Table of Contents

[Criteria 2](#_Toc447193109)

[Code 3](#_Toc447193110)

# Criteria

For task 3 I had to make a program that gave the user options to check previous estimates, displaying outstanding payments and display the company revenue.

# Code

I started by creating a function called search.



I had to access and read from a text file so I saved it in a variable.

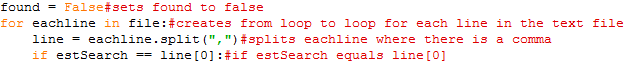


The “r” in the code allows to only read from the file.

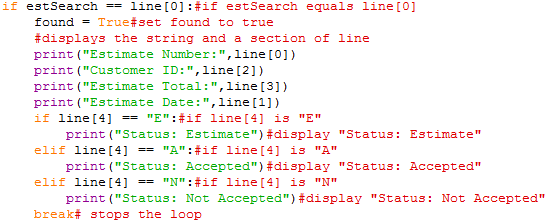
The user has to enter the customer number they are looking for and it is saved in the variable estSearch.



I created a variable called “found” that is set to false unless the customer the user is looking for is found. To search for the customer number I looped through each line in the text file. Each line would be separated at the commas and saved as the list “line”. To check if that line contained the customer number I used an if statement to compare the variable estSearch with the first item in the list line.

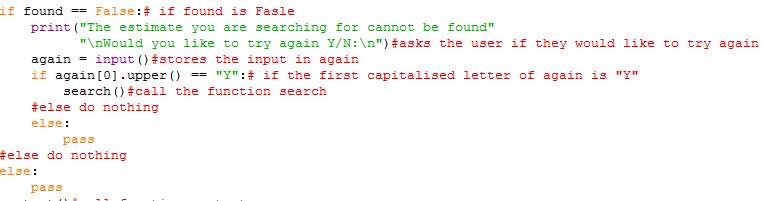


If estSearch is the same as the first item in the list “line”, found is set to “True” and the customers estimate details are displayed. Also the loop is stopped by the break in the code.



If estSearch is not the same as the first item in the list “line” then it passes and loops through the rest of the lines in the code until it is found or there are no more lines to loop through.

Once the program has finished looping through the lines in the text file it checks if the variable “found” is still false.



If “found” is still false then it displays a message telling the user that the customer they were looking for couldn’t be found and ask them if they would like to try again. The user’s input is saved in the variable “again”. To check the user’s input the first letter is capitalised and compared with the letter “Y”, if its “Y” then it calls the search function. If the first letter of the variable again is not “Y” then pass.

I then created a function called “revenue” that would generate the company revenue.



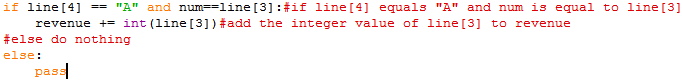
The “r” in the code allows to only read from the file.

The revenue would only be calculated using completed payments so I had to check which jobs were completely paid for. I did this by looping through the text file and saving each line to the variable “line”.





Once each line is split I need to get the fifth item so I can use it in and if and else statement later. The fifth element has “\n” at the end of it so it has to be removed and saved in the variable “num”.

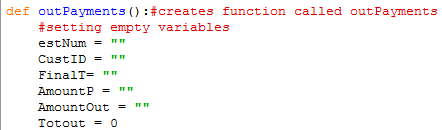


I check the 4th if it is equal to “A” (which means that the job is completed) and if variable num is equal to the 3rd item in the line (the 3rd item is the amount paid), if it is then the 3rd item in line is converted to an integer and added to the variable revenue.



The revenue is displayed and the restart function is called

I next created a function called “Outpayments” that would show all outstanding payments in a table and calculate the total outstanding payments. I also setup empty variables that I would be able to use later.



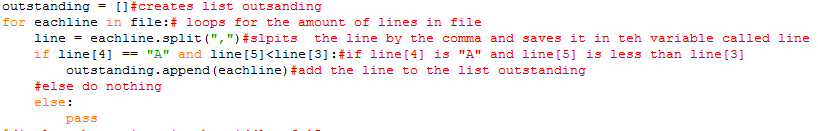
estNum is would be the estimate number, CustID would be the customer ID, FinalT would be the final total, AmountP is the amount paid, AmountOut is the amount outstanding and the Totout is the Total Outstanding.

I had to access and read from a text file so I saved it in a variable.



The “r” in the code allows to only read from the file.

To display the outstanding payments in a table I had to first found the lines that contained outstanding payments. I did this by looping through each line and checking if the 3rd item in the line was bigger than the 5th item (the 5th item is the amount paid and the 3rd item is cost of job) and if the 4th item was “A” (This meant that the job was accepted).



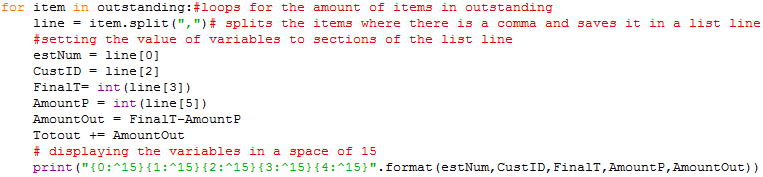
If the 4th item was “A” and the 3rd item was bigger than the 5th then it would be added to the list called outstanding.

Once all the outstanding payments were found they had to be displayed in a table. I did this by using string formatting to allocate a certain amount of space to each string.



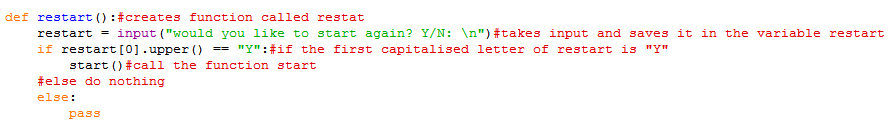
Inside the curly brackets the first number is the index and the second number is the amount of space, the symbol points in the direction where the string will be placed (< - this will place the string at the start of the space, ^ - this will place the string in the middle of the spaces, > - this will place the string at the end of the spaces). This setup my table headings, now the contents of the table had to be displayed.

I looped through the items in the list outstanding and split them at the commas and saved it in the variable “line”. I saved the details of the line in variables so it would be easier to understand what would be displayed when the program is run. Once the variables were setup I used string formatting again to get my variables to line up with the correct heading.



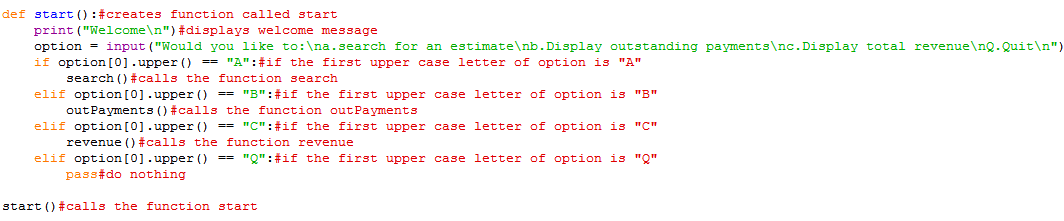
Before I used string formatting I tried escape sequences to line up the text but it was very hard to get the text in the correct position so I used string formatting because it was easier to do and easier to read.

I created a function called restart that would be called at the end of functions revenue, OutPayments and search. The function would give the user the option to restart the program if they choose to.



It asks the user if they want to restart and saves their input in the variable called “restart”. I capitalised the first letter of the variable restart and compare it the letter “Y”, if its “Y” then is calls the function start. If it’s not it does nothing.

I created a function called “start” that would be used to call other functions based on what the user would like to do.



The function first displays a welcome message and asks the user if they would like to search for an estimate, display outstanding payments, display revenue or quit. The user chooses by typing a,b,c or Q. It saves the input in the variable called “option”. To check the user input I capitalise the first letter and compare it to the letter A,B,C and Q. if it is “A” then it calls the search function, if it is “B” then the outPayments function, it is “C” it displays the revenue or “Q” it does nothing.

I separated my program into subprograms to make it easier to read and I would be able to use sections of my program separately making testing easier and faster. Splitting up my program into sections also made it easier to work with as I didn’t have to worry about ruining different sections of my code and made it easier to find a specific piece of code.

Conclusion

The program meets all the criteria as I can search for an estimate, display outstanding payments and display the company revenue without any problems. The interface is clean and easy to read so it is easy for the user to use. The only problem with the program is that there is not enough validation so the user can still cause errors.