. then look at the final data point 1 day before the date and do that * value% (for +%'s it'll be 1.value for – it'll just be value)
Do the exact same thing but for the price the rooms where sold at
And if all data points where calculated then average the new ones onto the old one

The 1st will calculate the difference between the BOB data point and its trends data. If BOB is below it will give a -% but if above it will give a +%.

The 2nd will increment a value by one as it moves through all the data points leading up to a day. This also has a value that is totalled up throughout the programs running (sum).

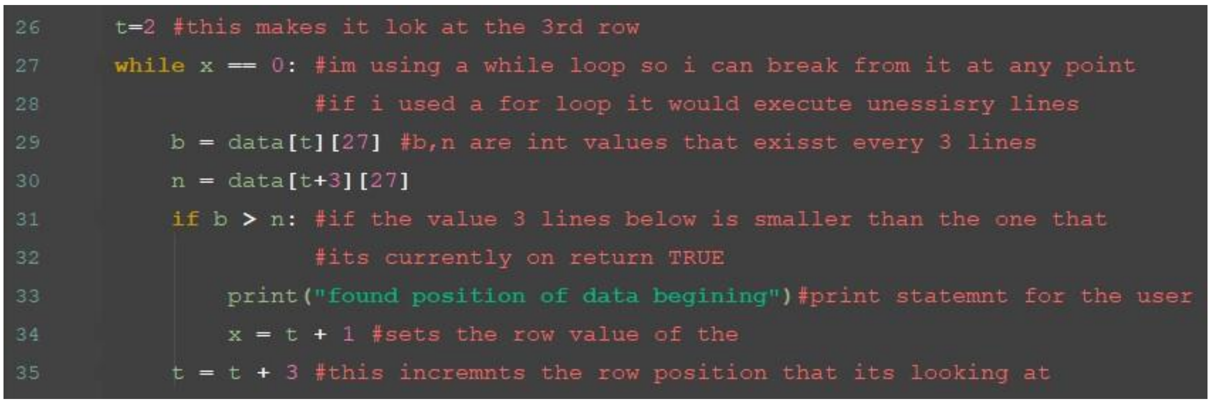
3rd is a check to see if all have been calculated and if not go back to top (1st sub-function).

4th dose sum/count = X this gives an average across the board. If the value is +tive you do 1.X * final data point for trend (even if you are not there for bob 28th data point) if the value is –tive then you do X * the final data point. The output will be a %.

5th the same will be done for the same days but with price, and will need a maximum room price inputted at the programs start.

6th once all the data points have been calculated then average the old one onto the new ones (there will be a count to show how many times it has been over it.

Pseudocode for 2nd 1/3 of 1st algorithm



```
t=2 #this makes it lok at the 3rd row
while x == 0: #im using a while loop so i can break from it at any point
              #if i used a for loop it would execute unessisry lines
    b = data[t][27] #b,n are int values that exisst every 3 lines
    n = data[t+3][27]
    if b > n: #if the value 3 lines below is smaller than the one that
              #its currently on return TRUE
        print("found position of data begining")#print statemnt for the user
        x = t + 1 #sets the row value of the
    t = t + 3 #this incremnts the row position that its looking at
```

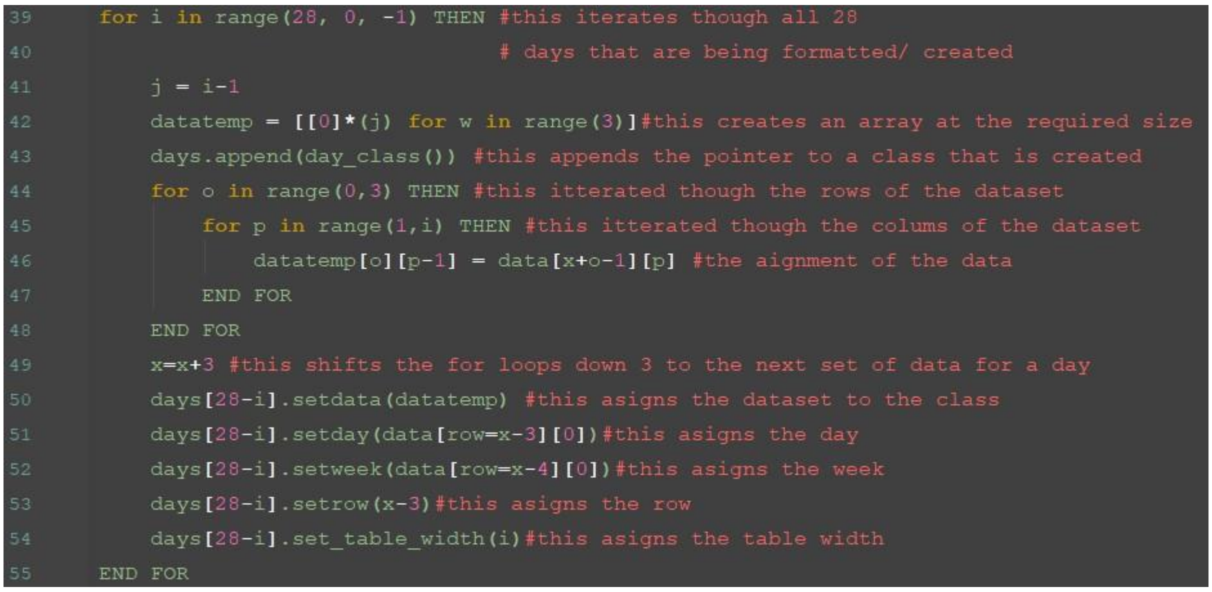
t	b	n	is b>n	is x = 0
2	1	1	N	N
5	1	1	N	N
8	1	1	N	N
11	1	1	N	N
14	1	1	N	N
17	1	1	N	N
20	1	1	N	N
23	1	1	N	N
26	1	1	N	N
29	1	1	N	N
32	1	1	N	N
35	1	1	N	N
38	1	1	N	N
41	1	1	N	N
44	1	1	N	N
47	1	1	N	N
50	1	1	N	N
53	1	0	Y	Y

11.4
11.39
1
19.92
19.93
1
12.8
12.81
1
12.79
12.81
1
12.83
12.81
1
18.5
18.51
1
7.13

7.12
1
21.34
21.36
1
32.74
32.75
1
38.46
38.44
1
34.16
34.17
1
9.98
9.97
1
21.35
21.36

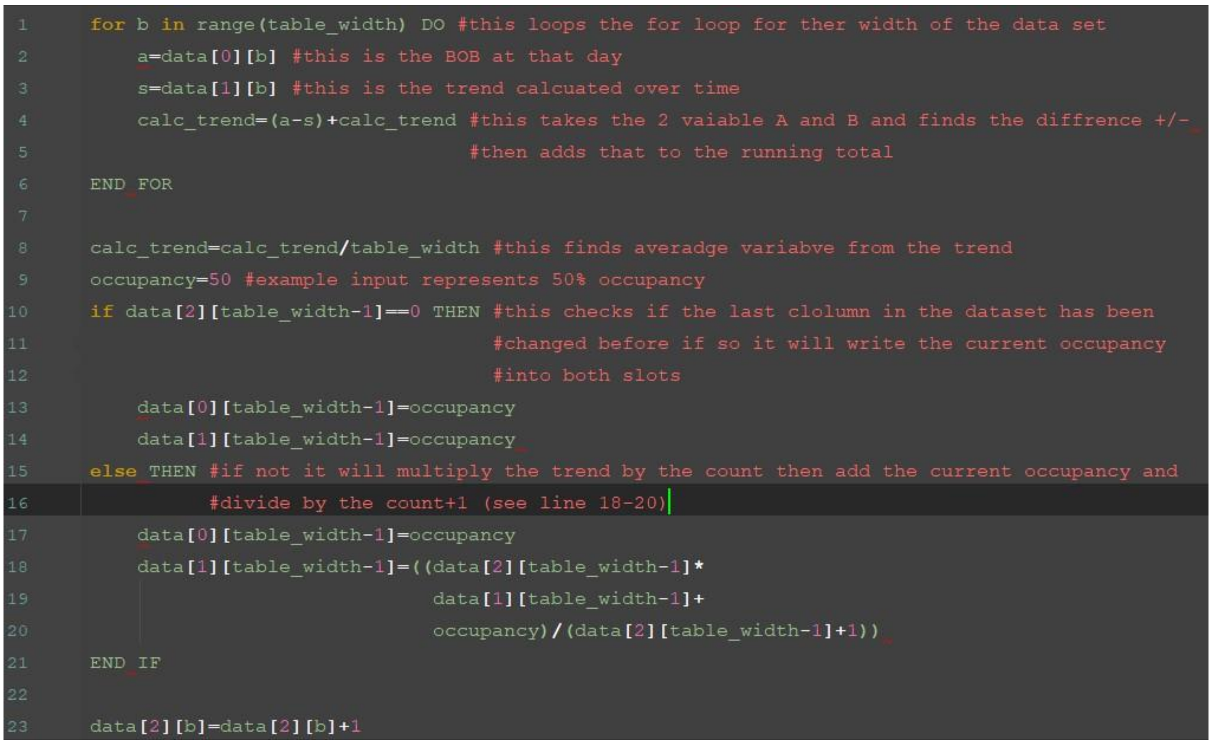
1
11.4
11.39
1
41.27
41.29
1
62.66
62.64
1
51.27
51.25
1
28.49
28.47
0

Pseudocode for 3rd 1/3 of 2nd algorithm

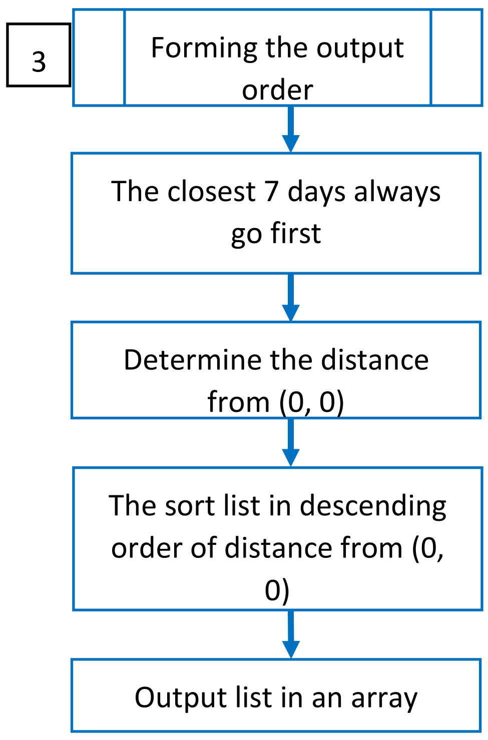


```
for i in range(28, 0, -1) THEN #this iterates though all 28
                                        # days that are being formatted/ created
    j = i-1
    datatemp = [[0]*(j) for w in range(3)]#this creates an array at the required size
    days.append(day_class()) #this appends the pointer to a class that is created
    for o in range(0,3) THEN #this itterated though the rows of the dataset
        for p in range(1,i) THEN #this itterated though the colums of the dataset
            datatemp[o][p-1] = data[x+o-1][p] #the aignment of the data
        END FOR
    END FOR
    x=x+3 #this shifts the for loops down 3 to the next set of data for a day
    days[28-i].setdata(datatemp) #this asigns the dataset to the class
    days[28-i].setday(data[row=x-3][0])#this asigns the day
    days[28-i].setweek(data[row=x-4][0])#this asigns the week
    days[28-i].setrow(x-3)#this asigns the row
    days[28-i].set_table_width(i)#this asigns the table width
END FOR
```

Pseudocode for the 2Nd algorithm



```
1   for b in range(table_width) DO #this loops the for loop for ther width of the data set
2       a=data[0][b] #this is the BOB at that day
3       s=data[1][b] #this is the trend calcuated over time
4       calc_trend=(a-s)+calc_trend #this takes the 2 vaiable A and B and finds the diffrence +/-
5                                      #then adds that to the running total
6   END FOR
7
8   calc_trend=calc_trend/table_width #this finds averadge variabve from the trend
9   occupancy=50 #example input represents 50% occupancy
10  if data[2][table_width-1]==0 THEN #this checks if the last clolumn in the dataset has been
11                                       #changed before if so it will write the current occupancy
12                                       #into both slots
13      data[0][table_width-1]=occupancy
14      data[1][table_width-1]=occupancy
15  else THEN #if not it will multiply the trend by the count then add the current occupancy and
16           #divide by the count+1 (see line 18-20)
17      data[0][table_width-1]=occupancy
18      data[1][table_width-1]=((data[2][table_width-1]*
19                              data[1][table_width-1]+
20                              occupancy)/(data[2][table_width-1]+1))
21  END IF
22
23  data[2][b]=data[2][b]+1
```

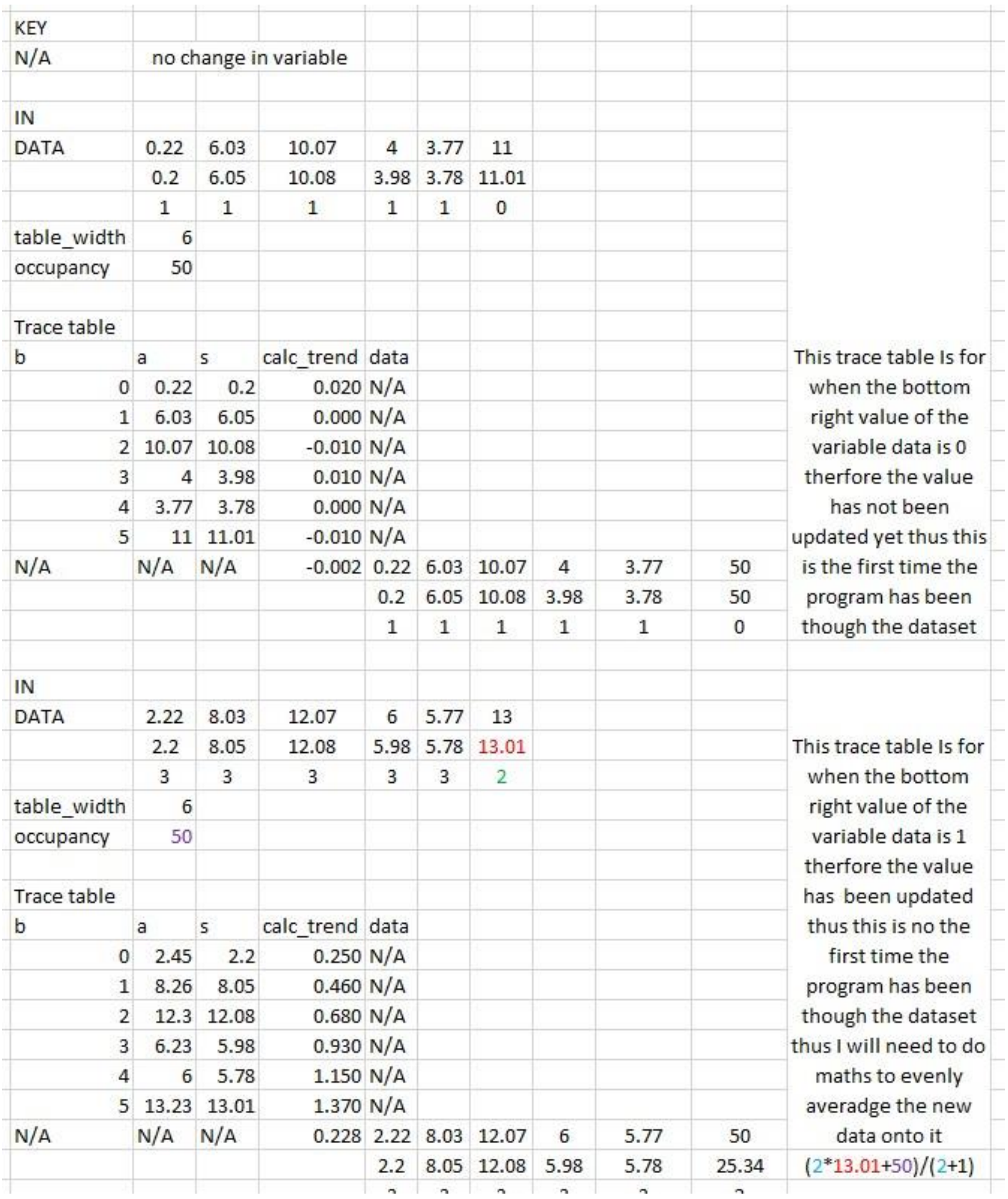

3
Forming the output order
The closest 7 days always go first
Determine the distance from (0, 0)
The sort list in descending order of distance from (0, 0)
Output list in an array

1st will pick the first 7 days that are closest (soonest to latest) and put their date into a list.

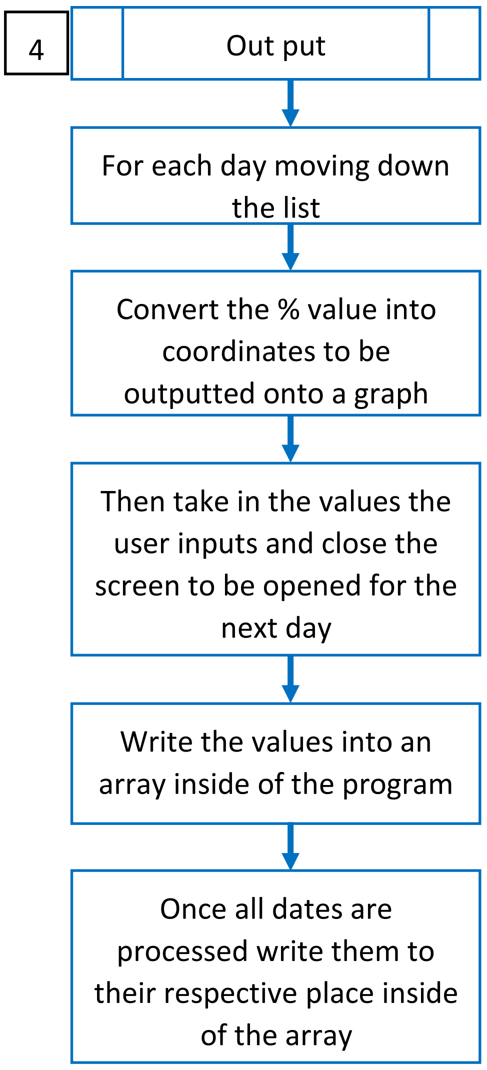
2nd will make all values +tive during the calculation but not change their actual values. $(X^2 + Y^2)^{1/2}$ that is $A^2 + B^2 = C^2$ but then we square root to find the distance.

3rd / 4th Use a liner search algorithm to output their dates in descending order.

I have decided not to do this step as the program is intended to be easy and quick to use. Thus making the user jump forwards and backwards in their books is not going to be convenient. So I want to make it consecutive thus they just read down the list and input values.



KEY										
N/A	no change in variable									
IN										
DATA	0.22	6.03	10.07	4	3.77	11				
	0.2	6.05	10.08	3.98	3.78	11.01				
	1	1	1	1	1	0				
table_width	6									
occupancy	50									
Trace table										
b	a	s	calc_trend	data						This trace table Is for
0	0.22	0.2	0.020	N/A						when the bottom
1	6.03	6.05	0.000	N/A						right value of the
2	10.07	10.08	-0.010	N/A						variable data is 0
3	4	3.98	0.010	N/A						therfore the value
4	3.77	3.78	0.000	N/A						has not been
5	11	11.01	-0.010	N/A						updated yet thus this
N/A	N/A	N/A	-0.002	0.22	6.03	10.07	4	3.77	50	is the first time the
				0.2	6.05	10.08	3.98	3.78	50	program has been
				1	1	1	1	1	0	though the dataset
IN										
DATA	2.22	8.03	12.07	6	5.77	13				
	2.2	8.05	12.08	5.98	5.78	13.01				This trace table Is for
	3	3	3	3	3	2				when the bottom
table_width	6									right value of the
occupancy	50									variable data is 1
										therfore the value
Trace table										has been updated
b	a	s	calc_trend	data						thus this is no the
0	2.45	2.2	0.250	N/A						first time the
1	8.26	8.05	0.460	N/A						program has been
2	12.3	12.08	0.680	N/A						though the dataset
3	6.23	5.98	0.930	N/A						thus I will need to do
4	6	5.78	1.150	N/A						maths to evenly
5	13.23	13.01	1.370	N/A						averadge the new
N/A	N/A	N/A	0.228	2.22	8.03	12.07	6	5.77	50	data onto it
				2.2	8.05	12.08	5.98	5.78	25.34	(2*13.01+50)/(2+1)
				[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

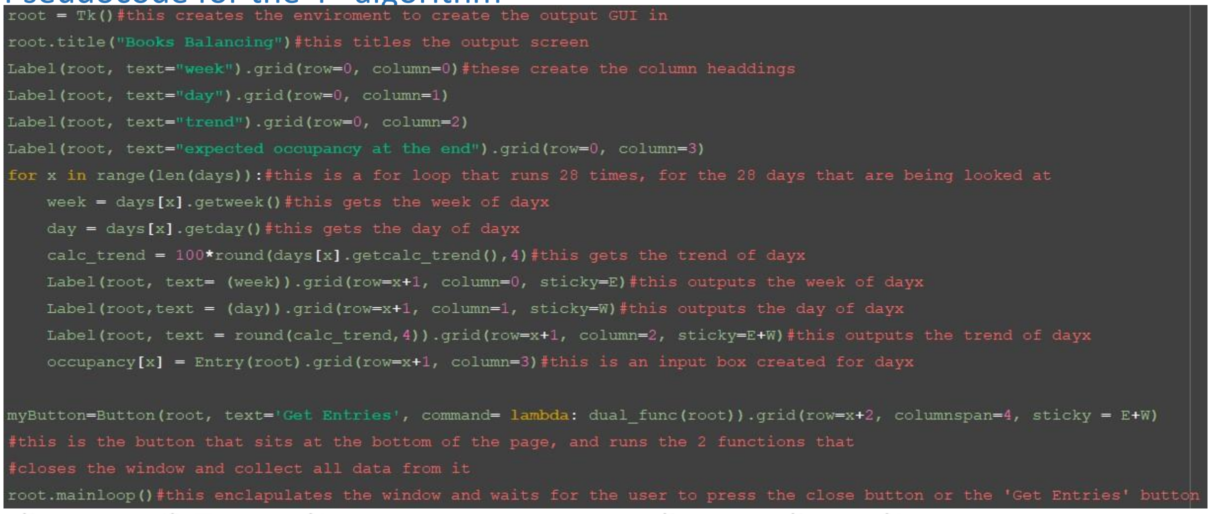

4
Out put
For each day moving down the list
Convert the % value into coordinates to be outputted onto a graph
Then take in the values the user inputs and close the screen to be opened for the next day
Write the values into an array inside of the program
Once all dates are processed write them to their respective place inside of the array

1st / 2nd this will be worked out once I have finalised how I'm going to do the output sheet but I will probably make it a 1000X1000 pixel image thus calculating where it goes on (little red dot). This will be a small image that will be overlaid on top.

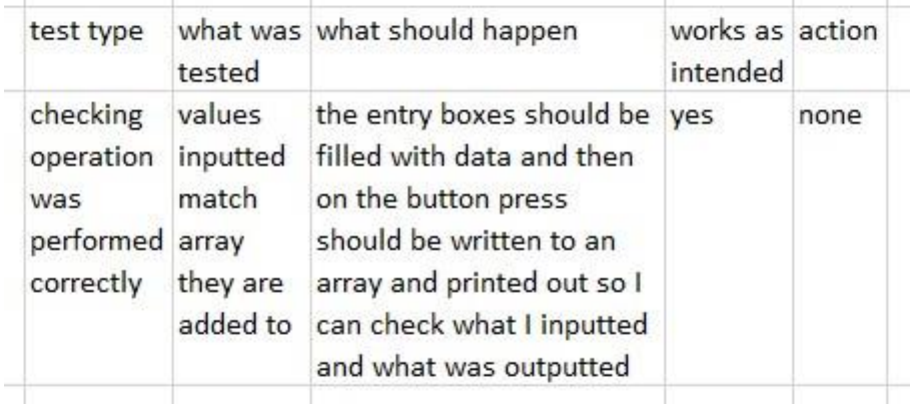
3rd / 4th this will take the 2 values the user inputs and write them back into the specific place in the array. It will then close the screen down and then open up the next date in the list.

5th once all of the dates have been processed then open up the output table and white each dates data into its relevant place inside of the array.

Pseudocode for the 4th algorithm



```
root = Tk()#this creates the enviroment to create the output GUI in
root.title("Books Balancing")#this titles the output screen
Label(root, text="week").grid(row=0, column=0)#these create the column headdings
Label(root, text="day").grid(row=0, column=1)
Label(root, text="trend").grid(row=0, column=2)
Label(root, text="expected occupancy at the end").grid(row=0, column=3)
for x in range(len(days)):#this is a for loop that runs 28 times, for the 28 days that are being looked at
    week = days[x].getweek()#this gets the week of dayx
    day = days[x].getday()#this gets the day of dayx
    calc_trend = 100*round(days[x].getcalc_trend(),4)#this gets the trend of dayx
    Label(root, text= (week)).grid(row=x+1, column=0, sticky=E)#this outputs the week of dayx
    Label(root,text = (day)).grid(row=x+1, column=1, sticky=W)#this outputs the day of dayx
    Label(root, text = round(calc_trend,4)).grid(row=x+1, column=2, sticky=E+W)#this outputs the trend of dayx
    occupancy[x] = Entry(root).grid(row=x+1, column=3)#this is an input box created for dayx

myButton=Button(root, text='Get Entries', command= lambda: dual_func(root)).grid(row=x+2, columnspan=4, sticky = E+W)
#this is the button that sits at the bottom of the page, and runs the 2 functions that
#closes the window and collect all data from it
root.mainloop()#this enclapulates the window and waits for the user to press the close button or the 'Get Entries' button
```



test type	what was tested	what should happen	works as intended	action
checking operation was performed correctly	values inputted match array they are added to	the entry boxes should be filled with data and then on the button press should be written to an array and printed out so I can check what I inputted and what was outputted	yes	none

Test Passed

Other Features

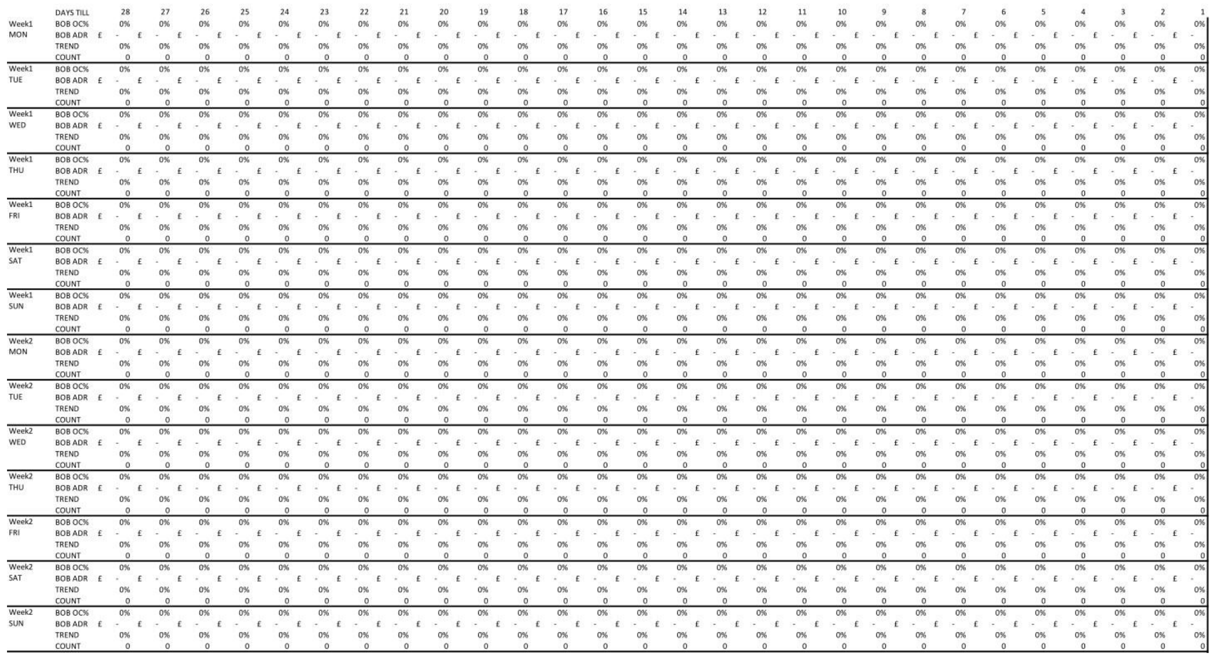
I and going to design my program to make it as simple as possible then add other features and elegance during its running

If I have time towards the end of my development, then I intend to implement these features:

When the program first runs it will ask the user how often it wants to input its data. This will be at the end of every week or every day. Then I will delete the other table and record the preference in the database. The idea of this is to make the program easier to use and seem more polished during use.

Input Data Format/ Test Data

The Database that I'm going to use will also double up as how I record the trends data. When the program first runs it will ask the user how often it wants to input its data. This will be at the end of every week or every day.



	DAYS TILL	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
MON	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
TUE	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
WED	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
THU	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
FRI	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SAT	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week1	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SUN	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
MON	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
TUE	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
WED	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
THU	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
FRI	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SAT	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Week2	BOB OC%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SUN	BOB ADR	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -	£ -
	TREND	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	COUNT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The first Row holds the Count for Days till, this is how many days until the current date that this data is being used for. So let's say that it's the 2nd and you're looking at the 13th your data goes in slot 11 as you are 11 days away. It runs from 28 days away to the day before, as my program is intended to help with getting last minute bookings, not large groups for weddings etc. There is no need to run the program for the day that is currently happening thus the numbers don't count down to 0. The reason why I have this is because once at the output stage I can say in 13 day(s) this is the trend etc.

For each set of 4 rows I have the week number; week1, week2, …, week53. I've gone up to 53 as that's how many Weeks there are in a year despite the 53rd not being filled.

Under that I have MON, TUE, … this is so that I can keep track of what day it is. The reason why I am not using dates is because 1/1/2018 is a Monday but that same date the next year is a Tuesday. This matters because the day heavily impacts the number of bookings received and thus I need to make sure that the days sync up rather than the dates.

Each of the 4 rows per day holds; BOB%, ADR, Trend%, Count

BOB% is the Business On Books as a % of total occupancy

ADR is the average daily rate that we are selling rooms at for that day

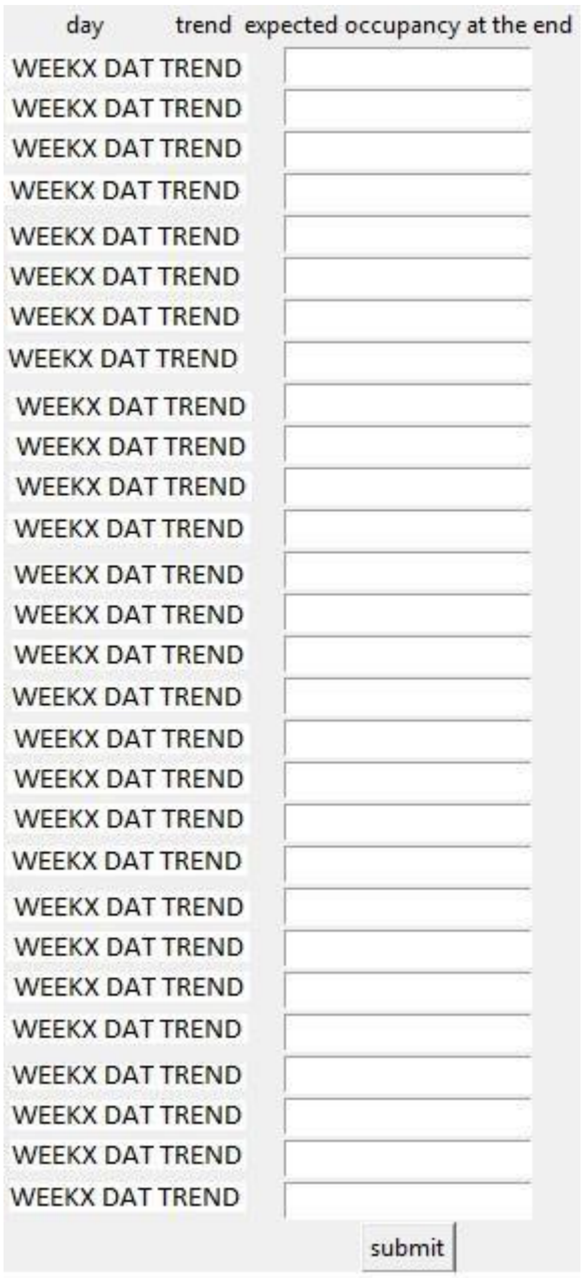
Trend % is how I keep track of trend data over the years

Count is how many times it has been averaged over thus the program can ensure that the data from 5 years ago is still as relevant as it is right now. After 5 years it may have an average of 74.3% then my program runs with its new piece of data 21% it dose 74.3% + 21% / 2 what this means is that that 21% is equally important as the 5 years of data before it. Thus I would use count to ensure that all the data is important:

(74.3*count + 21) / Count+1

This ensures that the newest piece of data is no more important than the data that currently exists.

Out Put Screen



day trend expected occupancy at the end

WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND
WEEKX DAT TREND

submit

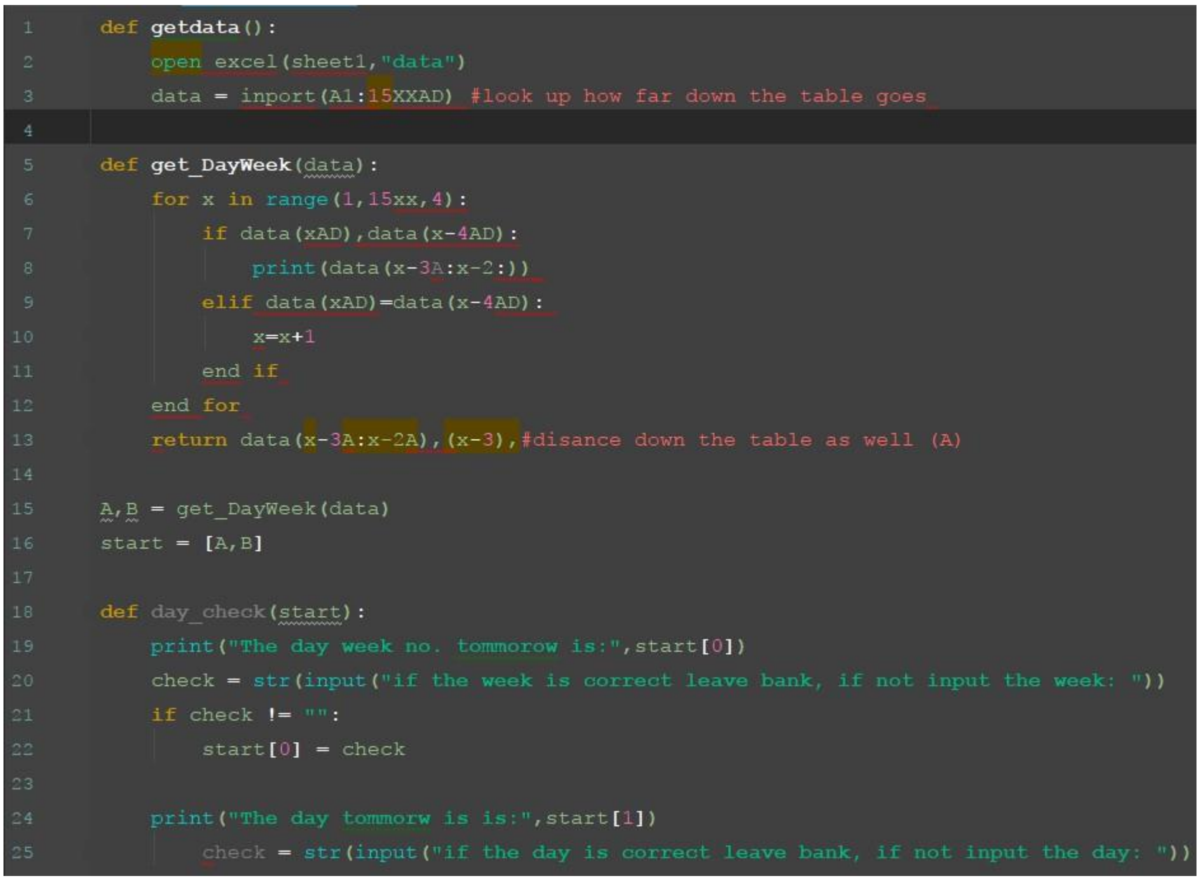
The output screen will not include a graph as my program needs to be quick an easy to use and thus a big graph will take up a lot of space and time to read it. So, to improve efficiency I will just list the final deviation from the trend data and the day and week of it. I will also exclude the idea of ordering it in order of how far from the trend it is as it will make the process longer where they have to look through their books for that day instead of looking at the next page or row down. It will speed up the process and they are going to go through all the days instead so ordering them has no impact.

The output screen will have 28 entry boxes and the data to the left of them. This data will be in-ported into an excel sheet separate from the rest of the data so that it can be looked at and processed separately from the trend data. This is so that it can be imported into a program that can update the prices automatically online.

DEV LOG

1.0

In this section I will be doing a write up of the pseudocode I don't expect it to work that's what the next section is for. I may also start working on a few of the functions that I didn't include in the design section. I won't be able to test them as some of the pseudocode algorithms won't work perfectly first time. Most of the errors and logic errors will come in the next section where I try to get it to complete its core function.

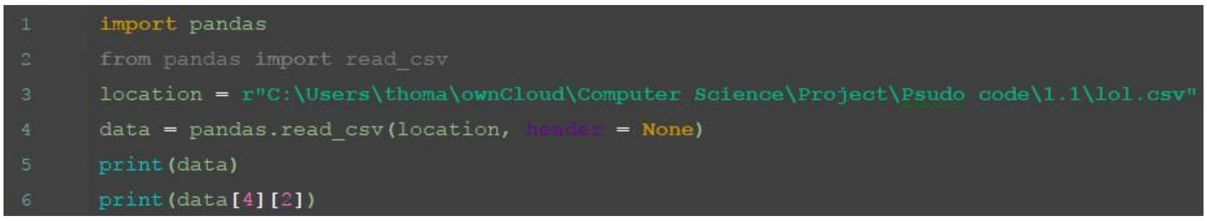


```
def getdata():
    open excel(sheet1,"data")
    data = inport(A1:15XXAD) #look up how far down the table goes

def get_DayWeek(data):
    for x in range(1,15xx,4):
        if data(xAD),data(x-4AD):
            print(data(x-3A:x-2:))
        elif data(xAD)=data(x-4AD):
            x=x+1
        end if
    end for
    return data(x-3A:x-2A),(x-3),#disance down the table as well (A)

A,B = get_DayWeek(data)
start = [A,B]

def day_check(start):
    print("The day week no. tomorow is:",start[0])
    check = str(input("if the week is correct leave bank, if not input the week: "))
    if check != "":
        start[0] = check

    print("The day tomorw is is:",start[1])
        check = str(input("if the day is correct leave bank, if not input the day: "))
```

This my initial write up from the pseudocode, it doesn’t work but the logic is there so once the data can be imported it can hopefully be easily adjusted to get it to work.

1.1

In the section I want to have working code that outputs the desired values it doesn’t have to use a GUI or save it to a file. If I can get it to just print the values on the screen then I would consider that a success.



```
1   import pandas
2   from pandas import read_csv
3   location = r"C:\Users\thoma\ownCloud\Computer Science\Project\Psudo code\1.1\lol.csv"
4   data = pandas.read_csv(location, header = None)
5   print(data)
6   print(data[4][2])
```

This is the 1st working version of my program it simply imports the data and a random point on the array to make sure it can be accessed. It works well and seems reliable, with the pandas module that can allow further data analysis and manipulation of my code.

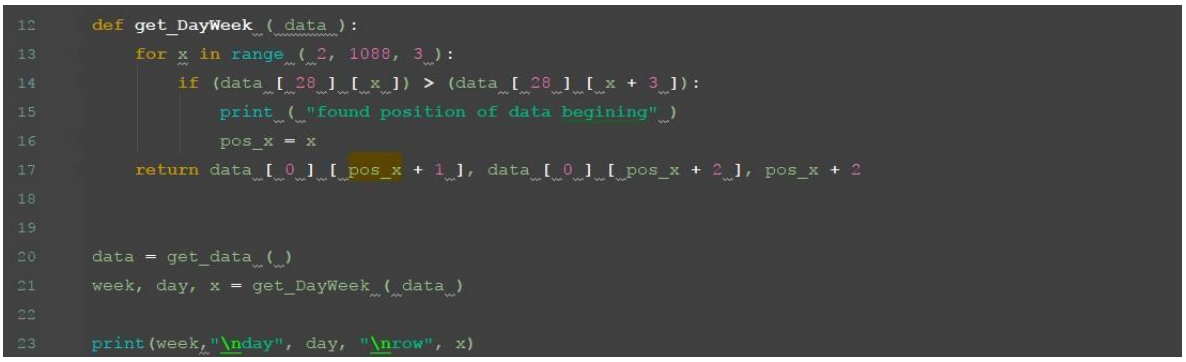
1 – This is the importing of the pandas module at allows me to access the excel sheet that it’s working off of

2 – This is for the importing of a sub module within pandas that can read CSV files

3 – This the assignment of the location of the file that I’m using

4 – This is where the spread sheet is imported into the database

5, 6 – This is where I out put the table and a single cell so I know that it works and that I can access it



```
12      def get_DayWeek ( data ):
13          for x in range ( 2, 1088, 3 ):
14              if (data [ 28 ] [ x ]) > (data [ 28 ] [ x + 3 ]):
15                  print ( "found position of data begining" )
16                  pos_x = x
17          return data [ 0 ] [ pos_x + 1 ], data [ 0 ] [ pos_x + 2 ], pos_x + 2
18
19
20      data = get_data ( )
21      week, day, x = get_DayWeek ( data )
22
23      print(week,"\nday", day, "\nrow", x)
```

This is the 1st version of my code with the formatting of its position etc. it finds where a certain column goes from a number to a smaller one. Then outputs the week and day number and row position of where the change occurs. It works well and is reliable so far.

12 – This is the initialisation of the function that will find the point in the data where the program will start

13 – This is the for loop that will go through the whole array it starts on line 3 because that is the first instance of the pointer and goes in steps of 3 as that's the gap between the pointers

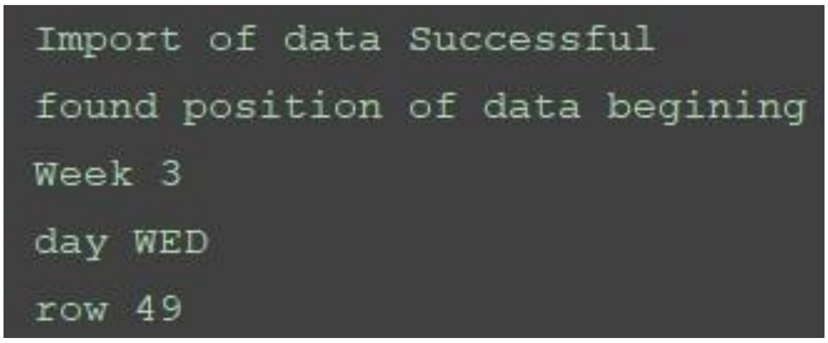
14 – This if statements compares 2 lines and if the 2nd is less than the 1st then it is true

15 – If the statement is true on line 14 then it will return a message that says that it has found the position of the data

16 – Then it will write the position of that data to a variable that will be used in the output of the data

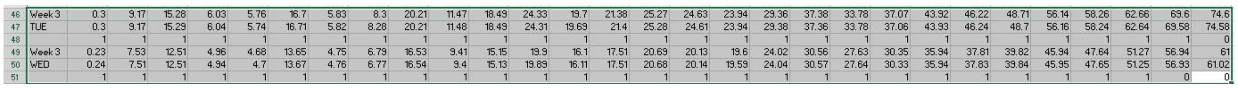
17 – This output statement will return the week the day and the row number of the day tomorrow

23 – This is a print statement that contains the week, day and row position to confirm it with me whilst in coding to make sure I have not messed up



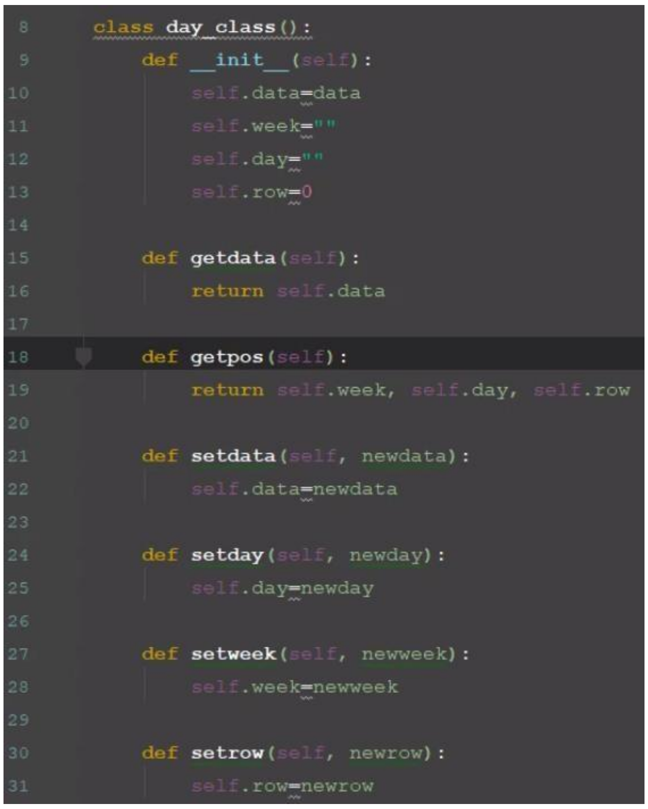
```
Import of data Successful
found position of data begining
Week 3
day WED
row 49
```

This is showing the outputted data the week number the day and the row position I know this works because it's where the 1's turn to 0's.



46	Week 3	0.3	9.17	15.28	6.03	5.76	16.7	5.83	8.3	20.21	11.47	18.49	24.33	19.7	21.38	25.27	24.63	23.94	29.36	37.38	33.78	37.07	43.92	46.22	48.71	56.14	58.26	62.66	69.6	74.6
47	TUE	0.3	9.17	15.29	6.04	5.74	16.71	5.82	8.28	20.21	11.48	18.49	24.31	19.69	21.4	25.28	24.61	23.94	29.38	37.36	33.78	37.06	43.93	46.24	48.7	56.16	58.24	62.64	69.58	74.58
48		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
49	Week 3	0.23	7.53	12.51	4.96	4.68	13.65	4.75	6.79	16.53	9.41	15.15	19.9	16.1	17.51	20.69	20.13	19.6	24.02	30.56	27.63	30.35	35.94	37.81	39.82	45.94	47.64	51.27	56.94	61
50	WED	0.24	7.51	12.51	4.94	4.7	13.67	4.76	6.77	16.54	9.4	15.13	19.89	16.11	17.51	20.68	20.14	19.59	24.04	30.57	27.64	30.33	35.94	37.83	39.84	45.95	47.65	51.25	56.93	61.02
51		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

As you can see in the 2nd from right hand column it has 1 recurring every 3 rows then at week 3 WED it changes to 0's, just like my program says it dose it also outputs row 49 same as the row that Week3 is on which is the 49th row. I've tested this multiple time and it works.



```
8   class day_class():
9       def __init__(self):
10          self.data=data
11          self.week=""
12          self.day=""
13          self.row=0
14
15      def getdata(self):
16          return self.data
17
18      def getpos(self):
19          return self.week, self.day, self.row
20
21      def setdata(self, newdata):
22          self.data=newdata
23
24      def setday(self, newday):
25          self.day=newday
26
27      def setweek(self, newweek):
28          self.week=newweek
29
30      def setrow(self, newrow):
31          self.row=newrow
```

This shows the class system that I will be using to hold all of the days that I will be performing the operation's on. Now that I have all of the data that I need I will be aiming to use classes as the way that I have implemented them is so that I can iterate through a list to call for each class thus I can just put it in a for loop and go through all of the them.

8 –> 13 – This is the initialisation of the class called day class it is made out of 4 pieces of data. Data, week, day, row:

Data is the day that it will be calculating for

Week is the week number of that day e.g. WEEK13

Day is the day that the data is being calculated for e.g. TUE

Row is the row position on the excel sheet that this day starts it's the same row as the week

15, 16 – This is the method that returns the data for the day to the running program

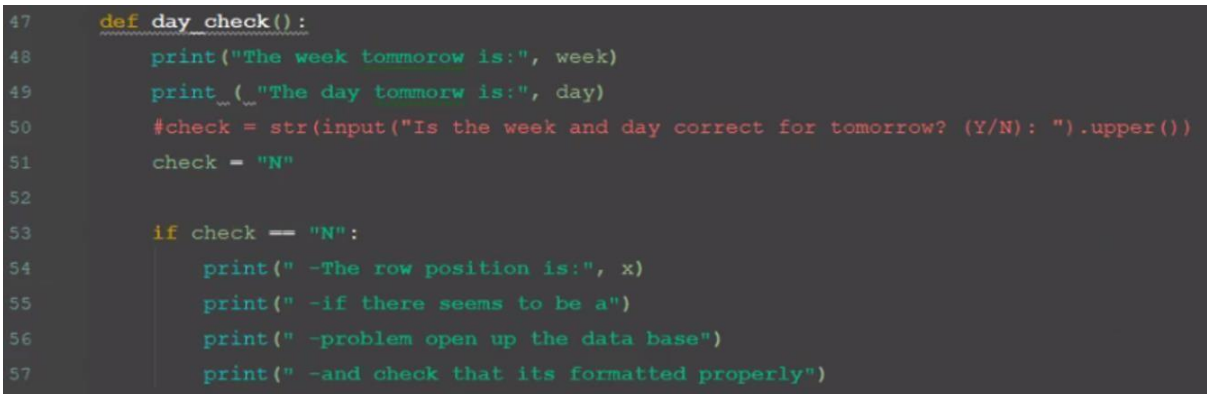
18, 19 – This is the method that returns the week, the day and the row of the day

21, 22 – This is the method that allows the program to write to it the data to be processed

24, 25 – This is the method that allows the program to write to it the day

27, 28 – This is the method that allows the program to write to it the week

30, 31 – This is the method that allows the program to write to it the row



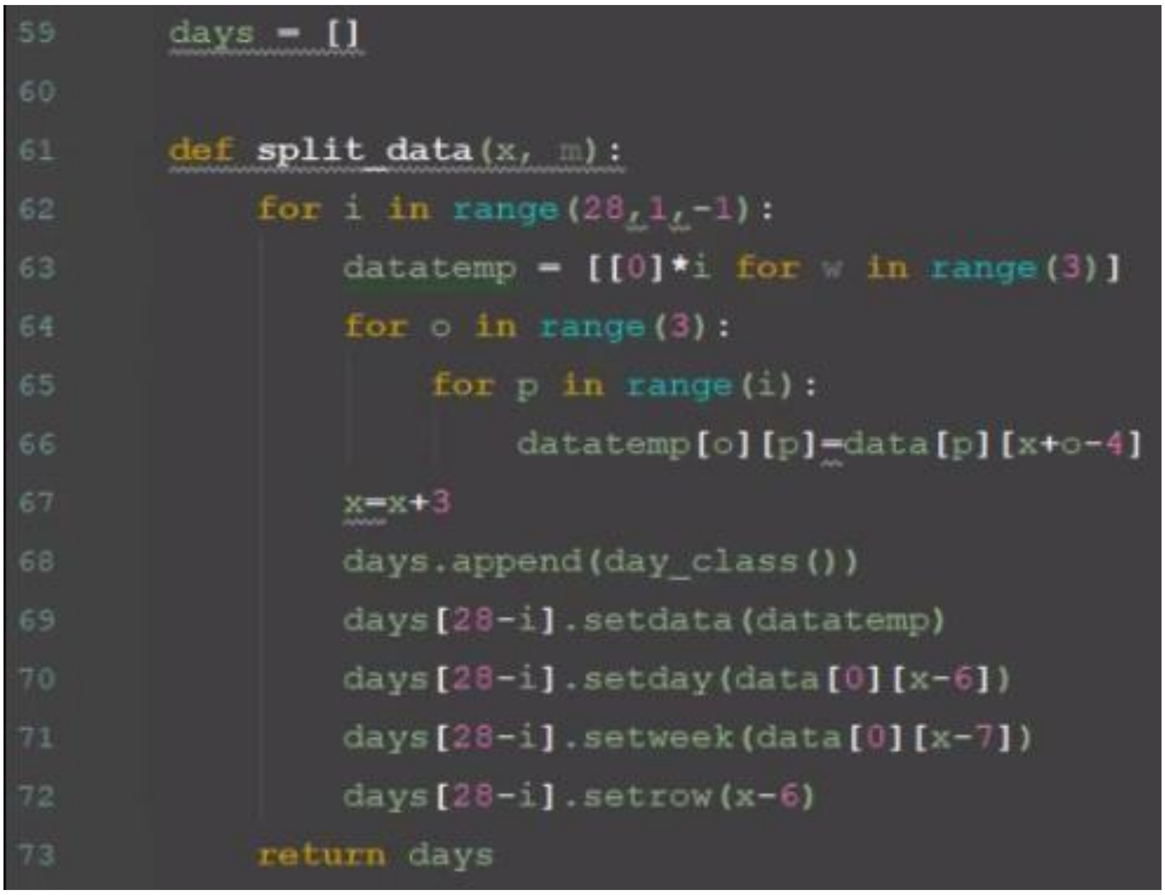
```
47      def day_check():
48          print("The week tommorow is:", week)
49          print ( "The day tommorw is:", day)
50          #check = str(input("Is the week and day correct for tomorrow? (Y/N): ").upper())
51          check = "N"
52
53          if check == "N":
54              print(" -The row position is:", x)
55              print(" -if there seems to be a")
56              print(" -problem open up the data base")
57              print(" -and check that its formatted properly")
```

This function makes sure that the user knows where the program is collecting its data from. If I have time I will add the ability to set the day, but at the moment it runs through the year and then stops and will then crash constantly.

48, 49 – This is how the program informs the user of where the program thinks tomorrow is then returns that to the user for them to check

50, 51 – This is a check statement, I have made it hashed through and set the value of check to "N" so that it automatically skips through that bit as I know it works and thus I don't need to press "N" every time I test my program.

53 –> 57 – This Is a print statement that's activated by the user keying in "N" it tells the user that they need to go and check the data that's being inputted as it contains a fault



```
days = []

def split_data(x, m):
    for i in range(28,1,-1):
        datatemp = [[0]*i for w in range(3)]
        for o in range(3):
            for p in range(i):
                datatemp[o][p]=data[p][x+o-4]
        x=x+3
        days.append(day_class())
        days[28-i].setdata(datatemp)
        days[28-i].setday(data[0][x-6])
        days[28-i].setweek(data[0][x-7])
        days[28-i].setrow(x-6)
    return days
```

This function splits the data into the days, these are the held in an array labelled days. I do this by creating a class at each point on the array and then I can reference them.

62 – This for loop defines how big the array can be, it does this by counting down from 28, I then creates a table with the size required and then lets the other for loops populate that.

63 – This is the creation of the array it uses the value "I" from line 62 to create the width and then dose that 3 times to create the depth.

64 – This is the initialisation of a for loop that counts the 3 rows

65 – This is the initialisation of a for loop that counts the "I" columns

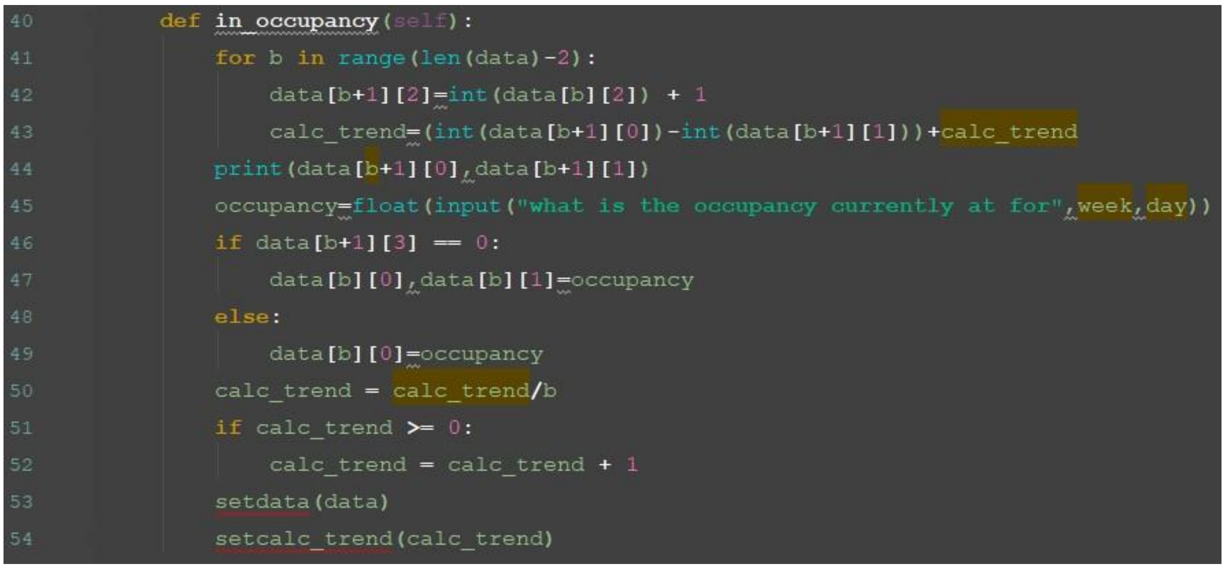
66 – This is the assignment of the data from the massive array to the days that have been split up.

67 – This is an increment to make sure that when it writes for the next day its 3 rows down so that it does not write the same day over and over again

68 – This is what happens once the day is completed a class is made and its address is appended to a list

69 –> 72 – This where it writes the data to the group using the functions that I made

73 – This is where the day's list is returned that holds all of the classes that have been created



```
40        def in_occupancy(self):
41            for b in range(len(data)-2):
42                data[b+1][2]=int(data[b][2]) + 1
43                calc_trend=(int(data[b+1][0])-int(data[b+1][1]))+calc_trend
44            print(data[b+1][0],data[b+1][1])
45            occupancy=float(input("what is the occupancy currently at for",week,day))
46            if data[b+1][3] == 0:
47                data[b][0],data[b][1]=occupancy
48            else:
49                data[b][0]=occupancy
50            calc_trend = calc_trend/b
51            if calc_trend >= 0:
52                calc_trend = calc_trend + 1
53            setdata(data)
54            setcalc_trend(calc_trend)
```

This the first version of the code that will take in the occupancy inputted by the user for that single day in the future and the previous data from the database, it then works out the trend difference. It then returns the updated data set for that particular day and the calculated trend data.

40 – This is the initialisation of the method in the class

41 – This is the for loop that goes though the width of the data set

42 – This line increases the count by 1 to show how many times it has been written over

43 – This is how I calculate the trend I do the occupancy – the trend so I can see how it is doing over time this is then made into a sum

44 – This line outputs the occupancy and trend so I can see that it is doing it right

45 – This line askes the user for the occupancy of the hotel currently for a future date

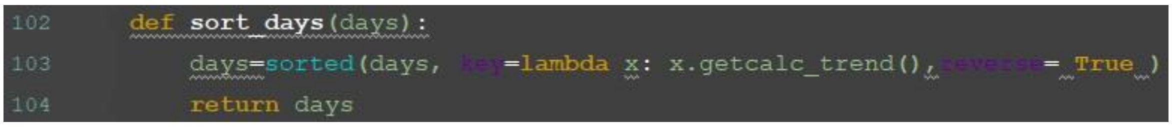
46, 47 – This is the if statement that checks if the count value is 0 and if true it will set occupancy and trend = to the occupancy inputted by the user

48, 49 – If it is false it will write occupancy to just the occupancy row not the trend row

50 – This line will take the calculated trend and divide it by the number of days it has processed it for giving the average trend

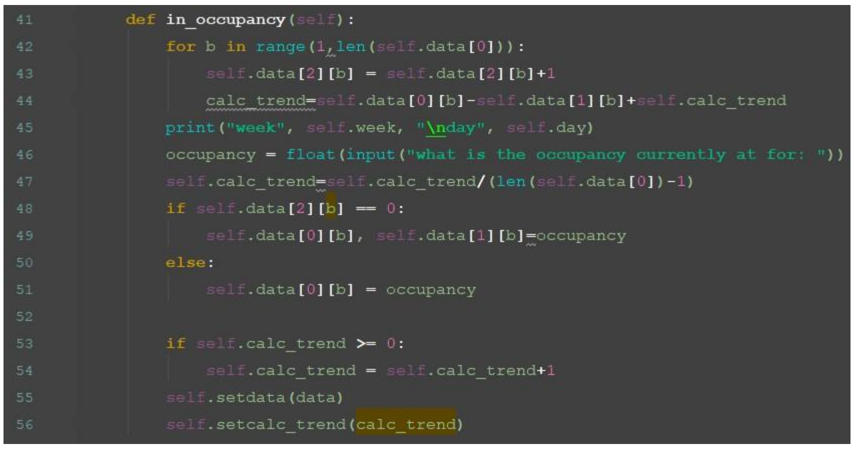
51 – This line asks if the value produced by the code for the trend data is above 0 if it is 100% gets added to It (need to fix)

52, 53 – These lines are the methods to write the data back to the class



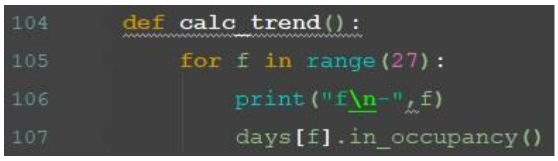
```
    def sort_days(days):
        days=sorted(days, key=lambda x: x.getcalc_trend(), reverse= True )
        return days
```

This function is used to sort the days by the trend data that is saved inside of the class. So within one line I am able to sort a list of appended classes by a selected attribute that is obtained by a function. I then reverse the order, so it goes from smallest to lowest. This then orders it showing the days that have the lowest average occupancy which you obviously want to look at first if you are in a rush as they are lacking behind where you want the occupancy to be ideally above the trend and thus if it's under then that's obviously cause for concern.

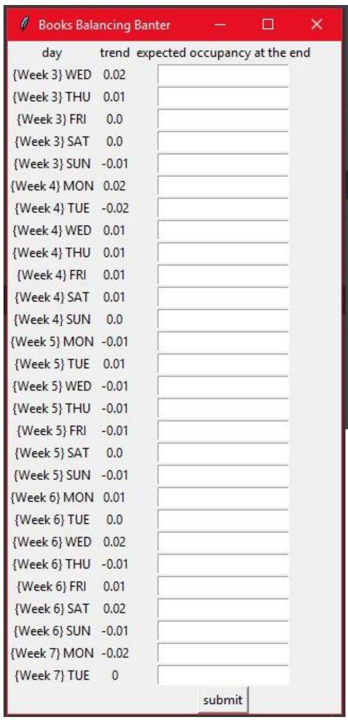


```
    def in_occupancy(self):
        for b in range(1,len(self.data[0])):
            self.data[2][b] = self.data[2][b]+1
            calc_trend=self.data[0][b]-self.data[1][b]+self.calc_trend
        print("week", self.week, "\nday", self.day)
        occupancy = float(input("what is the occupancy currently at for: "))
        self.calc_trend=self.calc_trend/(len(self.data[0])-1)
        if self.data[2][b] == 0:
            self.data[0][b], self.data[1][b]=occupancy
        else:
            self.data[0][b] = occupancy

        if self.calc_trend >= 0:
            self.calc_trend = self.calc_trend+1
        self.setdata(data)
        self.setcalc_trend(calc_trend)
```

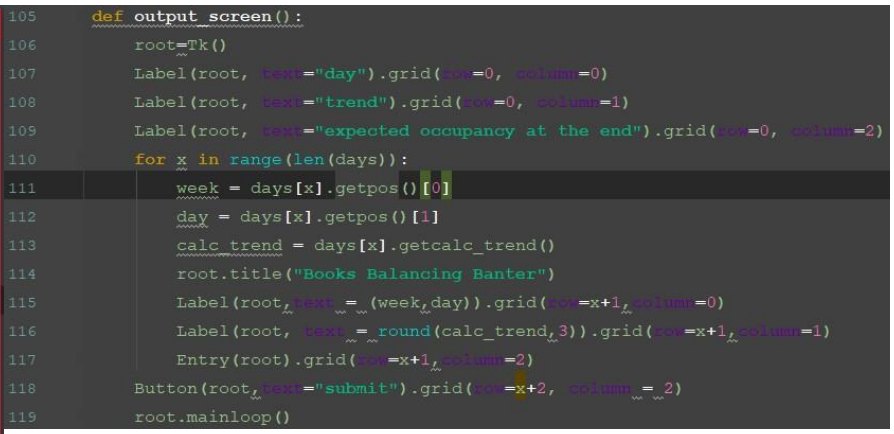
This is the first working version of this function where it uses an input statement that askes the user line by line. What the current occupancy is at for X day and X week, this is then used to calculate the trend for the occupancy of the room. Then the occupancy is written into the data point in the array that belongs to the class that sits on the array Days that is iterated though by a for loop.


```
def calc_trend():
    for f in range(27):
        print("f\n-",f)
        days[f].in_occupancy()
```

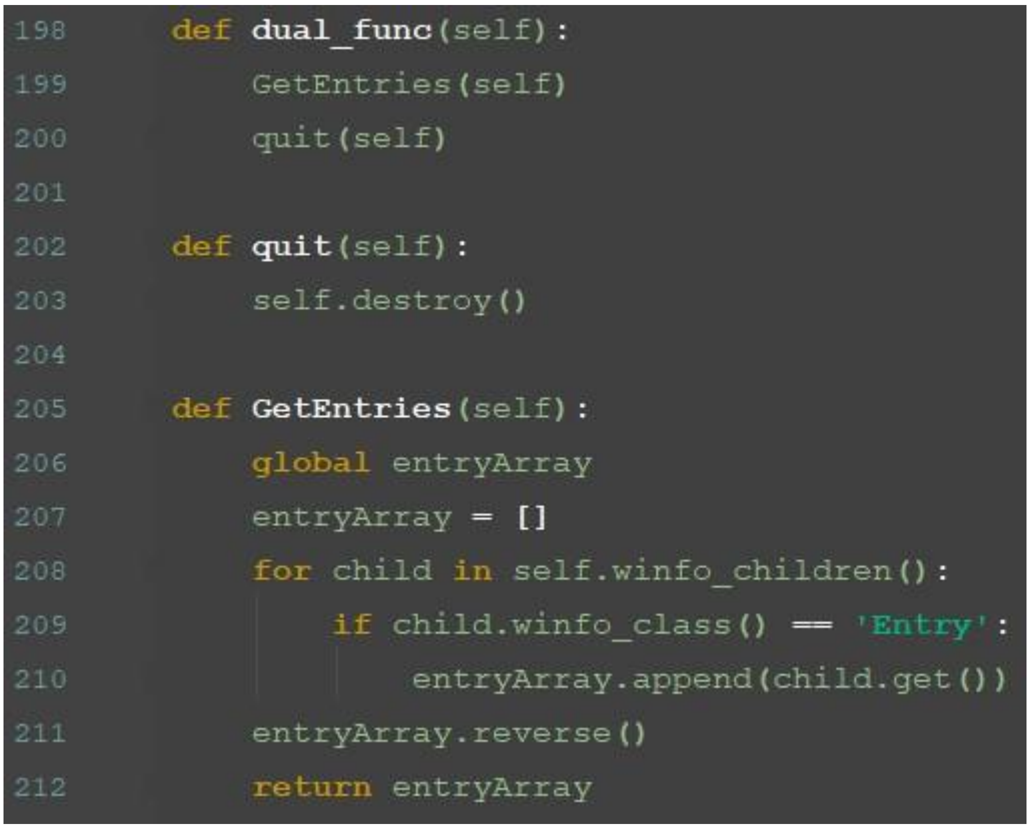
This is the function that calls the function above iterating though all the days and calculating their trend respectfully. This data is used in the final screen where it outputs the day and trend asking for what the user will set the new room price to be.



day	trend	expected occupancy at the end
{Week 3} WED	0.02	
{Week 3} THU	0.01	
{Week 3} FRI	0.0	
{Week 3} SAT	0.0	
{Week 3} SUN	-0.01	
{Week 4} MON	0.02	
{Week 4} TUE	-0.02	
{Week 4} WED	0.01	
{Week 4} THU	0.01	
{Week 4} FRI	0.01	
{Week 4} SAT	0.01	
{Week 4} SUN	0.0	
{Week 5} MON	-0.01	
{Week 5} TUE	0.01	
{Week 5} WED	-0.01	
{Week 5} THU	-0.01	
{Week 5} FRI	-0.01	
{Week 5} SAT	0.0	
{Week 5} SUN	-0.01	
{Week 6} MON	0.01	
{Week 6} TUE	0.0	
{Week 6} WED	0.02	
{Week 6} THU	-0.01	
{Week 6} FRI	0.01	
{Week 6} SAT	0.02	
{Week 6} SUN	-0.01	
{Week 7} MON	-0.02	
{Week 7} TUE	0	


```
def output_screen():
    root=Tk()
    Label(root, text="day").grid(row=0, column=0)
    Label(root, text="trend").grid(row=0, column=1)
    Label(root, text="expected occupancy at the end").grid(row=0, column=2)
    for x in range(len(days)):
        week = days[x].getpos()[0]
        day = days[x].getpos()[1]
        calc_trend = days[x].getcalc_trend()
        root.title("Books Balancing Banter")
        Label(root,text = (week,day)).grid(row=x+1,column=0)
        Label(root, text = round(calc_trend,3)).grid(row=x+1,column=1)
        Entry(root).grid(row=x+1,column=2)
    Button(root,text="submit").grid(row=x+2, column = 2)
    root.mainloop()
```

This function is the first working version of the code where is asks the user for their set of inputs, I have the headers wrong, but it’s supposed to be asking about what price they want to set the room to. I also need to get rid of the “{}” around all of the weeks and format the columns better. I will separate the week and day to be 2 separate columns thus I can format them easier.



```
    def dual_func(self):
        GetEntries(self)
        quit(self)

    def quit(self):
        self.destroy()

    def GetEntries(self):
        global entryArray
        entryArray = []
        for child in self.winfo_children():
            if child.winfo_class() == 'Entry':
                entryArray.append(child.get())
        entryArray.reverse()
        return entryArray
```

This function is what on the press of the button of the output screens will destroy/ close the window but also get the values entered by the user. Because of my limited understanding of programming I will have to make the entryArray variable global so that I can be edited by 2 different functions within functions.

Code Review

This is the first working demo of my code that I have, it lacks the aesthetic appeal nor is it easy to use but the very core fundamentals are there. It allows the user to input a3

CSV file and then go thought the stages of the program spitting out a set of values that are then used to input into the main program and be written to a list, for the user to copy and paste to a file.

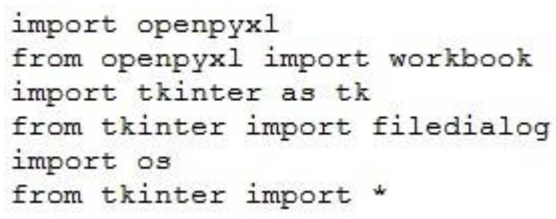
In this version I will improve the user interface and improve the reliability of the code, also once the user has inputted the sets of values I need to save them to the appropriate places. I also need to see if I can optimise my code, reducing the number of operations to achieve the same thing.

With the way that I have imported the file it means that I can’t save to it or write to it, thus I will need to research how I can make this possible. I may have to completely change how I import the data and that may mean changing the data format or using a different library.

The easiest way that I found to check if the code worked was for it to print out the values and then I either use my calculator or look at the XLSX sheet that the data came from and look to see if what I got was the same that I expected. For example, I checked that the XLSX sheet was successfully imported by checking a few random values with the sheet. When the code splits the data up for the different classes I needed to check the day week row position of the data.

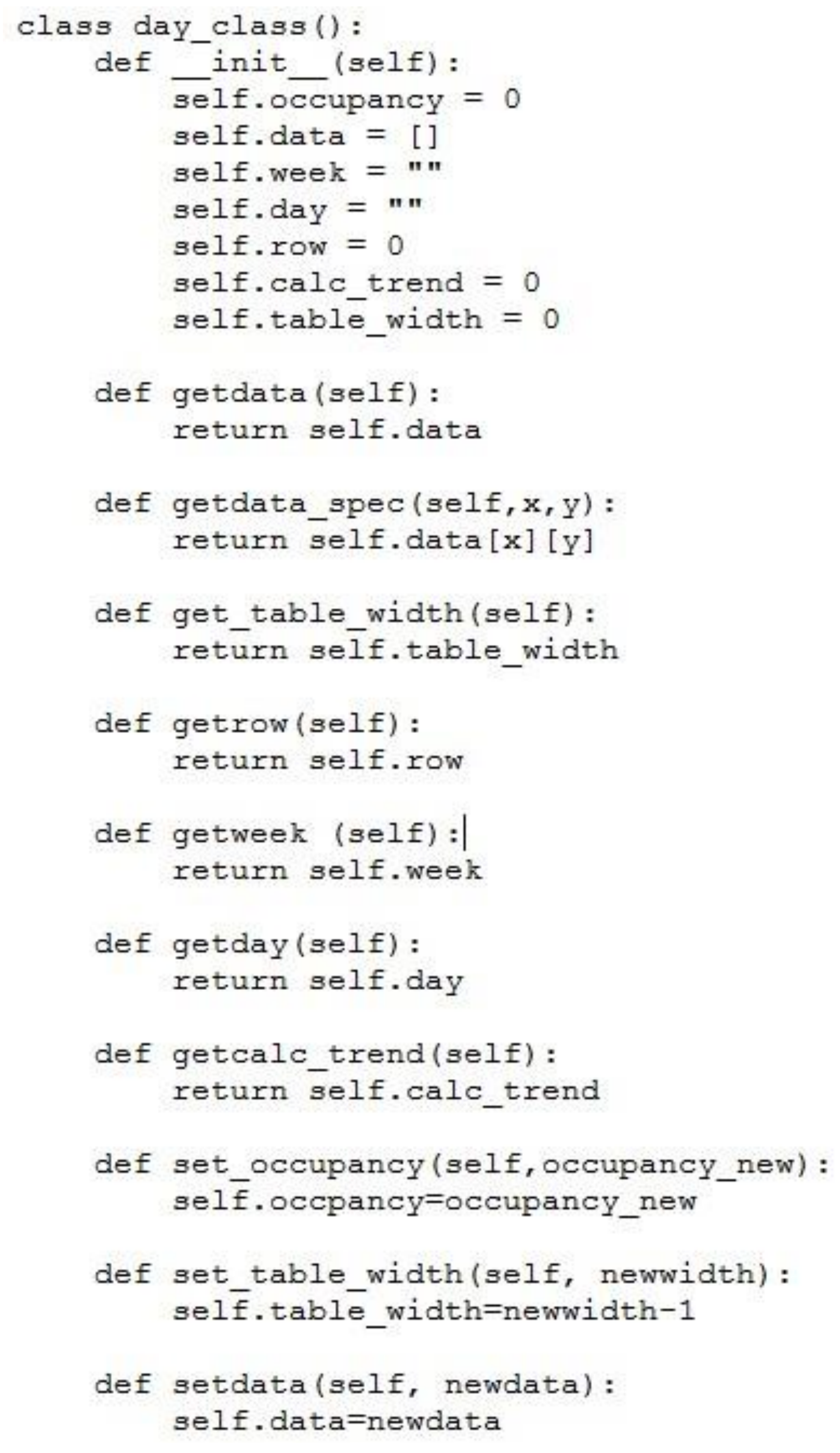
1.2

This is the final version of my code, where I have all wanted output screens working and returning the proper values. I also have the corresponding values being written into the XLSX sheet where desired or being written to a new blank one. This section I'll be mainly talking about the changes per function that I have made to do the outlined things in the code review of section 1.0 and 1.1.



```
import openpyxl
from openpyxl import workbook
import tkinter as tk
from tkinter import filedialog
import os
from tkinter import *
```

This is the import statements at the beginning of my program, as you can see it includes the openpyxl library that I use to open, edit and write XLSX sheets. I decided to use this one because of the large community behind it and thus the vast array of helpful tutorials. I also use the tkinter library, this is for the GUI, and I used this library because it's easy to use and quickly prototype a basic GUI. It also has a large variety of built in GUI functions that I can use for example I implement a way for the user to select a file to import. To program this on my own would be very difficult but because tkinter has a built in GUI function it makes the whole process so much easier.



```
class day_class():
    def __init__(self):
        self.occupancy = 0
        self.data = []
        self.week = ""
        self.day = ""
        self.row = 0
        self.calc_trend = 0
        self.table_width = 0

    def getdata(self):
        return self.data

    def getdata_spec(self,x,y):
        return self.data[x][y]

    def get_table_width(self):
        return self.table_width

    def getrow(self):
        return self.row

    def getweek (self):
        return self.week

    def getday(self):
        return self.day

    def getcalc_trend(self):
        return self.calc_trend

    def set_occupancy(self,occupancy_new):
        self.occpancy=occupancy_new

    def set_table_width(self, newwidth):
        self.table_width=newwidth-1

    def setdata(self, newdata):
        self.data=newdata
```

This is the first part of the code for the initialisation of the class that holds all of the data for the different days that are being used/ processed. Except the last routine all of them are for the storing or retrieval of data from the class. It's very simple but works well the only way an error occurs is if the data that gets inputted into it has errors.

```
    def setday(self, newday):
        self.day=newday

    def setweek(self, newweek):
        self.week=newweek

    def setrow(self, newrow):
        self.row=newrow

    def setcalc_trend(self, newtable):
        self.calc_trend=newtable

    def day_trend_calc(self):
        b=0
        for b in range(self.table_width):
            a = self.data[0][b]
            s = self.data[1][b]
            self.calc_trend=(a-s)+self.calc_trend


        self.calc_trend=self.calc_trend/self.table_width
        self.occupancy = entryArray[self.table_width-1]
        if self.data[2][self.table_width-1] == 0:
            self.data[0][self.table_width-1]=self.occupancy
            self.data[1][self.table_width-1]=self.occupancy
        else:
            self.data[0][self.table_width-1] = self.occupancy
            self.data[1][self.table_width-1]=((self.data[2][self.table_width-1]*
                                               self.data[1][self.table_width-1]+
                                               self.occupancy)/(self.data[2][self.table_width-1]+1))

        self.data[2][b] = self.data[2][b]+1
        self.setdata(self.data)
        self.setcalc_trend(self.calc_trend)
```

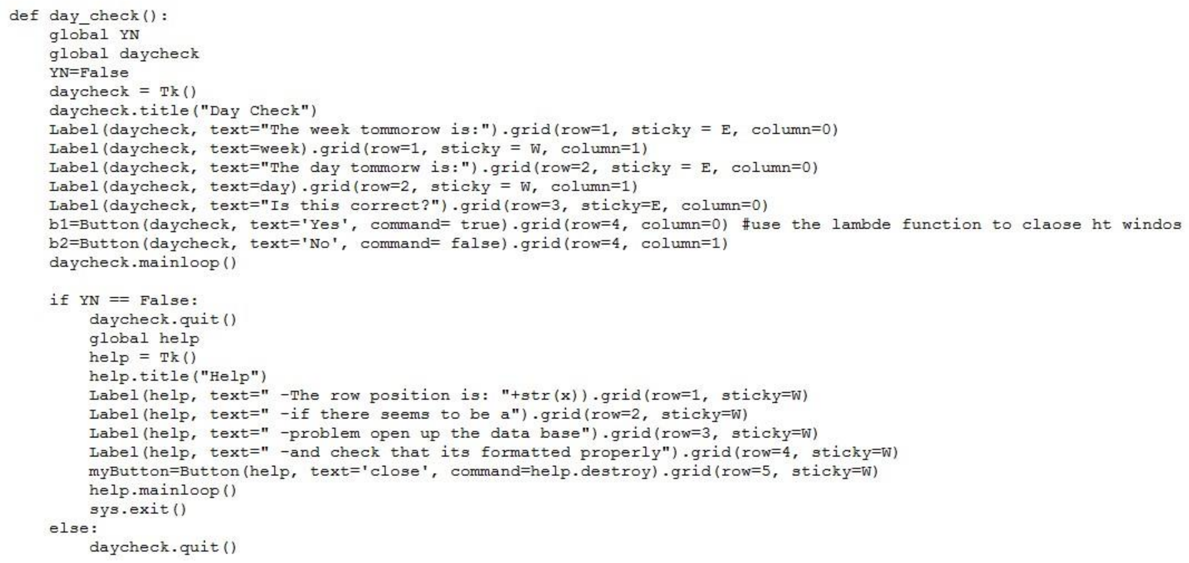
The last procedure of the class is the dat_trend_calc that uses the input from a later function and then performs the necessary calculations to return and write to the class the trend that has been calculated.

```
def get_data ():
    global book
    application_window=tk.Tk()
    my_filetypes=[('all files', '.*'), ('text files', '.txt')]
    location=filedialog.askopenfilename(parent=application_window,
                                         initialdir=os.getcwd(),
                                         title="Please select a file:",
                                         filetypes=my_filetypes)
    application_window.destroy()
    book = openpyxl.load_workbook(location)
    data = book.active
    return data
```

This is the first function that is run it open up a box that's identical to the kind of window that you would find if in word you wanted to find a file. The user then selects the wanted file and its location is returned to be imported by the program and then goes on to the next section.

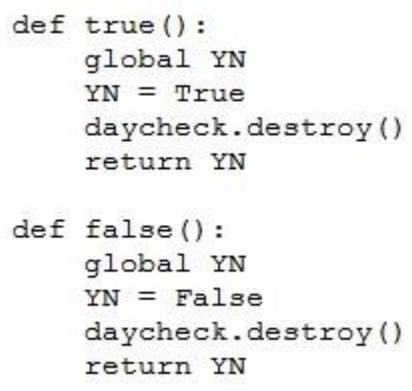
```
def get_DayWeek (data):
    t=3
    x = 0
    while x == 0:
        b = data.cell(row=t, column=28).value
        n = data.cell(row=t+3, column=28).value
        if b > n:
            print("found position of data begining")
            x = t + 1
        t = t + 3
    return (data.cell(row=x, column=1).value), (data.cell(row=x+1, column=1).value), (x)
```

This function scans though the data base looking for a certain change in value that would indicate where the next day to be worked on lies. It then returns the day, week and row value of that day. I assign the values of "b" and "n" separately of the "if" statement because I want to make code maintenance easier.



```
def day_check():
    global YN
    global daycheck
    YN=False
    daycheck = Tk()
    daycheck.title("Day Check")
    Label(daycheck, text="The week tommorow is:").grid(row=1, sticky = E, column=0)
    Label(daycheck, text=week).grid(row=1, sticky = W, column=1)
    Label(daycheck, text="The day tommorw is:").grid(row=2, sticky = E, column=0)
    Label(daycheck, text=day).grid(row=2, sticky = W, column=1)
    Label(daycheck, text="Is this correct?").grid(row=3, sticky=E, column=0)
    b1=Button(daycheck, text='Yes', command= true).grid(row=4, column=0) #use the lambde function to claose ht windos
    b2=Button(daycheck, text='No', command= false).grid(row=4, column=1)
    daycheck.mainloop()

    if YN == False:
        daycheck.quit()
        global help
        help = Tk()
        help.title("Help")
        Label(help, text=" -The row position is: "+str(x)).grid(row=1, sticky=W)
        Label(help, text=" -if there seems to be a").grid(row=2, sticky=W)
        Label(help, text=" -problem open up the data base").grid(row=3, sticky=W)
        Label(help, text=" -and check that its formatted properly").grid(row=4, sticky=W)
        myButton=Button(help, text='close', command=help.destroy).grid(row=5, sticky=W)
        help.mainloop()
        sys.exit()
    else:
        daycheck.quit()
```

This function returns to the user the day and week of tomorrow and asks the user if it's correct, if they press the no button then. It will display a message recommending them to check the data base for formatting faults and then run the program again, after this it closes the window and closes the program.



```
def true():
    global YN
    YN = True
    daycheck.destroy()
    return YN

def false():
    global YN
    YN = False
    daycheck.destroy()
    return YN
```

If the user presses the button labelled yes, then it will run the 1st function if no it will run the 2nd function. If they press the 1st then it will continue the running normally if they press the 2nd it will display the message showed earlier then then stop the running of the program completely, there will not be any problems with stopping the programs running abruptly as there are no values that have been edited or are being edited thus no data will be lost as it will return to the state of the program before it was run.

```
def split_data(x, days):
    for i in range(28, 0, -1):
        j = i-1
        datatemp = [[0]*(j) for w in range(3)]
        days.append(day_class())
        for o in range(0,3):
            for p in range(1,i):
                datatemp[o][p-1] = data.cell(row=(x+o), column=p+1).value
        x=x+3
        days[28-i].setdata(datatemp)
        days[28-i].setday(data.cell(row=x-2, column=1).value)
        days[28-i].setweek(data.cell(row=x-3, column=1).value)
        days[28-i].setrow(x-3)
        days[28-i].set_table_width(i)
    return days
```

This function is where the classes are created and their values assigned, it runs 28 times for the 28 days being calculated. It goes though and writes in the day, week, row, table width and the data that is being worked on. When I create the class, I could assign all of its values but I don't want a massively long and complex line, this again is trying to stick to the idea of keeping the code easy to maintain.

```
def in_occupancy():
    global input_current
    input_current = Tk()
    occupancy_tbl = [0]*len(days)
    input_current.title("Occupancy input")
    Label(input_current, text="week").grid(row=0, column=0)
    Label(input_current, text="day").grid(row=0, column=1)
    Label(input_current, text="Current occupancy").grid(row=0, column=2)
    for x in range(len(days)):
        day = days[x].getday()
        week = days[x].getweek()
        Label(input_current, text=week).grid(row=x+1, column=0)
        Label(input_current, text=day).grid(row=x+1, column=1)
        occupancy_tbl[x]=Entry(input_current).grid(row=x+1, column=2)

    myButton=Button(input_current, text='Get Entries', command = lambda: dual_func(input_current)).grid(row=x+2, columnspan=3, sticky = E+W)
    input_current.mainloop()

    for x in range(len(days)):
        days[x].set_occupancy(occupancy_tbl)
```

This function creates a screen that has all of the days and weeks in their respective place, ordered from top to bottom, soonest to latest. Then it has rows of boxes for the user to input the current occupancy for they then press the button at the bottom and the window closes writing all the values to an array.

```
def calc_trend():
    for f in range(27):
        days[f].day_trend_calc()
```

This function calls the day_trend_calc procedure I won't get any errors as it's a very simple procedure that calls a procedure with an incrementing value.

```
def dual_func(self):
    GetEntries(self)
    quit(self)

def quit(self):
    self.destroy()

def GetEntries(self):
    global entryArray
    entryArray = []
    for child in self.winfo_children():
        if child.winfo_class() == 'Entry':
            entryArray.append(child.get())
    entryArray.reverse()
    return entryArray
```

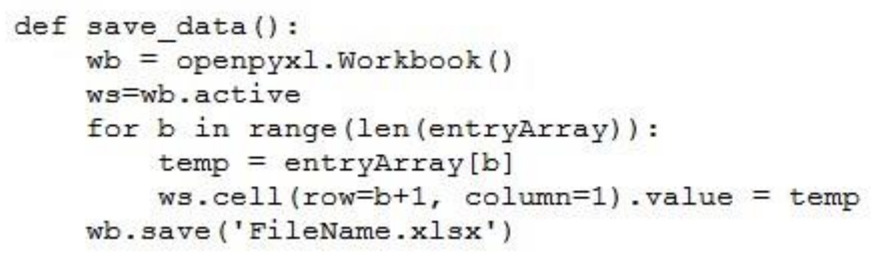
These functions get all the values from the input boxes in the GUI and writes them to an array and then closes the window. This is possible because the screen is considered a class and thus it can iterate though all widgets and text and buttons to get the values that exist in the entry boxes. There won't be any errors as it will accept any value inputted into the entry widget, thus the user can look though the XLSX sheet and edit or review the values inputted.

```
def return_data(x, days, data):
    for i in range(28):
        width = days[i].get_table_width()
        for o in range(0,3):
            for p in range(0,width):
                data.cell(row=x+o+1, column=p+2).value = days[i].getdata_spec(o,p)
        x=x+3
    book.save("memes.xlsx")
    return data, book
```

This function writes all of the class's data with edited values back into the XLSX sheet, it is then saved. This is the function that takes the inputs for the occupancy of the hotel and thus calculates the trend etc. I don't see any way that this can have a fault as there is no requirement for any of the data to be of a certain type, thus it can be monitored and over seen by the user.

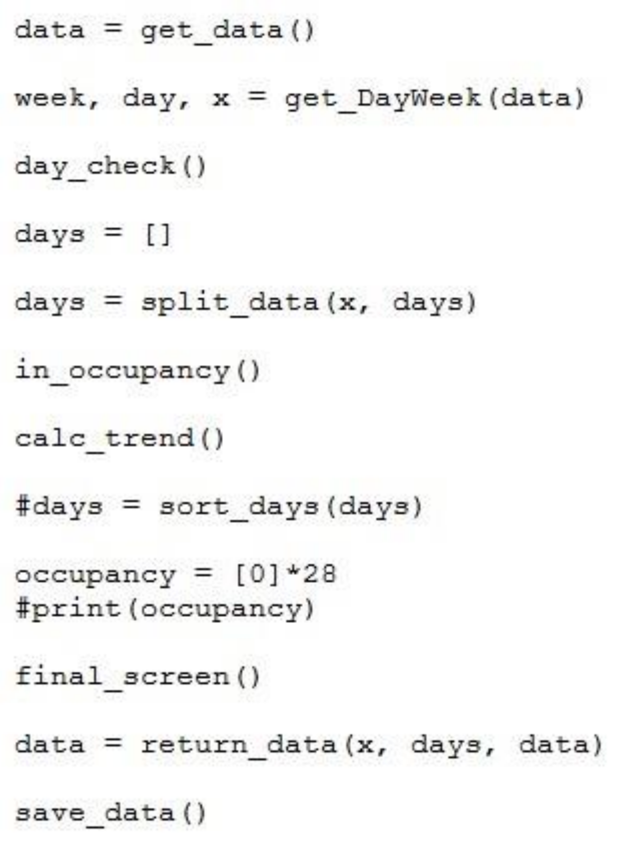
```
def final_screen():
    global root
    root = Tk()
    root.title("Books Balancing Banter")
    Label(root, text="week").grid(row=0, column=0)
    Label(root, text="day").grid(row=0, column=1)
    Label(root, text="trend").grid(row=0, column=2)
    Label(root, text="expected occupancy at the end").grid(row=0, column=3)
    for x in range(len(days)):
        week = days[x].getweek()
        day = days[x].getday()
        calc_trend = 100*round(days[x].getcalc_trend(),4)
        Label(root, text= (week)).grid(row=x+1, column=0, sticky=E)
        Label(root,text = (day)).grid(row=x+1, column=1, sticky=W)
        Label(root, text = round(calc_trend,4)).grid(row=x+1, column=2, sticky=E+W)
        occupancy[x] = Entry(root).grid(row=x+1, column=3)

    myButton=Button(root, text='Get Entries', command= lambda: dual_func(root)).grid(row=x+2, columnspan=4, sticky = E+W)
    root.mainloop()
```

This is the last screen the user sees where they input the new prices that they want to set for each day. The code above the for-loop is for the column headers and the naming of the window. The for-loop then gets the data from the classes on the variable days and then outputting them along with an input box.



```
def save_data():
    wb = openpyxl.Workbook()
    ws=wb.active
    for b in range(len(entryArray)):
        temp = entryArray[b]
        ws.cell(row=b+1, column=1).value = temp
    wb.save('FileName.xlsx')
```

This then takes all the values that the user inputted for the room prices and then writes them to a new XLSX sheet that it creates. There won't be any errors here as the program can
accept and write any type of data to the cells. The user may have written themselves a note and thus wanted to save it to the file.



```
data = get_data()

week, day, x = get_DayWeek(data)

day_check()

days = []

days = split_data(x, days)

in_occupancy()

calc_trend()

#days = sort_days(days)

occupancy = [0]*28
#print(occupancy)

final_screen()

data = return_data(x, days, data)

save_data()
```

This is the list of functions that get called during its running

Code review

My program work well and if all the values are inputted correctly then it runs with no errors, I have very little error handling in my program as this is a very early build (the final build for the project) and thus I'm very happy just to have my code work, and thus I leave the reliance for the working over the program to the user to input valid values. Despite the complex instructions and a large amount of for loops it still runs very quickly ½ a second or less in between the main screens closing and the next opening.

I find that if the outputted result matches what I was expecting then everything in the background must be going how I expect. So when the trend calculations and the data is outputted to the user if it matches up with what I would have gotten on my calculator then I'm happy if the days for the output screens are in chronological order that's a pass from me. When I input the values and then return them back to the program and they get written to the sheet exactly how I expected then that's a pass.

Testing

White/ black box testing

Test nº	Function	Input/ Condition	Expected result	Actual result
1	Import data	XLSX sheet selected by the user	The table is printed out for me to check and all the values align to what's in the sheet	As expected
2	Get day/ week	The data sheet from the function import data	A day, week, row position of the next day to be worked on	As expected
3	Check day/ week	The output from get day/ week	A dialogue box displaying the day and week to the user for them to say if its correct or incorrect	As expected
4	Split data	The values from the 1st 2 functions	28 classes appended to a list, that have the correct values stored in them	As expected
5	Input occupancy	A set of numbers between 0-100 but any number would be accepted	They would be written to an array	As expected
6	Calculate trend	The variable days which has all the classes on it, and the inputted occupancy from the user	New values for the trend value in the classes on the variable days. That are what they are expected to be	As expected
7	Final screen	A set of numbers that are the new prices for the rooms on the corresponding days	They are written to an array	As expected

8	Return data	The data in the classes on the array days	A set of updated values in the correct positions with the relevant values in the XSLX sheet	As expected
9	Save data	The data from the function final screen	This writes the array from the final screen function into a separate XSLX sheet in a column of numbers	As expected
10	Button check for the day check	The user clicks a button	If they click yes, the window close and the program continues running. If they click no it prints out a recommendation to check the formatting of the database and then closes the window and ends the program.	As expected
11	The file selector opens correctly	The program is ran	The window will open that allows the user to select a file	As expected

Client testing

I won't be involving the stake holders in the testing of the product as it's not ready to be used by them my client says. So, I sat down with the client and talked and showed him what I had created. He wrote out a bunch of changes and improvements which I've talked about it in a later section. Due to time constraints I haven't made it as user friendly as it could be thus he had some difficulty using it until I explained what each page did and how it worked. At the end he thought that it was more of an early prototype of the possibility of a greater concept. He was very happy with the product that had been delivered.

[illegible]

This is his signature showing that he believes that it meets his expectations and completes the task that I set out to do.

Evaluation

Meeting success criteria

Success criteria

1. My program will analyse large databases from hotels business on their books (BoB). It will use complex algorithms to spit out certain values that can be used by the user to set the price for the hotels rooms. They could also be used to provide business intelligence in decision making.

2. My program will have a minimalistic user interface this is so that it makes using the program easier and thus faster, as my program is designed to speed up the time it takes for the finance manager to do this.

3. The program must output its pricing in a desirable way such that it can be used by price parity programs that can update the hotels prices across many sites; travel agents, websites, local systems. If it wasn't outputted in a useable format, then the revenue manager would have to input all of the prices separately which takes time

Has the criteria been met?

1. My program over time builds up a large database/ history of the hotels occupancy and thus the finance manager can look though that and see the average occupancy of the hotel at X days away from day Y. thus he can see trends and look at on average how the hotel performs and how the occupancy increases in relation to days until. Because of this large data set I think that my program has passed this success criteria.
2. My user interface forces the user to complete each task step by step and thus there is no ambiguity as to what order things are completed in and thus it makes it easier and faster to use. Thus, I think that this criteria is completed.
3. The new prices are written in the 1st column of a newly created array for easy importation into in different software's and data structures. Because of the output format being a column of numbers in an XLSX format I think I have succeeded in the criteria.

Limitations/ criteria not met

The main limitation of my code is that there is no tutorial, if you haven't been told how the program works then it's very unintuitive as to what you do. The reason why something being easy to use is important is because the idea of the program is to be quick and easy to use thus if something takes a long time to figure out how it works then that's going against that idea. I would do this by adding a help button on each page so that the user can be prompted as to what each page is for and dose next, and at the begging a tutorial that goes through each page and explains what each one dose and what it's for.

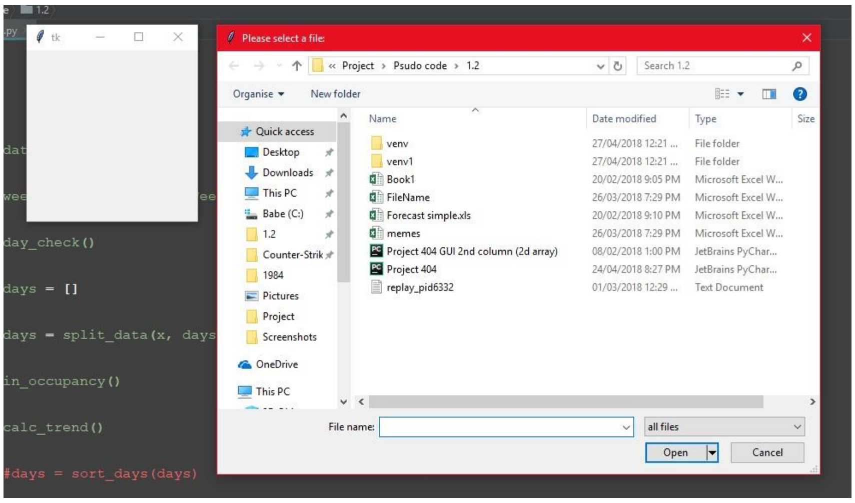
Another limitation of my code is that the data base cannot account for popular days such as Christmas. In the future I would plan to change the data base so that for example Christmas which would obviously be a very popular time of year, but it falls on random days one year its Monday the next a Thursday. This would obviously skew the data, so I would need my data structure to be able to hold data about data specific dates. This would probably be done using another table with pre-programmed date ranges and ones set by the user. For example, school holidays could be added by the user. But things like the week before Christmas and New Year's or Easter weekend, would be pre-programmed. When the program detected that the date was X days away it would load the desired range in the program and work from that instead, prompting the user that a custom date range was being loaded.

Another limitation of my code is that the main reason why it's so useful except for the speed of the admin that it does is the history of the data that it holds and thus for the 1st year the data that you have there is useless thus to make the program useful from the begging is by allowing the user to back date and create history. I would create a back-dating feature that would allow the user to fill in the occupancy prior to the date of the programs deployment. It could also be imported automatically; the user would state the import format and how the dates/data was structured. Because of this they would be able to easily and quickly create a vast history and thus improve the usability and usefulness of the program hugely in a very small amount of time.

User testing and client review

As stated earlier in its limitations there is no tutorial and thus when using the program if I didn't explain what to do and how it worked they had almost no idea as to what each page did and how it worked and what it accomplished. Once they understood what the program did and how I worked they were very happy with the resulting program, except for the lack of a tutorial.

They would also recommend that I add a date to the pricings that are outputted for better integration with price parity programs. The reason why adding a date is the solution is so that 1- it ensures that the user can look over the past pricings that have been submitted 2- it helps automate the process and speed it up thus improving the programs useful ness because the price parity program can automatically detect the date and then import the prices instead of them having to be entered by hand.



```
day_check()
days = []
days = split_data(x, days
in_occupancy()
calc_trend()
#days = sort_days(days)
```

Please select a file:

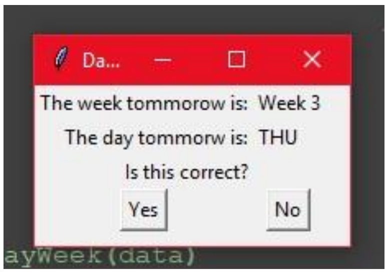
« Project › Psudo code › 1.2

Name	Date modified	Type	Size
venv	27/04/2018 12:21 ...	File folder	
venv1	27/04/2018 12:21 ...	File folder	
Book1	20/02/2018 9:05 PM	Microsoft Excel W...	
FileName	26/03/2018 7:29 PM	Microsoft Excel W...	
Forecast simple.xls	20/02/2018 9:10 PM	Microsoft Excel W...	
memes	26/03/2018 7:29 PM	Microsoft Excel W...	
Project 404 GUI 2nd column (2d array)	08/02/2018 1:00 PM	JetBrains PyChar...	
Project 404	24/04/2018 8:27 PM	JetBrains PyChar...	
replay_pid6332	01/03/2018 12:29 ...	Text Document	

File name:

all files

This first screen has no tutorial associated with it thus if the use dose not know what they're supposed to do with this screen then they will just become stuck. This would then make the program near impossible to use, but once my client learnt what this page was for and how it worked then he was fine. So, if I was going to sell this commercially then it would need a tutorial, or an instructional video associated with it.



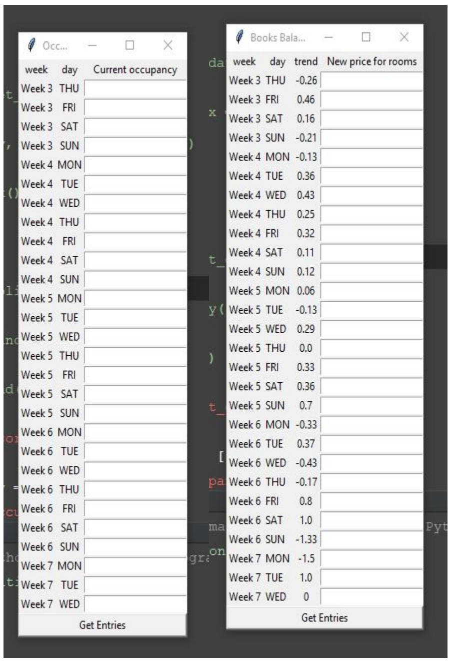
Da...

The week tommorow is: Week 3

The day tommorw is: THU

Is this correct?

This screen is easy to understand its asking the user a simple question. But I feel like the use would want to know what's happened and thus how did my program get to this situation. What would it mean if it was not correct, and how would it be sorted out.



Occ...

week	day	Current occupancy
Week 3	THU	
Week 3	FRI	
Week 3	SAT	
Week 3	SUN	
Week 4	MON	
Week 4	TUE	
Week 4	WED	
Week 4	THU	
Week 4	FRI	
Week 4	SAT	
Week 4	SUN	
Week 5	MON	
Week 5	TUE	
Week 5	WED	
Week 5	THU	
Week 5	FRI	
Week 5	SAT	
Week 5	SUN	
Week 6	MON	
Week 6	TUE	
Week 6	WED	
Week 6	THU	
Week 6	FRI	
Week 6	SAT	
Week 6	SUN	
Week 7	MON	
Week 7	TUE	
Week 7	WED	

Books Bala...

week	day	trend	New price for rooms
Week 3	THU	-0.26	
Week 3	FRI	0.46	
Week 3	SAT	0.16	
Week 3	SUN	-0.21	
Week 4	MON	-0.13	
Week 4	TUE	0.36	
Week 4	WED	0.43	
Week 4	THU	0.25	
Week 4	FRI	0.32	
Week 4	SAT	0.11	
Week 4	SUN	0.12	
Week 5	MON	0.06	
Week 5	TUE	-0.13	
Week 5	WED	0.29	
Week 5	THU	0.0	
Week 5	FRI	0.33	
Week 5	SAT	0.36	
Week 5	SUN	0.7	
Week 6	MON	-0.33	
Week 6	TUE	0.37	
Week 6	WED	-0.43	
Week 6	THU	-0.17	
Week 6	FRI	0.8	
Week 6	SAT	1.0	
Week 6	SUN	-1.33	
Week 7	MON	-1.5	
Week 7	TUE	1.0	
Week 7	WED	0	

these two pages are like the first one having no tutorial associated with it thus if the use does not know what they're supposed to do with this screen then they will just become stuck. This would then make the program near impossible to use, but once my client learnt what this page was for and how it worked then he was fine. So, if I was going to sell this commercially then it would need a tutorial, or an instructional video associated with it.

How easy is the code to maintain

I've made sure that my code uses a modular design to ensure that the code can be easily checked for errors and edited. Also, by using functions I can then loop though certain repetitive tasks instead of having to write them all out 28 times.

My code is lacking comments but It does come with lots of documentation as to how it works and the overall purpose of each function (this document). Since this program is developed individually and not in a team wreathe or not I comment my code doesn't really matter as I'm the only one that needs to know what each line and function.

Python is an interoperated language thus making changes is very easy to do I don't have to compile the program every time I want to change something about my program. Thus, editing my code and seeing the resulting changes it a far faster process which means that the overall development speed is far faster.

In conclusion I think that my code is easily maintained if you can understand it due to it all being in for loops instead of line by line, and it being in functions to group. To make my code easier to maintain and understand I should add comments as to show to anyone looking at my code what each line dose.

What would be changed in the future

The main limitation of my code is that there is no tutorial, if you haven't been told how the program works then it's very unintuitive as to what you do. The reason why something being easy to use is important is because the idea of the program is to be quick and easy to use thus if something takes a long time to figure out how it works then that's going against that idea. I would do this by adding a help button on each page so that the user can be prompted as to what each page is for and dose next, and at the begging a tutorial that goes through each page and explains what each one dose and what it's for.

In the future I would plan to change the data base so that for example Christmas which would obviously be a very popular time of year, but it falls on random days one year its Monday the next a Thursday. This would obviously skew the data, so I would need my data structure to be able to hold data about data specific dates. This would probably be done using another table with pre-programmed date ranges and ones set by the user. For example, school holidays could be added by the user. But things like the week before Christmas and New Year's or Easter weekend, would be pre-programmed. When the program detected that the date was X days away it would load the desired range in the program and work from that instead, prompting the user that a custom date range was being loaded.

I would create a back-dating feature that would allow the user to fill in the occupancy prior to the date of the programs deployment. It could also be imported automatically; the user would state the import format and how the dates/data was structured. Because of this they would be able to easily and quickly create a vast history and thus improve the usability and usefulness of the program hugely in a very small amount of time.

In the future I would try to implement different ways of analysing the data to begin to create predictions these different methods would either be represented individually or averaged to create a hopefully meaningful, useful and accurate prediction of the occupancy.

The reason why this would be done is part of the idea of doing all the time intensive work; long algorithms and data analysis that the finance manager doesn't have time to do and instead uses his/ her experience to make an educated guess. So, I would then try to get as much information in front of the user so that they can use their experience and reduce the guess work hopefully improving yield.

I sat down with my client and he stepped though the program with me talking about ways that I could improve it, except spelling errors and rewording columns/ buttons, he brought up some very interesting points and things to develop.

He thought that I should include another column when the user inputs the price which should hold a prediction as to what the program thinks the occupancy will be at roughly because all they have to go on is just a % that is based on how closely this year's occupancy follows a trend. I could very easily add a 2nd Column that would show a calculated estimate alongside the trend as another way of giving the user the maximum amount of information.

On top of not having a tutorial my program also lacks any form of help once the tutorial has been worked through; if you forget what a certain page dose you could press a help button and then receive a prompt as to what the page is for and what it does. This feature would aim to speed up the process because they wouldn't have to cancel the whole process and then go back though the tutorial, instead they can get help then and there where they needed it thus speeding up the whole process.

When testing the program, I didn't have this problem, but my client did; when the 2 main pages show they are often too big for the screen thus my client proposed 2 sets of columns of 14 that would then in total make up the 28 days being worked on. This would allow the whole screen to fit on any page thus solving the problem of not being able to see the full page.

My client recommend that I add a date to the pricings that are outputted for better integration with price parity programs. The reason why adding a date is the solution is so that 1- it ensures that the user can look over the past pricings that have been submitted 2- it helps automate the process and speed it up thus improving the programs useful ness because the price parity program can automatically detect the date and then import the prices instead of them having to be entered by hand.

In some large corporations you may have a finance manager over seeing multiple hotels and thus allowing the system to work for multiple hotels would be very important this could be done by running the same program multiple times but for different databases or could be ran once but then cycle though a multi-dimensional array that holds multiple hotels data this improves the usability of the program and speeds up the process which is the whole idea of it.