Scientific Calculator With Python 3

Project Final Phase

Professor(s):	Prof. Dr. Bart Jansen	
	Dr. Evgenia Papavasileiou	
Instructors:	Dr. Redona Brahimetaj	
	Dr. Taylor Frantz	
Team Members:	Mr. Hossein Dehghanipour (Team Leader)	0582897
	Mrs. Mahsa Naghedinia	0586285
	Mr. Iliya Hakani	0584989
	Mrs. Mahsa Alirezaee	0574173

January 2022

Contents

1 Introduction

The task that we are working on as our project is implementing a scientific calculator. This calculator contains several arithmetic and trigonometric operations. As a team with a strong mathematical background we found this project the suitable one to develop and expand. The implementation of this project is split into front-end and back-end. For the back-end part we use Python Programming Language and for the front-end part we use Python's Tkinter library.

1.1 Description

For work organization, From the beginning, we went through group meetings via the "Teams" interface. These meetings were held regularly in order to check the progress and update the tasks. In the first meeting which took place after subject confirmation, firstly we brainstormed and shared our ideas about the appearance of the calculator and respectively the object-oriented python code design of the project. All members had worked in python and OOP python was covered during the course, but Tkinter was new to some, so all members were given a 7-day chance to study Tkinter for front-end and design a small calculator to compare and conclude about the initial look of the calculator. We also created a git repository in which we could work and share and update our code. During the meetings, The work progress was checked and updated in the git environment. The tasks in both front-end and back-end were allocated equally to the team members. The workload details have been explained in the following report.

1.2 How To Run The Code

To run the code, you need to open and run the "main" file in Code directory.

1.3 Needed Packages

- Tkinter version 8.6 or Higher

1.4 Work division

Hossein Dehghanipour	Back End, Fixing bugs in Both Back and Front, Writing Report
Mahsa Naghedinia	Front End, Fixing bugs in Both Back and Front, Writing Report
Mahsa Alirezaee	Front End, Writing Report
Iliya Hakani	Front End, Writing Report

2 Front End

For the front-end, we decided to implement the Graphical User Interface of the project, using the Tkinter library. The fact that we made the decision of using this library out of other Python GUI libraries is for two explicit reasons: :

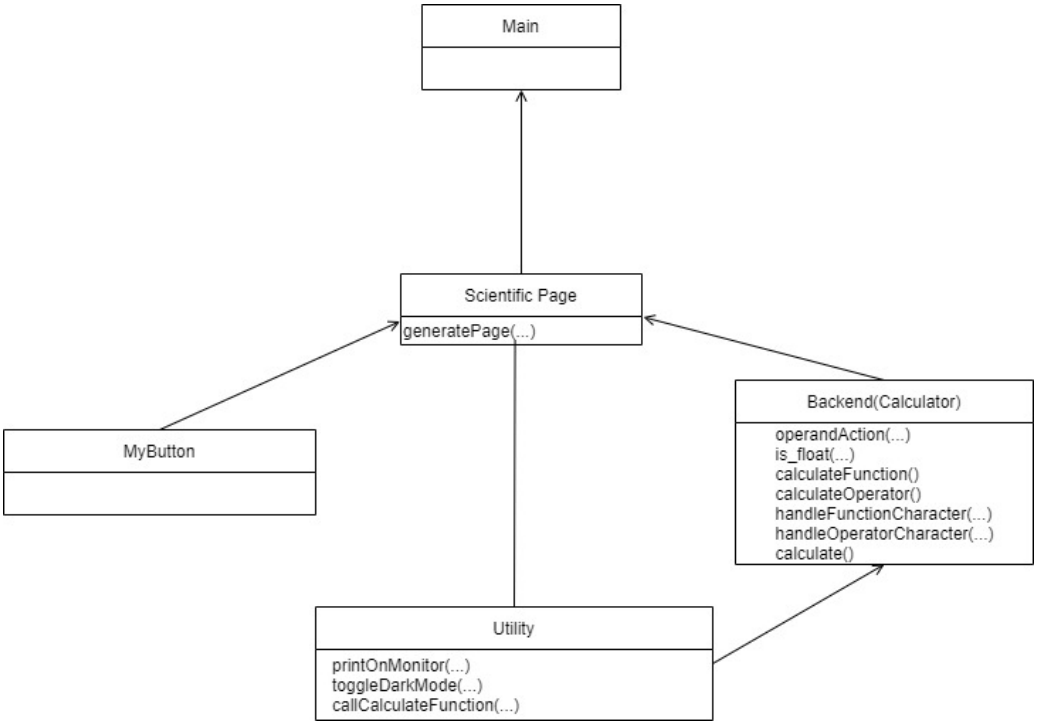
- Tkinter provides strong geometry managers such as “grid” which divides the window in even rows and columns and therefore is perfect for a calculator design.

- Another important factor was that Tkinter is one of the easy libraries in Python to use and work with; and since we were all beginners in front-end coding, we found this library to be easier to master and use.

Working with a new library, the first task for the whole team was to get familiar with the different functions and parameters of Tkinter. This step was challenging, since we did not know many features that the library offered, and all of our attempts were mostly faced with an error. Learning how to solve those errors got us familiar enough and enhanced our knowledge of the library.
In general, we succeeded in giving our program a beautiful look, along with practical features. The front-end part consists of the main window which is divided by the “grid” feature of Tkinter, the main label, and the buttons. Basically, all of the buttons are connected to one “PrintOnMonitor” function except for the equal button(=). The equal button is when we get the expression on the main label and send it to the back-end part to handle and return a result. Error handling is done by using the “try” and “except” in the front-end to avoid running into any unexpected out-of-nowhere errors while running the program. The function buttons can be done on any inputs and expressions the user enters. If an expression is being passed to a function, the code starts handling the answer from the most inner expression, and considering the priorities, it will give us the final result. We tried to have the highest user experience for this program. Meaning a user must not have any difficulties while working with the program. All the features for keeping the highest user experience have been compared to the Microsoft Windows calculator, in order to learn and improve our program.
In the following, Front-End classes are going to be explained.

2.1 UML Diagram

After getting used to its different features, the process of the UML diagram illustration for our project, identification of different classes and their connection to each other was begun. Said diagram is as follows:



As shown on the diagram, so far we have 4 classes for our program. The three classes which are related to front-end are “Button”, “ScientificPage” and “Utility”. We have a “Button” class which is the definition of the buttons that we use in the program. This way not only it eases the process of modifying and expanding the program, but also it keeps our program object-oriented. In the button class we have different parameters like width, height, background color, font size and features of all of our buttons.

2.2 Button Class

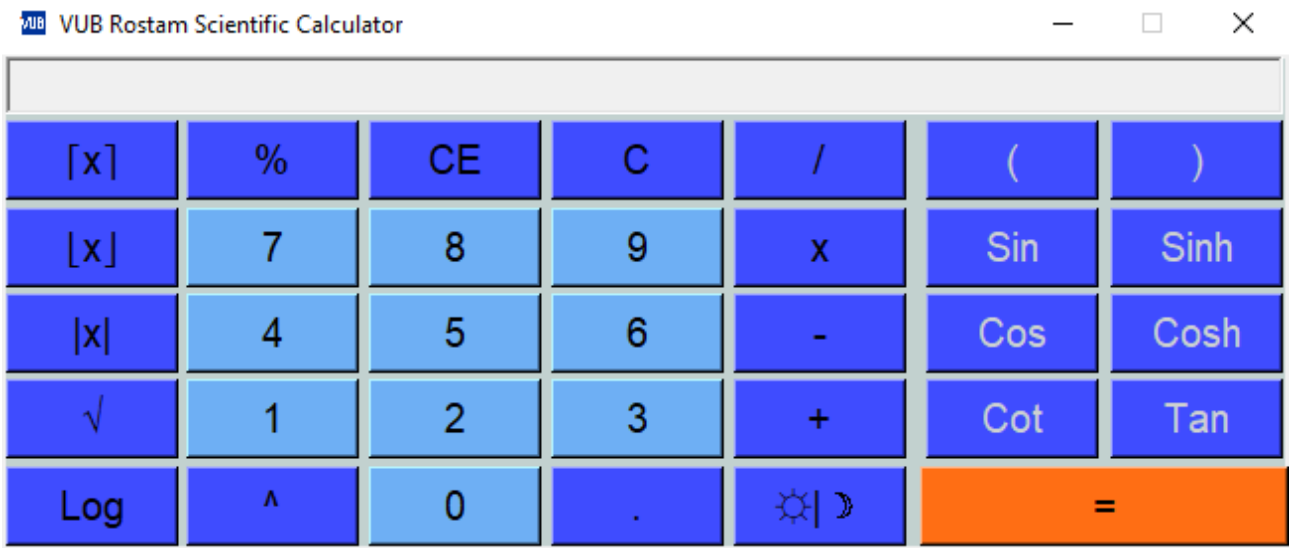
We have a “Button” class which is the definition of the buttons that we use in the program. This way not only eases the process of modifying and expanding the program but also keeps our program object-oriented. In the button class, we have different parameters like width, height, background color, font size, and features of all of our buttons. This class also handles buttons’ background color change with mouse hover, by binding the movements to a color-change function.

2.3 Scientific Page Class

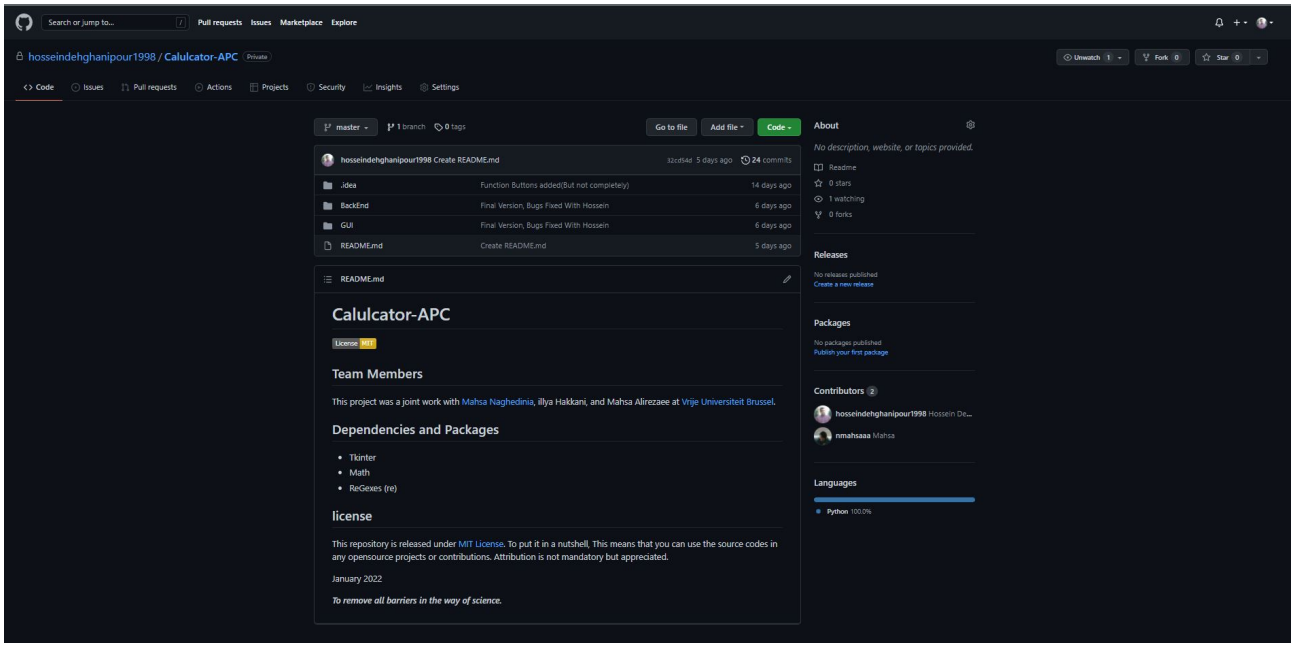
Since we wanted to keep the possibility for our program to contain different pages for different purposes, we defined the “ScientificPage” class. This way, if we ever decide to expand our program and have a Mathematical Page, for instance, we could easily define other classes so that we are able to design each page considering its purpose and users’ needs. “ScientificPage” basically is the main design of our form. In this class, we generate the labels and buttons with the help of our “Button” class. Also connecting each button to its function in the back-end is done in this class.

2.4 Utility Class

The main front-end class is the Utility class. In this class we handle button pushes, meaning every buttons' command is linked to one of the functions in this class. There are two functions. One is "printOnMonitor" which is linked to every button except for the equal button (=). This function does not call the back-end part, it only prints the value of each button on the screen(main label). The second function is "callCalculatorFunction" which only has the equal button assigned to it because it's only the equal button that needs the call for the back-end part. In all of these functions , "try" and "except" expressions are used properly to avoid any unwanted and errors while the user is working with our program.



Calculator User Interface



Mahsa and Hossein git repository

3 Back End

3.1 Methods

There are quite many challenges in this part. The most important of them is considering the Precedence of the operators in the calculation. The mathematical expression that we pass to a calculator is in a simple form that is readable and easily understandable by humans, which is called Infix. The computer, on the other hand, does not understand the same as humans do (obviously :D), therefore, we have to either teach it to interpret by itself (by designing a learnable model by using NNs 4 or by writing algorithms, like what we did here). In order to fix this problem, we can either use simple iterators to iterate over our expression and read the operators and operands one by one and try to understand them (which adds more complexity to our algorithm and is not suggested) or by converting the infix expression to either Prefix or Postfix. Converting the infix expression to postfix makes our job even easier to handle.

3.2 What We Used

There is also a third way that we used here: we did the exact operations the same as the ones we would do to convert an infix expression to postfix but instead of making a postfix expression and then iterating on it again, we calculated the outputs of each part while creating the postfix expression. You may ask why? Because it reduces the complexity of $\Theta(n)$ of iterating over out the whole expression again and starting to calculate. Our biggest challenge in calculating the results was handling the parentheses and mathematical functions.

3.3 Our Improvement and Additions Since Intermediate Upload

Now, our calculator is able to calculate any nested functions such as “cos(sin(5+2) + 9)” and also it has become more robust and user-friendly. We also used regular expressions to detect float numbers. The major improvement we had in the back end was handling nested operations and considering their preferences and higher priorities. It can now support the important functions in the Microsoft Calculator. Besides, the back end can determine whether the user has entered an extra operator or an empty function (error detection) and reports it back to the Front End in order to show a proper message to the user. The code has become more readable and using OOP has made it more robust and encapsulated. The code maintenance has also become easier, which allows us to add more functionality in the future if needed. To put it in a nutshell, the possibility of giving a false negative (wrong answer) is zero. We, as a team, held weekly meetings in order to debug the code and discuss more possibilities of adding new features. We also used a private Github repository to sync our code from the front end to the back end. Here is a brief list of mathematical operations that the back-end currently supports: ”+”, ”-”,”/”,”*”,”%” , ”powerFunction”, ”sin”,”cos”, ”tan”,”ctg”, ”ceil”, ”floor”, ”sqrt”, ”log”, ”exp”, ”cosh”, ”sinh”. In order to keep this report short, we refrained from bringing any code from the back-end here. All the codes are well-commented, therefore, feel free to go through them.