

Macquaire Fields High school

Software design and development

Year 11 Assessment task no.2

Mr Pingett – Bob He And Xiaotian(Rickey) Lin

Contents

[Project definition 2](#_Toc17140892)

[Planning: 2](#_Toc17140893)

[Gantt chart 2](#_Toc17140894)

[Logbook 2](#_Toc17140895)

[Context Diagram 3](#_Toc17140896)

[Data Flow Diagram 4](#_Toc17140897)

[Structure chart 4](#_Toc17140898)

[Data Dictionary 5](#_Toc17140899)

[IPO 6](#_Toc17140900)

[Pseudocode 6](#_Toc17140901)

[Flowchart 8](#_Toc17140902)

[Deskcheck with appropriate Data Set 11](#_Toc17140903)

[Storyboard of the GUI 12](#_Toc17140904)

[Python codes 13](#_Toc17140905)

[Prototype version without Graphical user interface 13](#_Toc17140906)

[Version with Graphical user interface 14](#_Toc17140907)

[Error correction techniques 17](#_Toc17140908)

[Differences between syntax, runtime and logic errors 17](#_Toc17140909)

[When would I use stubs and flags? 17](#_Toc17140910)

[Test Python code 18](#_Toc17140911)

[Desk checking of the Python code (GUI) 18](#_Toc17140912)

[Library of codes 19](#_Toc17140913)

[Social Issues 19](#_Toc17140914)

[User’s Manual 21](#_Toc17140915)

[Self-Evaluation 21](#_Toc17140916)

[References 22](#_Toc17140917)

# Project definition

General outlines of project: create a program that can solve arithmetic and geometric sums problems through the use of a Graphical User Interface. With an option to choose which type of sum the user would like to solve.

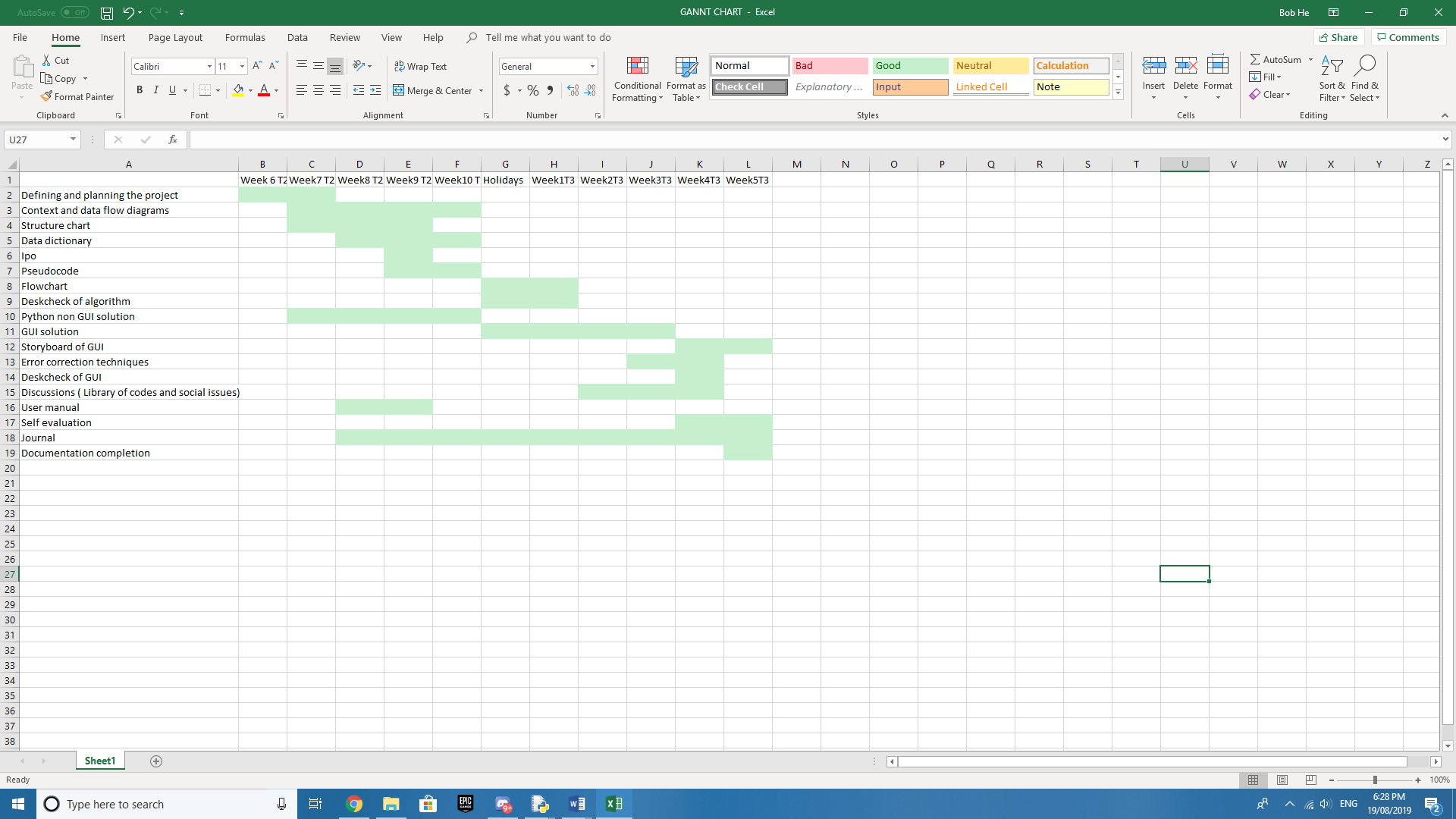
**Objectives:**

Planning:

* Use a Gant chart and Logbook to record the work done over time
* Create a context diagram to illustrate how the user will interact with software to provide a solution to either Arithmetic or Geometric sums
* Create a data flow diagram to show how the program works with enhanced detail
* Create a structure chart to further enhance illustration to show interactions between the user, hardware and software.

# Planning:

## Gantt chart



## Logbook

31/7/2019

Bob completed some coding for arithmetic and geometric sums (code planning).

Rickey worked on creating the Documentations needed for the project:

- Project definition

- Gantt Chart

- Logbook

- Context and data flow diagram

- Structure chart and Data dictionary

1/8/19

Bob did more code and Rickey worked on project definition.

2/8/19

Bob was away. Rickey worked continued on the project definition.

6/8/19

Bob continued to improve the code with Rickey, and then Rickey continued to

complete the Documentation.

7/8/19

Bob added exemptions to the code and helped Rickey to understand how they

worked as Rickey created the structure chart and completed the data dictionary.

8/8/19

Bob developed the Graphical User Interface and Rickey began creating the Gannt

chart and updated both the data dictionary and data flow diagram

14/8/19

Bob started to develop exemptions for the Graphical User Interface and Rickey began working on the documentation.

15/8/19

Bob continued to work on the exemptions on the Graphical User Interface. It proved to be more complex than originally planned.

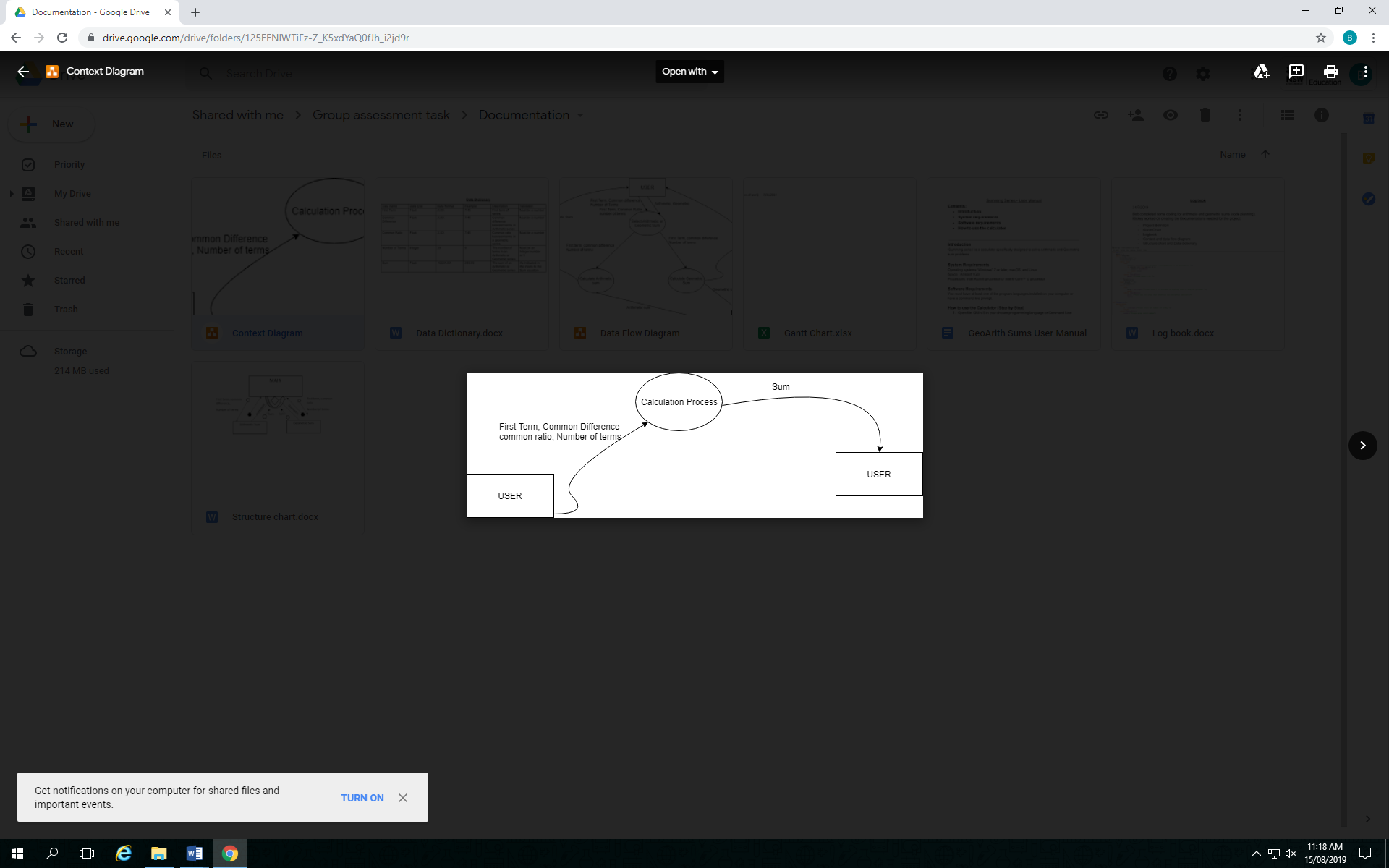
16/8/19

Bob finished the Graphical user interfacing and is working on it to be an executable file. Rickey finalised on the discussion areas in the documentation.

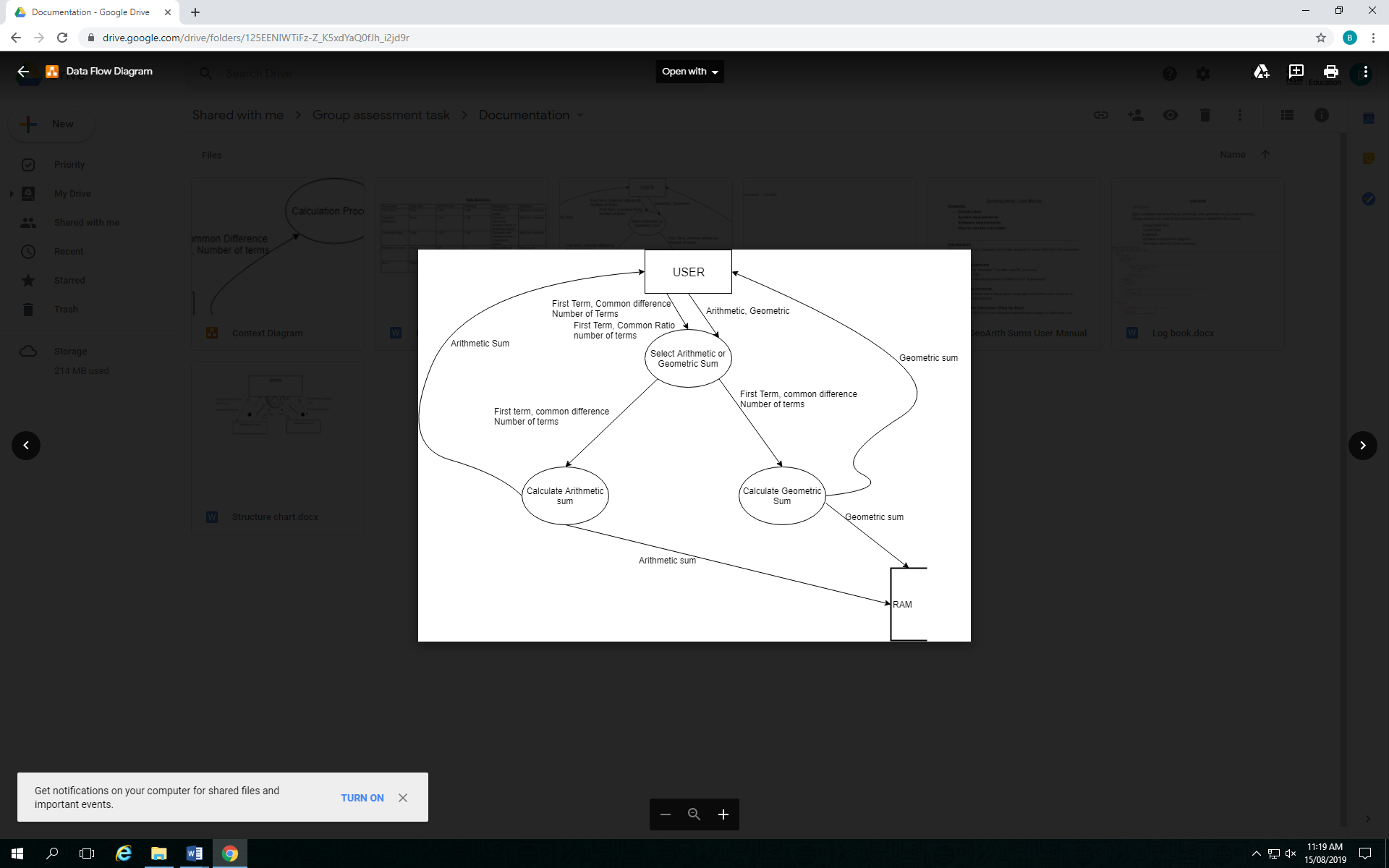
20/8/19

Nearly all items were completed. The documentation was printed as well as creating a set of references.

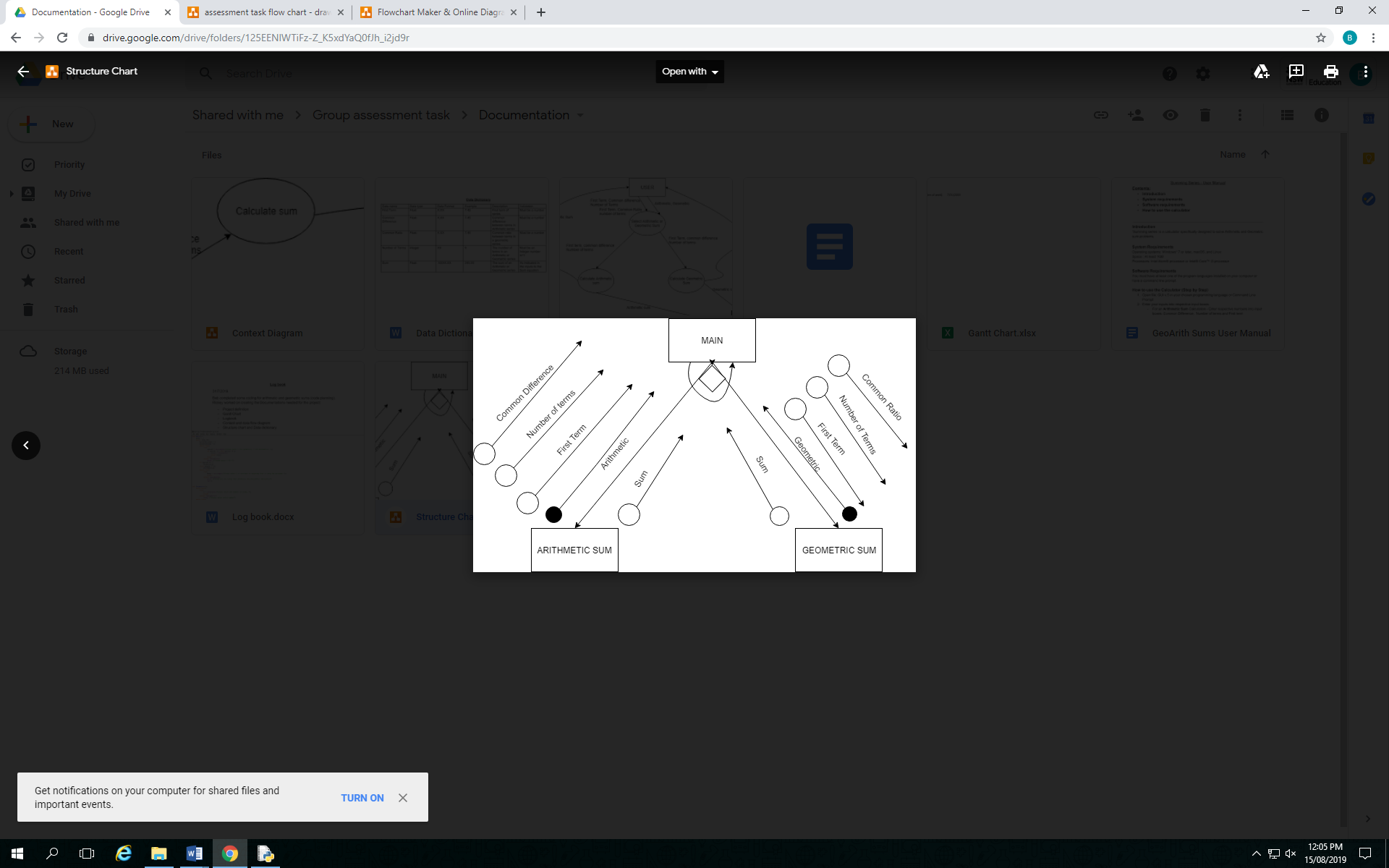
# Context Diagram



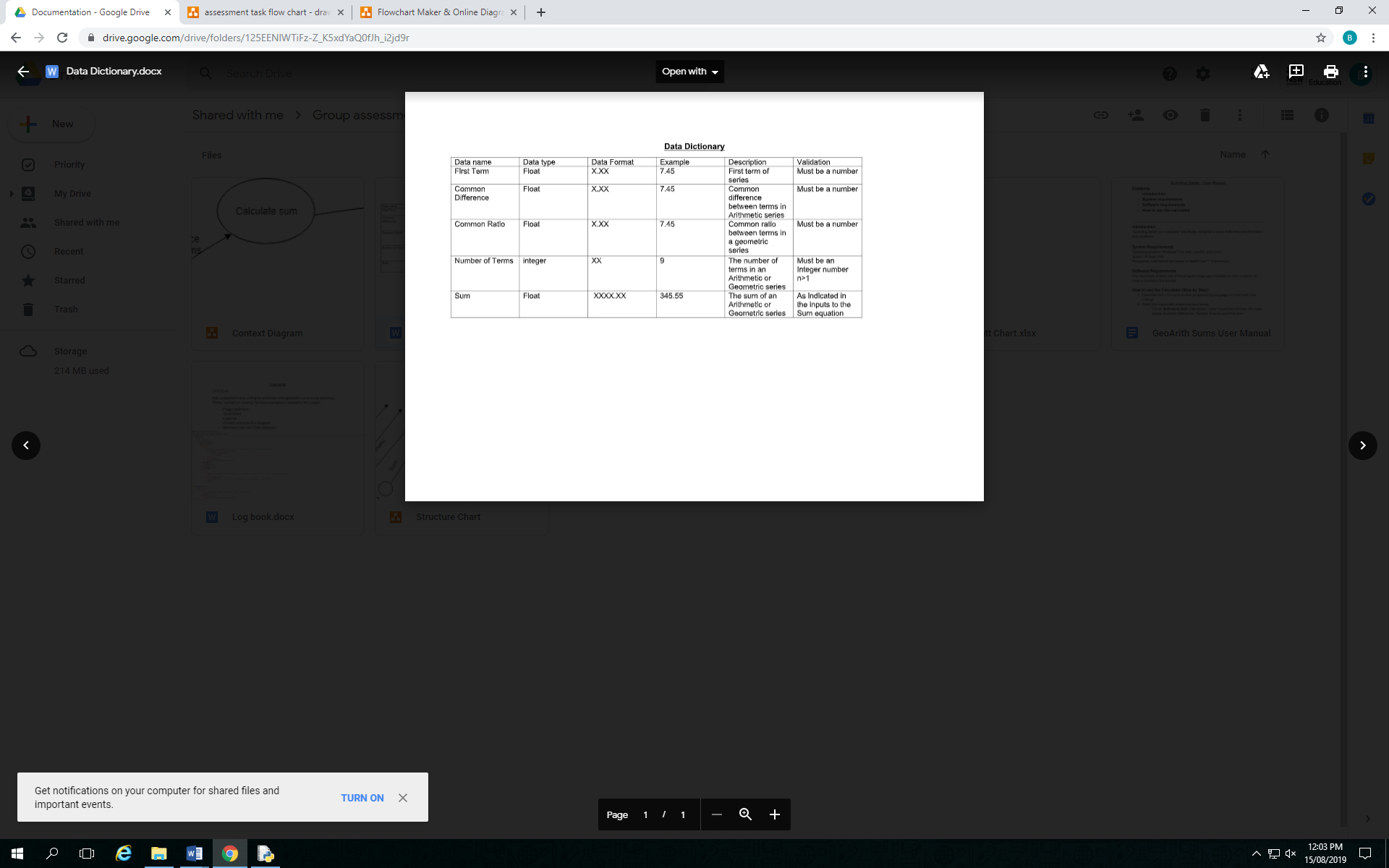
# Data Flow Diagram



# Structure chart



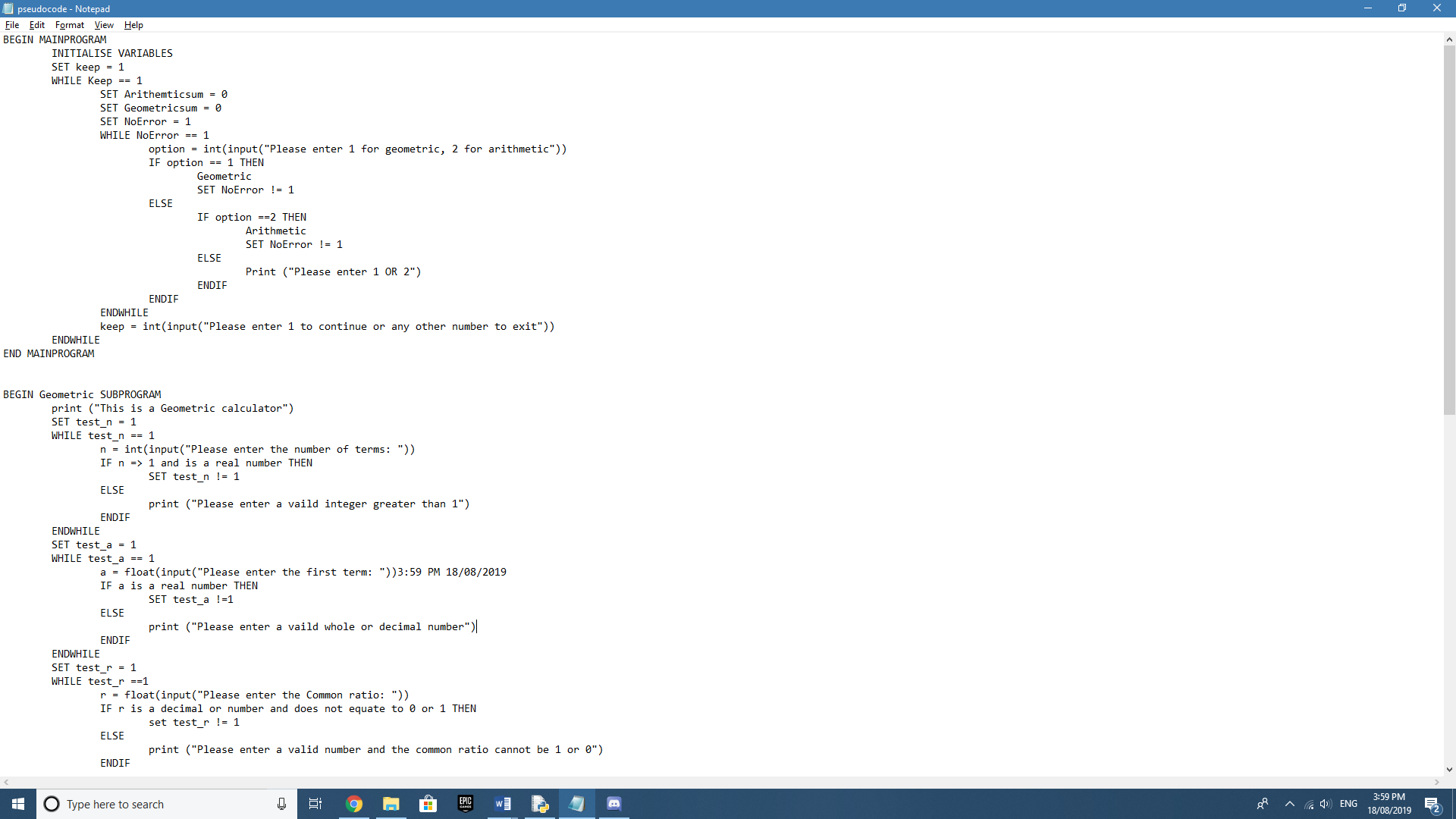
# Data Dictionary

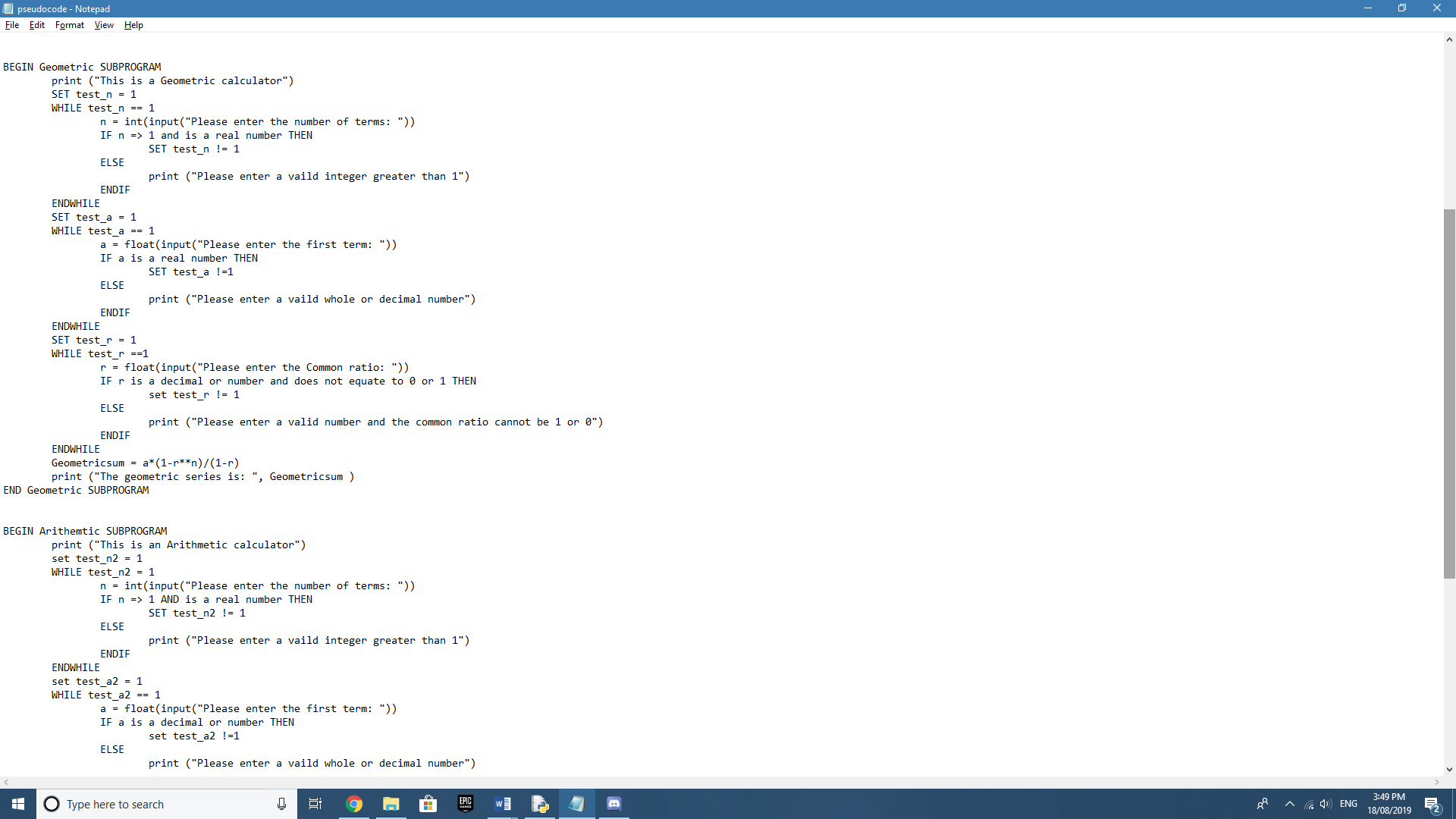


# IPO

|  |  |  |
| --- | --- | --- |
| INPUT | PROCESS | OUTPUT |
| Option (for geometric or arithmetic)  n (number of terms)  a (First term)  r (Common ratio)  d (difference)  keep (to keep or exit the loop) | SET keep = 1  WHILE keep == 1  Set Arithmeticsum = 0  Set Geometricsum = 0  IF option = 1 THEN  Geometricsum = a\*(1-r\*\*n)/(1-r)  ELSE  IF option = 2 THEN  Arithemticsum = (n/2)\*(2\*a+(n-1)\*d)  ELSE  Print error message  ENDIF  ENDIF  ENDWHILE | Geometricsum  Arithemticsum  Error |

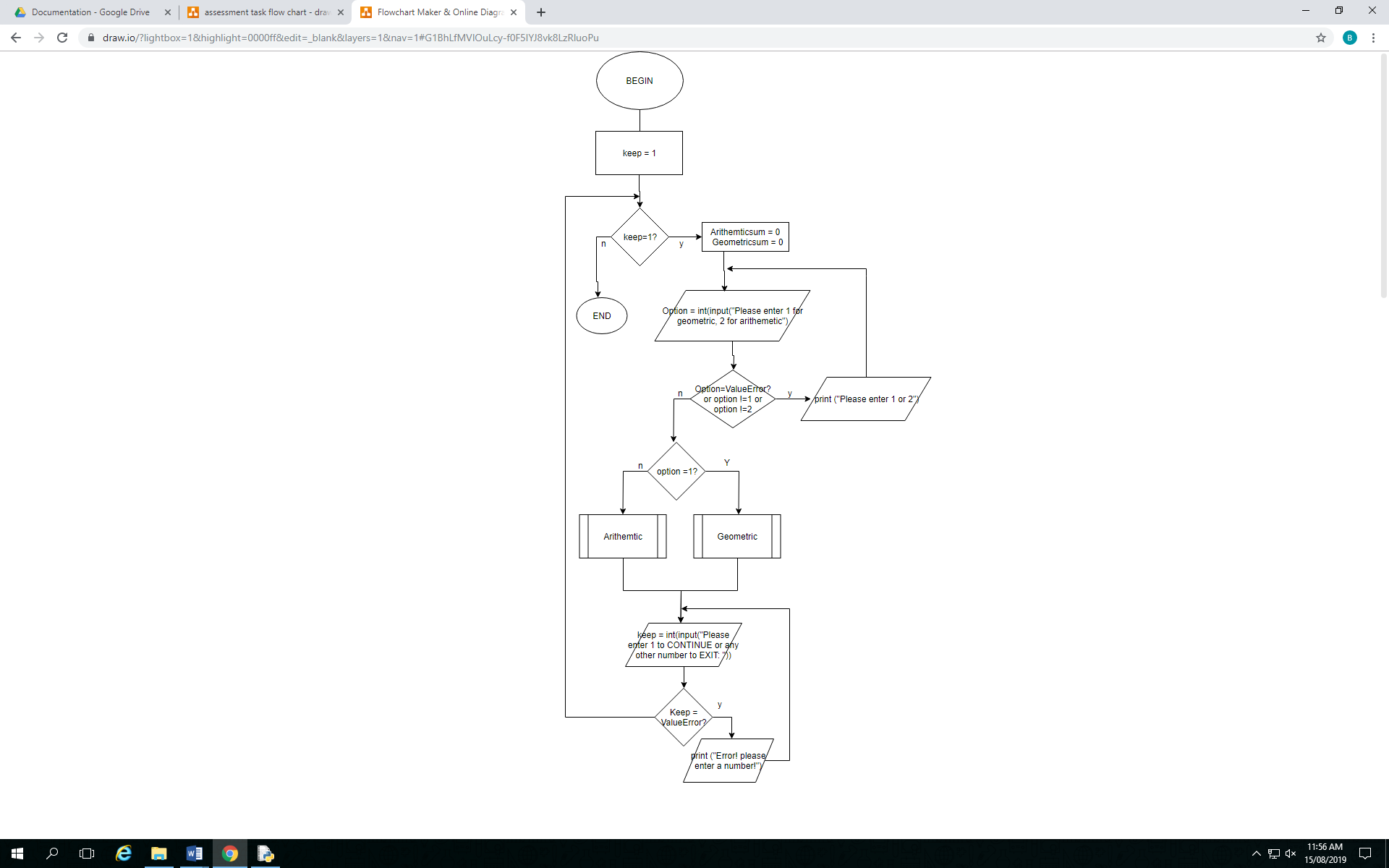
# Pseudocode

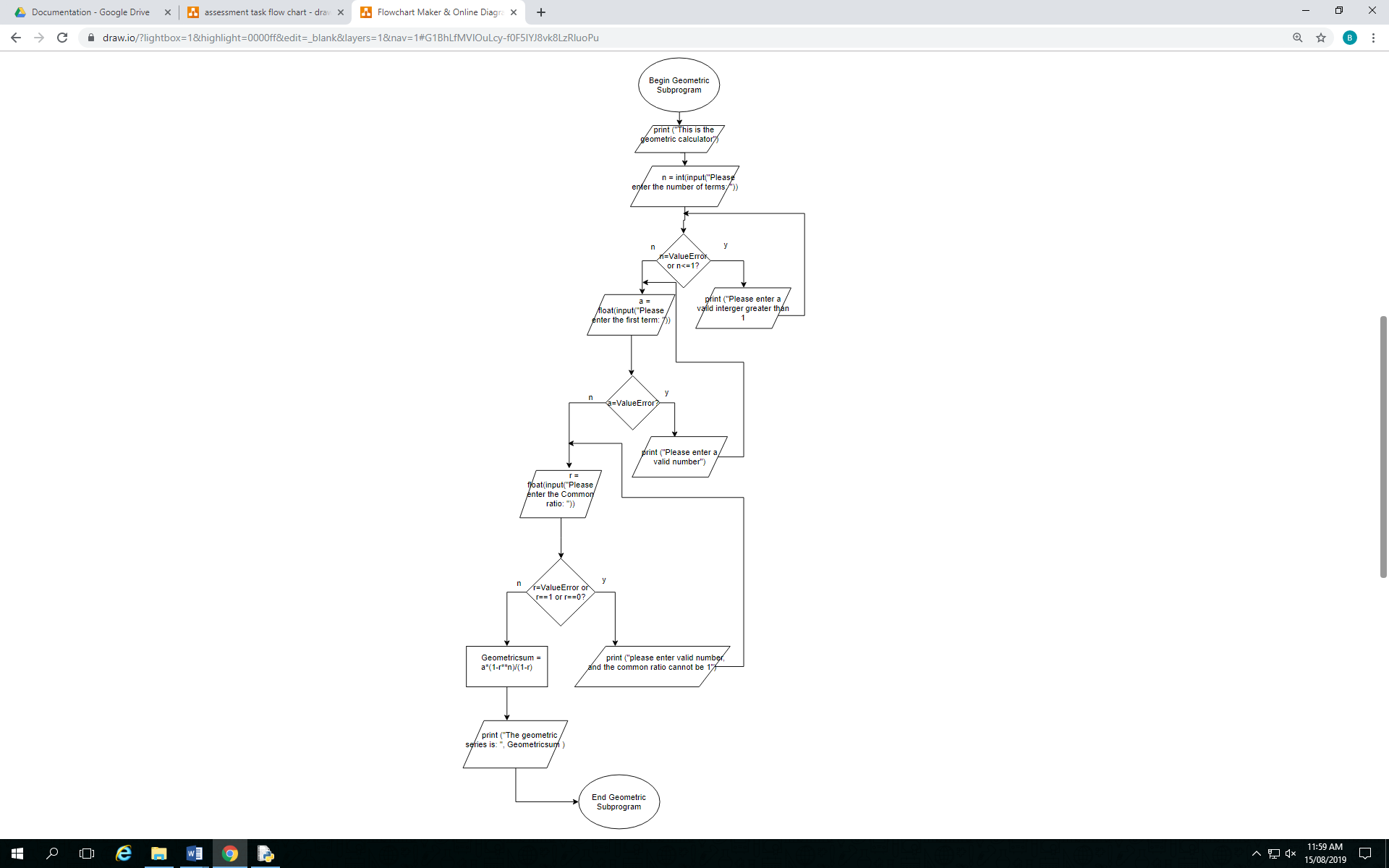


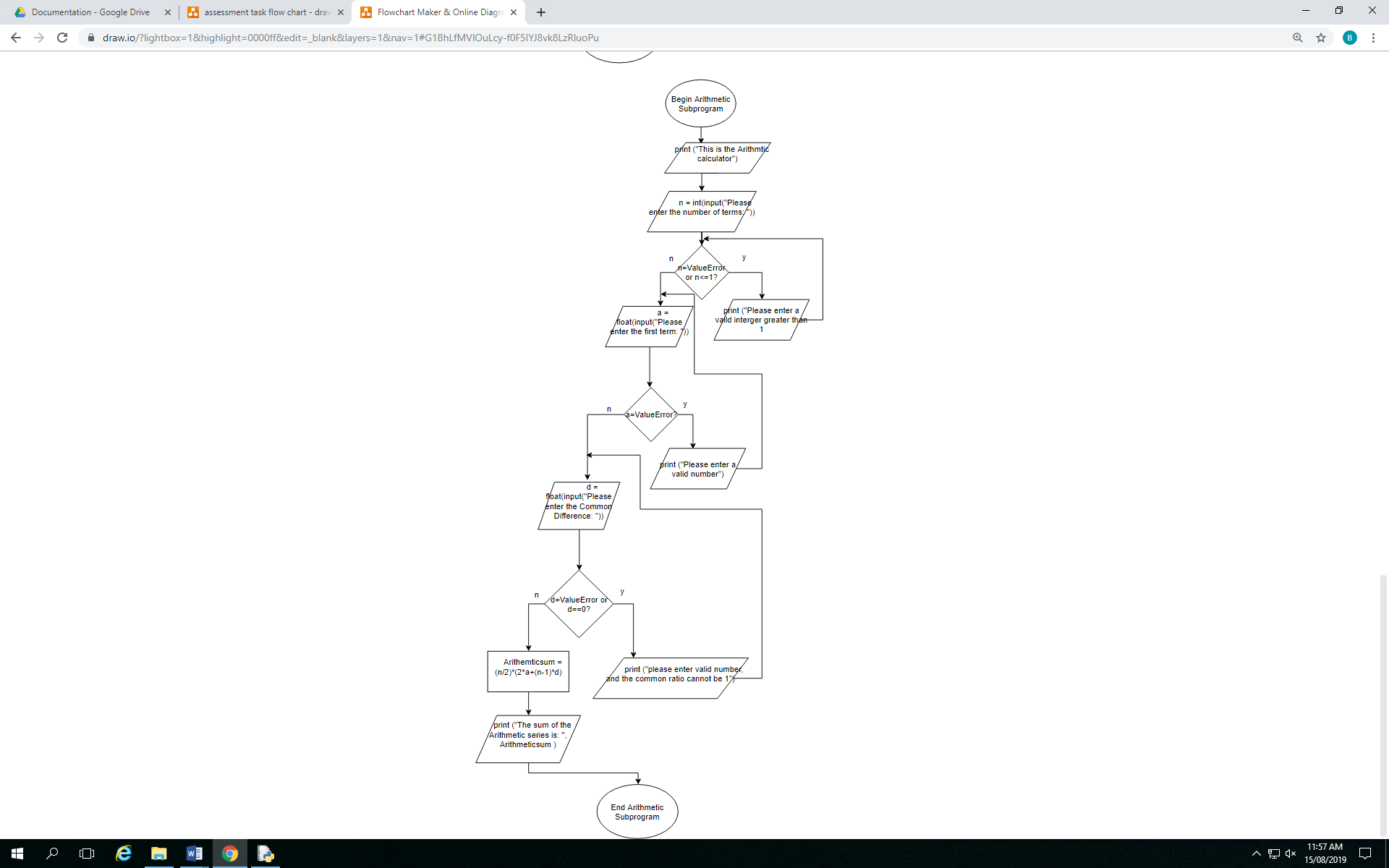




# Flowchart



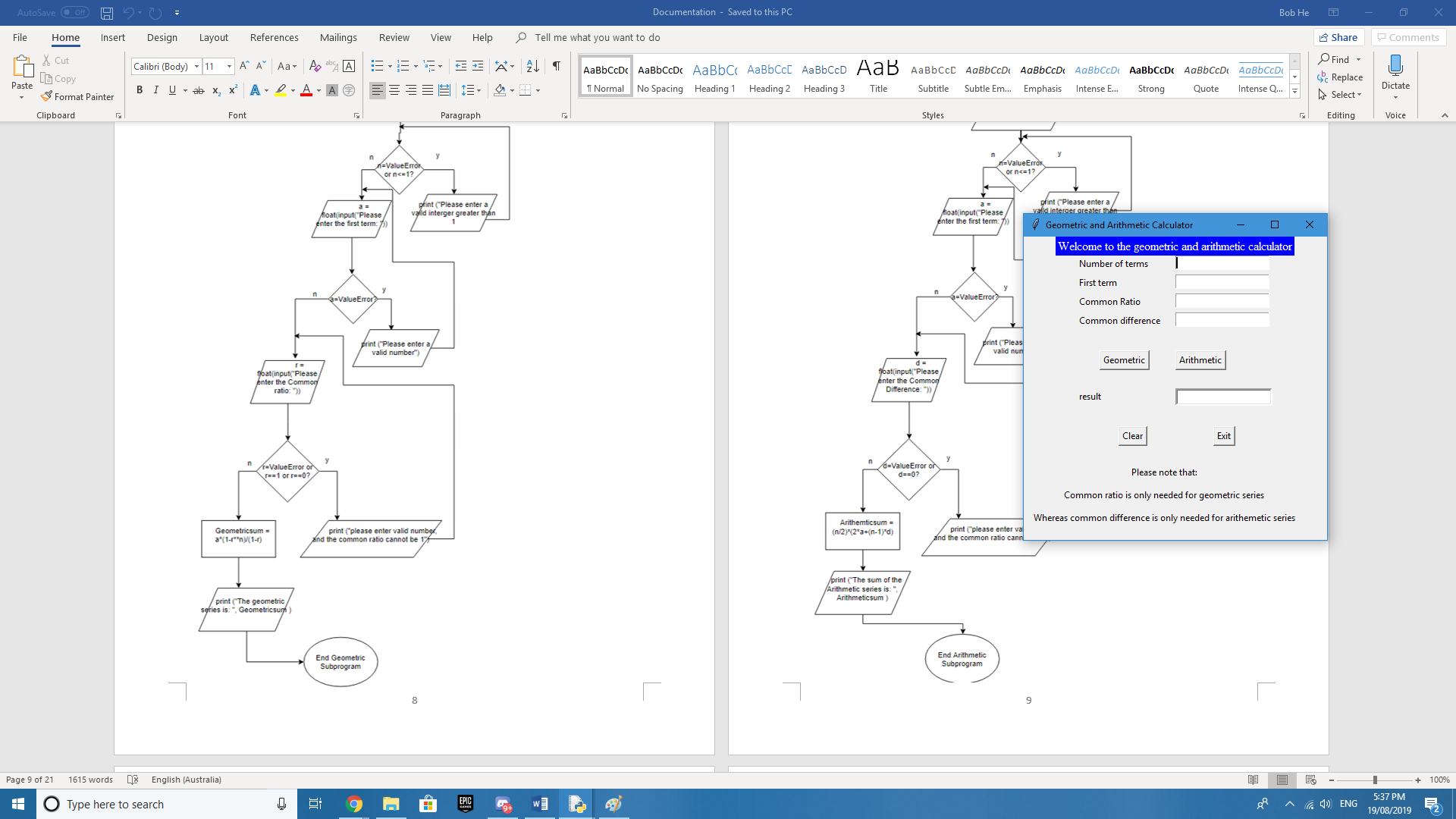
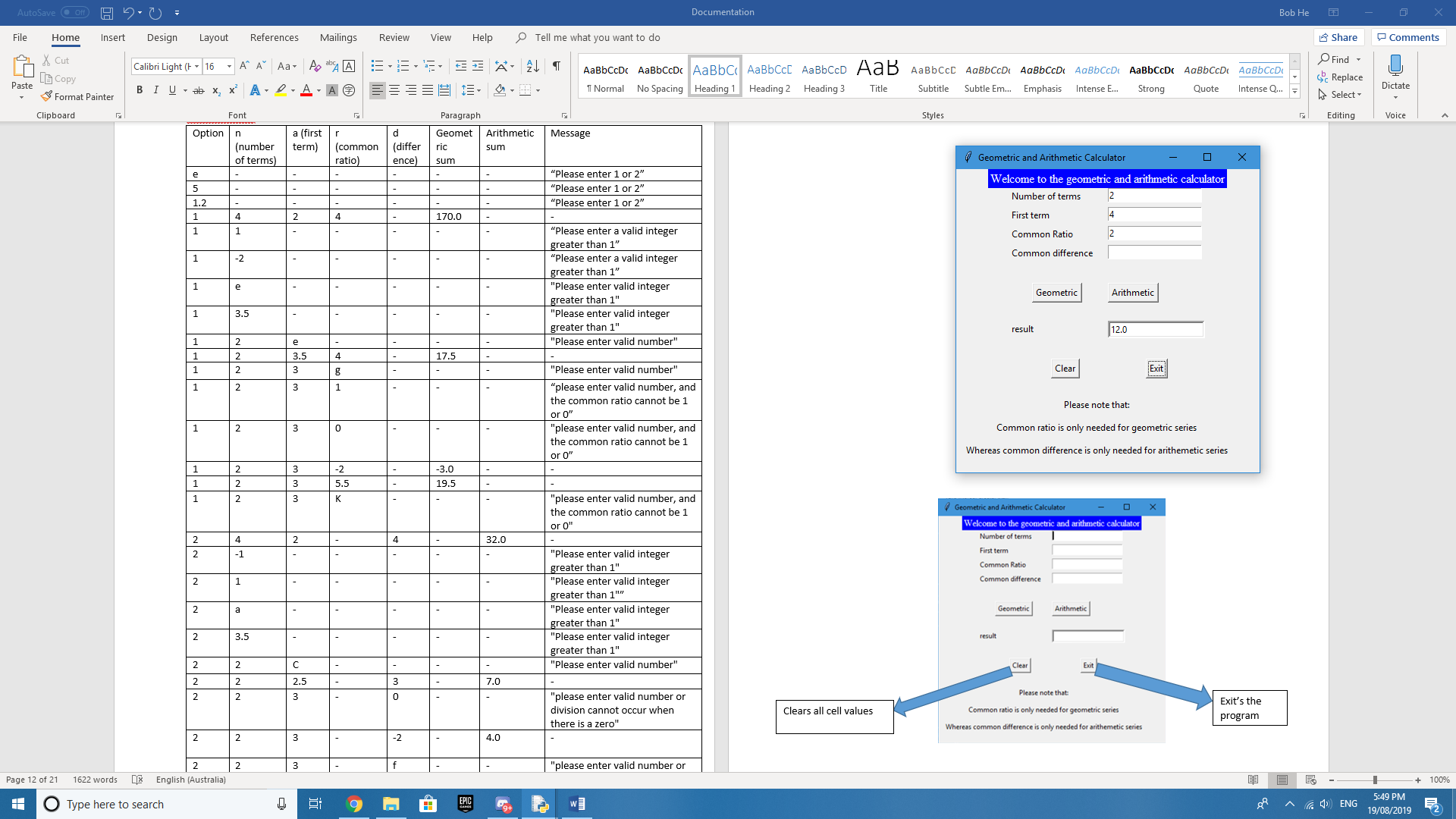
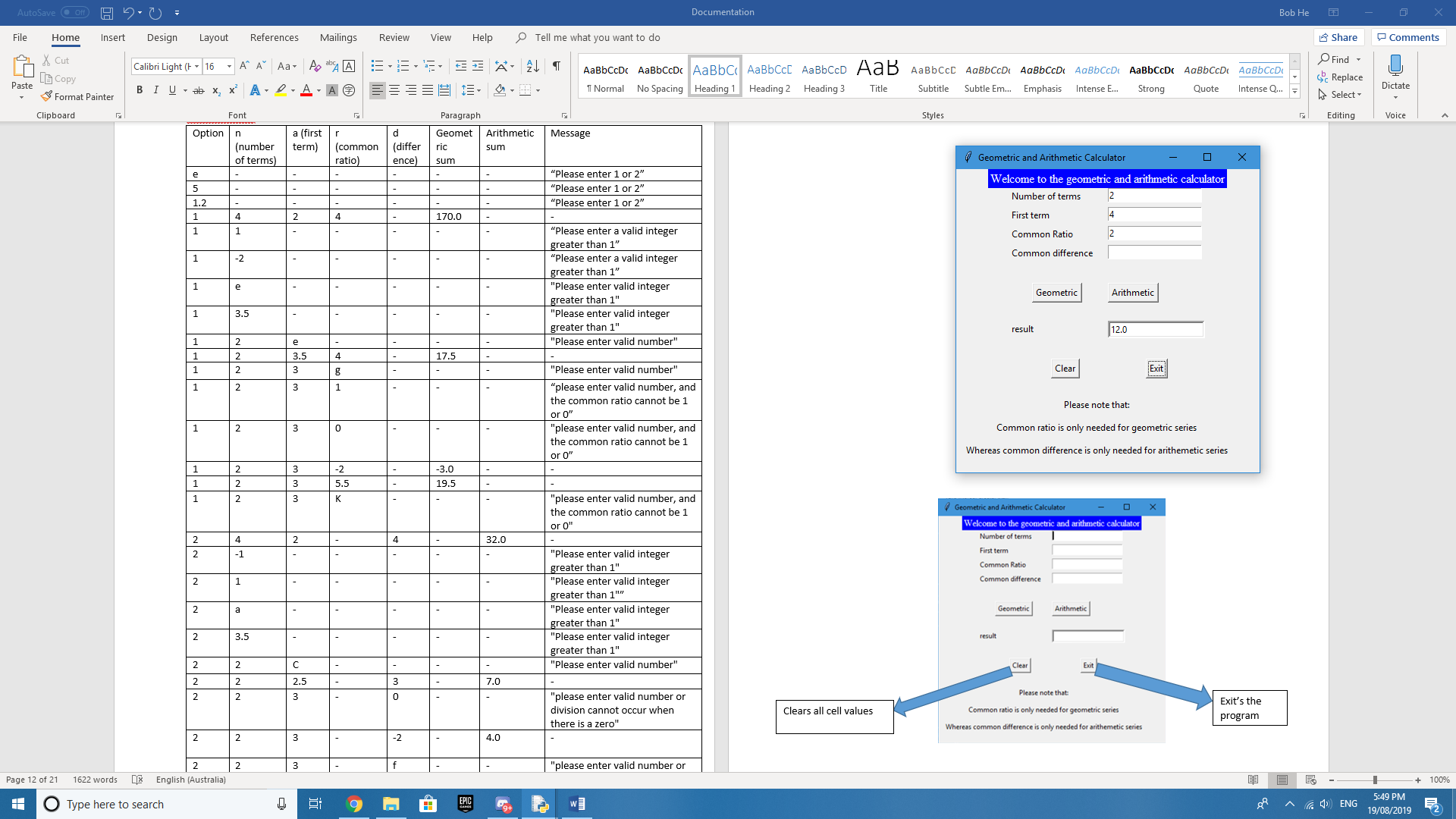
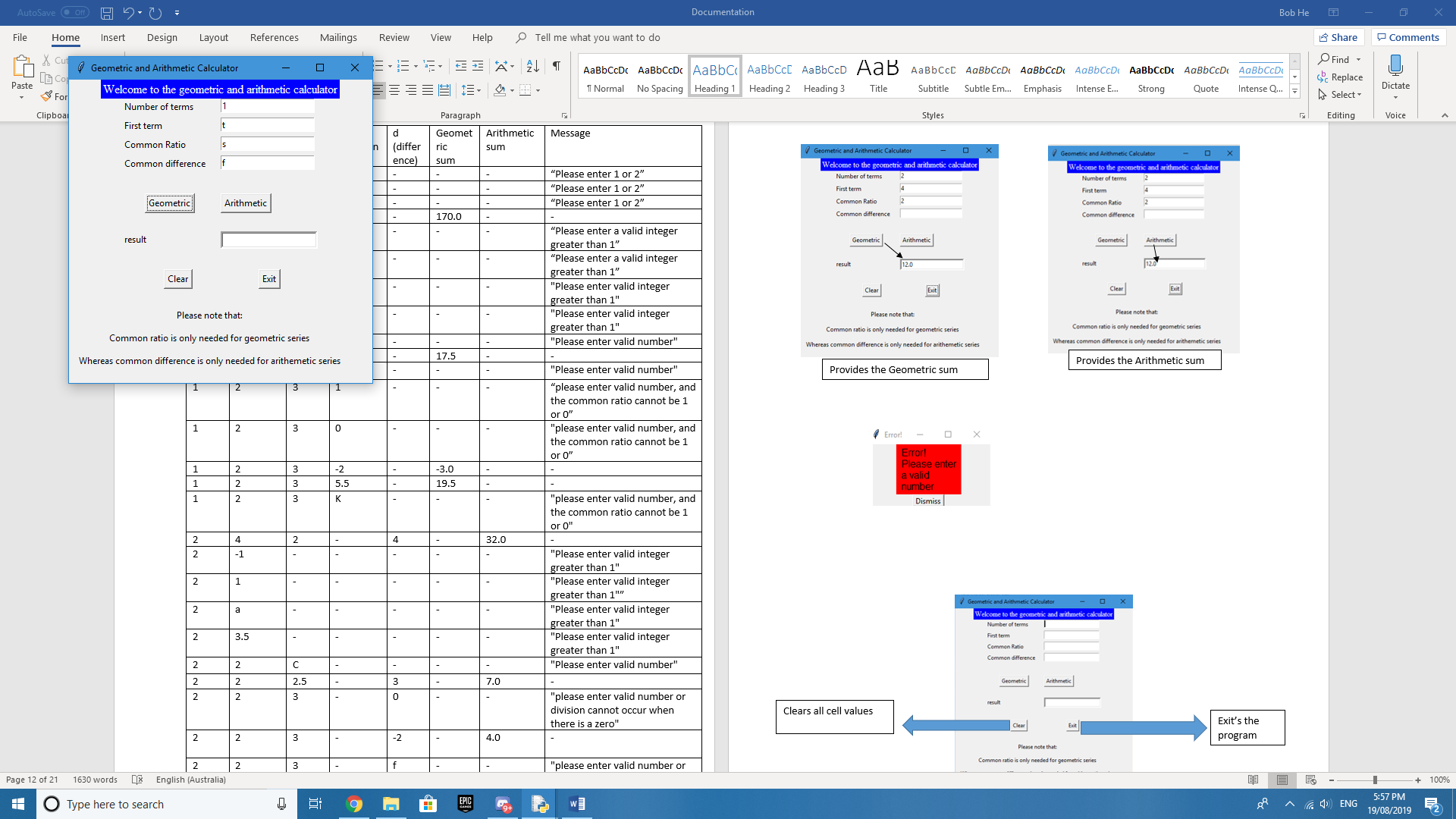
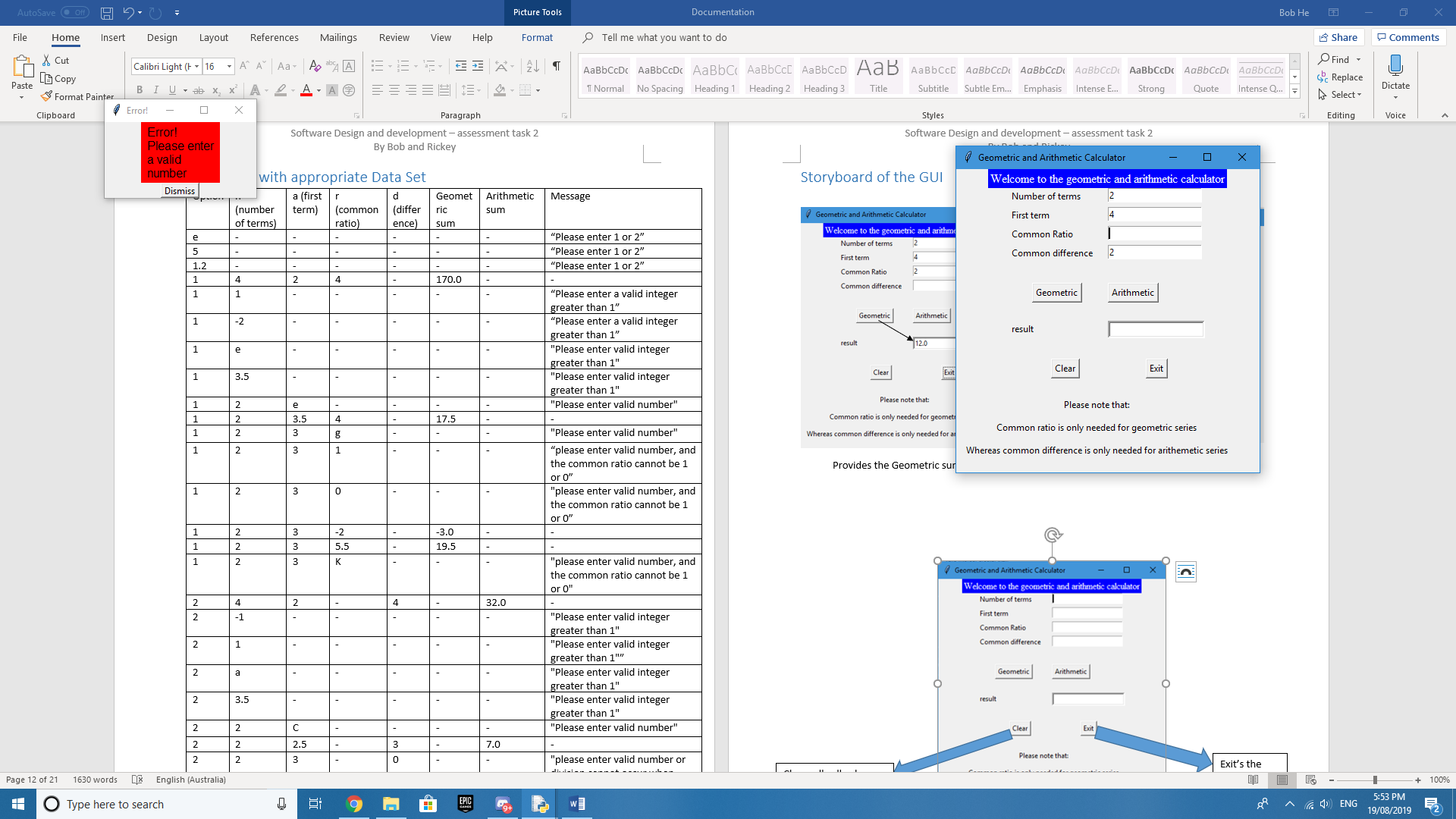




# Deskcheck with appropriate Data Set

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Option | n (number of terms) | a (first term) | r (common ratio) | d (difference) | Geometric  sum | Arithmetic  sum | Message |
| e | - | - | - | - | - | - | “Please enter 1 or 2” |
| 5 | - | - | - | - | - | - | “Please enter 1 or 2” |
| 1.2 | - | - | - | - | - | - | “Please enter 1 or 2” |
| 1 | 4 | 2 | 4 | - | 170.0 | - | - |
| 1 | 1 | - | - | - | - | - | “Please enter a valid integer greater than 1” |
| 1 | -2 | - | - | - | - | - | “Please enter a valid integer greater than 1” |
| 1 | e | - | - | - | - | - | "Please enter valid integer greater than 1" |
| 1 | 3.5 | - | - | - | - | - | "Please enter valid integer greater than 1" |
| 1 | 2 | e | - | - | - | - | "Please enter valid number" |
| 1 | 2 | 3.5 | 4 | - | 17.5 | - | - |
| 1 | 2 | 3 | g | - | - | - | "Please enter valid number" |
| 1 | 2 | 3 | 1 | - | - | - | “please enter valid number, and the common ratio cannot be 1 or 0” |
| 1 | 2 | 3 | 0 | - | - | - | "please enter valid number, and the common ratio cannot be 1 or 0” |
| 1 | 2 | 3 | -2 | - | -3.0 | - | - |
| 1 | 2 | 3 | 5.5 | - | 19.5 | - | - |
| 1 | 2 | 3 | K | - | - | - | "please enter valid number, and the common ratio cannot be 1 or 0" |
| 2 | 4 | 2 | - | 4 | - | 32.0 | - |
| 2 | -1 | - | - | - | - | - | "Please enter valid integer greater than 1" |
| 2 | 1 | - | - | - | - | - | "Please enter valid integer greater than 1"” |
| 2 | a | - | - | - | - | - | "Please enter valid integer greater than 1" |
| 2 | 3.5 | - | - | - | - | - | "Please enter valid integer greater than 1" |
| 2 | 2 | C | - | - | - | - | "Please enter valid number" |
| 2 | 2 | 2.5 | - | 3 | - | 7.0 | - |
| 2 | 2 | 3 | - | 0 | - | - | "please enter valid number or division cannot occur when there is a zero" |
| 2 | 2 | 3 | - | -2 | - | 4.0 | - |
| 2 | 2 | 3 | - | f | - | - | "please enter valid number or division cannot occur when there is a zero” |

# Storyboard of the GUI



Provides the Arithmetic sum

Exit’s the program

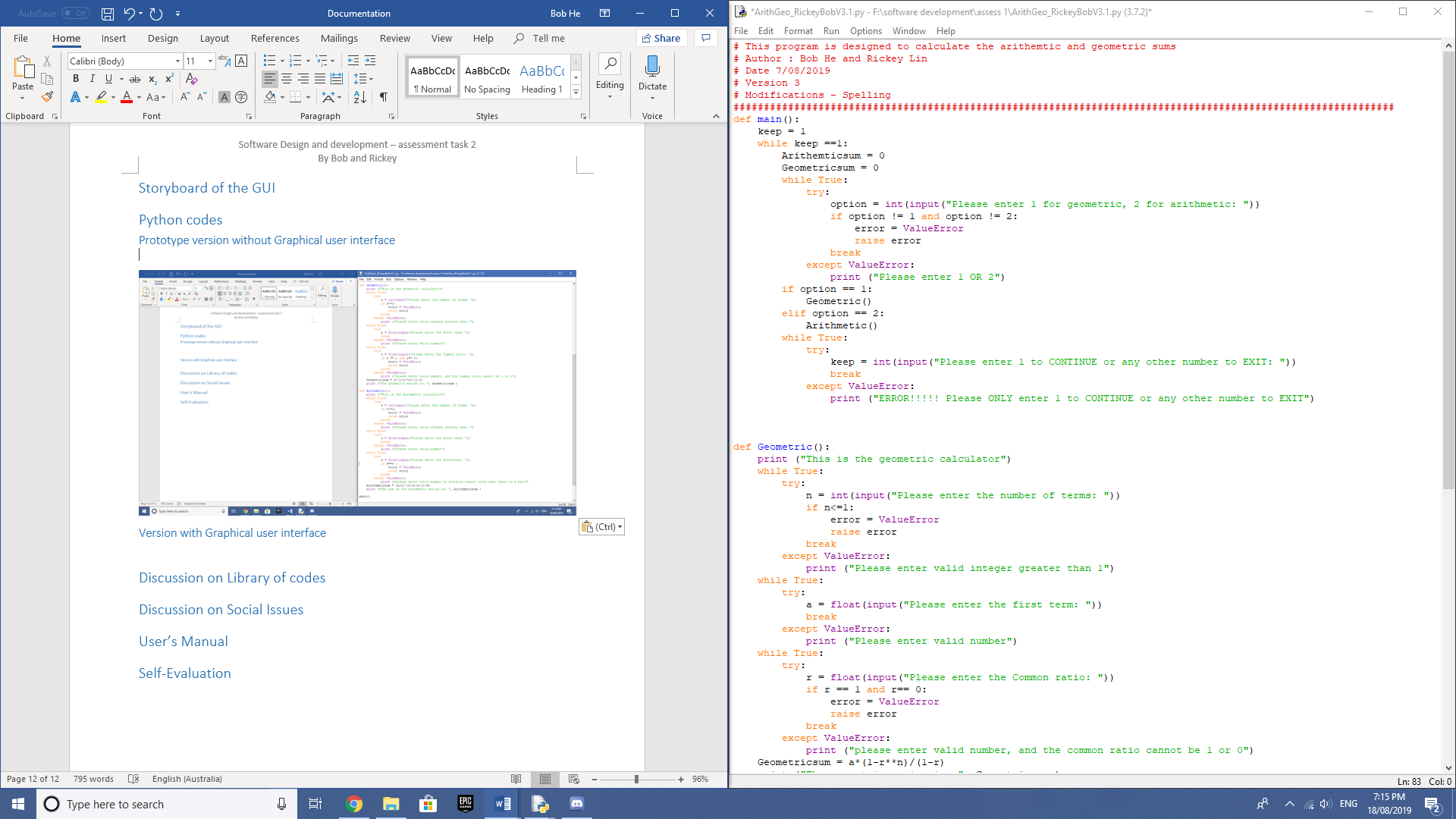
When a ValueError or exemption is detected

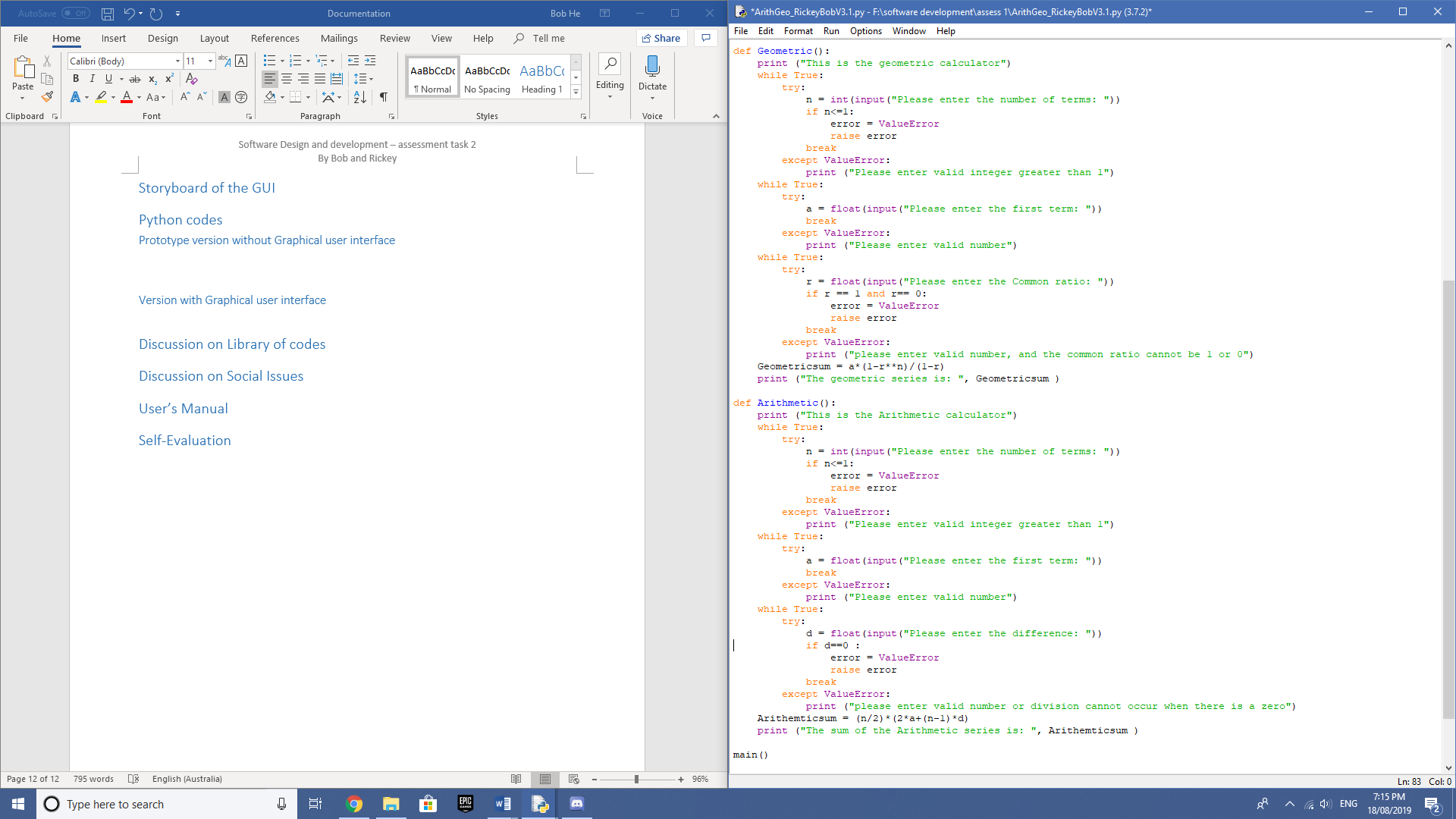
Provides the Geometric sum

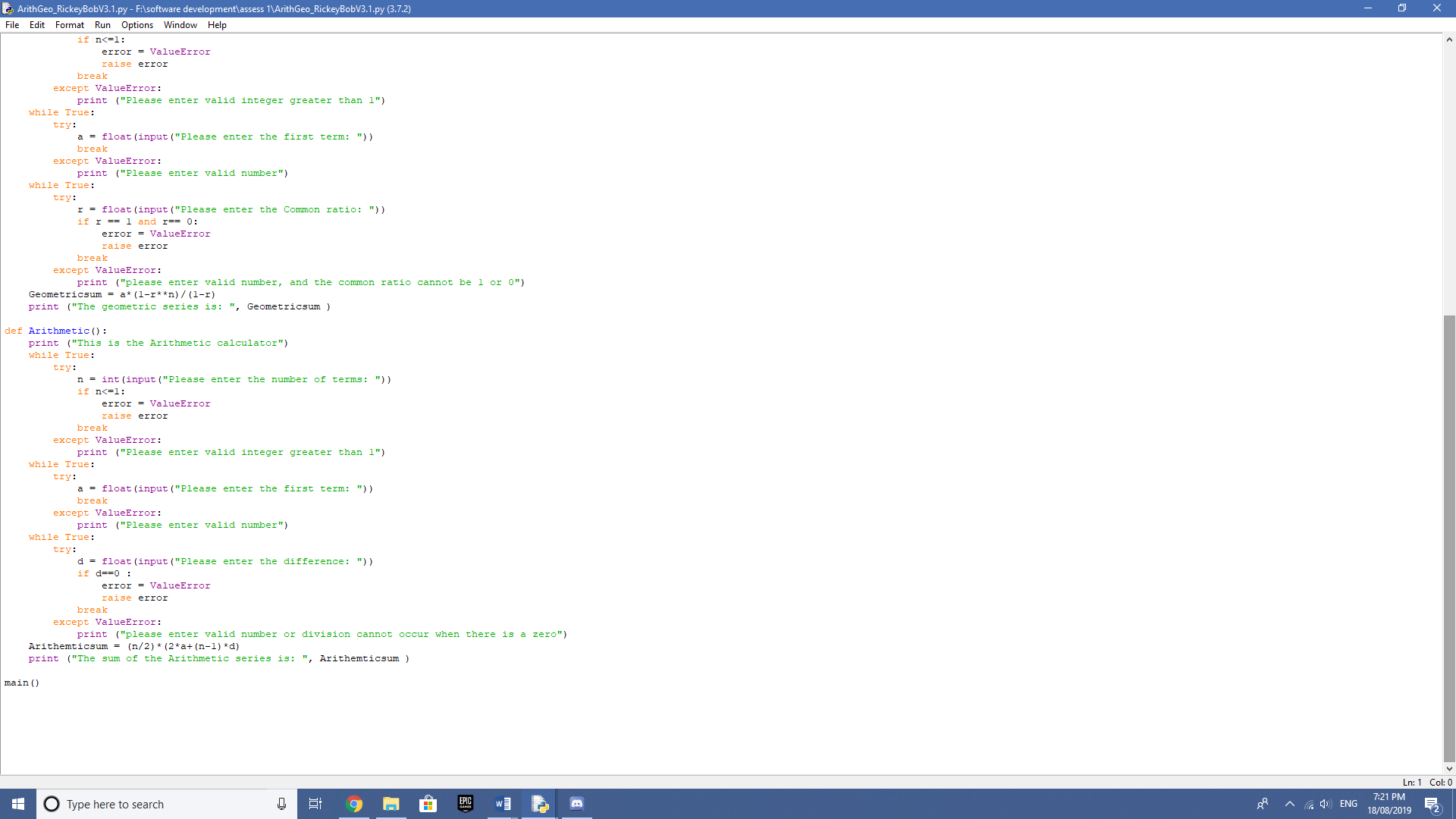
Clears all cell values

# Python codes

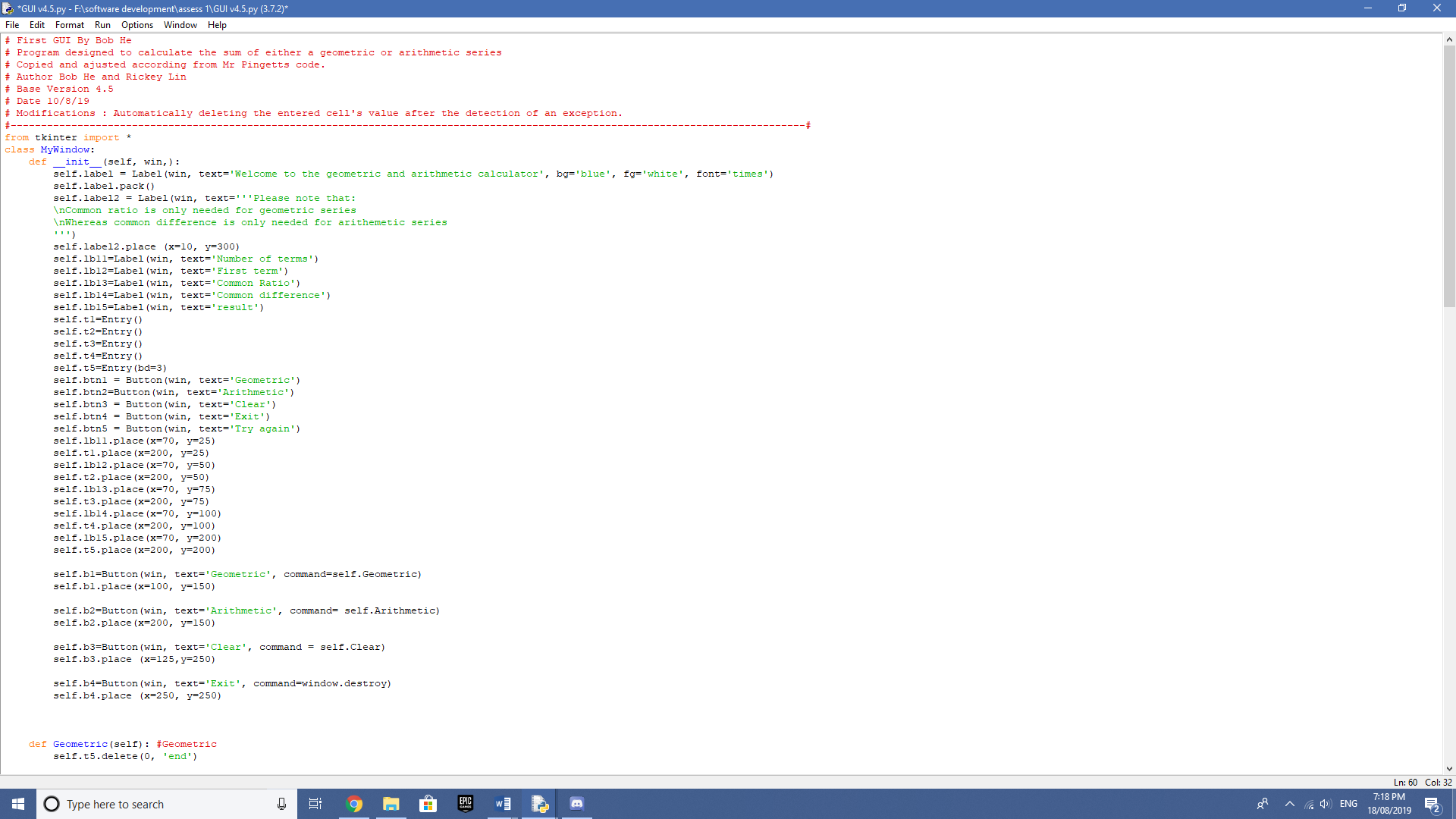
## Prototype version without Graphical user interface

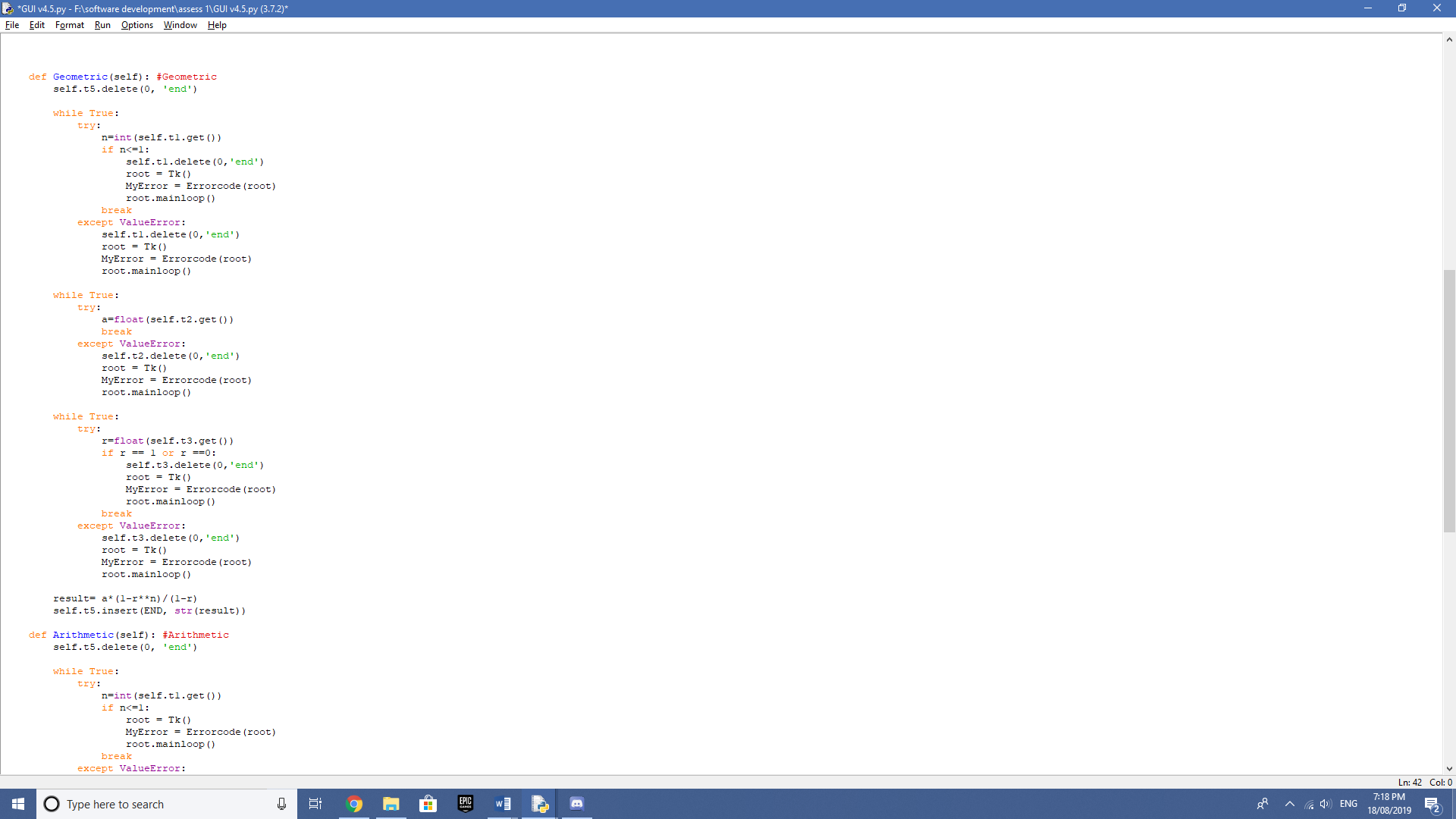


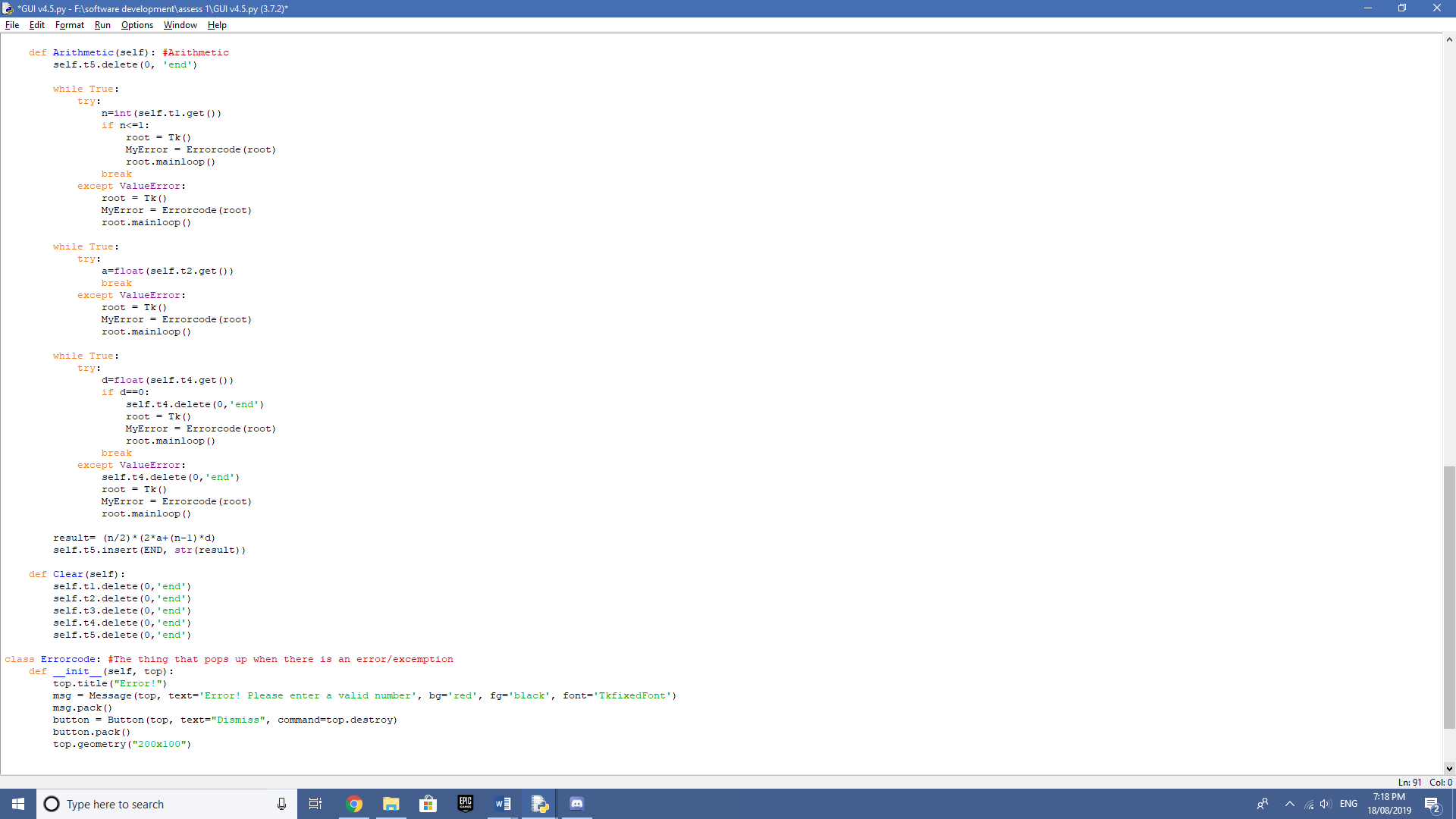


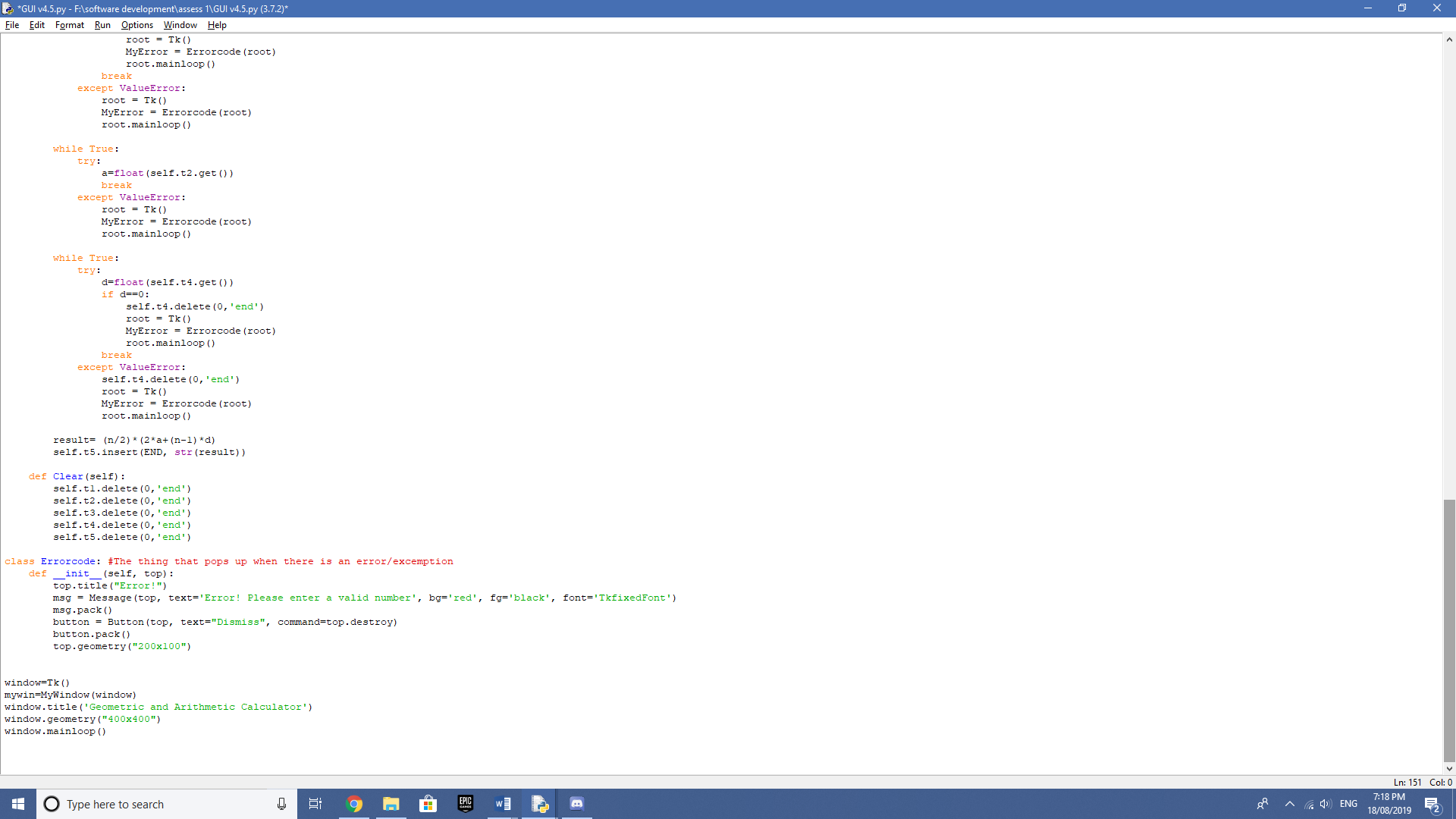


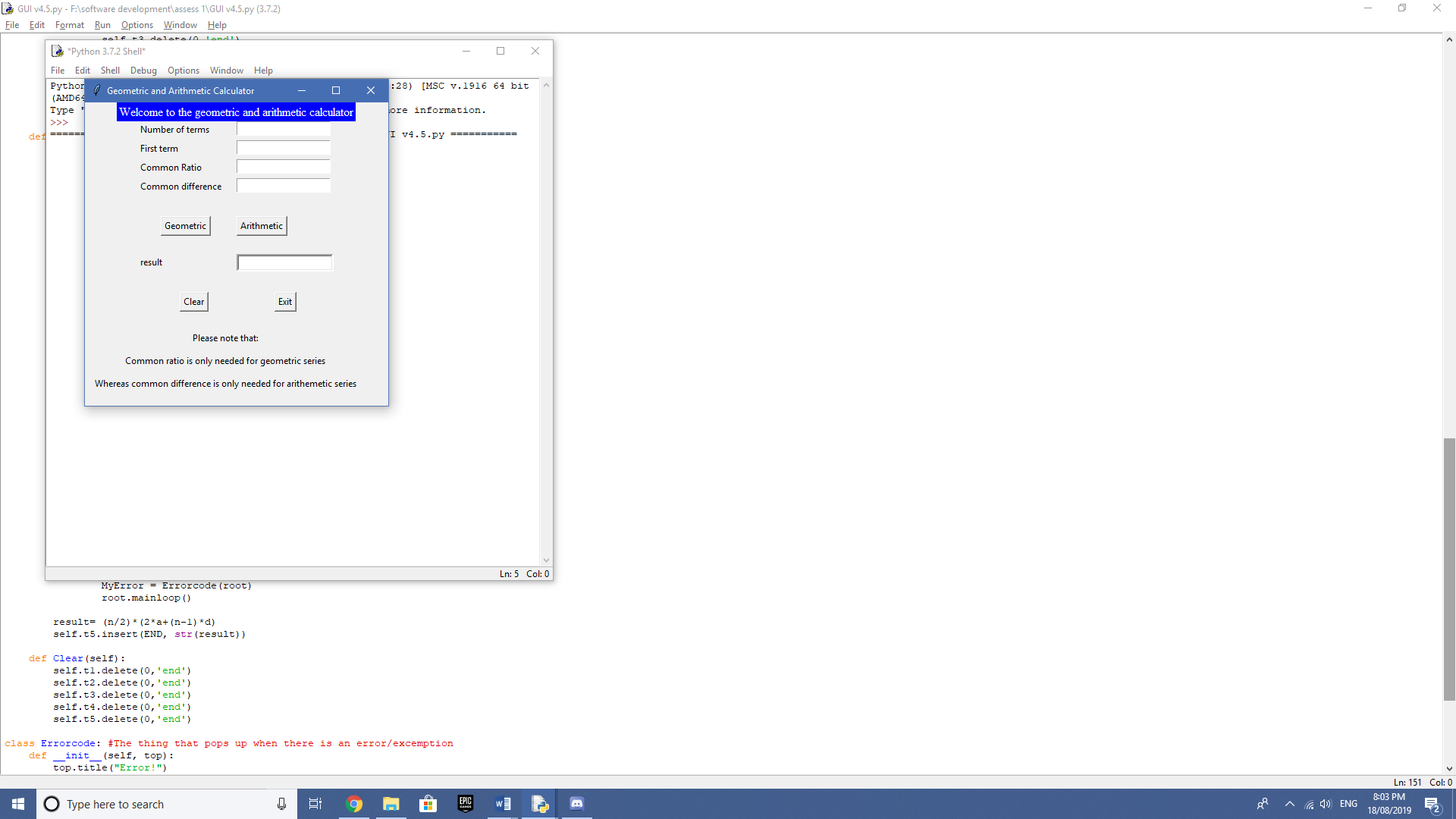
## Version with Graphical user interface











# Error correction techniques

## Differences between syntax, runtime and logic errors

Syntax errors -  This error occurs when there is a mistake in the arrangement of characters in the programs code. Some common syntax errors include:

* Misspelt words
* Missing characters (letters and semicolons)

Runtime Errors - This type of error occurs when the program has no syntax errors and when the program tells the computer to do something it cannot do. For example:

* Array index out of bounds
* Dividing two floats and storing it in an integer

Logic Errors - A logic error is when there is a mistake in the program’s source code that causes the program to behave in an incorrect way or give an unexpected result. For example: Dividing two numbers together instead of subtracting them.

## When would I use stubs and flags?

***Stubs*** - Stubs are modules in which allows us to test a particular portion of the program. We would use it when a particular part of the program has been completed. In this task, we could have used it for when we completed the while loops, the geometric and arithmetic subprograms, or in many stages in doing the GUI. In particular with the GUI, we used it to test the x and y coordinates of the buttons, the text, the boxes to type in and the window size of the GUI.

***Flags*** - Flags on the other hand are something known as Boolean variables and were used to record whether or not a certain condition had been met, which are displayed in output statements. We would use flags in my program to test whether the subprograms have been called, being the arithmetic and geometric subprograms. Furthermore, we would test whether the code has entered or exited the while programs within my code. We have also used flags to test whether the parameters had been correctly set, and whether it printed. This included the classes in the GUI and defining the rest of the code using \_init\_, to test whether the section was met.

# Test Python code

## Desk checking of the Python code (GUI)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of terms | First term | Common Ratio | Common difference | Button clicked | Result | Message |
| - | - | - | - | Geometric | - | Error! Please enter a valid number |
| - | - | - | - | Arithmetic | - | Error! Please enter a valid number |
| - | - | - | - | Clear | - | - |
| - | - | - | - | Exit | - | - |
| 2 | 2 | 2 | 4 | Geometric | 6.0 | - |
| 2 | 2 | 2 | 4 | Arithmetic | 8.0 | - |
| 2 | 2 | 2 | - | Geometric | 6.0 | - |
| 2 | 2 | - | 4 | Arithmetic | 8.0 | - |
| 2 | 2 | a | 4 | Arithmetic | 8.0 | - |
| 2 | 2 | a | 4 | Geometric | - | Error! Please enter a valid number |
| 2 | 2 | 2 | a | Geometric | 6.0 | - |
| 2 | 2 | 2 | A | Arithmetic | - | Error! Please enter a valid number |
| 1 | 2 | 3 | 4 | Geometric | - | Error! Please enter a valid number |
| 1 | 2 | 3 | 4 | Arithmetic | - | Error! Please enter a valid number |
| 0.5 | 2 | 3 | 4 | Arithmetic | - | Error! Please enter a valid number |
| -1 | 2 | 2 | 4 | Geometric | - | Error! Please enter a valid number |
| A | 3 | 2 | 4 | Arithmetic | - | Error! Please enter a valid number |
| A | 3 | 2 | 4 | Geometric | - | Error! Please enter a valid number |
| 3 | c | 3 | 5 | Geometric | - | Error! Please enter a valid number |
| 3 | c | 3 | 5 | Arithmetic | - | Error! Please enter a valid number |
| 3 | 2 | 1 | - | Geometric | - | Error! Please enter a valid number |
| 3 | 2 | 0 | - | Geometric | - | Error! Please enter a valid number |
| 3 | 2 | a | - | Geometric | - | Error! Please enter a valid number |
| 3 | 2 | - | 0 | Arithmetic | - | Error! Please enter a valid number |
| 3 | 2 | - | c | Arithmetic | - | Error! Please enter a valid number |
| 2 | 3.5 | 4.6 | - | Geometric | 15.59 | - |
| 2.1 | 3.5 | 4.6 | - | Geometric | - | Error! Please enter a valid number |
| 3 | 3.5 | - | 4.6 | Arithmetic | 24.29 | - |
| 3.5 | 3.5 | - | 4.6 | Arithmetic | - | Error! Please enter a valid number |

# Library of codes

How can we reuse this code to make it part of a bigger solution package?

This piece of code is a small bit pre-written code and therefore could be used in another solution package of a bigger all encompassing calculator. For example, if we were to perhaps be requested to make a calculator for all mathematical calculations, we could reuse this snippet of code and plug it into our new program. There are many more options to reuse this code that we have written. We could take out parts of our program and plug it into other pieces of code. For example, part of our  program that denotes the graphic user interface could be taken out and placed in another calculator, and the code that controls the calculations of the Arithmetic and Geometric sums can simply be changed around a little bit to make other calculations and could be changed around a little bit to suit the needs of the other programmer. Also, we could potentially upload this online and release this to the public as a reusable pre-written snippet of code to aid in their production of a new program.

# Social Issues

**Summing Series - Social Issues**

**Plagiarism:**

To prevent the plagiarising of our program and to protect our intellectual property. We have considered many options such as:

* Putting the Author and the date of completion within our code
* Encrypting our program and the folder that it’s contained in
* The usage of software license agreements and
* The usage of the End User License Agreement (EULA)

Encrypting the Author details and date of completion within our code:

This prevents plagiarism of our code since if someone were to copy the code and try to claim the code was their own, there would be evidence that it clearly is not theirs. Thus, protecting our Intellectual property.

Software license agreements:

A software license agreement is a document that denotes the formal permission to use a product. Within the agreement contains:

* Term: the time period that the agreement is valid
* Limited use: an agreement that says the product can’t be altered, copied and redistributed
* Liability : The limited liability of the software developer. This means that if a customer/consumer were to sue the company, the software developer would be free from this lawsuit.

Copyright laws:

During our production of the software, we also considered the copyright laws that protect the software license agreement. Such laws include:

* The copyright act of 1968
* The Copyright Amendment Act 1993

End User License Agreement (EULA):

The EULA is a document that contains details on:

* How the software can and can’t be use
* And any restrictions the manufacturers have imposed upon the user

**Inclusivity:**

Inclusivity in terms of Software development is the practice of constructing a piece of software include as many people as possible from every walk of life, that would otherwise be marginalised from using the software because of things such as: a physical or mental disability and the affiliation with a certain group of people e.g. minority groups.

How we have considered inclusivity when creating our program:

Whilst considering inclusivity issues related to our program we incorporated these elements to improve inclusivity and ergonomics:

* Big, easy to read font to assist those with poor eyesight
* The use of a variety of colours to highlight text and improve visibility
* Separating the elements of the GUI to improve the ease of use and reduce confusion
* Clear instructions in the user manual

Other things we could include:

There are many other things we could have included into our program to further its inclusivity. We could have considered a multitude of different things that we could do to improve our program. Such things include:

* Incorporating an a voice assist to narrate the options presented on the screen
* Bold font
* Bigger buttons

# User’s Manual

Summing Series - User Manual

**Contents:**

* **Introduction**
* **System requirements**
* **Software requirements**
* **How to use the calculator**

**Introduction**

‘Summing series’ is a calculator specifically designed to solve Arithmetic and Geometric sum problems.

**System Requirements**

Operating systems: Windows\* 7 or later, macOS, and Linux

Space : At least 1GB

Processors: Intel Atom® processor or Intel® Core™ i3 processor

**Software Requirements**

You must have at least one of the program languages installed on your computer or have a command line prompt.

**How to use the Calculator (Step by Step)**

1. Open file: GUI v.5 in your chosen programming language or Command Line Prompt
2. Enter your inputs into respective input boxes.

* For an **Arithmetic Sum** Calculation - Enter respective numbers into input boxes: Common Difference,  Number of terms and First term
* Press Arithmetic Button
* If you want to do another calculation, press clear
* For a **Geometric Sum** Calculation - Enter respective numbers into input boxes: Common Ratio, Number of terms and First term.
* Press Geometric Button
* If you want to do another calculation, press clear

# Self-Evaluation

Bob – I found this task very straightforward any quite enjoyable in contrast to the tasks in other subjects. I found coding python solutions had allowed me to learn significantly more about python. In particular, coding the GUI using classes posed as a real challenge, but with online research and trial and error, I was able to complete it to my satisfaction. The Graphical user interface, at first thought appeared to be impossible, as the code was filled with foreign lines that I’ve yet to understand. Using classes and making edits to the code was a mixture of trial and error and just straight up time consuming. I was surprised to see such detailed documentations containing many of the new skills we’ve learnt this year compared to the documentations from the previous years, such as the system/data flow chart and diagrams. Yet with already some knowledge of the layout and items needed, I found that the documentation, while it was very large, I still had a very grasp of it from the similar one’s last year. What I would do differently if I had to start a new project is to allocate more time to this task. While I certainly did not rush this task, I felt like I could have checked and edited each code and documentation component in significantly more detailed.

# References

Introduction to GUI programming with tkinter — Object-Oriented Programming in Python 1 documentation. 2019. Introduction to GUI programming with tkinter — Object-Oriented Programming in Python 1 documentation. [ONLINE] Available at: https://python-textbok.readthedocs.io/en/1.0/Introduction\_to\_GUI\_Programming.html. [Accessed 19 August 2019].

The Tkinter Toplevel Class. 2019. The Tkinter Toplevel Class. [ONLINE] Available at: http://effbot.org/tkinterbook/pythondoc-tkinter-toplevel.htm#Tkinter.Toplevel-class. [Accessed 19 August 2019].

Stack Overflow. 2019. How do I create child windows with Python tkinter? - Stack Overflow. [ONLINE] Available at: https://stackoverflow.com/questions/15306631/how-do-i-create-child-windows-with-python-tkinter. [Accessed 19 August 2019].

24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.16 documentation. 2019. 24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.16 documentation. [ONLINE] Available at: https://docs.python.org/2/library/tkinter.html. [Accessed 19 August 2019].

Widget Styling. 2019. Widget Styling. [ONLINE] Available at: https://effbot.org/tkinterbook/tkinter-widget-styling.htm. [Accessed 19 August 2019].