# Project summary

During the past week, I put all my effort into the Alatest exercise assignment, which is a nice experience full of enrichment and progress. Meanwhile, I found that the actual coding project is much more different from the daily self-learning. Although there were more challenges, it made me progress more quickly.

The first step of the project is to get the knowledge of the requirement. During the coding process, I really know why you recommend me to read and re-read the instructions. To be honest, I wrote several solutions, but I dropped the first two versions. Each time when I re-read the instruction, I came up with a new idea about the project. So, I had to improve the logic of the design of the code.

The second step is the code design of the project. I divided it into two functions, one is to save all of the price lists from each operator, another is to match the telephone with the prefix and find the cheapest price. Followed flow charts demonstrate the main logic of the second function:

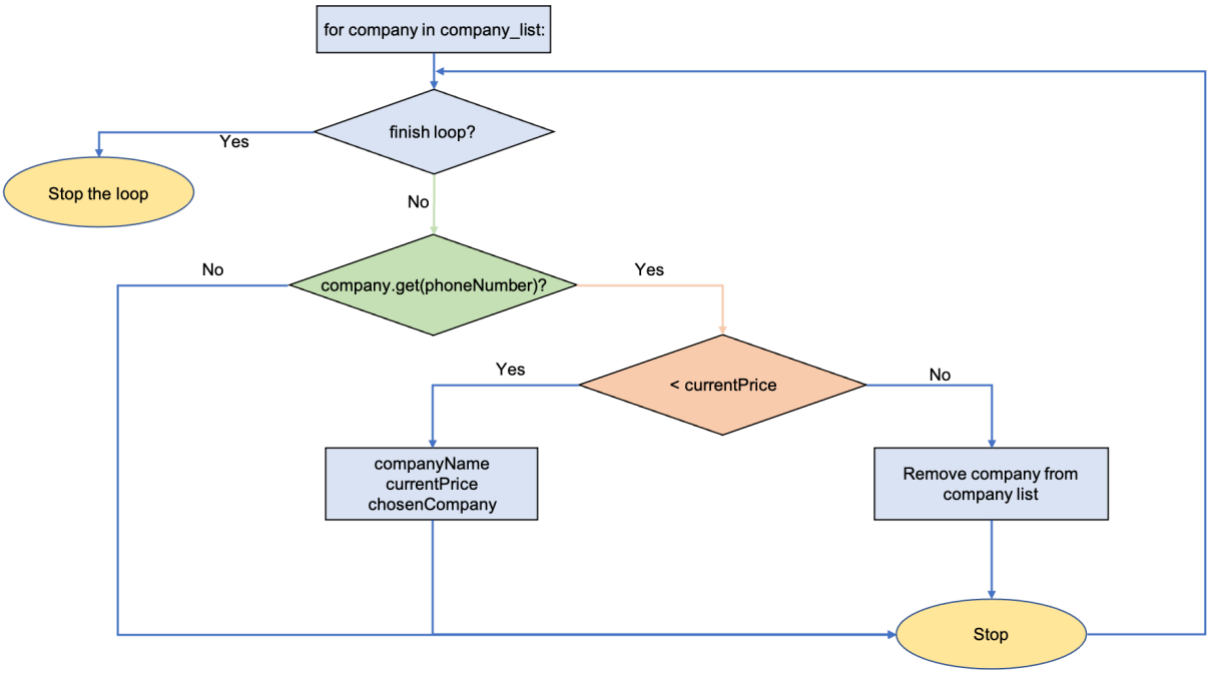


*Figure 1: For loop flow chart*

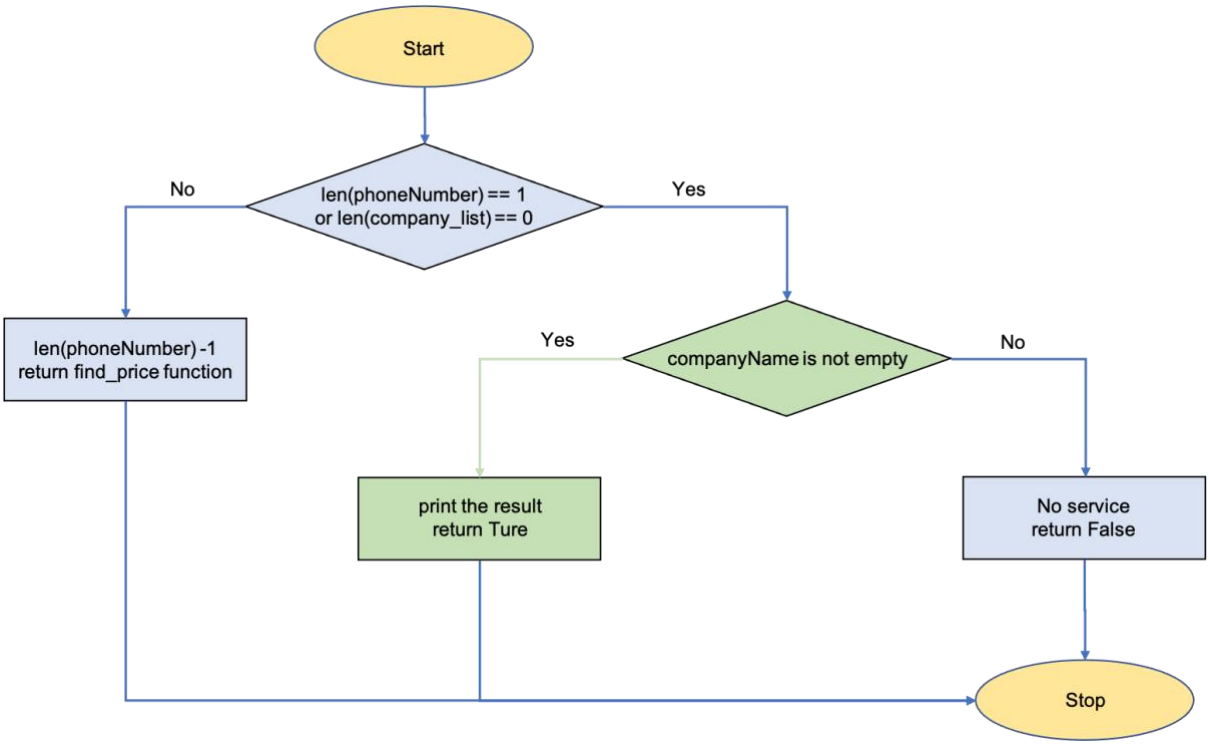


*Figure 2: Nested if else flow chart*

- In the first *get list* function, in order to save all of the prefixes and price from each operator efficiently, I created the dictionary data structure to store the data, which is input from CSV files. In addition, I set the operator's name as the first key value in each dictionary, when we find the matched prefix and price. We can just return the first value of the dictionary to get the name of the operator. In order to prepare for the next function, I also put all of the dictionaries in a list, which is easy to traversal all of the price lists to find the cheapest price. Meanwhile, I also set a special variable to store the *max length* of the prefix, which can avoid the meaningless search loop in the second function.
- In the second *find price* function. The logic of the design is to input three parameters to finish the search: *telephone number*, *operator name*, and *initial price*. The *telephone number* has already been cut into the suitable length before input the function, the operator name is empty, and the *initial price* is 999.
    - During the match, I used a for loop to traversal all of the dictionaries to find the cheapest price. The top priority of the search is from the *max length* of the phone

- If we cannot find the matched prefix this time, then cut the last digit and return the function itself to continue the search.
- If we can find a matched prefix, the first time the corresponding price must be smaller and the *initial price*. Then we can save the price and the name of the operator. Meanwhile, we also put the whole price list into a new dictionary. The next step is that if we have already found a price in an operator, we need to remove it from the dictionary list, because the introduction aims that the match follows the longest prefix, but the operator name and price are saved in the variables. During the traversal, if we find another matched operator, we need to compare the prices and remove the expensive one from the dictionary list.
- When the length of the telephone has already been cut to one digit, or there is no price dictionary in the list. The match is almost finished, if we saved an operator name, then output the cheapest price and the operator name. if not, there is no service about this telephone among the operators.

The third step is to test the code. I set different inputs and check the output is the same as my expectation or not. Actually, it is the first time for me to hear the test unit. I found some online tutorials to learn, and then write the test file. However, because I am not proficient in *unittest* module in Python, there is something wrong when I ran the test file. I think the test of the first function works well, but the second function is not ok. I have already cut the end-to-end test into slides by saving each function as a new module and import them into a new file to do the test by *unittest* module. But I do not know why, when I ran the test file, the original file will be running as well automatically at the same time. Specifically, I have already set the special value of the *telephone number*, which will traversal all of the codes and all situations, such as different if and else. During the test, it let me input the t*elephone number* again, which made me very confused. I think this is a factor that cause the failure of the second function test.

The next step is to optimize the code structure by PEP 8, which makes it easy to read. I changed the name of the function into snake case style, and add the blank lines and whitespace. However, as for the name of the variable, I think the set the first digit of

the second word into uppercase is easier to read and the lowercase style, I have no idea about why PEP8 recommend me to put all of them into lowercase.

Last but not least, I think there are also improvements to my code and logic. But I have no time to achieve them before the deadline.

- The first improvement point is in the get list function. When I list all of the price lists into the dictionary, it will be more formal and logical if I use the nested dictionary and achieve the 2D data structure. In the current version, I used a tricky solution to save the operation name.
- The second improvement point is the search method in the second function. My method needs to traversal all of the prefixes by for loop. When I searched on the website, I found other search methods which I think it will decrease the calculated magnitude and make the process more efficient.
  Such as the depth-first search: https://en.m.wikipedia.org/wiki/Depth-first_search, I think my current method is breadth-first search, and it will be more logical and suitable for the assignment if I change the search logic.
  Another one is the binary tree: https://en.m.wikipedia.org/wiki/Binary_tree, if I can use it to classify the prefix before the search, when I compare the first digit of the phone number, if will have an accurate location of the prefix, which I do not need to traversal all of the prefixes. The algorithm must be more efficient than the current one.
- The third improvement point is the search direction, the direction of the current method is from end to front, I have no idea about from front to end method. When I check the price list, the majority of the prefix is short, so I think to change the direction maybe increase the calculation efficiency.
- The other improvement points are about the usage of new search modules, I got the knowledge of some new modules in Python, such as *gasket* function, *pandas* module, *startswith* function, and so on. I have no time to achieve them one by one. So, I have no idea which one is the best.

This is a good experience for me to do a real small project, I also realize there is a big space for me to improve myself. I would like to see your comments and feedback, and if any suggestions can be given, it will be really appreciated!

Reference list


1. Read the CSV files to python relevant code example:
   https://realpython.com/python-csv/
   https://www.geeksforgeeks.org/python-os-path-splitext-method/
   https://docs.python.org/3/tutorial/datastructures.html
2. Find the match in dictionary:
   https://www.w3schools.com/python/ref_dictionary_get.asp
3. Write the unit tests:
   https://www.youtube.com/watch?v=6tNS--WetLI&t=442s
4. Individual Guihub: https://github.com/zitong666