Aaron Galonis

Advanced Python Final Project Journal

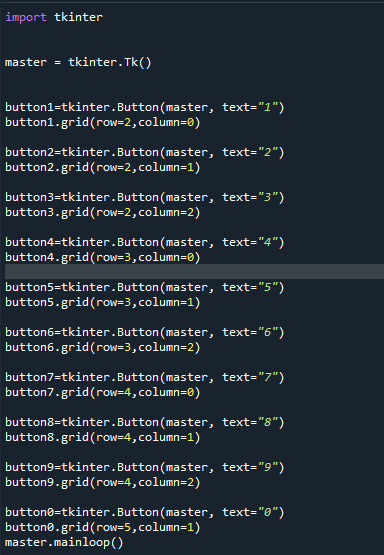
**4/1/2021**

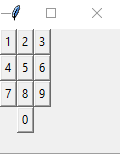
Did some research on how I would go about creating a GUI in python and what are the best ways to do so. Tkinter was one of the most recommended GUI application builders so I settled on that. I then created my first version of my calculator. This one didn’t have any math operator buttons and only had numbers; it was purely to get myself familiar with the grid method for button placement in tkinter.

I’ve been using this website mostly for now for help with buttons.

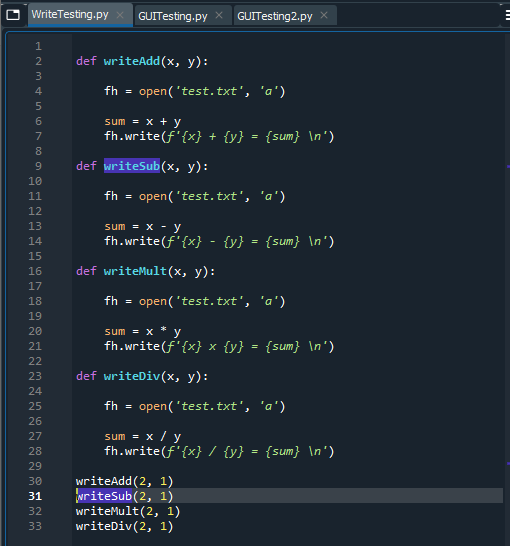
<https://www.python-course.eu/tkinter_buttons.php>

I had a few issues figuring out how to run tkinter, some of the ways I should go about placing buttons and how to create them. My biggest issues came with getting tkinter installed and working for importation but one I got that it didn’t take me long to create this.

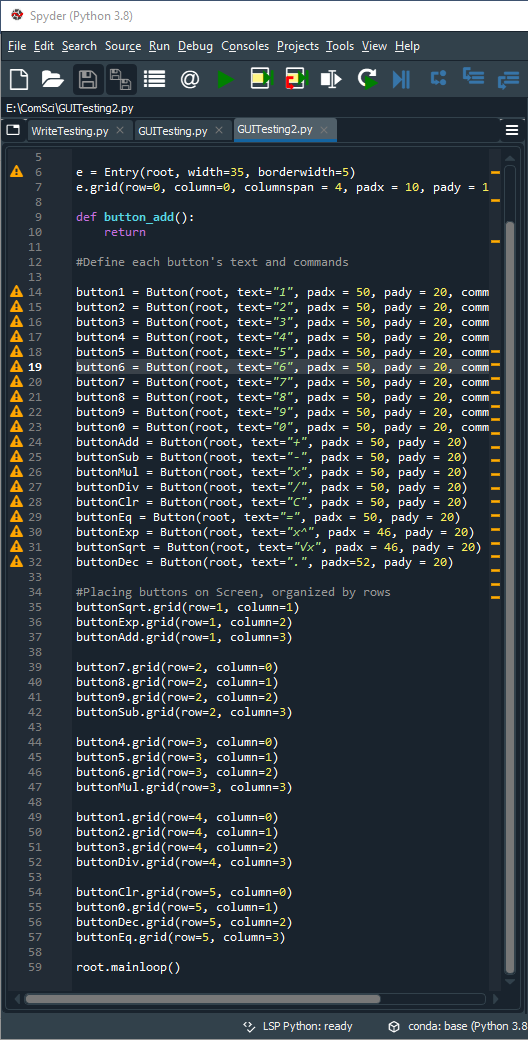


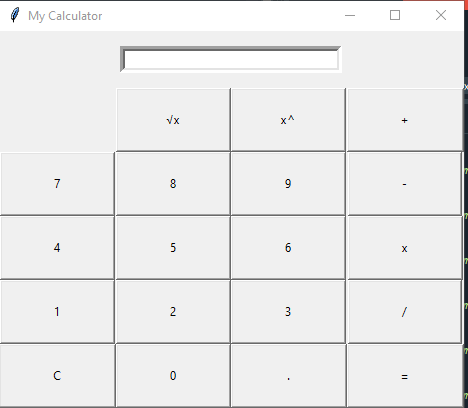


**4/3/2021**

 Created my initial methods that I’d like to use to help add and subtract numbers. I’m unsure if it’ll work with button commands or how I’ll implement them into my code but so far the methods work. I also managed to implement file writing when passing the method in python so that history can be stored in some way for now.

**4/8/2021**

 Created a much more organized and complete looking calculator using tkinter. It took me a lot of planning to figure out each buttons positioning I liked on the grid. Also made a title for the calculator window but for now it is just “My Calculator”. I’ve added commands that don’t really do anything to the number buttons in hopes that all of the commands I use for those buttons will be the same. I also had issues figuring out the correct spacing on my buttons, especially using the padx and pady, which help make the buttons bigger. Some of the symbols required different lengths. I also added a text box at the top that spans 4 columns.



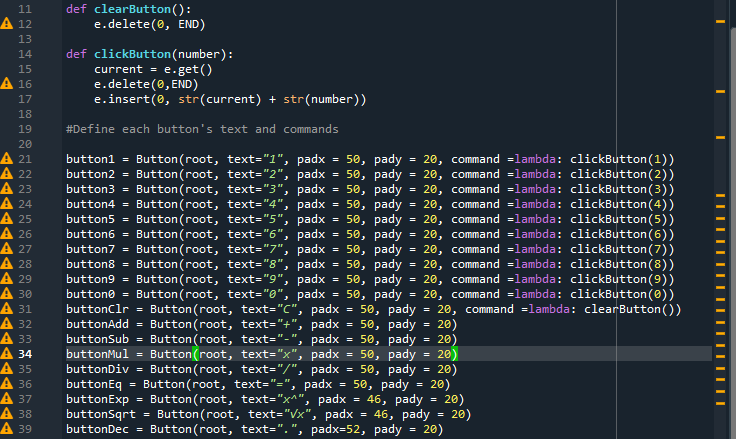
**4/11/2021**

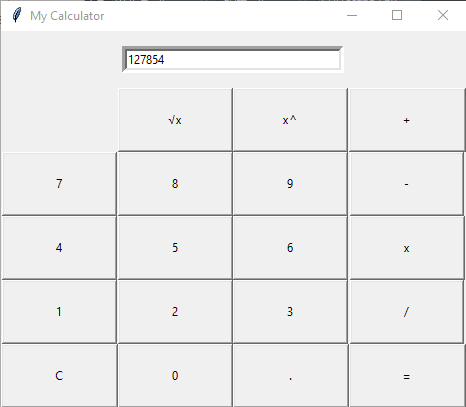
I worked on creating some commands for my calculator, primarily focusing on the number buttons for now since I believe I can use one command for all of the buttons. My biggest issue here was that after I did some research I realize it can become very tricky to pass arguments in button commands. With a fair amount of research going into this, it seems that most people use the lambda functions to help pass arguments to commands with tkinter.

This was the website/question that helped point me in the direction of using lambda:

<https://stackoverflow.com/questions/6920302/how-to-pass-arguments-to-a-button-command-in-tkinter>

Even though I had found some information about this online, it still took some troubleshooting to understand how arguments, methods, commands and parameters all worked in unison with lambda and tkinter buttons. I eventually created the clickButton method, with parameter number that allows me to click on numbers and as they’re clicked they are shown in the calculator. While that worked smoothly, it automatically concatenates the numbers together in the order they’re pressed which calculators do not do. So I created the method so that the numbers appear correctly as a calculator should. I also created the clearButton method which allows the text area on the calculator to be cleared when the “C” button is pressed. I’m thinking about creating a different file for my methods.



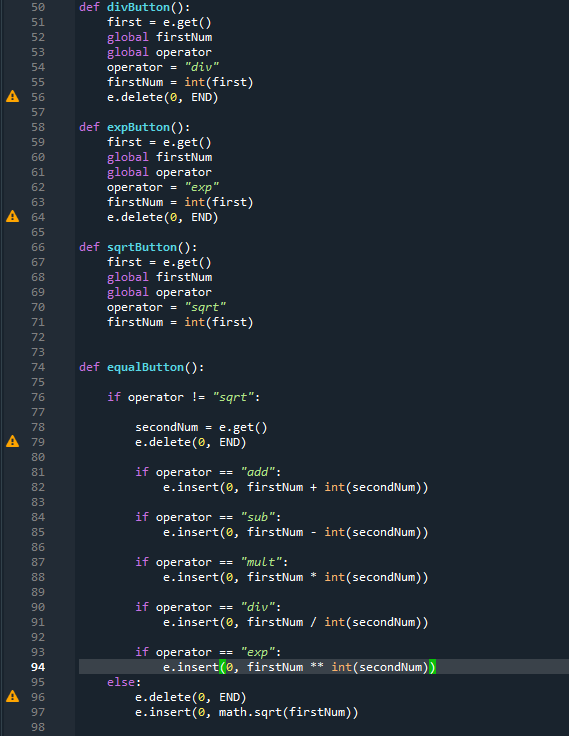


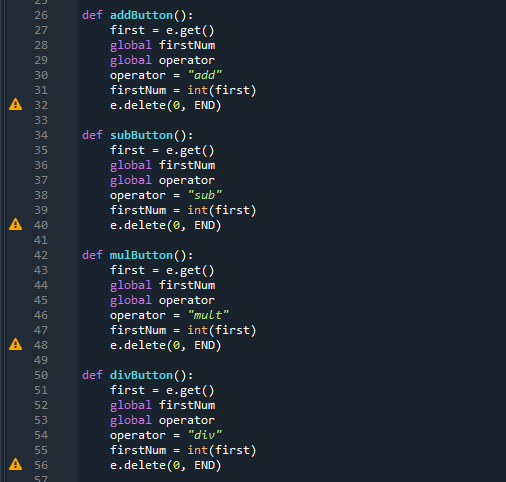
**4/14/2021**

Today I worked on getting all of my math operators to work on my calculator. I first focused on creating addition and subtraction which already proved to be kind of difficult since I was unsure of how I would help the calculator realize which operator to use. I couldn’t put the actual calculations into the add/subtract methods and instead made it so that the calculations are made when the equal sign is pressed. It took some time but I realized that global variables would be my simplest solution to this answer. I created firstNum and operator variables.

The firstNum variable stores the number that is in the calculator textbox when the operator is pressed. Then the operator variable sets the variable to a string based on what button is pressed. Then in the equalButton method, it tests for what operator is necessary selected and does the calculations based on that.

I had a few errors creating this, especially with the global variables as I had many errors that they weren’t defined. I eventually realized that the global variables needed to be called in every function correctly.

Lastly, the square root button had to be treated a little differently. Since square roots do not need a second number, we don’t t need to call delete, and rather than call delete after every operator test, I decided to test for the square root button first and if it’s not selected then proceed with every other operator test.

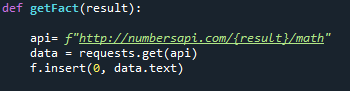
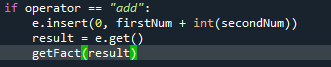


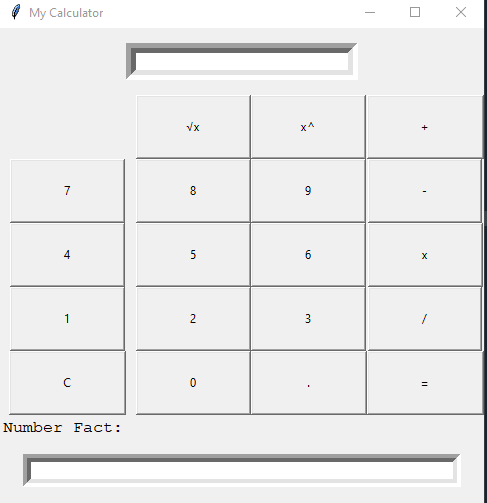
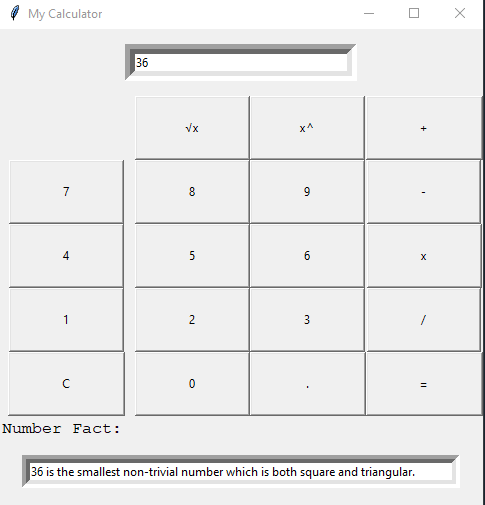
**4/19/2021**

Today I focused mostly on getting my API implemented. I had originally planned to use Math Tools Numbers REST API (found here: <https://api.math.tools>), but was unable to due to the apikey. That said, I had found another API that allows me to accomplish similar things that I had intended to accomplish before. The API is the numbersAPI (found here: http://numbersapi.com#42). My new plan is to use this to draw random facts about a number and display that fact to the user. I will then test for my prime numbers using my own method I’ve created.

Currently, I only have the API implemented roughly. Now, when the equal sign is pressed, a fact will display in a text box underneath all of the numbers on the calculator. There is also some text above displaying that it is showing the facts. I first created a method that takes the result of the equation, keeps it a string and then uses fstrings to append it to the URL of the API.

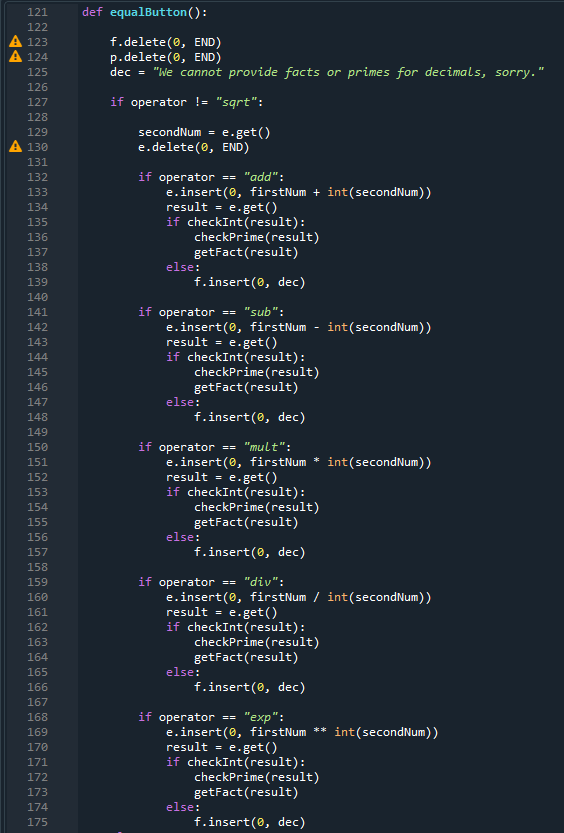
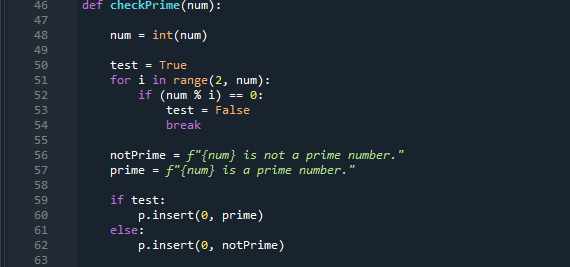
I had a lot of problems getting this to work. Firstly, I was unsure of how to retrieve the information from the API that I wanted. I eventually solved this problem by storing the information given in a variable called data and then inserting that into my text box using data.txt which shows the plain text for the API, which is what we want now. I also was unable to figure out how to use the API for numbers that use decimals and was getting an error that stopped the calculator a lot. This is because the website cannot have another decimal/period in its URL than it already has. Unfortunately, the sqrt method causes the result to always be a decimal. I’m not sure how to fix this yet.

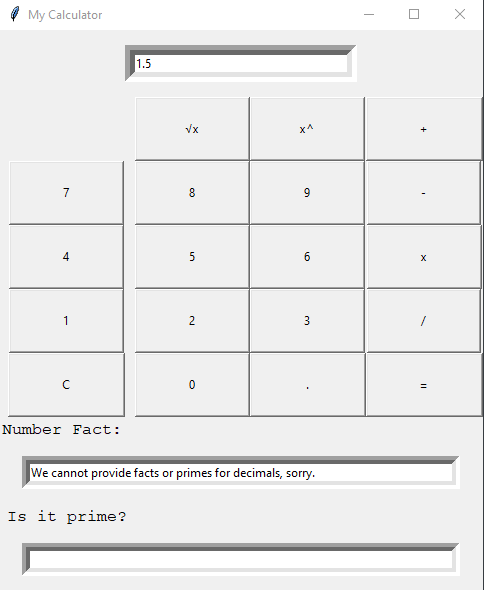


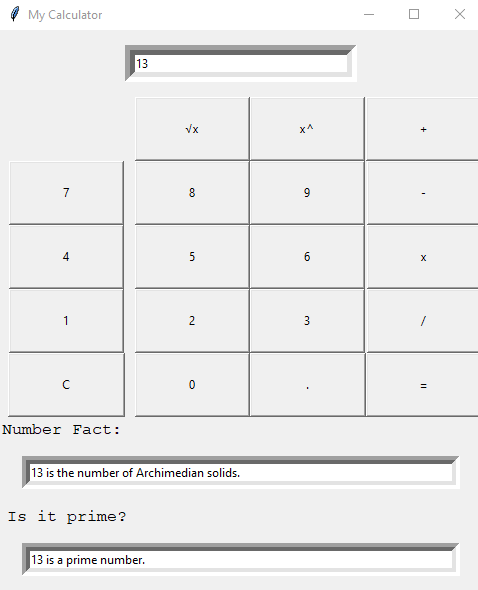


**4/22/2021**

I started today off by deciding that I was going to find a way to allow the calculator to display facts if it’s an integer, and writing an error message in the fact box if a decimal is displayed. I did this by creating the checkInt method. This method traverses a string and then checks if there is a period/decimal in the string. I did it this way because when using the .get method for a text box in tkinter it becomes a string. I had attempted to use some other methods when doing this but many problems came up when trying to convert between strings, integers and decimals so I decided to do it this way as it works well for my program.

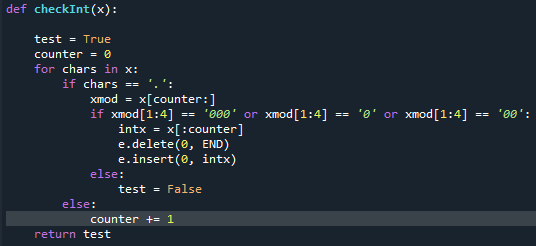
 I then created the last text box for my calculator, which I was actually unsure of if I wanted to make. I almost put my number fact text box to the right of the calculator but once I decided to do 2 more I put them underneath since the calculator application is not supposed to be a cluttered app. The last text box is for the prime numbers. I first created a method that checks if a number is prime or not. Then, similarly to the getfact methods, I put it into my equalButton method and whenever the equal button is pressed, it should display a fact and a prime. If it is not an integer, nothing is entered into the prime text box and I changed the number fact message to display that a prime number couldn’t be found from a decimal either.



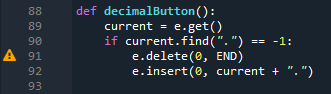


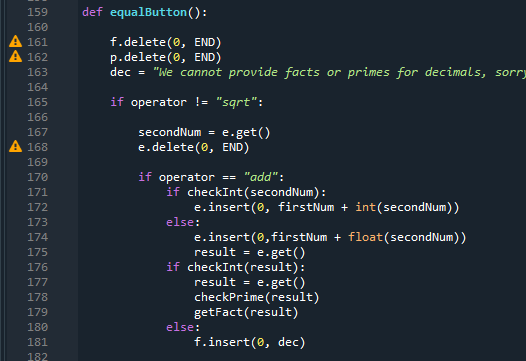
**4/23/2021**

I’m now nearing the end of my project, but I had a few different bugs and kinks to work out before I can add my last thing. The first thing I did today was round down any decimals that had more than 3 zeros after the number to integers. This is so that I can provide number facts for more numbers along with prime numbers. The logic behind creating this is what took me the longest time and gave me many errors.

 It involved multiple variables and traversing the string correctly but now that I’ve gotten it to work I can get simple math such as division and square roots to always show as integers unless they are truly decimals.

I also had an issue where pressing a number and an operator after pressing the C button, the program would still think I had an old number stored and goes on to perform a calculation even though the button was cleared. I fixed this by just clearing my global variable firstNum.

Lastly, I created my decimalButton method, but this created a whole new array of issues since until now my code was always trying to put the strings into integers. To counter this I had checkInt all of my operator buttons, along with the math itself in the equals button.



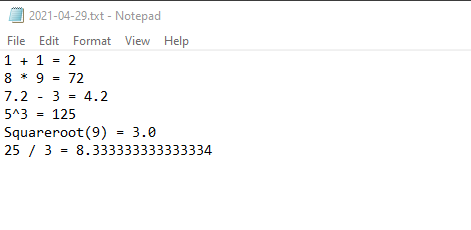
**4/28/2021**

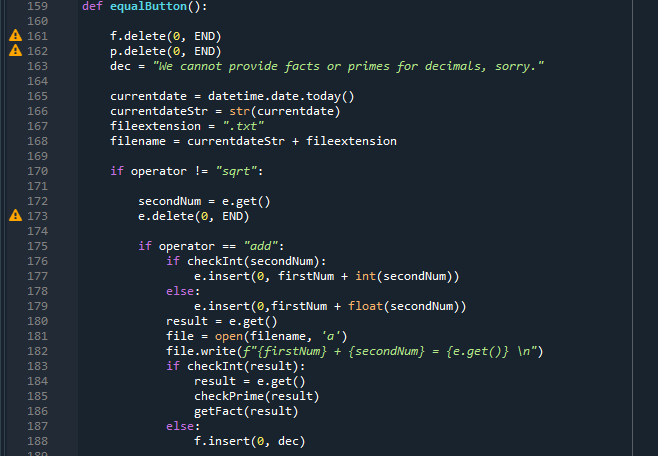
Today I added functionality of writing to a text file as calculations are made. I’ve already done multiple python programs to do this so I wasn’t struggling too much in figuring out how I wanted to do it. The biggest issue was making the code as minimal as possible and also making the file have the name of the date. I also had issues with getting the calculations on different lines but realized I could fix that by just adding \n to the end of my strings.

As a brief note, the txt file is just created in the directory the file is in.

I’ve also decided to not prompt the user to ask them to make a file and instead just create/open the file when the equal sign is pressed, it’s easier this way, saves time for the user who may not want to be prompted and .txt files take up very little space.

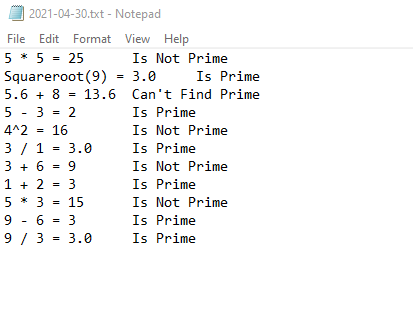
Lastly, I’ve also decided to not include the API facts in the text file as they can vary way too often and length and I believe it’d look too cluttered for me to include it there.





**4/30/21**

For my last day of the project I’ve decided to touch up my text file inputs and include whether the result is prime in the text file. At first, I tried doing this by just adding a line into my checkPrime method but I couldn’t get this to work due to the order of my code. While I may have been able to rearrange it, I came up with another idea and changed up my equalButton method. At first, I had created more conditionals after displaying all of the data on the calculator, these conditionals tested checkPrime, which I added a return True/False to. Once they test, they add a line in the file that tells you if it’s prime. Unfortunately, with my method of checkPrime, I have it so it displays the prime in the calculator everytime its called, so I was having my prime label repeat the prime statement multiple times. Eventually, I rearranged my code and allowed for only one checkPrime to be called each time the equal button is pressed. Now, when calculations are called, they look like this in the text file.



I feel as this is helpful information to display in the file in the event you don’t want to restart the calculator.

 Lastly, I’ve also made the calculator just look a little better, I kind of went with colors I like since I’m mostly going to be using it but the colors can be easily switched. I also changed the lengths of the buttons a little so that there’s not gaps between them but getting all of them the same size was really difficult due to how grids work in tkinter.

**Conclusions:**

This project took me quite a lot of work to put together due to all the research needed, I was unfamiliar with all of the modules and some of the concepts that I needed to know in order to complete it. It was fun to be challenged like this and must do my own research and find my own solutions to the problems I was having.

I am extremely happy with how the calculator turned out, it is actually more sophisticated than I thought it would be and so far since completion I haven’t run into any errors. I was not expecting everything to work so well when creating the text files and pulling information from the API, but I was very surprised.

Of course, there are a few bugs that I have found that I think do not really hinder anything, such as the decimals still not being able to get facts. Unfortunately, there is no way around this since the API I have does not allow for this. Also, sometimes the facts that I have pulled from the API are just a little too long for the text box. I have decided that this is okay because if I were to make the box any longer, I do not think the calculator would look as neat. Also, the same fact is not always displayed so the same number will not always leave the text box.

Lastly, I could not get my calculator to run in google colab, which is what I generally use for getting .pynb files. That said, I will upload both the .pynb and .py file with my submission.

Thanks for this project, I enjoyed the time I spent on it.