Assignment 2

1. **Description of Implementation**

This application demonstrates the required functions shown in **Table 1**, whilst applying coding best practices. Functions were built to be simple and readable, minimising redundancy whilst maximising efficiency by using helper functions (Reitz & Schluser, 2016). Functions are named for the actions they perform, with variables represented as nouns to increase the readability of code (Martin 2008).

Table : Assignment 2 Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Function | Line | Description |
| Enter at least five records | add\_company()  input\_company\_info() | 94  127 | add\_company() uses a helper function called input\_company\_info() which collects input data, stores it in a dictionary, and appends it to master\_list (which stores all company dictionaries) |
| Delete and search records based on keyword/ string entry | search\_company()  delete\_company() | 47  162 | search\_company() prompts users to enter a company name, then displays the key/value pairs for the matching company and returns the index value  delete\_company() takes the returned index value from search\_company(), asks if the user wishes to delete that company, then deletes the company’s dictionary from master\_list |
| Sort records in a specific order and display them on the screen | sort\_companies() | 63 | sort\_companies() prompts users to select from three sorting options (Company Name, POC Name, or Revenue) and prints the ordered key/value pairs for each company |
| Select an option on the screen to perform a function | main\_menu()  sort\_companies() | 30  63 | Both main\_menu() and sort\_companies() print options on the screen with instructions for the user to select a corresponding function (e.g., “To ADD a new company, type ADD”) |
| Get a prompt on the screen before an action is performed | delete\_company()  update\_company() | 162  106 | Both delete\_company() and update\_company() prompt the user to confirm update or delete operations on a record by printing the current data and asking for a Y/N input |

This application required the implementation of two novel concepts: the declaration of a *global* variable, and the use of a *lambda* function. The global variable allows the input\_company\_info() function to modify the variable company\_count beyond the local scope of the function (Python Software Foundation, 2022). This enables the application to count how many companies have been added to the directory, instead of having the count reset every time the add\_company() function runs. The lambda function enables sorting a list according to constituent dictionary keys and values by passing the “Company ID” key to the sort method through abstraction, instead of having to generate many layers of code (Burgaud 2018). This makes the sort function easier to read.

1. **Instructions to Execute the Code**

Running *Mennell\_Assignment\_2.py* will execute the run\_app() function. A screenshot follows in **Figure 1**.

Figure : Main menu

Text

Description automatically generated

The application will accept commands regardless of capitalisation by using the .lower() method on user input. The run\_app() method then matches the user input to the corresponding function.

1. **Testing and Validation**

The test plan in Assignment 1 was modified to account for the added sorting function. In addition to the actions described below, each function was tested for error handling, positive and negative responses to confirmation messages, and termination commands.

* 1. **add\_company() Test**

The initial record was successfully input and saved using the following data:

|  |  |
| --- | --- |
| Company Name: | ABC Corp |
| DUNS Number: | 123456789 |
| POC Name: | Joe Bob |
| POC Phone: | +1 (555) 123-7890 |
| POC Email: | joe@abc.com |
| Est Annual Revenue: | 1000000.00 |

To test search and sort functions, the following records were added:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Company Name | DUNS Number | POC Name | POC Phone | POC Email | Est Annual Revenue |
| XYZ Ltd | 100112233 | Aardvark Jones | +1 555 123 1234 | aj@aj.co.uk | 100000.00 |
| Mediocre Co | 200445566 | Molly Middle | 555 456 7890 | molly@mc.com | 500000.00 |
| Delta Group | 100778899 | Pete Triangle | (555) 111-3333 | pete@d.com | 1500.00 |
| Py Company | 200778899 | Zu Smith | +44 1206 873333 | zu@py.co | 9999000.00 |

* 1. **sort\_companies() Test**

**Text, letter

Description automatically generated**Selecting *SORT* from the main menu brings up a following menu that prompts the user with sorting options shown in **Figure 2**.

Figure : Sort menu

Each of the three sorting operations performed correctly, with results following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sorted by: | Result 1 | Result 2 | Result 3 | Result 4 | Result 5 |
| Company (Ascending) | ABC Corp | Delta Group | Mediocre Co | Py Company | XYZ Ltd |
| POC Name (Ascending) | XYZ Ltd (Aardvark Jones) | ABC Corp  (Joe Bob) | Mediocre Co (Molly Middle) | Delta Group (Pete Triangle) | Py Company (Zu Smith) |
| Revenue (Descending) | Py Company (9999000.00) | ABC Corp (1000000.00) | Mediocre Co (500000.00) | XYZ Ltd (100000.00) | Delta Group (1500.00) |

* 1. **search\_company() Test**

Searching for *Py Company* returned the result shown below in **Figure 3**. This test was successfully replicated for each of the five companies.

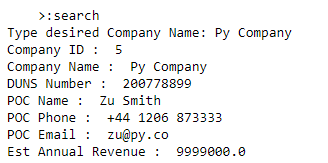


Figure : Search result for Py Company

* 1. **delete\_company() Test**

Instructing the function to delete *Delta Group* printed Delta Group’s information and confirmed the user’s desire to delete that company before deleting.

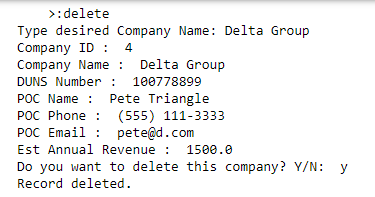


Figure : Delete function operation

* 1. **update\_company() Test**

Instructing the function to update *Mediocre Co* printed Mediocre Co’s information and confirmed the user’s desire to update that company before prompting for updated information.

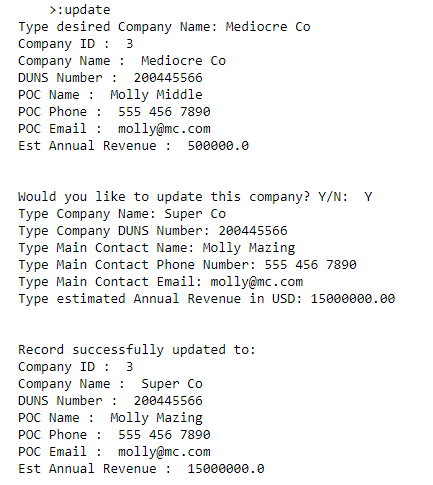


Figure : Update function operation

**References**

Burgaud, A. (2018) *How to Use Python Lambda Functions.* Available from: https://realpython.com/python-lambda/ [Accessed 27 January 2022].

Martin, R. (2008) *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall.

Python Software Foundation. (2022) *Programming FAQ.* Available from: https://docs.python.org/3/faq/programming.html [Accessed 15 January 2022].

Reitz, K. & Schlusser, T. (2016) *The Hitchhiker’s Guide to Python*. USA. O’Reilly Media, Inc. Available from: https://books.google.co.uk/books?id=nHDtDAAAQBAJ&lpg =PP1&pg=PP1#v=onepage&q&f=false [Accessed 27 January 2022].