Before I start making this free hand calculator I made a non GUI calculator. This other calculator s functionality was very limited and its flow chart couldn’t be more simple and straightforward. It is as follows:

1 –The user is prompted for an input of a number

2 –The user inputs the first number

3 –The user is prompted for an input for an operation. Example: +, -, \*, /

4 –The user gives an input for the operation

5 –The user is prompted for the input of the second number

6 –The user gives the second number

7 –The user gets the result printed out and is asked if he wants to do another operation

I am using a programming style called Ad-hoc. It is when I figure out what I have to do only after I have made the previous step. For my level of programming skills I find it more convenient. In the beginning I made an approximate plan of what I have to do step by step. With each step I figured that the next step can’t be the same as what I had written before. That’s why I started programming in this Ad-hoc programming style.

For example:

My first idea was to make a function called ‘char\_correc’ which basically compares the given string with a list of allowed characters. If the given string contains even one symbol that is not allowed then it returns false. If all symbols are contained in the list of allowed characters then return true.

After that function I started thinking what other essential function do I need in order to recognize the user input as a mathematical expression. The other function is called ‘check\_bal\_parenthesis’ and it determines whether we have used the parenthesis symbol correctly. It does it in 3 steps described in the comments inside the code of the application.

Then I created the function that checks whether two operators repeat after one another. For example we can’t have +- or \*/ or ++ etc.

After that I created another function called check\_symbols\_around\_pars which checks whether a correct symbol is used around a parenthesis in our expression. It makes sure that the given expression is not something like (1+)2 -1(blala). In other words infront open par we can only have another open par, an interval or all the operators. And for the other cases is vise versa. The function uses 3 arguments. Based on the 2nd and 3rd arguments we determine which case we examine.

By this moment the development of the program was within 1 file called development. After that I decided that I wanted to implement a more divided approach with all essential functions divided into different modules. At that point I created the main\_test.py where I tested the program functionality. All new functions were made inside a different module. All tests made inside the main\_test