Y13 Jake Brackley (Queensbury)

Properties of Waves Simulator.

Programming Project

H446 03/04

Contents

[Analysis 6](#_Toc166063930)

[Problem Identification 6](#_Toc166063931)

[Clients ID and Stakeholders 6](#_Toc166063932)

[Project Research 7](#_Toc166063933)

[Client Interview 10](#_Toc166063934)

[Interview Analysis 10](#_Toc166063935)

[Essential features of solution 11](#_Toc166063936)

[Suitable approaches based off research 11](#_Toc166063937)

[Proposed Solution 12](#_Toc166063938)

[Limitations Of Proposed Solution 12](#_Toc166063939)

[Requirements 13](#_Toc166063940)

[Total Success Criteria 14](#_Toc166063941)

[Which Developments Occur at each Iteration. 16](#_Toc166063942)

[Iteration 1 16](#_Toc166063943)

[Design: Iteration 1 16](#_Toc166063944)

[Success Criteria 16](#_Toc166063945)

[Equations 17](#_Toc166063946)

[Decomposition of program 18](#_Toc166063947)

[Whole program flow diagram 18](#_Toc166063948)

[Algorithm pseudocodes 19](#_Toc166063949)

[Usability Features 20](#_Toc166063950)

[Key Variable Table 22](#_Toc166063951)

[Validation Strategy 24](#_Toc166063952)

[Approach To Testing 25](#_Toc166063953)

[Test Strategy 26](#_Toc166063954)

[Implementation: Iteration 1 35](#_Toc166063955)

[Testing: Iteration 1 39](#_Toc166063956)

[Evaluation: Iteration 1 44](#_Toc166063957)

[Testing Review 44](#_Toc166063958)

[Success Criteria evaluation of Iteration 1 44](#_Toc166063959)

[Client feedback of Iteration 1 45](#_Toc166063960)

[My feedback of Iteration 1 45](#_Toc166063961)

[Iteration 1 Final Code 45](#_Toc166063962)

[Iteration 2 46](#_Toc166063963)

[Design: Iteration 2 46](#_Toc166063964)

[Success Criteria 46](#_Toc166063965)

[Equations 48](#_Toc166063966)

[Decomposition of program 48](#_Toc166063967)

[Usability Features 53](#_Toc166063968)

[UI Design 54](#_Toc166063969)

[Key Variable Table 55](#_Toc166063970)

[Validation Strategy 65](#_Toc166063971)

[Approach To Testing 66](#_Toc166063972)

[Test Strategy 66](#_Toc166063973)

[Implementation: Iteration 2 78](#_Toc166063974)

[Testing: Iteration 2 91](#_Toc166063975)

[Evaluation: Iteration 2 106](#_Toc166063976)

[Test Review 106](#_Toc166063977)

[Success Criteria evaluation of Iteration 2 106](#_Toc166063978)

[Client Feedback of Iteration 2 107](#_Toc166063979)

[My feedback of Iteration 2 107](#_Toc166063980)

[Iteration 2 Final Code 107](#_Toc166063981)

[Final Evaluation 109](#_Toc166063982)

[Final Success Criteria Evaluation 109](#_Toc166063983)

[Final Key Variable Table 110](#_Toc166063984)

[Final Usability Features 117](#_Toc166063985)

[Evaluation 117](#_Toc166063986)

[List of Entry Boxes (used for input at some point): 117](#_Toc166063987)

[List of Radio Buttons 117](#_Toc166063988)

[Maintenance 120](#_Toc166063989)

[Limitations and future developments 120](#_Toc166063990)

[Final client review 120](#_Toc166063991)

[Bibliography 121](#_Toc166063992)

[Appendices 122](#_Toc166063993)

[Appendix A – Iteration 1 Test Screenshots 123](#_Toc166063994)

[A1 123](#_Toc166063995)

[A2 123](#_Toc166063996)

[A3 123](#_Toc166063997)

[A4 123](#_Toc166063998)

[A5 123](#_Toc166063999)

[A6 123](#_Toc166064000)

[A7 123](#_Toc166064001)

[A8 123](#_Toc166064002)

[A9 123](#_Toc166064003)

[A10 123](#_Toc166064004)

[A11 124](#_Toc166064005)

[A12 124](#_Toc166064006)

[A13 124](#_Toc166064007)

[A14 124](#_Toc166064008)

[A15 124](#_Toc166064009)

[A16 124](#_Toc166064010)

[A17 124](#_Toc166064011)

[A18 124](#_Toc166064012)

[A19 124](#_Toc166064013)

[A20 125](#_Toc166064014)

[A21 125](#_Toc166064015)

[A22 125](#_Toc166064016)

[A23 125](#_Toc166064017)

[A24 125](#_Toc166064018)

[A25 125](#_Toc166064019)

[A26 125](#_Toc166064020)

[A27 125](#_Toc166064021)

[A28 126](#_Toc166064022)

[A29 126](#_Toc166064023)

[Appendix B – Iteration 1 Full Code 127](#_Toc166064024)

[Appendix C – Iteration 2 Test Screenshots 133](#_Toc166064025)

[C1 133](#_Toc166064026)

[C2 133](#_Toc166064027)

[C3 134](#_Toc166064028)

[C4 134](#_Toc166064029)

[C5 135](#_Toc166064030)

[C6 135](#_Toc166064031)

[C7 135](#_Toc166064032)

[C8 135](#_Toc166064033)

[C9 135](#_Toc166064034)

[C10 136](#_Toc166064035)

[C11 136](#_Toc166064036)

[C12 136](#_Toc166064037)

[C13 137](#_Toc166064038)

[C14 137](#_Toc166064039)

[C15 138](#_Toc166064040)

[C16 138](#_Toc166064041)

[C17 138](#_Toc166064042)

[C18 138](#_Toc166064043)

[C19 139](#_Toc166064044)

[C20 139](#_Toc166064045)

[C21 140](#_Toc166064046)

[C22 140](#_Toc166064047)

[C23 141](#_Toc166064048)

[C24 141](#_Toc166064049)

[C25 142](#_Toc166064050)

[C26 142](#_Toc166064051)

[C27 143](#_Toc166064052)

[C28 143](#_Toc166064053)

[C29 144](#_Toc166064054)

[C30 144](#_Toc166064055)

[C31 145](#_Toc166064056)

[C32 145](#_Toc166064057)

[C33 146](#_Toc166064058)

[C34 146](#_Toc166064059)

[C35 147](#_Toc166064060)

[C36 147](#_Toc166064061)

[C37 148](#_Toc166064062)

[C38 148](#_Toc166064063)

[C39 149](#_Toc166064064)

[C40 149](#_Toc166064065)

[C41 150](#_Toc166064066)

[C42 150](#_Toc166064067)

[C43 151](#_Toc166064068)

[C44 151](#_Toc166064069)

[C45 152](#_Toc166064070)

[C46 152](#_Toc166064071)

[C47 153](#_Toc166064072)

[C48 153](#_Toc166064073)

[C49 154](#_Toc166064074)

[C50 154](#_Toc166064075)

[Appendix D – iteration 2 Full Code 155](#_Toc166064076)

# Analysis

## Problem Identification

Waves are a big part of physics. There can be a big differentiation in what waves do and how they behave in relation to their properties. With them being taught from a young age up until they end education means there is a lot of content to cover, and it can take a lot of time and resources for the teachers to build the lesson plans. Whereas if there was a single source they could constantly refer to for their examples and allow the students to experiment with to further their understanding of the topic, this would take a lot of pressures off teachers and give them more time to be able to structure the learning around this source and more efficiently spend time in and out of the classroom.

The idea is to provide a program that can cover the waves topic up to an A-Level standard. With the idea being that teachers teaching levels all throughout high school can and will refer to a singular program that gives a theoretical and interactive experience with the topic. I plan to add different levels of learning so, for example, if it’s a Year 7 class that is being taught, the user can pick a simpler view of waves that won’t need to involve the extra details you’d get at A-Level Physics. It will also come with theoretical explanations for the various properties and contents on a basic level for the teacher to expand upon in lesson, but enough so that the program is intuitive and useable to younger years.

Being able to simulate this on a computer machine makes sense as most waves aren’t able to be seen in detail with the naked eye, so ‘in real life observation’ in a school setting isn’t a highly accessible option, and when it is it can be inaccurate subject to error, whereas a program is coded to be correct so unless students misuse the software, their experiments can be more accurate.

## Clients ID and Stakeholders

The client for the ‘Properties of Waves Simulator’ will be teachers, and particularly my Physics teacher Mr Holbrook, it’ll be used by her and her students to provide an interactive educational program to better understand the relationship between a wave’s properties (wavelength, frequency, amplitude etc) and will be a versatile and reliable way to teach the waves topics to students throughout the years (ages 12-18).

## Project Research

(Phet Interactive Simulations, 2022)

Phet Interactive – Wave on a String

Graphical user interface, chart

Description automatically generatedThe Wave on a String simulation crafted by Phet Interactive shows how features such as Amplitude, Frequency, Damping and Tension can all have effects on a wave as demonstrated by a string. Interactive elements such as sliders for the Amplitude, Frequency, Damping and Tension and buttons for various addons like timers, rulers and a reference line mean it has a lot of useable and interactivity with the user. Being able to pick through many options such as having a fixed, loose or no end to the user’s string means the movement of the wave can be showcased in more ways than if the user had no decision over it. Picking between different ‘customisations’ changing the visuals accordingly gives a clear and logical distinction between all the ‘modes’. Having methods where the user can manually move the end or choose between it automatically oscillating or having a pulse helps with the interactivity and further ways a wave can move.

Timeline

Description automatically generatedAlthough I believe the ‘String’ gimmick of the site is also holding it back as the website doesn’t demonstrate any sort of longitudinal waves and having settings such as ‘Damping’ and ‘Tension’ are specific to the scenario of a string and don’t reflect on the reality of ways through different properties.

The yellow colour scheme is simple and non-distracting and having objects such as wrenches and clamps cater to a younger audience by having tools they’ll recognise from real life.

Having buttons and sliders where appropriate do make for an easy and simple user experience with things being rather self-explanatory, except the actual concepts themselves.

Concepts and ideas that I’ve taken from this research is to have many customisable features like manual movement, pulses, different types of ends and features like rulers and timers to increase the interactivity and versatility of my program, though I may stay away from the real-life objects and string gimmick as I found them rather distracting and limiting.

(the Physics Classroom, 2022)

The Physics Classroom – Simple Wave Simulator

The Simple Wave Simulator by the Physics Classroom demonstrates the relationships between frequency, period, Wave Speed, Wavelength and amplitude in both longitudinal and transverse waves. It has a timer built in that is perpetually running so the user can see how the period of the wave translates to how the wave moves.

Table

Description automatically generated with low confidenceGraphical user interface, table

Description automatically generatedA grid background behind the wave can be used to show off the wave amplitude. They have sliders to control the Frequency, Wave Speed and they have a box showing the frequency, period, wave speed, wavelength and amplitude so the user can see how changing the sliders affects the other properties of the wave. However the sim is limited in the fact that the amplitude slider snaps to 0.5cm and can only be between 0.5cm – 2cm. The other sliders also snap to values though at a much larger range. However this means that this simulator isn’t useful for anything past vague experiments as an example of concept rather than a true wave model.

Having a button where the user can change between the ‘wave as rope’ and ‘wave as sound’ is handy as the user can see how the particles move in different ways between the two types of waves. And having buttons for ‘Pause’, ‘Slow motion’, ‘Real Time’ and ‘Fast Motion’ is also good as the user (teacher) can show students how they move and explain throughout various movement speeds. The colour scheme is simple with bold blues, oranges and greys. Blues and oranges make the boxes stand out but overall, it is a standard design.

The speed settings and the button to switch between transverse and longitudinal waves are the two best and most stand out features along with the statistics box that shows the relations between the settings. These will probably feature in my final program design.

(O'Neill, 2015)

AS/A Level Physics A (Second Edition) – Textbook

A white text with pink and white text

Description automatically generatedModule 4.4 and Module 4.5 in the A Level Physics Textbook are on waves and their properties and behaviours. For this reason, a lot of the mathematics and equations of my program will be referenced from this textbook. The book gives clear definitions for Wavelength, Time Period, Frequency, Displacement and Amplitude (as seen in Fig. to the right). As well as providing the equations that relate velocity, frequency and wavelength; and frequency and the time period. In the quantum physics module (4.5) it also gives an equation which relates the energy of each photon in the EM wave to the wave’s frequency. These equations will be used in the project as they will be the way the program will calculate the missing variables from the ones the user has inputted, and these variables will be used to create the visualisation of the wave that will feature in the program.

## Client Interview

J.B (Jake Brackley): What are the core principles of the program that are a requirement to include?  
P.H (Paul Holbrook): The program must include and have inputted/output: wavelength, velocity, period, amplitude, frequency.

J.B: Does the GUI of the program have to appeal to a certain audience? or do you have any specific preferences of the look?

P.H: The interface needs to ideally appeal to a teenage audience, between the ages of 12-17. It needs to be clear and readable for students with sight loss, not too harsh on sight as this will be used often by a varying range of characters. The program needs to be scalable to big and small screens as the use will vary from teacher presentations to homework set for students.

J.B: How would you like the user to be able to interact with the interface?

P.H: The user should be able type inputs into the program so they can have a near-unlimited reign to the variables of their wave. There should be info boxes you can hover over for pop ups that give information and explanation to the wave properties.

J.B: Do you have any preferences and features that will make the program stand out from others like itself?

P.H: Having a way to choose between different units for the variables, something I have found that often are missing from online programs of such, being able to demonstrate how some waves movements are better recorded in milliseconds than seconds and millihertz than hertz. This would excel the program and make it have much more practical uses in the classroom. It should also be able to provide with education and a way to access explanations of concepts shown in it, possibly with bubbles that explain how the variables relate to one another.

## Interview Analysis

From this interview I can now have a whole idea of to the requirements of my program and what preferences my client wants in the program.

My client’s requirements for the inputs and outputs of the program are wavelength, velocity, period, amplitude, frequency in the program. They want these to be inputted via entry boxes to have students have a limitless way to test the boundaries of reality and the theoretical. This means my program will make great use of a variety of variables and inputs, the graphical interface will also now have to be highly interactive and have a layer of depth and details beyond those comparable to it.

The program should have a way to be able to choose the units of each its variables and have a wider birth of inputs compared to other programs. This will mean I need a way in my code to be able to convert the inputted values between the orders of magnitude for almost every inputted and outputted variable in the program.

All this will mean that my program is going to have to use the graphical interface to its extents and I will need to find a way to balance all the information in a way it isn’t overwhelming and off-putting to the user. I may need to have multiple iterations for the two individual sections of my program as well so I can focus on having each part put to its limits and be as good as they can be when I put them together into one program.

I think the essential features will be my inputs, outputs, equations and having high usability with a concise GUI.

## Essential features of solution

* Inputs for wavelength, amplitude, frequency, and velocity and other key variables – This is for the simulating waves section so the user can implement and test against them against one another.
  + This is so the user can input their desired wave properties and interact with the program. These were prominent features of the programs in my research, although programs often only allowed 2 inputs. So, if my program is valid to input many different sets of variables then it will give the program more utility over others on market.
* An output on the desired values that weren’t inputted by the user.
  + The user needs to get outputted the results of their ‘experiments’ so they can view them and then change their inputs based on what they desire. This is something that I found from research to be common as the program will output the variables to the users to show the relationships between variables.
* A way to view the equations being used and the correlations between the properties investigated in the program.
  + So that the user can experiment with the waves but also learn about the physics behind it so it can be used as an educational program. This is something that I noticed was missing from programs in my research and I think would be a good part of an educational program.
* The program should have a visual representation of the wave.
  + This is something that was in both the programs I researched, and this makes the program seem more friendly to younger audiences, which is important as this program will be used with a plethora of age groups in school.

## Suitable approaches based off research

1st Approach:

Diagram

Description automatically generated with medium confidenceUsers can edit and interact with the waves’ variables, and it displays a table of statistics and properties. To input these variables (e.g. wavelength, frequency, period and wave speed) they will be a set of sliders, that the user can move and set between a min and max value and the table of statistics will update live alongside these user changes.

Throughout the program next to every variable name there should be a small ‘info’ box that the user can interact with, and it will output a text box that explains the variable and its relationship with the other variables in the program.

This would work as an approach as it includes all of the essential details and features of the project whilst still keeping it user friendly and without overwhelming details and distracting visuals.

2nd Approach:

Diagram

Description automatically generatedWhen you load the program you will see a stagnant line on the screen with a selection of sliders and input boxes, each corresponding to a variable that can be set (e.g. Wavelength, Frequency, Velocity etc.). These input box values will live update as the user edits the other input boxes or slides the sliders. The most ‘commonly taught’ variables like wavelength and wavespeed will be at the top for user accessibility. There should also be a graphic of the standing wave that moves in live relative accuracy to the properties that have been selected.

Throughout the program next to every variable name there should be a small ‘info’ box that the user can interact with and it will output a text box that explains the variable and its relation with the other variables in the program.

A settings box should also be available so the users can set any accessability preferences for the program.

## Proposed Solution

Approach 2 has been chosen for my basis as the project as I believe having the live graphics that update will be better suited to all ages as having a visual representation of gives the students a better practical idea of the variables they’re inputting and editing throughout. It provides all the necessary details and is simplistic enough to not be distracting yet engaging and ‘easy on the eye’ and not overwhelming with text and details (atleast not as the main attention of the audience)

## Limitations Of Proposed Solution

* Whilst I can code an operational, mathematic driven program in python – I’m still not super experienced and there will be a learning curve in how to create convincing graphics with the Tkinter library. This will add time onto the programming process as I’ll have to learn and research the best and most optimal ways to code in a functional graphical interface.
  + The GUI is needed so the program is more intuitive and easier to use for all users as they won’t be overwhelmed by the sight of a terminal window and things will be able to be laid out in a manner that makes sense to users.
* The mathematics and equations behind the refraction of waves can vary depending on a lot of minute variables, so to try and balance the program to have a sophisticated level of detail, whilst also not being overwhelming and confusing – the variables I do use are going to have to be carefully picked as to still have an accurate simulation that also doesn’t confuse users.
  + This is to keep the program accessible to the largest range of audience as it needs to be useful to younger years (down to about 12 y/o) all the way up to college/sixth form age (up to 18).

## Requirements

* Software and Hardware requirements for school system use:
  + Software:
    - OS – windows 10 (64-bit)
    - Python – IDLE
    - Python - Spyder
  + Hardware:
    - Intel i5-8500 @ 3.2 GHz
    - 8 GB RAM
    - SSD – 1.95 shared application drive
    - Onboard sound and graphics
    - Input – Mouse, keyboard
    - Outputs – Monitor
  + Justification:
    - These are bare minimum requirements as this is a program made for use within the school so it must be able to run upon the systems available throughout the school. IDLE is needed to be able to run the python code and execute the program.
* Software and Hardware requirements for home laptop use:
  + Software:
    - OS – windows 10 (64-bit)
    - Python – IDLE
  + Hardware:
    - Intel i7 @ 2.8GHz
    - 16 GB RAM
    - 550 GB SSD storage
    - Onboard sound and graphics
    - Inputs – Trackpad, mouse, keyboard, webcam, microphone
    - Outputs – Inbuilt monitor, Laptop docking station.
  + Justification:
    - These requirements cannot be surpassed as my laptop is what I’m going to code the majority of my program on and it’s a pretty standard laptop specs so it should mean my program should run smoothly on the majority of peoples personal devices.
* Software and Hardware requirements for home computer use:
  + Software:
    - OS – Windows 11 Pro (64-bit)
    - Python - Spyder
  + Hardware:
    - Intel i7-10700 @ 2.9GHz
    - 64 GB RAM
    - 1TB SSD storage
    - Onboard sound
    - ASUS ROG Strix 8gb - GPU
    - Inputs – Mouse, Keyboard, Webcam, Microphone
    - Outputs – 16:9 24” Monitor
  + Justification:
    - My home desktop is the other device a majority of my programming will take place and due to it’s high-end specs it should be able to run my program perfectly and it’s different aspect ratio of monitor will mean I will be able to view how the scaling of my programs GUI is working.

## Total Success Criteria

1. The program should have an input for wavelength.
   * This is as this is one of the independent variables the user will be able to input a value for that will be used by the program to calculate the dependent variables.
2. The program should have an output for wavelength.
   * This is as the user can decide to have the wavelength as a dependent variable, in which case the program will calculate and output this value back to the user using the inputs they used for the independent variables.
3. The program should have an input for velocity.
   * As this is one of the independent variables of the program that the user will be able to input a value for that will be used by the program to calculate the dependent variables.
4. The program should have an output for velocity.
   * This is as the user can decide to have the velocity as a dependent variable, in which case the program will calculate and output this value back to the user using the inputs they used for the independent variables.
5. The program should have an input for frequency.
   * This is one of the independent variables that the user will be able to input a value for and will be used by the program to calculate the dependent variables.
6. The program should have an output for frequency.
   * This is as the user can decide to have the frequency as a dependent variable, in which case the program will calculate and output this value back to the user using the inputs they used for the independent variables.
7. The program should have an input for amplitude.
   * This is one of the key independent variables of the wave that the user will be able to edit and see the impact it has on the wave graph’s y-axis.
8. The program should have an output for amplitude.
   * This is so the program can feed the amplitude value back to the user in a uniform manner with the rest of the independent and dependent variables of the wave.
9. The program should output other dependent variables of the wave.
   * This is to give the program a potential advantage over others such as the ones in my research as they often give limited information so I plan on outputting other variables with my program such as time period or photon energy.
10. The user should be able to choose which variables of wavelength, velocity and frequency are their desired dependent variable.
    * This is so that the program is able to calculate the dependent variable of the program based off the two independent variables inputted by the user.
11. The program should have preset wave variables that are based off common waves found in education.
    * As the users are likely to be younger students that are in school, being able to have presets of popular EM spectrum waves that the users may be familiar with the properties of would show users how the waves variables can affect the properties it holds.
12. The program should use the equations found in the textbook research to determine the dependent variables.
    * This is so the program can function as it is intended to by determining dependent variables of a wave from independent values provided by a user.
13. The program should have a functioning GUI that the user interacts with.
    * This is so the users can interact with the program without going through a python console window which can be confusing and a GUI means the information can be organized and interactive for the user to interact with.
14. The GUI should be able to fit on a plethora of different sized devices.
    * This is as the program is going to be used by a plethora of students of different ages they may have different devices such as computers or laptops with different sized peripherals, so having the program fit to size means it can be used on a wider range of devices.
15. The GUI should be easy to follow and use.
    * This is so students don’t get confused or potentially distracted when using the software in a classroom or educational setting.
16. The program should have a visualisation of a wave imbedded in the GUI.
    * This is so the user can see the effects of the changes they’re making to their variables and how it changes the visualisation of the wave.
17. The program should have information boxes that give user the scientific explanation behind variables.
    * This helps this be an educational software and gives explanation to the logic and calculations behind the variables to give users a better understanding of what’s happening.

## Which Developments Occur at each Iteration.

Iteration 1 will not have a tkinter GUI and will be a simple text input/output in the Python shell. This iteration will mainly be a learning curve of implementing the formulas in the correct ways and make sure the mathematics behind the variables is working. Establishing the key core features and variables of the program.

Iteration 2 will include the introduction of a GUI into my program. The GUI will be rough and potentially incomplete and will lead to further developments in the third and final iteration of my program.

Iteration 3 of the program should be the completed and final production of the project. It will be a culmination of all the clients features and have a completed GUI with extra features over its predecessor.

# Iteration 1

# Design: Iteration 1

## Success Criteria

1. The user has inputs for wavelength.
   * This is justified as this is one of the main variables of a wave that the user will be able to manipulate to form their waves. In my research I found past programs haven’t included all these properties so it will add an extra level of detail to my program.
2. The user has inputs for frequency.
   * This is justified as this is one of the main variables of a wave that the user will be able to manipulate to form their waves. In my research I found past programs haven’t included all these properties so it will add an extra level of detail to my program.
3. The user has inputs for velocity.
   * This is justified as this is one of the main variables of a wave that the user will be able to manipulate to form their waves. In my research I found past programs haven’t included all these properties so it will add an extra level of detail to my program.
4. The user has inputs for amplitude.
   * This is justified as this is one of the main variables of a wave that the user will be able to manipulate to form their waves. In my research I found past programs haven’t included all these properties so it will add an extra level of detail to my program.
5. The user needs an input to be able to decide which wave property they are inputting.
   * So the user can specify which variables are the known variables and be able to select which they can enter to be put through the program and have the correct unknowns calculated.
6. Uses correct equations to calculate the properties of the waves after select input properties are inputted.
   * This is as the purpose of the program is to give users the missing variables corresponding to the wave properties they’ve inputted, so using correct and appropriate equations will mean my program functions correctly.
7. The program has an output for wavelength.
   * This is needed as the purpose of the program is to output the user the unknown and calculated variables of the wave. My research shows that these are essential for programs of this sort.
8. The program has an output for frequency.
   * This is needed as the purpose of the program is to output the user the unknown and calculated variables of the wave. My research shows that these are essential for programs of this sort.
9. The program has an output for velocity.
   * This is needed as the purpose of the program is to output the user the unknown and calculated variables of the wave. My research shows that these are essential for programs of this sort.
10. The program has an output for amplitude.
    * This is needed as the purpose of the program is to output the user the unknown and calculated variables of the wave. My research shows that these are essential for programs of this sort.
11. The program has an output for time period.
    * This is needed as the purpose of the program is to output the user the unknown and calculated variables of the wave. My research shows that these are essential for programs of this sort.
12. Program should have fail safes in the case users have input errors.
    * In such a manually operated program (given the nature of having to watch spelling and data types when inputting) having fail safes to cover all bases of input will mean the user cannot break the program and end up in an inescapable error message and the program can flow smoothly.

## Equations

* wavelength = velocity / frequency
  + This is the equation I found in my research in (O'Neill, 2015), the A-Level physics textbook, this will be used to find the missing variable of the program that the user has not inputted.
* period = 1 / frequency
  + This was also found in (O'Neill, 2015), and is going to be used in my program so that it can output the time period of the wave that is being calculated and will help fulfil the success criteria of this iteration.

## Decomposition of program

### Whole program flow diagram

A diagram of a diagram

Description automatically generatedThis flowchart shows the general process of the whole program. It will ask the user to input the two independent variables and then the amplitude. It will then perform Calculation and Output (see Algorithm pseudocodes below), Which will determine the dependent variables and output them to the user. The program then loops so the user can determine another wave’s properties if they desire.

### Algorithm pseudocodes

|  |  |
| --- | --- |
| Algorithm Pseudocode | Justification |
| procedure Calculation()  global wlVAR  global velVAR  global freqVAR  global period  IF wlVAR > 0 THEN  IF freqVAR > 0 THEN  velVAR = wlVAR \* freqVAR  period = 1 / freqVAR  Output()  ELIF freqVAR = 0 THEN  freqVAR = velVAR / wlVAR  period = 1 / freqVAR  Output()  ENDIF  ELIF wlVAR = 0 THEN  wlVAR = velVAR / freqVAR  period = 1 / freqVAR  Output()  ENDIF  endprocedure | This is the procedure that calculates the dependent variable of the program. This covers Success Criteria 6 as it uses the equations from the textbook to deduce the dependent variable that the user will be using the program to calculate. The procedure starts by loading the global variables into the procedure it will then deduce which variables the user inputted into the software by checking which variables have greater values than 0. So if wlVAR and freqVAR are greater than 0, that means the program needs to calculate velocity first. If wlVAR and velVAR are greater than 0, then it needs to calculate frequency. Then if wlVAR is 0, that means it needs to be calculated. So it will calculate the needed variable and the time period of the wave and then it will run the Output() procedure. |
| procedure Output()  global wlVAR, velVAR, freqVAR, ampVAR, period  print wlVAR, velVAR, freqVAR, ampVAR, period  endprocedure | This is the procedure that will print and output the variables into the console. This satisfies Success Criteria 7 to 11 as it will output all the key variables containing the properties of the users desired wave. It will load the global variables for the properties of the wave and then it will print them all out so the user can review the variables they entered and see the dependent variable that was calculated. |
| function Validation(inpMessage)  while True:  userInp = float(input(inpMessage))  IF error THEN  output("Please input valid value")  continue  ELSE  IF userInp > 0 THEN  return userInp  ELSE  output("enter value greater than 0")  continue  ENDIF  ENDIF  endwhile  endfunction | This function ensures that the value inputted by the user is a valid float value. This satisfies Success Criteria 12 as this means a user cannot input a value that would provide an error message if it was used in the maths in the calculation procedure. This function will take a user’s input and attempt to cast the string input as a floating point value. If this would then halt the program with an error it will instead redirect the program through the IF statement which will give an output prompting the user what they entered is not a valid input and then redirect the program back to the top of the while True loop. If the value is successfully casted to be a float the if statement then will make sure the value is greater than 0 as this is an erroneous value for a wave property. Then if the input fits these criteria, the value is returned back to the main program to be stored in the variable. |

## Usability Features

* ‘option’ input; is going to be the input that runs the users through the order of the program and inputting the variables.
  + This is an essential input as it guides the user through the order of variables to input and will eventually execute the calculate function which is the total point of the program.
* Wavelength input.
  + This is needed as wavelength is one of the variables that users can alter (Success Criteria 1) and will mean that my program offers more useability than others online and of public use.
* Velocity input.
  + This is needed as velocity is one of the variables that users can alter (Success Criteria 3), and this is often a key variable in the programs I researched.
* Frequency input.
  + This is needed as frequency is one of the variables that users can alter (Success Criteria 2), and this is often a key variable in the programs I researched.
* Amplitude input.
  + This is needed as amplitude is one of the variables that users can alter (Success Criteria 4), and can be picked to affect the desired wave.

## Key Variable Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable / Structure  name | Data type | Local or global | Description of what stored | Where used | Justification and Success Criteria |
| wlVAR | Float | Global | The wavelength of the desired wave | Main Program, Calculation, printOutput | I need the variable to satisfy Success Criteria 1 and 7, as it will store the wavelength of the wave on display as it is one of the key factors in a wave’s composition and will be one of the factors the user can control so I need a way to store their desired value or the calculated value by the program. |
| velVAR | Float | Global | The velocity of the desired wave | Main Program, Calculation, printOutput | I need the variable to satisfy Success Criteria 3 and 9, as it will store the velocity of the wave on display as it is one of the key factors in a wave’s composition and will be one of the factors the user can control so I need a way to store their desired value or the calculated value by the program. |
| period | Float | Global | The period of the desired wave | Main Program, Calculation, printOutput | This will satisfy Success Criteria’s 6 and 11 as period needs to be stored as it’ll be calculated by the program and be outputted as apart of the program. |
| freqVAR | Float | Global | The frequency of the desired wave | Main Program, Calculation, printOutput | I need the variable to satisfy Success Criteria 2 and 8, as it will store the frequency of the wave on display as it is one of the key factors in a wave’s composition and will be one of the factors the user can control so I need a way to store their desired value or the calculated value by the program. |
| ampVAR | Float | Global | The amplitude of the desired wave | Main Program, Calculation, printOutput | I need the variable to satisfy Success Criteria 4 and 10, as it will store the amplitude of the wave on display as it is one of the key factors in a wave’s composition and will be one of the factors the user can control so I need a way to store their desired value or the calculated value by the program. |
| option | String | Local | The users input on what variable they are editing | Main Program | This is satisfying Success Criteria 5, as it is what will be the input where the users will decide which variable they are editing with their input |
| varInput | Float | Local | The user’s input | inputFloat | This is important as it satisfies Success Criteria 8, as it is the variable that will store the users input whilst in the varInput function as it will attempt to cast the string to a float to see if it’s an allowed casting. |

## Validation Strategy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Validation entry | Description of what could go wrong | Strategy | Return message | Justification |
| Enter wavelength, velocity or frequency | Could accept something other than a way to indicate to those 3 variables | Make sure the user only enters something appropriate to one of those 3 variables | Please enter one of wavelength, velocity, or frequency | This is so the user doesn’t return with an error message and stopping the program by entering an invalid input into the first input |
| Enter wavelength | Program could try to process an input that cannot be used in the mathematics of the program which would then return an error and stop the program | Make sure user only inputs something that can be casted as a floating-point data type | Please enter numerical value | This is so the user doesn’t return with an error message and stopping the program by entering an invalid input that won’t work with the calculations |
| Enter velocity | Program could try to process an input that cannot be used in the mathematics of the program which would then return an error and stop the program | Make sure user only inputs something that can be casted as a floating-point data type | Please enter numerical value | This is so the user doesn’t return with an error message and stopping the program by entering an invalid input that won’t work with the calculations |
| Enter frequency | Program could try to process an input that cannot be used in the mathematics of the program which would then return an error and stop the program | Make sure user only inputs something that can be casted as a floating-point data type | Please enter numerical value | This is so the user doesn’t return with an error message and stopping the program by entering an invalid input that won’t work with the calculations |
| Enter amplitude | Program could try to process an input that cannot be used in the mathematics of the program which would then return an error and stop the program | Make sure user only inputs something that can be casted as a floating-point data type | Please enter numerical value | This is so the user doesn’t return with an error message and stopping the program by entering an invalid input that won’t work with the calculations |

## Approach To Testing

As this iteration is just a simple text input and text output the tests will be majority about testing the validation capabilities of my code, and then the mathematical accuracy. The first large set of tests will be validating all my various input points, e.g. selecting which variable the user is inputting into, the tests will be of all the accepted inputs and then of miscellaneous inputs to make sure the correct validation code runs and returns an error message within my code and not one from python halting my program, this is to make sure the users experience is seamless within the program and it doesn’t need to be re-run with potential user error that is made. The next set of tests after that will be the validation of my variable inputs, making sure they only accept inputs that can be casted as a float and that if not, the appropriate code is ran and if so that the variable is then stored properly ready for use in the calculations. This is to make sure that the program is properly storing the values of its variables as if it were to store something incorrectly it could lead to the program halting due to a syntax error attempting to run the mathematics or can lead to a logical error and get the incorrect results outputted to the user from the calculations.

## Test Strategy

Test numbers denoted as ‘D1 – (test number)’, meaning D1 = design iteration 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | What is being tested? | Test Data | Type of test  (Normal, Boundary, Erroneous) | Test description | What should happen | Justification |
| D1-1 | ‘option’ input | Wavelength | Normal | Entering ‘Wavelength’ into the option input in the program | It should accept the input and prompt the user to enter the wlVAR input. | Tests Success Criteria 1, 5 and 12. Some users may instinctively capitalise ‘Wavelength’ when entering it so having this means that the user won’t be penalised for an input that is the correct input in the incorrect case. |
| D1-2 | ‘option’ input | wavelength | Normal | Entering ‘wavelength’ into the first program input. | It should accept the input and prompt the entry of the wlVAR. | Tests Success Criteria 1, 5 and 12. Some users may not capitalise ‘wavelength’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-3 | ‘option’ input | W | Normal | Entering ‘W’ into the first program input. | It should accept the input and prompt the entry of the wlVAR. | Tests Success Criteria 1 and 12. Some users may capitalise the entry of ‘W’ referring to wavelength so having this means that the user won’t be penalised for a logical input |
| D1-4 | ‘option’ input | w | Normal | Entering ‘w’ into the first program input. | It should accept the input and prompt the entry of the wlVAR. | Tests Success Criteria 2, 5 and 12. Some users may not capitalise ‘w’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-5 | ‘option’ input | Frequency | Normal | Entering ‘Frequency’ into the first program input. | It should accept the input and prompt the entry of the freqVAR. | Tests Success Criteria 2, 5 and 12. Tests Success Criteria 1, 5 and 12. Some users may instinctively capitalise ‘Frequency’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-6 | ‘option’ input | frequency | Normal | Entering ‘frequency’ into the first program input. | It should accept the input and prompt the entry of the freqVAR. | Tests Success Criteria 2, 5 and 12. Some users may not capitalise ‘frequency’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-7 | ‘option’ input | F | Normal | Entering ‘F’ into the first program input. | It should accept the input and prompt the entry of the freqVAR. | Tests Success Criteria 2, 5 and 12. Some users may capitalise the entry of ‘F’ referring to frequency so having this means that the user won’t be penalised for a logical input |
| D1-8 | ‘option’ input | f | Normal | Entering ‘f’ into the first program input. | It should accept the input and prompt the entry of the freqVAR. | Tests Success Criteria 2, 5 and 12. Some users may not capitalise ‘f’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-9 | ‘option’ input | Velocity | Normal | Entering ‘Velocity’ into the first program input. | It should accept the input and prompt the entry of the velVAR. | Tests Success Criteria 3, 5 and 12. Some users may capitalise the entry of ‘Velocity’ so having this means that the user won’t be penalised for a logical input |
| D1-10 | ‘option’ input | velocity | Normal | Entering ‘velocity’ into the first program input. | It should accept the input and prompt the entry of the velVAR. | Tests Success Criteria 3, 5 and 12. Some users may not capitalise ‘velocity’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-11 | ‘option’ input | V | Normal | Entering ‘V’ into the first program input. | It should accept the input and prompt the entry of the velVAR. | Tests Success Criteria 3, 5 and 12. Some users may capitalise the entry of ‘V’ so having this means that the user won’t be penalised for a logical input |
| D1-12 | ‘option’ input | v | Normal | Entering ‘v’ into the first program input. | It should accept the input and prompt the entry of the velVAR. | Tests Success Criteria 3, 5 and 12. Some users may not capitalise ‘v’ when entering it so having this means that the user won’t be penalised for a logical input |
| D1-13 | ‘option’ input |  | Erroneous | Leaving input blank into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may leave entry blank accidentally |
| D1-14 | ‘option’ input | True | Erroneous | Entering Boolean value into input into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may accidentally input an incorrect data type in the entry |
| D1-15 | ‘option’ input | 15 | Erroneous | Entering an integer into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may accidentally input an incorrect data type in the entry |
| D1-16 | ‘option’ input | hello | Erroneous | Entering an invalid string input into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-17 | ‘option’ input | wavelengt | Boundary /Erroneous | Entering an almost accepted value ‘wavelength’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-18 | ‘option’ input | wavelengthf | Boundary /Erroneous | Entering an almost accepted value ‘wavelength’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-19 | ‘option’ input | frequenc | Boundary /Erroneous | Entering an almost accepted value ‘frequency’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-20 | ‘option’ input | frequencyf | Boundary /Erroneous | Entering an almost accepted value ‘frequency’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-21 | ‘option’ input | velocit | Boundary /Erroneous | Entering an almost accepted value ‘velocity’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-22 | ‘option’ input | velocityf | Boundary /Erroneous | Entering an almost accepted value ‘velocity’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-23 | ‘option’ input | ww | Boundary /Erroneous | Entering an almost accepted value ‘w’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-24 | ‘option’ input | ff | Boundary /Erroneous | Entering an almost accepted value ‘f’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-25 | ‘option’ input | vv | Boundary /Erroneous | Entering an almost accepted value ‘v’ into the first input | Print a line of validation reiterating what to input and repeating loop | Tests Success Criteria 5 and 12. Users may misspell or write the incorrect thing at the incorrect stage in inputs |
| D1-26 | ‘inputFloat’ input | 15 | Normal | Entering a valid input into the inputFloat function of my program to ensure it works | Program should accept value | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |
| D1-27 | ‘inputFloat’ input | 5.5 | Normal | Entering a valid input into the inputFloat function of my program to ensure it works | Program should accept value | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |
| D1-28 | ‘inputFloat’ input | hello | Erroneous | Entering an invalid input into the inputFloat function of my program to ensure it works | Program should return message and loop to input accepted value | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |
| D1-29 | ‘inputFloat’ input |  | Erroneous | Entering an invalid input into the inputFloat function of my program to ensure it works | Program should return message and loop to input accepted value | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |
| D1-30 | ‘inputFloat’ input | -15 | Erroneous | Entering a value that works in the float detection part of the function but not the part limiting the input to above 0 | Program should return ‘please enter numerical value greater than 0’ and loop | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |
| D1-31 | ‘inputFloat’ input | 0 | Boundary | Entering the minimum possible value accepted by the function, yet should be rejected as it isn’t above 0 | Program should return ‘please enter numerical value greater than 0’ and loop | Tests all Success Criteria 1 – 4 and 12. As all my numerical inputs are run through this single function only one set of testing should be necessary to make sure that the float casting and value check are working |

# Implementation: Iteration 1

A screenshot of a computer program

Description automatically generatedFirst the global variables for the waves properties that can be input and manipulated by the user needed to be declared so they can be used throughout the programs various procedures and functions.

A computer code with text

Description automatically generated with medium confidence

A white background with black text

Description automatically generatedThe first part of the program that was coded was the function for the whole validation of the program (Success Criteria 12) of when the user inputs values for the wavelength properties. This was necessary as if it was not used then if the user misinputs a non-float value the program would produce an error message which would halt the program. This function attempts to cast the users input as a float, then if this were to produce an error message it will intercept and instead continue the program loop and allow the user to reattempt entering a value. If the casting would not produce an error (e.g. the user entered an integer or float) then the program would check that the value of the input is greater than zero and then return the input value or continue the program loop respectively.

A white background with red text

Description automatically generatedThis is how the output of this function looks in practice. As you can see it successfully keeps the program loop running without causing an error that would’ve halted the program running if it had tried to use an erroneous non float value in a calculation later.

The next part of the program that was coded was the procedure for outputting the final variables. (Success Criteria 7-11)

A close-up of a computer screen

Description automatically generated

This was simple as the global variables for the outputs had already been declared and I knew later in the program the values would be wrote into these variables, so I coded this pre-emptively to be used for later after the calculations. The output is simply several print statements which will tell the user which variable is attributed to which value.

A white background with blue text

Description automatically generatedThis is the resulting output of inputting 5 for the wavelength, 10 for the velocity and 5 for the amplitude. This is simply and conveys all the important data to the user in the output and shows the calculated dependent variable and period to the user.

A screenshot of a computer code

Description automatically generated

A white background with black text

Description automatically generatedThe main build of the program is all built within the ‘while noEnt < 2’ loop as this means the code will loop indefinitely meaning the user can input multiple different sets of wave properties and have different results outputted for each set of variables inputted. The program will ask the user and take the input for what variable they want to input for (wavelength, velocity or frequency) (Success Criteria’s 1-3 and 5) and then the if statement will determine which one of these was inputted with an ’else’ clause used as defensive design (Success Criteria 12) in the case the user inputs an invalid value. The program then uses the coded ‘inputVARFloat’ function to input floating point values for the mathematics into the desired variable. The ‘noEnt’ is then incremented and the loop is continued under the parameter that ‘noEnt’ is now equal to 1 and will run the next part of the program. There is a large amount of outputs in this part of the program that outputs the backend variables to make the process easier to keep track of during testing.

A close up of words

Description automatically generatedThese screenshots show the result to a user typing either ‘wavelength’ or ‘hello’ into this section of code. ‘hello’ is an erroneous value and hence the else clause of the if statement triggered leading a loop of the while loop as well as an outputted error message. ‘wavelength’ is an accepted value by the program and hence prompts the user to input a desired value into the code and for testing purposes the program also prints the prevOp function as to make the program’s code easier to follow during the testing phase as this iteration is not designed for public use.

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generatedAfter this step in the program it made most sense to code the next step of the program logically. The ‘elif noEnt == 1’ is used to after the previous step when noEnt was incremented and the loop was continued this part of the if statement is used. Imbedded another if statement in this that checks the value of the prevOp variable is crucial as this will mean that the user cannot create an error such as entering in 2 values for wavelength and breaking the program mathematically causing an error (Success Criteria 12). This also means that the section of code in the above screenshot is essentially copied and altered for 3 times for the 3 potential prevOp values (wavelength (wl), velocity (vel), frequency (freq)) as pictured below. Inside this if statement is a section of code similar to the first iteration of this while loop in which it asks for a input of which the result is run through an if statement to see which variable the user wants to alter. The difference in this iteration is dependent on the prevOp if statement where if it was wavelength then it would only offer an input between frequency and velocity. After this it will then offer the user to input an amplitude (Success Criteria 4) as this is a variable independent of the other 3, hence not being able to be calculated. And it then runs the Calculation procedure and sets noEnt to 0, restarting the while loop. The prevOp if statement also has an else clause for defensive design although this should be impossible to trigger it is defensive design to re attempt the while loop. That means the input of the program was completed and by the calling of Calculation in the last section, this begins the A white background with blue text

Description automatically generatedrunning of the mathematics of the program.

This is a result of the above code if the user was to insert ‘velocity’ when ‘prevOp = wl’. It will then ask the user to input a value for velocity then again a value for amplitude. The reason the program asks for amplitude at the end if due to this being a separate variable from the other 3 properties of a wave and isn’t related by any equation. The program then prints ‘Calculating…’ as another landmark output for testing purposes despite the calculations happening in such a short time period there isn’t a real load time.

A white text with red and black text

Description automatically generated

Starting the procedure by declaring the variables to be used in the calculations. Then, for efficiency, I figured out there can only be 3 different combinations of inputted variables when 2 out of 3 variables have been inputted. Hence the selection of if statements leading to 3 potential outcomes. Using the equations found from (O'Neill, 2015) the program simply calculates the missing 3rd variable as well as the waves period (Success Criteria 6). This procedure then calls the previously made output procedure which will run and then the while loop will iterate and the user will be able to reinput the values for a new wave if they may. This means users can compare the properties of a wave without having to reboot the program.

A close-up of numbers

Description automatically generatedThis is an example of the calculation procedure being used when wavelength != 0 and velocity != 0, so the else clause of the imbedded if statement. This means freqVAR is found by 14/89 which is 0.157 (to 3 s.f.) and period is calculated by 1 / 0.157, which is 6.357 (to 3 s.f.). So the manipulation of the provided equations is working and being outputted correctly.

# Testing: Iteration 1

Test numbers denoted as ‘T1 – (test number)’, meaning T1 = testing iteration 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Type of test | Description | What should happen | What did happen | Pass /Fail | Screen shot No. |
| T1-1 | “ “ | Erroneous | Testing to see if when the user leaves the input (Line 54) blank, the iterative loop loops. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A1](#_B1) |
| T1-2 | W | Normal | Testing to see if the .lower() command and the input (Lines 54-56) works correctly. | It should prompt the user to input a value for the wavelength. | It prompted the user to enter a value for the wavelength. | Pass | [A2](#_B2) |
| T1-3 | Wavelength | Normal | Testing to see if the .lower() command and the input (Lines 54-56) works correctly. | It should prompt the user to input a value for the wavelength. | It prompted the user to enter a value for the wavelength. | Pass | [A3](#_B3) |
| T1-4 | V | Normal | Testing to see if the .lower() command and the input (Lines 54-55, 72) works correctly. | It should prompt the user to input a value for the velocity. | It prompted the user to enter for the velocity. | Pass | [A4](#_B4) |
| T1-5 | Velocity | Normal | Testing to see if the .lower() command and the input (Lines 54-55, 72) works correctly. | It should prompt the user to input a value for the velocity. | It prompted the user to enter for the velocity. | Pass | [A5](#_B5) |
| T1-6 | F | Normal | Testing to see if the .lower() command and the input (Lines 54-55, 64) works correctly. | It should prompt the user to input a value for the frequency. | It prompted the user to enter for the frequency. | Pass | [A6](#_B6) |
| T1-7 | Frequency | Normal | Testing to see if the .lower() command and the input (Lines 54-55, 64) works correctly. | It should prompt the user to input a value for the frequency. | It prompted the user to enter for the frequency. | Pass | [A7](#_B7) |
| T1-8 | 72 | Erroneous | Testing to see if the input (Line 54) will accept an erroneous integer value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A8](#_B8) |
| T1-9 | 0.432 | Erroneous | Testing to see if the input (Line 54) will accept an erroneous float value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A9](#_B9) |
| T1-10 | False | Erroneous | Testing to see if the input (Line 54) will accept an erroneous Boolean value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A10](#_B10) |
| T1-11 | Hello | Erroneous | Testing the inputVARFloat function (Line 7-20). | It should prompt the user to enter an input of the correct data type and loop the program | It prompted the user to enter a valid input and looped the program. | Pass | [A11](#_B11) |
| T1-12 |  | Erroneous | Testing the inputVARFloat function (Line 7-20). | It should prompt the user to enter an input of the correct data type and loop the program | It prompted the user to enter a valid input and looped the program. | Pass | [A12](#_B12) |
| T1-13 | True | Erroneous | Testing the inputVARFloat function (Line 7-20). | It should prompt the user to enter an input of the correct data type and loop the program | It prompted the user to enter a valid input and looped the program. | Pass | [A13](#_B13) |
| T1-14 | 0 | Erroneous/Boundary | Testing the inputVARFloat function (Line 7-20). | Prompt the user to enter a value above 0 and loop the program | Prompted the user to enter a value greater than 0 and looped the program | Pass | [A14](#_B14) |
| T1-15 | 0.000001 | Boundary | Testing the inputVARFloat function (Line 7-20). | It should accept the value and continue the program | It accepted the value and continued the program | Pass | [A15](#_B15) |
| T1-16 | 16 | Normal | Testing the inputVARFloat function (Line 7-20). | It should accept the value and continue the program | It accepted the value and continued the program | Pass | [A16](#_B16) |
| T1-17 | 20.52523 | Normal | Testing the inputVARFloat function (Line 7-20). | It should accept the value and continue the program | It accepted the value and continued the program | Pass | [A17](#_B17) |
| T1-18 | “ “ | Erroneous | Testing to see if when the user leaves the input (Line 85) blank, the iterative loop loops. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A18](#_B18) |
| T1-19 | V | Normal | Testing to see if the .lower() command and the input (Lines 85-86, 95) works correctly. | It should prompt the user to input a value for the velocity. | It prompted the user to enter for the velocity. | Pass | [A19](#_B19) |
| T1-20 | Velocity | Normal | Testing to see if the .lower() command and the input (Lines 85-86, 95) works correctly. | It should prompt the user to input a value for the velocity. | It prompted the user to enter for the velocity. | Pass | [A20](#_B20) |
| T1-21 | F | Normal | Testing to see if the .lower() command and the input (Lines 85-87) works correctly. | It should prompt the user to input a value for the frequency. | It prompted the user to enter for the frequency. | Pass | [A21](#_B21) |
| T1-22 | Frequency | Normal | Testing to see if the .lower() command and the input (Lines 85-87)works correctly. | It should prompt the user to input a value for the frequency. | It prompted the user to enter for the frequency. | Pass | [A22](#_B22) |
| T1-23 | 72 | Erroneous | Testing to see if the input (Line 85) will accept an erroneous integer value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A23](#_B23) |
| T1-24 | 0.432 | Erroneous | Testing to see if the input (Line 85) will accept an erroneous float value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A24](#_B24) |
| T1-25 | False | Erroneous | Testing to see if the input (Line 85) will accept an erroneous Boolean value. | It should prompt the user to enter a valid input and loop the program. | It prompted the user to enter a valid input and looped the program. | Pass | [A25](#_B25) |
| T1-26 | -15 | Erroneous | Testing the inputVARFloat function (Line 7-20). | It should prompt the user to enter a value above 0 and loop program. | It prompted the user to enter a value above 0 and loop program. | Pass | [A26](#_B26) |
| T1-27 | 10,100,  15 | Normal | Testing the Calculation procedure and it produces the correct number. (Wl = 10, Vel = 100, Amp = 15) | The program should calculate frequency as 10Hz and period as 0.1s and output all these values correctly. | It correctly calculated 10Hz, 0.1s and outputted these values | Pass | [A27](#_B27) |
| T1-28 | 25.5, 5, 100 | Normal | Testing the Calculation procedure and it produces the correct number. (Freq = 25.5, Vel = 5, Amp = 100) | The program should calculate wavelength as 0.1960784m and period as 0.039216s and output all these values correctly. | It correctly calculated the values for wavelength and period and outputted these values | Pass | [A28](#_B28) |
| T1-29 | 5.553, 10.123, 0.00001 | Normal | Testing the Calculation procedure and it produces the correct number (Wl = 5.553, Freq = 10.123, Amp = 0.00001) | The program should calculate velocity as 56.213 m/s and period as 0.098784s and output all these values correctly. | It correctly calculated the values for wavelength and period and outputted these values | Pass | [A29](#_B29) |

As all the tests passed first try there was no need to perform any retests during this iteration. I ended up having less tests in my main testing than my strategy due to having less expected inputs after optimising.

# Evaluation: Iteration 1

## Testing Review

In the testing of this program there were no failed tests, this is good as it means that the main crux of the calculations of the final program are going to work, as the calculation procedure of the program is essentially going to be the same in the final program as this program.

## Success Criteria evaluation of Iteration 1

|  |  |  |
| --- | --- | --- |
| Success Criteria | Pass or Fail | Evidence of Pass or Fail |
| Criteria 1 | Pass – There is a valid way for the user to input a value for the desired wavelength of the wave. | T1-2, T2-3, T1-16, T1-17 |
| Criteria 2 | Pass – There is a valid way for the user to input a value for the desired frequency of the wave. | T1-6, T2-7, T1-16, T1-17 |
| Criteria 3 | Pass – There is a valid way for the user to input a value for the desired velocity of the wave. | T1-4, T2-5, T1-16, T1-17 |
| Criteria 4 | Pass – There is a valid way for the user to input a value for the desired amplitude of the wave. | T1-16, T1-17 (as the amplitude input calls inputVARFloat at separate points) |
| Criteria 5 | Pass – The ‘option’ variable allows the user to input what variable they are inputting a value for. | T1-1, T1-2, T1-3, T1-4, T1-5, T1-6, T1-7, T1-8, T1-9, T1-10 |
| Criteria 6 | Pass – The program uses the correct equations to produce the desired results from the inputs the user has inputted. | T1-27, T1-28, T1-29 |
| Criteria 7 | Pass – The program has an output for the calculated value of wavelength using the correct equation. | T1-27, T1-28, T1-29 |
| Criteria 8 | Pass – The program has an output for the calculated value of frequency using the correct equation. | T1-27, T1-28, T1-29 |
| Criteria 9 | Pass – The program has an output for the calculated value of velocity using the correct equation. | T1-27, T1-28, T1-29 |
| Criteria 10 | Pass – The program has an output for the user inputted value of the amplitude. | T1-27, T1-28, T1-29 |
| Criteria 11 | Pass – The program outputs a calculated value for time period using the necessary equation. | T1-27, T1-28, T1-29 |
| Criteria 12 | Pass – The program is crafted inside a While loop which keeps the program from crashing and producing error messages, and the ‘InputVARFloat’ validates the users input to keep it as an integer or float value. As well as having multiple else statements for defensive design in its if statements | Every erroneous test will be testing defensive design as the program did not halt.  T1-1, T1-8, T1-9. T1-10, T1-11, T1-12, T1-13, T1-14, T1-15, T1-18, T1-23, T1-24, T1-25, T1-26 |

## Client feedback of Iteration 1

The client is pleased that all success criteria has been met. All calculations performed by the program have given correct results and the program is robust and is a good proof of concept for the envisioned final product.

## My feedback of Iteration 1

As my client is happy with the program and it has successfully reached all of the success criteria of the iteration, I now believe I am ready to move onto iteration 2 which will include creating a tkinter GUI for the program.

## Iteration 1 Final Code

[Appendix B – Iteration 1 Full Code](#_Appendix_C_–)

# Iteration 2

# Design: Iteration 2

## Success Criteria

1. The user has inputs for frequency.
   * This is needed as this is one of the dependent variables a user can input to find the independent variable of the wave and in my research, I found this is a common input in similar programs.
2. The user has inputs for velocity.
   * This is needed as this is one of the dependent variables a user can input to find the independent variable of the wave and in my research, I found this is a common input in similar programs.
3. The user has inputs for amplitude.
   * This is useful as it is an independent variable of a wave and will affect the look of the wave visualiser. This was a commonly found input in my research however usually using limited sliders so using a text input means this program will be more versatile with its input.
4. The user has inputs for wavelength.
   * This is needed as this is one of the dependent variables a user can input to find the independent variable of the wave and in my research, I found this is a common input in similar programs.
5. The program should output wavelength.
   * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was a set back in other programs that my client thinks would be beneficial in all ages of learning.
6. The program should output frequency.
   * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was a set back in other programs that my client thinks would be beneficial in all ages of learning.
7. The program should output velocity.
   * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was something missing in other programs that my client thinks would be beneficial in all ages of learning.
8. The program should output amplitude.
   * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was a set back in other programs that my client thinks would be beneficial in all ages of learning.
9. The program should output time period.
   * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was a set back in other programs that my client thinks would be beneficial in all ages of learning.
10. The program should output a number of other undecided dependent variables to the user.
    * This program is going to have a dedicated output box which will output the values of all variables of the wave, this is necessary as it will provide a clear way to present the information so the program can be used easier for educational purposes, in my research this was a set back in other programs that my client thinks would be beneficial in all ages of learning.
11. The program should have a GUI.
    * This is justified as the client wants the user to interact with the program with a GUI to make it easier to use with students.
12. Uses correct equations to calculate the properties of the waves after select input properties are inputted.
    * This is needed so the user can use the program to find accurate unknowns of their desired waves.
13. The window size of the program’s GUI should be big enough to take up a whole screen size.
    * This means the program should be easy to read as the inputs and outputs should be large enough and means the user experience of the program will be better. In my research, this is not apparent as most programs are inbuilt into webpages, this is a downside as some students may have seeing difficulties and this means this program can be used on a big monitor and be better saw around a classroom setting.
14. The program should have presets available for the wave’s properties for the user to select.
    * This is a good addition as it will give students presets to waves they would typically learn about on their courses and give them examples of how the program shows waves that they are familiar with. This was missing in programs I researched and something that my client would find useful in a classroom scenario.
15. The GUI should be logically designed and follow my clients wants to show a rough example of the final product.
    * Necessary as my client has said the program will be used for all year groups in physics so the program needs to be intuitive for youngers however still be able to give adequate information for older years.
16. The program should have a level of customisation with the inputs.
    * In the client interview my client said that something lacking from existing programs is a variation in units being able to be used so this program should be able to have the units of a few variables be picked and chosen for the inputs.
17. The program should have a visualisation of the wave graphed onto the GUI.
    * This is so the user can see what the different independent variables have effects on the wave.
18. The client should be able to select the desired dependent variable out of wavelength, velocity and frequency.
    * This is as the program is designed to determine the dependent variables of the users desired wave and one of these three is dependent on the other two.

## Equations

* wavelength = velocity / frequency
  + This is the equation I found in my research in (O'Neill, 2015), the A-Level physics textbook, this will be used to find the missing variable of the program that the user has not inputted.
* period = 1 / frequency
  + This was also found in (O'Neill, 2015), and is going to be used in my program so that it can output the time period of the wave that is being calculated and will help fulfil the success criteria of this iteration.
* energy of the individual photons = Planck’s constant \* frequency
  + This is the equation I found in my research in (O'Neill, 2015), the A-Level physics textbook, this will be used to find the energy of the individual photons of the wave.

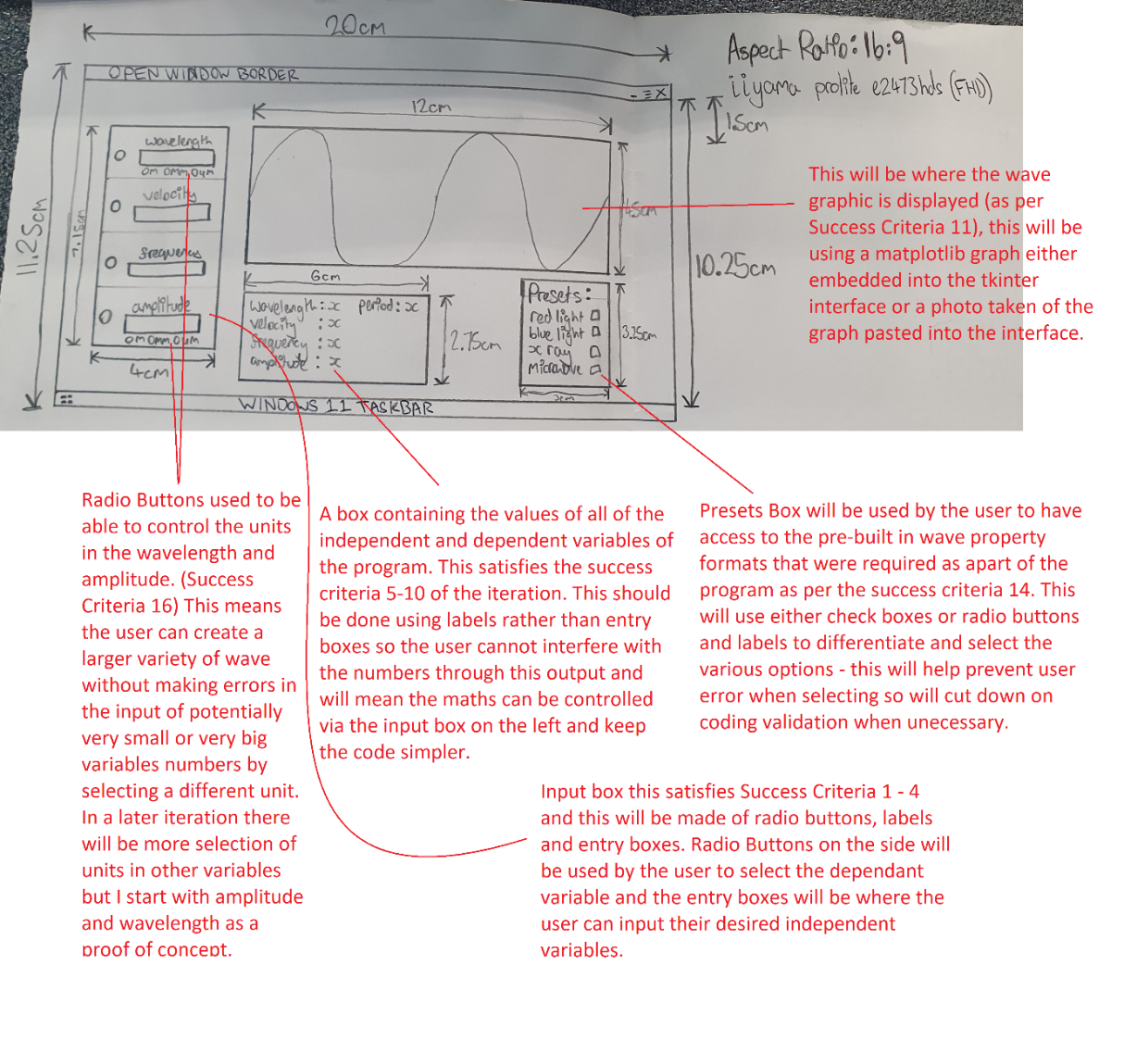
## Decomposition of program

|  |  |
| --- | --- |
| Algorithm Pseudocode | Justification |
| procedure getWavelength  global wlVAR  global wlUnit  wlVAR = input(wlEntry)  wlUnit = input(wlRadioButtons)  IF wlUnit = "meter" THEN  wlCalculation()  ELIF wlUnit = "nano meter" THEN  wlVAR = wlVAR \* (10^-9)  wlCalculation()  ELIF wlUnit = "micro meter" THEN  wlVAR = wlVAR \* (10^-6)  wlCalculation()  END IF  endprocedure | Procedure for fetching value user entered into wavelength entry box. This helps fulfil Success Criteria 4 as this is the main function will assign the input from the Tkinter entry box (wlEntry) to the variable that is used throughout the calculation of the program. It also helps fulfil Success Criteria 16 as it takes the units that has been selected by the user using the radio buttons to select between meter, micro meter or nano meter. |
| procedure wlCalculation  global wlVAR  global velVAR  global freqVAR  global period  global depVAR  IF depVAR = "frequency" THEN  freqVAR = velVAR / wlVAR  period = 1 / freqVAR  Output()  ELIF depVAR = "velocity" THEN  velVAR = wlVAR \* freqVAR  period = 1 / freqVAR  Output()  ENDIF  endprocedure | Procedure for attempting to calculate the dependent variable after defining a value of wavelength. This fulfils Success Criteria 12 as it uses the equations found from the (O'Neill, 2015) textbook from my research. This procedure loads the global variables for the dependent and independent wave properties as well as the variable that designates the dependent variable. It will then check which variable is the selected dependent variable and using the wavelength value that was just inputted (causing the calling of this procedure) and the other independent variable to calculate the dependent variable and the period. It then calls the Output() procedure to show this result to the user. |
| procedure getVelocity  global velVAR  velVAR = input(velEntry)  velCalculation()  endprocedure | Procedure for fetching value user entered into velocity entry box. This is for Success Criteria 2 as it is the procedure that is executed allowing the user to input the ‘velVAR’ via the tkinter GUI entry box. As the program doesn’t allow the use of different units for the velocity, this procedure just loads the variable, assigns the value of the entry box to it, then runs the procedure for calculations using velocity as one of the independent variables. |
| procedure velCalculation  global wlVAR  global velVAR  global freqVAR  global period  global depVAR  IF depVAR = "wavelength" THEN  wlVAR = velVAR / freqVAR  period = 1 / freqVAR  Output()  ELIF depVAR = "frequency" THEN  freqVAR = velVAR / wlVAR  period = 1 / freqVAR  Output()  ENDIF  endprocedure | Procedure for attempting to calculate the dependent variable after defining a value of velocity. This procedure helps towards Success Criteria 12 as it uses the equations to calculate the values the dependent variables (wavelength or frequency and period). It loads the global variables for the independent and dependent variables locally as well as the variable storing what has been assigned as the dependent variable. The procedure then uses an IF statement to check which is the dependent variable and then will use the independent variables to calculate the dependent variable as well as the period. It will then call the Output() procedure to output these values to the user. |
| procedure getFrequency  global freqVAR  freqVAR = input(freqEntry)  freqCalculation()  endprocedure | Procedure for fetching value user entered into frequency entry box. This procedure will satisfy Success Criteria 1 as it will read the users input from a tkinter entry box to assign a value for the frequency. This procedure loads the global variable ‘freqVAR’ locally as this it the variable that the value for frequency will be stored in. It is then assigned by the value the user has entered into the tkinter entry box and then freqCalculation() is run. |
| procedure freqCalculation  global wlVAR  global velVAR  global freqVAR  global period  global depVAR  IF depVAR = "wavelength" THEN  wlVAR = velVAR / freqVAR  period = 1 / freqVAR  Output()  ELIF depVAR = "velocity" THEN  velVAR = wlVAR \* freqVAR  period = 1 / freqVAR  Output()  ENDIF  endprocedure | Procedure for attempting to calculate the dependent variable after defining a value of frequency. This helps satisfy Success Criteria 12 as it uses the equations sourced from the textbook research and determines the dependent variables as according to what other independent variable was entered alongside frequency. The procedure loads all the global variables for the dependent and independent variables as well as the variable storing the identity of the dependent variable. Then it will determine the dependent variable using the if statement and using the equation ‘wavelength = velocity / frequency’ and rearranging it depending on the independent’s it will calculate the third dependent variable then the period of the wave. Then it will execute the Output() procedure for the user. |
| procedure getAmplitude  global ampVAR  global ampUnit  ampVAR = input(ampEntry)  ampUnit = input(ampRadioButtons)  IF ampUnit = "meter" THEN  PlotGraph()  ELIF ampUnit = "milli meter" THEN  ampVAR = ampVAR \* (10^-3)  PlotGraph  ELIF ampUnit = "micro meter" THEN  ampVAR = ampVAR \* (10^-6)  PlotGraph()  END IF  endprocedure | Procedure for fetching value user entered into amplitude entry box. This satisfies Success Criteria 3 and 16 as this procedure takes a user inputted value for amplitude as well as the amplitude’s unit. As amplitude is a completely independent variable from the other 3 this ‘get’ procedure is structured separately from the other 3 as it doesn’t need it’s own ‘calculation’ procedure to run. Hence it will take the users input for amplitude through the tkinter Entry box and then will use the radio buttons value to determine whether it needs to translate this value into it’s standard form or if it has already been inputted in as the calculatable base unit of meters. |
| procedure Output  global wlVAR  global wlUnit  global velVAR  global freqVAR  global depVAR  IF depVAR = "wavelength" THEN  IF wlUnit = "meter" THEN  wlEntry = empty  wlEntry = input(wlVAR)  PlotGraph()  ELIF wlUnit = "nano meter" THEN  wlEntry = empty  wlEntry = input(wlVAR \*10^9)  PlotGraph()  ELIF wlUnit = "micro meter" THEN  wlEntry = empty  wlEntry = input(wlVAR \* 10^6)  PlotGraph()  ENDIF  ELIF depVAR = "velocity" THEN  velEntry = empty  velEntry = input(velVAR)  PlotGraph()  ELIF depVAR = "frequency" THEN  freqEntry = empty  freqEntry = input(freqVAR)  PlotGraph()  ENDIF  endprocedure | Procedure for outputting the dependent variable that’s been calculated by the program and inputting it into the correct entry box. This satisfies Success Criteria 5 – 10 as this is the part of the program that will output all the independent and the calculated dependent variables. For this design section as I am yet to fully decide the amount of variables the program will output in this iteration I have designed the code for outputting the 2 main independent variables and the calculated dependent variable. So this procedure starts with loading the global variables for the wavelength and its unit, velocity, frequency and the dependent variable locally. It then determines which variable is the dependent variable using the IF statement. It will then empty the entry box associated with said independent variable and input the new calculated value using the variable it was assigned. In the case of wavelength as this has multiple radio buttons the user can select to pick which unit their wavelength is displayed or inputted in there is a need for an imbedded IF statement to determine what that unit is and then convert the answer to that unit prior to entry. |
| procedure dependentVAR  global depVAR  IF depVAR() = "frequency" THEN  freqEntry.config(state=DISABLED)  wlEntry.config(state=NORMAL)  velEntry.config(state=NORMAL)  Output()  ELIF depVAR() = "velocity" THEN  velEntry.config(state=DISABLED)  wlEntry.config(state=NORMAL)  freqEntry.config(state=NORMAL)  Output()  ELIF depVAR() = "wavelength" THEN  wlEntry.config(state=DISABLED)  freqEntry.config(state=NORMAL)  velEntry.config(state=NORMAL)  Output()  ENDIF  endprocedure | Procedure for selecting dependent variable depending on the selected dependent variable radio button. This is the main crux of the program as if this will block off the user from entering in a value for their desired dependent variable meaning that the program will not be able to calculate anything with no second independent variable value. The procedure simply takes the value of the radio button that is associated with the dependent variable using an IF statement then will lock disable inputs into that entry box and make sure the other 2 entry boxes are able to be entered into. |
| procedure PlotGraph  global ampVAR  global wlVAR  global waveCanvas  waveCanvas = Matplotlib(Figure)  x\_coord = (start=0, end=10000)  y\_coord = (sine{x/1.591549431} \* ampVAR)  y\_axis = “Amplitude (m)”  x\_axis = “Wavelength (m)”  x\_limit = (0, wlVAR)  IF waveCanvas != None THEN  waveCanvas.forget()  waveCanvas.draw(x, y)  endprocedure | This is the procedure used for the creation of the graph that will model the users inputs as a sine wave. This satisfies Success Criteria 17 as this is the graph that will be imbedded into the tkinter GUI. The algorithm imports the values of amplitude, wavelength and the canvas on the GUI locally in the procedure. It will then create a matplotlib Figure, which is a type of graph that can be used to be imbedded into tkinter and it will identify the x and y co ordinates of the wave. The x co-ordinates are the limits that the program will generate the wave for as generating an infinite sine wave would use a lot of memory so it will calculate values up to 10,000 as this is a reasonable range for students to use. The y co-ordinates are the modelling of the mathematical sine wave. The wave is a sine(x/1.591549431) wave as this is used from creating a 1 to 1 scale wave of the unit meter wavelength to the 1 wavelength mark on a sine wave (2 pi radians) then multiplying this by 10 as each wavelength will be modelled as 10 cycles to create a higher ‘visibility’ range in the limited graph. The axis are then appropriately labelled so students can identify what the wave on the graph is and then the canvas is checked that there isn’t an existing wave on it. If so that wave is deleted and a new one is drawn onto the graph. |

## Usability Features

* The program will have an entry box for wavelength that the user can input their desired wavelength value into.
  + This is essential as the main source of input in this program will be via entry boxes rather than sliders, this is so users have more freedom with the use of the program and the program can have a wider range of use cases within a school environment.
* The program will have a set of radio buttons to select the unit that they will be entering their wavelength in. Either metres, micrometres or nanometres.
  + Within an education space wavelength is often used in a range of units so being able to select the unit of wavelength using radio buttons will make the program more suitable to be used in education.
* The program will have an entry box for velocity that the user can input their desired velocity value into.
  + This is essential as the main source of input in this program will be via entry boxes rather than sliders, this is so users have more freedom with the use of the program and the program can have a wider range of use cases within a school environment.
* The program will have an entry box for frequency that the user can input their desired frequency value into.
  + This is essential as the main source of input in this program will be via entry boxes rather than sliders, this is so users have more freedom with the use of the program and the program can have a wider range of use cases within a school environment.
* The program will have an entry box for amplitude that the user can input their desired amplitude value into.
  + This is essential as the main source of input in this program will be via entry boxes rather than sliders, this is so users have more freedom with the use of the program and the program can have a wider range of use cases within a school environment.
* The program will have a set of radio buttons to select the unit that they will be entering their amplitude in. Either metres, millimetres or micrometres.
  + Within an education space amplitude is often used in a range of units so being able to select the unit of wavelength using radio buttons will make the program more suitable to be used in education.
* There will be radio buttons to pick which one of the three related variables (wavelength, velocity and frequency) the program will be calculating. The corresponding entry box to this radio button will be disabled and not able to be entered into. (See the dependent Variable procedure in decomposition).
  + Within an education space wavelength is often used in a range of units so being able to select the unit of amplitude using radio buttons will make the program more suitable to be used in education.
* There will be a set of radio buttons that the user can select to choose a preset of wave variables (e.g. red light, microwaves, UV light etc.).
  + This is because this is going to be used as an educational tool it is useful to have examples of popular waves that come up in the textbooks or topics so users can see what the variables will look like for real, commonly found waves that they learn about.

## UI Design

As this is designed to be an educational tool, in the design and creation of the GUI, to make sure that the software would boot and fit to the screen of the device it is being used on (this was highlighted due to the discrepancy of display sizes over the various systems I was coding this on). This led to the decision to design and code the GUI so objects are placed according to the percentage of the screen size. 

## Key Variable Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable name | Data type | Local or global | Description of what stored | Where used | Justification |
| wlVAR | Float | Global | Stores the users input of wavelength. | Main Program, getWavelength(), wlCalculation(), velCalculation(), freqCalculation(), Output(), PlotGraph() | This is a key variable as it will store the wavelength that is inputted by the user or calculated by the program. This helps satisfy Success Criteria’s 4 and 5. |
| wlUnit | String | Global | Stores the users input of the wavelength unit. | Main Program, getWavelength(), Output() | This variable is needed as it will store the user’s decision as to what unit of measurement the wavelength is converted to when it’s stored. Using radio buttons this will restrict the value to either meters, micro meters or nano meters. This satisfies Success Criteria 16 as it means the users can easily enter in a range of small or large values for the wavelength. |
| velVAR | Float | Global | Stores the users input of the velocity. | Main Program, getVelocity(), wlCalculation(), velCalculation(), freqCalculation(), Output() | This variable is also key as it stores the value of velocity of the users input or if it was calculated by the program. This will satisfy 2 and 7. |
| freqVAR | Float | Global | Stores the users input of the frequency. | Main Program, getFrequency(), wlCalculation(), freqCalculation(), velCalculation(), Output() | This variable is key to the program as it stores the value of the frequency that the has entered or has been calculated by the program. This satisfies Success Criteria 1 and 6. |
| ampVAR | Float | Global | Stores the users input of the amplitude. | Main Program, getAmplitude(), Output(), PlotGraph() | This variable is key to the program as it stores the amplitude inputted by the user and is used in the wave graph and in the output box. This variable is critical to Success Criteria 2, 8 and 11 |
| ampUnit | String | Global | Stores the users input of the amplitude unit. | Main Program, getAmplitude(), Output() | This variable is  key to the  program as it  stores the value  of the unit the  amplitude is  measured in.  This satisfies  Success Criteria  16 as it gives  the user more  customisation  in their input of  variables. |
| depVAR | String | Global | Stores the selected dependent variable. | Main Program, dependentVAR(), Output(), wlCalculation(), velCalculation(), freqCalculation() | This variable is important as the program’s core function is to determine and calculate the value of the dependent variable of the users desired wave. So this variable will store the value of which variable is the intended dependent one. |
| period | Float | Global | Stores the calculated value of the period. | Main Program, wlCalculation(), freqCalculation(), velCalculation(), Output() | Period is one of the static dependent variables in the program that will be calculated and outputted to the user. This variable will hold the calculated value and satisfies Success Criteria 9. |
| phEnergy | Float | Global | This will store the value of the calculated photon energy of the wave. | Main Program, Output() | This is going to be one of the dependent variables mentioned in Success Criteria 10. |
| root | Tkinter window | Global | This is the window in which all the tkinter GUI will be stored. | Main Program | root is the tkinter window that the GUI will run inside and will hold all of the programs tkinter elements and inputs and outputs. This is a part of Success Criteria 11. |
| waveFrame | Tkinter Frame | Global | This is the frame that the calculated graph will be imbedded in into the tkinter GUI. | MainProgram, PlotGraph() | The wave frame is the tkinter frame object that can have the plotted graph that models the wave in imbedded into it. This satisfies Success Criteria 17. |
| wGraph | Matplotlib x Tkinter Graph | Global | This is the graph that the variables will be displayed and translated into. | MainProgram, PlotGraph() | This graph will have the wave modelled onto it, it will then be displayed on the wave frame, imbedding it into the tkinter GUI. This satisfies Success Criteria 17. This will be modelled using the sine function in the python math library. |
| wlRadioButton | Tkinter Radio Button | Global | This is the radio button that will select whether wavelength is the dependent variable. | Main Program | This is the radiobutton that selects the wavelength as the users desired dependent variable, this is the main purpose of the program. |
| freqRadioButton | Tkinter Radio Button | Global | This is the radio button that will select whether frequency is the dependent variable. | Main Program | This is the radiobutton that selects the frequency as the users desired dependent variable, this is the main purpose of the program. |
| velRadioButton | Tkinter Radio Button | Global | This is the radio button that will select whether velocity is the dependent variable. | Main Program | This is the radiobutton that selects the velocity as the users desired dependent variable, this is the main purpose of the program. |
| wlLabel | Tkinter Label | Global | This is the label that will sit next to the entry box for that corresponds to the wavelength variable. | Main Program | This Label will identify to the user which entry box in the GUI is used to input a value for their wavelength. |
| wlEntry | Tkinter Entry Box | Global | This is the entry box that will allow the user to enter their desired variable for the wlVAR. | Main Program, getWavelength() | This Entry box is needed for the user to input a value for the wavelength and will show the output of the wavelength in the case it is selected as the dependent variable. Hence it satisfies Success Criteria 4 and 5. |
| wlMeter | Tkinter Radio Button | Global | This radio button is used to select what unit the wavelength that is entered into the entry box is processed through in the program | Main Program, getWavelength() | This radiobutton is used to select what unit of measurement the wavelength is entered in or outputted in. This satisfies Success Criteria 16. |
| wlMicro | Tkinter Radio Button | Global | This radio button is to be used by users to be able to change the unit that the wavelength in the wlEntry box is inputted and outputted in. | Main Program, getWavelength() | This radiobutton is used to select what unit of measurement the wavelength is entered in or outputted in. This satisfies Success Criteria 16. |
| wlNano | Tkinter Radio Button | Global | This radio button is to be used by users to be able to change the unit that the wavelength in the wlEntry box is inputted and outputted in. | Main Program, getWavelength() | This radiobutton is used to select what unit of measurement the wavelength is entered in or outputted in. This satisfies Success Criteria 16. |
| velLabel | Tkinter Label | Global | This is the label that will be next to the entry box used for velocity to show the user what it’s used for. | Main Program | This label is used in the GUI to show the user which entry box corresponds with the value of the waves velocity. |
| velEntry | Tkinter Entry Box | Global | This is the tkinter entry box that the user will be able to use to input their desired value for the velocity of the wave. | Main Program | This Entry Box is used in the GUI as the input and output of the velocity variable for the user. This therefore satisfies Success Criteria 2 and 7. |
| freqLabel | Tkinter Label | Global | This label will show the user which entry corresponds to the variable to input frequency. | Main Program | This label is used in the GUI to show the user which entry box corresponds with the value of the waves frequency. |
| freqEntry | Tkinter Entry Box | Global | This is the entry box that can be used by the user to enter their desired value of the frequency. | Main Program | This Entry Box is used in the GUI as the input and output of the frequency variable for the user. This therefore satisfies Success Criteria 1 and 6. |
| ampLabel | Tkinter Label | Global | This label is used in the GUI to show which entry box is used to enter in the users desired value for the amplitude. | Main Program | This label is used in the GUI to show the user which entry box corresponds with the value of the waves amplitude. |
| ampEntry | Tkinter Entry Box | Global | This is the entry box that the user can enter the desired value of the wave’s amplitude. | Main Program | This Entry Box is used in the GUI as the input of the amplitude variable for the user. This therefore satisfies Success Criteria 3. |
| ampMeter | Tkinter Radio Button | Global | This is the radio button on the tkinter GUI that can be used by the user to select the unit of the amplitude. In this case for it to be in meters. | Main Program, getAmplitude() | This radiobutton is used to select what unit of measurement the amplitude is entered in. This satisfies Success Criteria 16. |
| ampMilli | Tkinter Radio Button | Global | This radio button is used to select the unit that the amplitude is measured in. This button selects milli meters. | Main Program, getAmplitude() | This radiobutton is used to select what unit of measurement the amplitude is entered in. This satisfies Success Criteria 16. |
| ampMicro | Tkinter Radio Button | Global | This radio button is used in the tkinter GUI to select the amplitude’s unit. This in particularly is for micro meters. | Main Program, getAmplitude() | This radiobutton is used to select what unit of measurement the ampltiude is entered in. This satisfies Success Criteria 16. |
| presetsLabel | Tkinter Label | Global | This is a label to mark the presets radio buttons as pre existing sets of data. | Main Program | This label is used to show the user that the radiobuttons present within the box it’s placed within in the GUI correspond to the presets available in the program. This helps satisfy Success Criteria 14. |
| redLightRadioB | Tkinter Radio Button | Global | This is the radio button that is used to select the preset defined variables coded into the program. | Main Program | This radiobutton is used by the user to be able to preset wavelength, velocity and frequency of a common shade of red light on the EM spectrum. This helps satisfy Success Criteria 14. |
| gammaWaveRadioB | Tkinter Radio Button | Global | This is the radio button that is used to select the preset defined variables coded into the program. | Main Program | This radiobutton is used by the user to be able to preset wavelength, velocity and frequency of a common type of Gamma Radiation on the EM spectrum. This helps satisfy Success Criteria 14. |
| microWaveRadioB | Tkinter Radio Button | Global | This is the radio button that is used to select the preset defined variables coded into the program. | Main Program | This radiobutton is used by the user to be able to preset wavelength, velocity and frequency of a common type of Microwave on the EM spectrum. This helps satisfy Success Criteria 14. |
| customWaveRadioB | Tkinter Radio Button | Global | This is the radio button that is allowing the user to input their own variables into the programs entry boxes. | Main Program | This radiobutton is used by the user to be able to preset wavelength, velocity and frequency back to the default options when the program is reloaded. |

## Validation Strategy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Validation entry | Description of what could go wrong | Strategy | Return message | Justification |
| Disabling of the dependent entry box | If a user tries to enter an input into an entry box when it is defined as a dependent variable, this means that the incorrect script would run and could result in a math error and halt the program | When a variable is defined to be a dependent variable then the entry box for it will be disabled and not allow for user input so they cannot directly interact and affect that variable. | no return message as it simply will not accept an input | This means that the user will only be able to affect the independent variables which means the program will then calculate the dependent variable without the user (Success Criteria 18). |
| Inputting non-floating-point values into Entry Boxes | The user could try and enter a non-floating-point or integer value in which the mathematical algorithm’s could not process and could lead to the program halting or not functioning as intended. | When the user is incorrectly inputting letters or symbols, the input will not accept any values and no calculations will take place | ‘Please input a floating-point value’ | This contributes to Success Criteria 12 as it helps the program run and perform the calculations without potential issues trying to use strings or other data types in mathematical calculations |

## Approach To Testing

In this iteration, as it has a Tkinter GUI, the user input is now done through Tkinter entry boxes rather than the python console. Therefore, most of the tests in this iteration will be towards the use of the entry boxes. Other tests will involve the functionality of the tkinter radio buttons, ensuring that they produce expected results. It will also be testing the output of the graphing of the wave on the matplotlib graph.

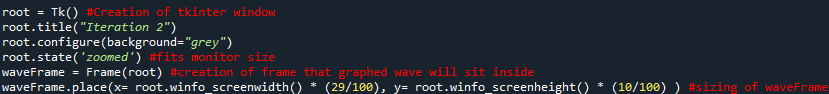
## Test Strategy

Test numbers denoted as ‘D2 – (test number)’, meaning D2 = design iteration 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | What is being tested? | Test Data | Type of test  (Normal, Boundary, Erroneous) | Test description | What should happen | Justification |
| D2-1 | Opening the Program | Run the program | Normal | This test is running the program to make sure that the GUI works and runs. | The program should run in a full zoomed state with all the elements in correct places | This is to ensure the program works and links to all Success Criteria, specifically 11, 13 and 15 as it is making sure the program has a GUI that will fill up the devices screen as well as it being structured and layed out logically. |
| D2-2 | Selecting wavelength as the dependent variable | Clicking wavelength dependent radio button | Normal | This is selecting the radio button that is associated with making the wavelength be the dependent variable of the program | The wavelength Entry box should become disabled, no longer allowing user entry into it. The entry box that was previously deemed ‘dependent’ and disabled should become enabled | This means the program will be able to calculate a value for the wavelength given velocity and frequency as the independent variables. This help satisfy Success Criteria 5 and 12 as if the wavelength is dependent, it will be outputted when there are values for velocity and frequency and uses the equations that were found during my research and listed in the equations section. |
| D2-3 | Selecting velocity as the dependent variable | Clicking velocity dependent radio button | Normal | This is selecting the radio button that is associated with making the velocity the dependent variable of the program | The velocity Entry box should become disabled, no longer allowing user entry into it. The entry box that was previously deemed ‘dependent’ and disabled should become enabled | This means the program has the capacity to calculate and output a value for the velocity. This helps satisfy Success Criteria 7 and 12 as the program will be using the equations detailed in the equation section to calculate and output a value for the velocity. |
| D2-4 | Selecting Wavelength micrometre unit | Clicking the wavelength micro metre unit radio button | Normal | The selecting of the radio button that is associated with making the unit of wavelength in micrometres | The program should output the value for the wavelength converted from micrometres into metres in the console and the additional info outputs on the GUI | This is necessary to the Success Criteria 5 and 16 as it will show an output for the value of the wavelength as well as give the user the ability to customise the units of the wavelength (e.g. metres, micrometres and nanometres) as these are common measurements used for wavelength as found in my research |
| D2-5 | Selecting wavelength nanometre unit | Clicking the wavelength nano metre unit radio button | Normal | The selecting of the radio button that is associated with making the unit of wavelength in nanometres | The program should output the value for the wavelength converted from nanometres into metres in the console and the additional info outputs on the GUI | This is necessary to the Success Criteria 5 and 16 as it will show an output for the value of the wavelength as well as give the user the ability to customise the units of the wavelength (e.g. metres, micrometres and nanometres) as these are common measurements used for wavelength as found in my research |
| D2-6 | Selecting Wavelength meter unit | Clicking the wavelength metre unit radio button | Normal | The selecting of the radio button that is associated with making the unit of wavelength in metres | The program should output the value for the wavelength in metres in the console and the additional info outputs on the GUI | This is necessary to the Success Criteria 5 and 16 as it will show an output for the value of the wavelength as well as give the user the ability to customise the units of the wavelength (e.g. metres, micrometres and nanometres) as these are common measurements used for wavelength as found in my research |
| D2-7 | Selecting Amplitude millimetre unit | Clicking the amplitude milli metre unit radio button | Normal | The selecting of the radio button that is associated with making the unit of amplitude in millimetres | The program should output the value for the amplitude converted from millimetres into metres in the console and the additional info outputs on the GUI | This is necessary to the Success Criteria 8 and 16 as it will show an output for the value of the amplitude as well as give the user the ability to customise the units of the amplitude (e.g. metres, micrometres and millimetres) as these are common measurements used for amplitude as found in my research |
| D2-8 | Selecting Amplitude micrometre unit | Clicking the amplitude micro metre unit radio button | Normal | The selecting of the radio button that is associated with making the unit of amplitude in micrometres | The program should output the value for the amplitude converted from micrometres into metres in the console and the additional info outputs on the GUI | This is necessary to the Success Criteria 8 and 16 as it will show an output for the value of the amplitude as well as give the user the ability to customise the units of the amplitude (e.g. metres, micrometres and millimetres) as these are common measurements used for amplitude as found in my research |
| D2-9 | Selecting Preset 1 radio button | Clicking the radio button that is associated with Preset 1 | Normal | The selecting of the radio button that is associated with changing the wave variables to match that of a preselected value | The values of the wlVAR, velVAR and freqVAR should be altered to correspond with the values of whatever sort of wave preset 1 is, as well as the graph being replotted to reflect these values and the variable output box being up to date | This satisfies Success Criteria 14 as it will give the user the choice to have pre-selected examples of real life EM waves. This is something from the research I found missing as it makes sense if students can use examples like ones they’ve learnt about in class and compare these to ones they’d learn about in class. |
| D2-10 | Selecting Preset 2 radio button | Clicking the radio button that is associated with Preset 2 | Normal | The selecting of the radio button that is associated with changing the wave variables to match that of a preselected value | The values of the wlVAR, velVAR and freqVAR should be altered to correspond with the values of whatever sort of wave preset 2 is, as well as the graph being replotted to reflect these values and the variable output box being up to date | This satisfies Success Criteria 14 as it will give the user the choice to have pre-selected examples of real life EM waves. This is something from the research I found missing as it makes sense if students can use examples like ones they’ve learnt about in class and compare these to ones they’d learn about in class. |
| D2-11 | Selecting Preset 3 radio button | Clicking the radio button that is associated with Preset 3 | Normal | The selecting of the radio button that is associated with changing the wave variables to match that of a preselected value | The values of the wlVAR, velVAR and freqVAR should be altered to correspond with the values of whatever sort of wave preset 3 is, as well as the graph being replotted to reflect these values and the variable output box being up to date | This satisfies Success Criteria 14 as it will give the user the choice to have pre-selected examples of real life EM waves. This is something from the research I found missing as it makes sense if students can use examples like ones they’ve learnt about in class and compare these to ones they’d learn about in class. |
| D2-12 | Selecting the standard preset radio button | Clicking the radio button that is associated with the standard preset | Normal | The selecting of the radio button that is associated with changing the wave variables to match that of a preselected value (probably something simple like 10, 1, 0.1 and 1) | The values of the wlVAR, velVAR, freqVAR ad ampVAR should be altered to correspond with the values I predefine for the program (probably 10, 1, 0.1 and 1 as these are close to the lowest bounds), as well as the graph being replotted to reflect these values and the variable output box being up to date | This is something that will be used by the program as a baseline that is the variable values that happen when you boot the program. Essentially a reset button so users don’t get lost in how to use the program when using the preset buttons. |
| D2-13 | Selecting frequency entry when disabled | Clicking the entry box that is associated with frequency when frequency is selected as the dependent variable | Erroneous | This is attempting to mouse click the Entry box on the GUI that is disabled | The program should not allow any editing within the text box | This is a validation strategy test as the user should not be able to enter any values into the entry box when it is being used as an output as that would lead to the program appearing incorrect once the two independent values have calculated a value that could be altered. |
| D2-14 | Selecting velocity entry when disabled | Clicking the entry box that is associated with velocity when velocity is selected as the dependent variable | Erroneous | This is attempting to mouse click the Entry box on the GUI that is disabled | The program should not allow any editing within the text box | This is a validation strategy test as the user should not be able to enter any values into the entry box when it is being used as an output as that would lead to the program appearing incorrect once the two independent values have calculated a value that could be altered. |
| D2-15 | Selecting wavelength entry when disabled | Clicking the entry box that is associated with wavelength when wavelength is selected as the dependent variable | Erroneous | This is attempting to mouse click the Entry box on the GUI that is disabled | The program should not allow any editing within the text box | This is a validation strategy test as the user should not be able to enter any values into the entry box when it is being used as an output as that would lead to the program appearing incorrect once the two independent values have calculated a value that could be altered. |
| D2-16 | Entering float value for wavelength when frequency is dependent | 15.6 | Normal | Entering a valid value into the wavelength Entry Box | The program should use this value to calculate the value of the frequency and output this value to the program | This checks Success Criteria 4 and 6 as this is ensuring that the user can input wavelength and receive an output for the frequency. |
| D2-17 | Entering float value for velocity when frequency is dependent | 14.2 | Normal | Entering a valid value into the velocity Entry Box | The program should use this value to calculate the value of the frequency and output this value to the program | This checks Success Criteria 2 and 6 as this is ensuring that the user can input velocity and receive an output for the frequency. |
| D2-18 | Entering float value for wavelength when velocity is dependent | 11.5 | Normal | Entering a valid value into the wavelength Entry Box | The program should use this value to calculate the value of the velocity and output this value to the program | This checks Success Criteria 4 and 7 as this is ensuring that the user can input wavelength and receive an output for the velocity. |
| D2-19 | Entering float value for frequency when velocity is dependent | 14.6 | Normal | Entering a valid value into the frequency Entry Box | The program should use this value to calculate the value of the velocity and output this value to the program | This checks Success Criteria 1 and 7 as this is ensuring that the user can input frequency and receive an output for the velocity. |
| D2-20 | Entering float value for velocity when wavelength is dependent | 14.3 | Normal | Entering a valid value into the velocity Entry Box | The program should use this value to calculate the value of the wavelength and output this value to the program | This checks Success Criteria 2 and 5 as this is ensuring that the user can input velocity and receive an output for the wavelength. |
| D2-21 | Entering float value for frequency when wavelength is dependent | 12.4 | Normal | Entering a valid value into the frequency Entry Box | The program should use this value to calculate the value of the wavelength and output this value to the program | This checks Success Criteria 1 and 5 as this is ensuring that the user can input frequency and receive an output for the wavelength. |
| D2-22 | Entering float value for amplitude | 7.6 | Normal | Entering a valid value into the amplitude Entry Box | The program should use this value to output a valid value for amplitude to the console and program | This checks success criteria 3 and 8 as it is ensuring that the user can input a value for amplitude and it will be output by the program. |
| D2-23 | Entering erroneous value for wavelength when frequency is dependent | hello | Erroneous | Entering an erroneous value into the wavelength Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-24 | Entering erroneous value for velocity when frequency is dependent | Hello | Erroneous | Entering an erroneous value into the velocity Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-25 | Entering erroneous value for wavelength when velocity is dependent | hello | Erroneous | Entering an erroneous value into the wavelength Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-26 | Entering erroneous value for frequency when velocity is dependent | hello | Erroneous | Entering an erroneous value into the frequency Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-27 | Entering erroneous value for velocity when wavelength is dependent | hello | Erroneous | Entering an erroneous value into the velocity Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-28 | Entering erroneous value for frequency when wavelength is dependent | Rayat | Erroneous | Entering an erroneous value into the frequency Entry Box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-29 | Entering erroneous value for amplitude | Do not | Erroneous | Entering an erroneous value into the amplitude entry box | The program should output an error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-30 | Entering 0 value for wavelength when frequency is dependent | 0 | Erroneous /Boundary | Entering an erroneous value into the wavelength Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-31 | Entering 0 value for velocity when frequency is dependent | 0 | Erroneous /Boundary | Entering an erroneous value into the velocity Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-32 | Entering 0 value for wavelength when velocity is dependent | **0** | Erroneous /Boundary | Entering an erroneous value into the wavelength Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-33 | Entering 0 value for frequency when velocity is dependent | 0 | Erroneous /Boundary | Entering an erroneous value into the frequency Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-34 | Entering 0 value for velocity when wavelength is dependent | 0 | Erroneous /Boundary | Entering an erroneous value into the velocity Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-35 | Entering 0 value for frequency when wavelength is dependent | 0 | Erroneous /Boundary | Entering an erroneous value into the frequency Entry Box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-36 | Entering 0 value for amplitude | 0 | Erroneous /Boundary | Entering an erroneous value into the amplitude entry box | The program should output a math error message to the console and not accept the value for any calculation. | This ensures the program does not break and halt if the user inputs an erroneous value. |
| D2-37 | Entering boundary value for wavelength when frequency is dependent | 0.00000001 | Boundary | Entering a boundary value into the wavelength Entry Box | The program should accept this value for the calculations and calculation frequency and output this to the program | This checks Success Criteria 4 and 6 as this is ensuring that the user can input a valid although very small value for wavelength and receive an output for the frequency. |
| D2-38 | Entering boundary value for velocity when frequency is dependent | 0.00000001 | Boundary | Entering a boundary value into the velocity Entry Box | The program should accept this value for the calculations and calculation frequency and output this to the program | This checks Success Criteria 2 and 6 as this is ensuring that the user can input a valid although very small value for velocity and receive an output for the frequency. |
| D2-39 | Entering boundary value for wavelength when velocity is dependent | 0.00000001 | Boundary | Entering a boundary value into the wavelength Entry Box | The program should accept this value for the calculations and calculation velocity and output this to the program | This checks Success Criteria 4 and 7 as this is ensuring that the user can input a valid although very small value for wavelength and receive an output for the velocity. |
| D2-40 | Entering boundary value for frequency when velocity is dependent | 0.00000001 | Boundary | Entering a boundary value into the frequency Entry Box | The program should accept this value for the calculations and calculation velocity and output this to the program | This checks Success Criteria 1 and 7 as this is ensuring that the user can input a valid although very small value for frequency and receive an output for the velocity. |
| D2-41 | Entering boundary value for velocity when wavelength is dependent | 0.00000001 | Boundary | Entering a boundary value into the velocity Entry Box | The program should accept this value for the calculations and calculation wavelength and output this to the program | This checks Success Criteria 2 and 5 as this is ensuring that the user can input a valid although very small value for velocity and receive an output for the wavelength. |
| D2-42 | Entering boundary value for frequency when wavelength is dependent | 0.00000001 | Boundary | Entering a boundary value into the frequency Entry Box | The program should accept this value for the calculations and calculation wavelength and output this to the program | This checks Success Criteria 1 and 5 as this is ensuring that the user can input a valid although very small value for frequency and receive an output for the wavelength. |
| D2-43 | Entering boundary value for amplitude | 0.00000001 | Boundary | Entering a boundary value into the amplitude entry box | The program should accept this value and output this to the program | This checks success criteria 3 and 8 as it ensures a user can input a valid (although very small value) for amplitude and have it outputted back to the program. |

# Implementation: Iteration 2

A screen shot of a computer

Description automatically generatedThe first step of coding this iteration was to import the essential required libraries for this project. Tkinter is the library that will be used in the program to create the GUI front end interface for interacting with the program. Importing ‘\*’ from tkinter means that during the coding of the program I will have access to use the whole of the main tkinter library. Whilst this will use additional memory in the runtime of the program, this is not a significant enough of a difference to warrant having to warrant the loading of the library in pieces. ‘numpy’ is the python library for accessing the math such as trigonometric functions. Which will need to be used in the program as the wave on the graph will be modelled using the sine function. Matplotlib’s ‘FigureCanvasTkAgg’ is the extension of Matplotlib that allows the embedding of a matplotlib graph into a tkinter screen. And figure is the type of graph that this program will use from the matplotlib library, this is because it is the graph that is easiest embedded into a tkinter frame. The next step of coding the program is the creation of the tkinter window and the use of frames to place where the wave graph is going to go in the program’s window, fulfilling Success Criteria 11. Tk() is the window widget in tkinter, Assigning that to the variable ‘root’ means that it can be used to have the GUI elements embedded into it. The next two lines are changing the window name and window background colour so that the program is usable and keeps the program design simple for earlier iterations such as this. ‘root.state(“zoomed”)’ will mean the program will size to whatever screen size is applicable to the device it is being run on, meaning this iteration meets Success Criteria 13. The creation of the waveFrame variable is the tkinter frame that is going to be used when having the matplotlib graph be imbedded into the GUI window. This is then placed using the percentages of the window size of the root window. This is to ensure that no matter the size or aspect ratio of the screen that the program is run on, that every object within the window will be placed and sized in proportion to the screen size. This is in accordance with Success Criteria 13. These numbers used are based roughly on the UI design sketch that was A screenshot of a computer

Description automatically generatedmade, hence the unit measurements on the sketch so that I could convert them into percentages for the screen. On the right, this is what the code looked like when running, as the tkinter frame is invisible you cannot see where it is placed on the screen, but the root window is opened in a zoomed state and the background is grey and this will be the basis of the program’s GUI. A computer code on a black background

Description automatically generatedThe next part of the code is the creation and placement of the boxes that I want in the program to sort of the various inputs and outputs as according to the UI design sketch. Each canvas is defined as a variable and then placed on the tkinter window by a percentage in accordance with Success Criteria 13. These canvas’ are sized according to the GUI window size and then they are coloured in as a light grey, this is so they stand out on the darker grey background so the text and objects later can be more visible on them. waveCanvas is designated NULL as this will be later defined during the creation of the wave graph in a function to be then placed in the wave frame created earlier. When run the program’s GUI A screenshot of a computer

Description automatically generatednow has the 3 main boxes that the rest of the GUI’s elements will be structured around.

A screen shot of a computer code

Description automatically generatedThe first input that I added was the inputs for wavelength (Success Critera 4 and 5). This needed 1. A label for identifing the inputs, 2. An entry box for inputting values, 3. Radio buttons for selecting the units of wavelength, 4. A radio button for assigning it as the dependent variable. The first part of the coding that I did was the creation of the label and the entry box. First I created a variable that was designated to the wlEntry box called ‘wlEntryVAR’, this variables is designed to store what value is inputted into the Entry box for wavelength. Then I created the label for the entry box by assigning the label to the ‘wlLabel’ variable, then using the UI design measurements and the tkinter .place() method in order to position to label on my tkinter GUI window inside the left most created canvas. Creating the entry box used the same method of assigning it the ‘wlEntry’ variable then making the variable of the Entry box ‘wlEntryVAR’. I then used the tkinter trace feature that detects changes to variables and added a trace to the variable ‘wlEntryVAR’ so that when the user inputs a value into the entry box it will trigger the procedure ‘getWl’ which I planned in the algorithms section and will code later.

Then I coded in the radio buttons that will be used to select the unit of measurement that wavelength is inputted/outputted in (Success Criteria 16). To do this I needed to code 3 radio buttons, each corresponding to meter, micro meter and nano meter, these need to be linked by a common variable. They then need to be placed around the rest of the wavelength input and there needs to be a trace on them to activate a function/procedure to change the value of the wavelength when they’re interacted with. A screen shot of a computer code

Description automatically generatedI had to create the variable ‘wlUnit’ to be able to link the radio buttons together and then I could add a variable trace on it set to execute the ‘getWl’ procedure when the user will select one of the radio buttons. ‘wlMeter.select()’ is used so that when the program is executed the selected unit of the wavelength is set to meters as this is the base unit for distance measurement. To finish off the code for the user’s input of the wavelength, next was to code the procedure ‘getWl’ – which as planned in my algorithms sections, will take the value from the wavelength entry box and the value of the unit radiobuttons and converts the value and stores it in the ‘wlVAR’ variable to be used in calculations and the program later on.

Then I created the 3 other inputs for the 3 variables: frequency, velocity and ampltiude. Most of this code is identical in structure to the wavelength input code, simply with changing the positioning of the elements and the variables that they’re associated with.A screenshot of a computer code

Description automatically generated From the screenshot you can see that the creation of all the inputs followed the same procedure of: creating a variable for the entry to assign the inputted value to, creating and placing the label, creating and placing the entry box then adding a variable trace on the variable that is associated with the entry box in order to be able to track when a user inputs and changes the contents of the entry box. For amplitude as well I created an extra set of radio buttons to designate the unit of measurement it is inputted as. These sets of code help cover Success Criteria 1-3 and 16, as these are the crux of the usability of the program that allow the user to input their desired independent values. At this point in the program with the main entry boxes fully coded the program looked like this:A grey and white rectangular shapes

Description automatically generated with medium confidenceAs you can see this GUI is so far pretty simple which I’m hoping will make it accessible to users of all ages and will pass Success Criteria 15.

The next part of the program that is needed before the calculation’s can be coded is the way for the user to select the dependent variable out of wavelength, velocity and frequency. This is going to use radio buttons that are placed next to the entry boxes.

A screenshot of a computer

Description automatically generatedA computer code on a black background

Description automatically generatedA variable that is attributed to the radiobuttons to link them together and to be able to track the value of what radio button is selected. The code to create the radio buttons is pretty simple as this code above is just the creation of the radiobuttons, assigning them to run the procedure SelectGrayedInput() when engaged, then they are placed into the canvas to be placed next to the entry boxes in the tkinter GUI. ‘FreqSelectRB.select()’ means that when the program is run the program will initially mark the frequency radio button, so that there is always a selected dependent variable to stop a user mistakenly not being able to use the program incorrectly. On the left is the output of the placement of the radiobuttons, and then I had to code the procedure for SelectGrayedInput() which will disable the entry box as determined by the radio button selected.

A screenshot of a computer

Description automatically generatedA screen shot of a computer code

Description automatically generatedThis code starts with importing the global variable associated with the radio buttons (varNonSelect) then the if statement will determine which of the entry boxes are desired to be disabled by the user by the value of the variable ‘varNonSelect’, it then will disable the entry box associated with this value and normalise the state of the other two entry boxes to ensure they are properly enabled. This procedure is then executed at the end of the program to ensure it runs on startup of the program so the user can misuse the program before a dependent variable has been selected. As you can then see, when the program is run it is loaded with the frequency entry box grayed out and the user is unable to interact or input any values into it.

Now that the program on boot had a working set of entry boxes for the variables and a way that the user can define the dependent variable of the calculation (Success Criteria 18), we can work on the program taking and processing these inputs. So I started with the procedures to read the value of each entry box and tie them to a calculation.

A screenshot of a computer program

Description automatically generatedFirst for wavelength we have getWl() procedure (Success Criteria 4). This procedure passes the parameters ‘var, index, mode’, which are the parameters which are provided by the ‘\_trace’ tkinter method when passed through a procedure. This procedure starts with loading wlVAR, wlUnit as local variables so they can be used in the procedure. ‘wlEntryVAR.get()’ is then assigning the value of the wlEntryVAR variable to the wlVAR variable so this then can be used in the calculation procedures later. The if statement then will use ‘wlUnit.get()’ which is the variable that is assigned to the unit radiobuttons, so it will take the value that is given by the various radio buttons and it will perform the mathematical translation given by what unit it is. So nanometer becomes ‘x10-9’ meters and micro meter becomes ‘x10-6’ meters. This function then will perform the wlCalculation() procedure which will be use the independent value of wavelength and one other A computer screen shot of a program

Description automatically generatedindependent value (either velocity or frequency).

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedThe wlCalculation() function takes the two scenarios in which the wavelength will have been a independent variable. It then will check the value of varNonSelect(), then it will use the equations from my research (Success Criteria 12), to determine the value of the dependent variable. These are basically the basis of the procedures for all 3 of the variables wavelength, velocity and frequency. There is validation in the if statement as the ‘else’ clause will produce an error message in the case that something goes wrong but it won’t fault the program. Below are the rest of the ‘get’ procedures for velocity, frequency and amplitude. This then covers the inputs of all 4 of the selected independent variables of the program (Success Criteria 1 – 4). The program then has the two calculation procedures for the frequency and velocity. The next part of the program was to code the procedure that outputs the calculated dependent values back to the user after they have been calculated by the program. This procedure at this stage of programming had 3 main purposes. To output the value of the calculated dependent variable back into the entry box that the user has designated using the radio buttons assigned to varNonSelect. Eventually it shall also calculate and ouput the various other dependent variables (such as time period and photon energy) as well as the independent variables to an additional info A screen shot of a computer program

Description automatically generatedbox in the tkinter box that will be coded in later, as well as plot the wave graph visualisation This procedure loads the variables locally and then uses an if statement to check which variable has been selected as the users desired dependent variable. This if statement also uses validation as it has an ‘else’ clause that will output an error into the console for maintenance reasons and means the program will not hault and continue running for the user to user. It then enables the entry box so the program can edit it’s value, it deletes the existing contents of the entry box and inputs the new calculated value. It A screen shot of a computer program

Description automatically generatedthen re-disables the entry box so the user cannot interfere with it. In the case of wavelength, as there is the option for multiple units of A screenshot of a computer

Description automatically generatedmeasurement, the program uses an if statement to determine the unit and then converts the wlVAR into that unit before outputting it back to the user. It was at this point that I coded the default values of the independent variables in the program so that when the user opens the program it will have preloaded variables and not lead to any math errors on program launch. So on program launch the value of the wavelength and velocity are set to 10 and 1, then the program will run the trace’s on these variables and resolve the frequency. As well as the amplitude being set to 1 so when the wave visualisation is in the program it is visible from start. This is now what the output of the program displays when it is ran, the program determines the value of frequency as 0.1 and will output that to the disabled entry box. The next part of the program that was coded was the presets that the user will be enabled to use in the program (Success Criteria 14). This will be a set of labels and radiobuttons that will input values for wavelength, velocity and amplitude into their respective entry boxes. This way the program’s trace’s on each entry boxes variables will lead to the program calculating the final dependent variable so the program can output the results back to the user. This will be used by changing the value of the wavelength’s unit last in the order to activate the getWl() procedure, which calls the wlCalculation() procedure, which then calls A screen shot of a computer program

Description automatically generatedA computer screen with colorful text

Description automatically generated with medium confidencethe Output() procedure. The code to initialise these radiobuttons is made simpler as we defined the canvas that they will be placed into the GUI earlier so we had the co-ordinates they need to be placed within before this. Then the radiobuttons are programmed to a variable radioButtonSelect and to execute the procedure SelectWaveInput() when they are selected. This procedure was simple as it was just having to determine the value of radioButtonSelect() using an if statement, it would then enable all the entry boxes for the variables and A screen shot of a computer program

Description automatically generatedinput the values needed to replicate that of the wave that the user clicked the radio button designated to. It then disables the frequency box and selects this as the dependent variable and selects the appropriate unit of measurement for the wavelength. This is also to run the trace on the wlUnit variable so it can run a daisy chain of procedures up to Output() (to see outputs, see tests T2-11 through T2-15).

A screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generatedThis was the last of the inputs and usability features that the program will feature. The last two parts of the program now are the final info output box and the wave visualisation output. The easier of those two will be the output box so that’s the part I coded first (Success Criteria 9 and 10). The code to have the outputs of these variables on the was included in the Output() procedure as these entry boxes are designed to always have the disabled state so the user cannot interact these elements as they’re purely for output. I used entry boxes as these are easy to insert values into so therefore are well used for output displays of values. This big chunk of code is the creation of all the entry boxes and labels that will be in the box. From the code you can see there are 6 different variables that the program will display, with 3 of them inputted by the user and 3 of them dependent variables the program has calculated. These all being: wavelength, amplitude, velocity, frequency, photon energy and the time period of the wave. After this I then added extra lines of code to the output procedure, as this is executed when the program is run due to the variable traces in the program. The procedure now calculates energy using a const called plancks constant, found from my textbook research, that I added as a global constant in the program. It then converts this value using the conversion rate of joules to electronvolts, as that is the unit of power used at a quantum level. It then ensure each entry box in that we just created in enabled and able to be edited and then clears them and inserts the value of the variable they are associated with. It then disables the entry box so the user can’t interact and change the value to make the output untrueA screenshot of a computer

Description automatically generated. This means this is now what the output box for the program now looks like. The final part of the program to code is the wave visualisation. This initially in my design and preparation for this iteration was going to be a stationary wave that just showed a change in the variables. But after a number of attempts and further research, I found it would be easier and simpler to code a moving graph of a sine wave using the matplotlib’s animation library. A screenshot of a computer code

Description automatically generatedThe above code is the code that is used to plot the moving graph on the tkinter interface. The procedure contains 2 functions which are defined as core parts of the FuncAnimation class from tkinter’s animation library. The procedure starts with the loading of the global variables waveCanvas, as this is the predefined tkinter canvas that the matplotlib figure is going to be imbedded in for it to be shown on the GUI. The first part of the code is the creation of a matplotlib graph called WaveGraph, it is defined by ‘wGraph = plt.subplots()’, this is the tuple containing the tkinter figure and the matplotlib pyplot that the co-ordinates will be displayed on. Using ‘WaveGraph, wGraph = plt.subplots()’ returns the tuple into 2 variables. WaveGraph being the figure, and wGraph being the array of axes that the x and y values can be plotted on. Next the code sets the height and width of the figure on the GUI in accordance to the screen size. Then it sets the axis label as amplitude and time. Then the ‘if waveCanvas’ statement simply checks to see if the waveCanvas variable exists prior to the procedure being executed, then if so, will remove it from the GUI. This means everytime the procedure is run it will overwrite the prior graph so they aren’t stacked on top of one another leading to an extortionately large memory load. The waveCanvas is then defined as a ‘FigureCanvasTkAgg’, which is just the type of matplotlib figure which can be embedded into a tkinter window. This is then packed and the limits of the y-axis are set to just either side of the amplitude to lead to a better looking wave. There is then defined a function ‘animation\_fig’, this is the function that provides the co-ordinates for the graph to the FuncAnimation class. It uses the wavelength, velocity and amplitude of the wave to model it on a graph as a sine wave. First the x co-ordinates are defined by the numpy linspace class, This is used to choose a start and end point for your x co-ords, then pick the number of generated co-ordinates between these 2 points. In my code, the start point of the wave is the value of the first frame of the animation, obviously, this is then multiplied by the velocity of the wave so the speed of the waves scrolling is affected by the calculated velocity. The final frame is then 10 frames later, hence frame+10. The amount of points generated between them can be changed depending on the system it is primarily going to be used on, for example, on my home pc I can run around 1000 loaded co-ordinates but on a school computer you’d want to load closer to 500-600 frames. This is due to the memory limitations and having a smaller number will lead to a lower resolution wave at higher numbers but is necessary for less powerful systems. The y co-ordinate is then modelled as a sine wave that is tranformed be a factor of 2\*pi / wavelength, this is because this is how frequency is modelled, it is then multiplied by x to get the time period of the x-axis. This is then all multiplied by the amplitude of the wave as this makes the y axis scale with the inputted amplitued of the wave. This data is then assigned to the line variable ‘sineWave’ and then an x limit is put at the minimum and max of the x variable so that the graph has an illusion to be scrolling as the frame number increases. This is then returned to the FuncAnimation class. The next function, ‘anim\_init()’ is the initialisation function and it sets the value of the co-ordinates of the to be plotted line to be blank, meaning theres a blank slate for the x and y values to be wrote into. The next stage of the PlotGraph() procedure is to plot the empty co-ords onto the matplotlib figure to ensure it’s blank following the plotting of the animated graph. Then the FuncAnimation class is called onto the waveGraph figure. This runs the initialisation function to clear the graph prior to animating on it, then it declares the ‘animation\_fig()’ function we just defined which will be called every frame to create the x and y co-ordinates that will be drawn onto the figure. ‘blit’ is a parameter that leads to smoother animations when the animation has many frames as it only redraws parts on the graph that need to be changed rather than resetting the whole graph each frame. And the interval parameter is another one variable to performance, it is the millisecond interval between showing each frame. I have currently set this to 50 milliseconds as that seems to tie smooth animation with a decent performance computer. This video shows the ouput of this code as a result of just running the program so with wlVAR = 10, velVAR = 1 and ampVAR = 1. The final part of the program that I coded was some tkinter.tix balloon widgets, this was something my client discussed and it is just a popup that will show on the screen when a user hovers over a GUI widget it is bound to. So I did one of these on what parts of the program I thought could do with some extra A screen shot of a computer

Description automatically generatedexplanation. This code simply will take a variable, and then assign it a message that will appear after a short time of hovering over it. For the final view of the program see the ‘Wave moving vid’ in the additional files.

(The school has not reinstalled the tix module when resetting the python libraries as far as I’ve found so I will provide a second version of the program without this library loaded or this part of the code)

# Testing: Iteration 2

Test numbers denoted as ‘T2 – (test number)’, meaning T1 = testing iteration 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Number | Test Data | Type of test | Description | What should happen | What did happen | Pass /Fail | Screen shot No. |
| T2-1 | Opening Program | Normal | I am simply going to run the program. | The tkinter root window will load in a zoomed state and all the tkinter objects are placed correctly with the correct numbers and graph loaded. | The root windows loads with all the objects placed correctly with the numbers 10, 1, 0.1, 1 loaded into wavelength, velocity, frequency and amplitude respectively. The graph loaded with the axis reading 1 and 10 and the sine graph modelled correctly. | Pass | [C1](#_C1) |
| T2-2 | Selecting the wlSelectRB | Normal | Clicking the radio button that is associated with selecting wavelength to be the dependent variable | The program should run the SelectGrayedInput() procedure which will disable the wlEntry and enable the freqEntry and velEntry as well as run Output() | The program did run the SelectGrayedInput() and Output() procedures successfully and now edits to the 2 independent variables affect the wlVAR and wlEntry values | Pass | [C2](#_C2) |
| T2-3 | Selecting the velSelectRB | Normal | Clicking the radio button that is associated with selecting velocity to be the dependent variable | The program should run the SelectGrayedInput() procedure which will disable the velEntry and enable the freqEntry and wlEntry as well as run Output() | The program did run the SelectGrayedInput() and Output() procedures successfully and now edits to the 2 independent variables affect the velVAR and velEntry values | Pass | [C3](#_C3) |
| T2-4 | Selecting the freqSelectRB | Normal | Clicking the radio button that is associated with selecting frequency to be the dependent variable | The program should run the SelectGrayedInput() procedure which will disable the freqEntry and enable the wlEntry and velEntry as well as run Output() | The program did run the SelectGrayedInput() and Output() procedures successfully and now edits to the 2 independent variables affect the freqVAR and freqEntry values | Pass | [C4](#_C4) |
| T2-5 | Selecting wlMicro radio button | Normal | Clicking the radio button under the wavelength entry box that controls the unit that the value of the entry is written in. | The program should run the getWl() procedure and in the if statement it will run the else statement and divide the inputted wavelength value by 1000000 and ouput to the console and the info box on the GUI | The program did run the getWl() procedure and successfully converted and outputted the inputted value of wavelength into micrometres | Pass | [C5](#_C5) |
| T2-6 | Selecting wlNano radio button | Normal | Clicking the radio button under the wavelength entry box that controls the unit that the value of the entry is written in. | The program should run the getWl() procedure and in the if statement it will run the elif statement and divide the inputted wavelength value by 1000000000 and ouput to the console and the info box on the GUI | The program did run the getWl() procedure and successfully converted and outputted the inputted value of wavelength into nanometres | Pass | [C6](#_C6) |
| T2-7 | Selecting wlMeter radio button | Normal | Clicking the radio button under the wavelength entry box that controls the unit that the value of the entry is written in. | The program should run the getWl() procedure and in the if statement it will run the if statement and ouput to the console and the info box on the GUI | The program did run the getWl() procedure and successfully outputted the inputted value of wavelength | Pass | [C7](#_C7) |
| T2-8 | Selecting ampMilli radio button | Normal | Clicking the radio button that is associated with the changing of the unit of the amplitude in the entry box | The program should run getAmp() and in the if statement it will run the elif statement and output to the console and info box in the GUI | The program did run the getAmp() procedure and converted the amplitude to millimetres and outputted the value in the console and on the GUI | Pass | [C8](#_C8) |
| T2-9 | Selecting ampMicro radio button | Normal | Clicking the radio button that is associated with the changing of the unit of the amplitude in the entry box | The program should run getAmp() and in the if statement it will run the else statement and output to the console and info box in the GUI | The program did run the getAmp() procedure and converted the amplitude to micrometres and outputted the value in the console and on the GUI | Pass | [C9](#_C9) |
| T2-10 | Selecting ampMeter radio button | Normal | Clicking the radio button that is associated with the changing of the unit of the amplitude in the entry box | The program should run getAmp() and in the if statement it will run the if statement and output to the console and info box in the GUI | The program did run the getAmp() procedure and outputted the value in the console and on the GUI | Pass | [C10](#_C10) |
| T2-11 | Selecting redLightRadioB radio button | Normal | Clicking the radio button in the presets box that is associated with changing the properties to be that of a demonstration red light wave | The program should run the SelectWaveInput() procedure which will run through the “if radioButtonSelect.get() == ‘Red’:” of the if statement. | The program successfully ran the SelectWaveInput() procedure. | Pass | [C11](#_C11) |
| T2-12 | Selecting blueLightRadioB radio button | Normal | Clicking the radio button in the presets box that is associated with changing the properties to be that of a demonstration blue light wave | The program should run the SelectWaveInput() procedure which will run through the “elif radioButtonSelect.get() == ‘Blue’:” of the if statement. | The program successfully ran the SelectWaveInput() and getWl() procedures | Pass | [C12](#_C12) |
| T2-13 | Selecting microWaveRadioB radio button | Normal | Clicking the radio button in the presets box that is associated with changing the properties to be that of a demonstration microwave wave | The program should run the SelectWaveInput() procedure which will run through the “elif radioButtonSelect.get() == ‘Micro’:” of the if statement. | The program successfully ran the SelectWaveInput() and getWl() procedures | Pass | [C13](#_C13) |
| T2-14 | Selecting gammaWaveRadioB radio button | Normal | Clicking the radio button in the presets box that is associated with changing the properties to be that of a demonstration gamma wave | The program should run the SelectWaveInput() procedure which will run through the “elif radioButtonSelect.get() == ‘Gamma’:” of the if statement. | The program successfully ran the SelectWaveInput() and getWl() procedures | Pass | [C14](#_C14) |
| T2-15 | Selecting the customWaveRadioB radio button | Normal | Clicking the radio button in the presets box that is associated with changing the properties to be that of the default settings when you first open the program | The program should run the SelectWaveInput() procedure which will run through the “elif radioButtonSelect.get() == ‘Custom’:” of the if statement. | The program successfully ran the SelectWaveInput() and getWl() procedures | Pass | [C15](#_C15) |
| T2-16 | Selecting frequency entry when disabled | Erroneous | Clicking the entry box for frequency when it has been disabled by the program | Nothing should happen as the program should not allow the user to interact with the entry box or freqVAR directly without inputting independent variables. | Nothing happened – which is good | Pass | [C16](#_C16) |
| T2-17 | Selecting velocity entry when disabled | Erroneous | Clicking the entry box for velocity when it has been disabled by the program | Nothing should happen as the program should not allow the user to interact with the entry box or velVAR directly without inputting independent variables. | Nothing happened – which is good | Pass | [C17](#_C17) |
| T2-18 | Selecting wavelength entry when disabled | Erroneous | Clicking the entry box for wavelength when it has been disabled by the program | Nothing should happen as the program should not allow the user to interact with the entry box or wlVAR directly without inputting independent variables. | Nothing happened – which is good | Pass | [C18](#_C18) |
| T2-19 | 12.5 | Normal | This is entering a floating point value into the ‘wlEntry’ box when the value of varNonSelect() is ‘Freq’. | The program should successfully run the getWl() procedure and pass the floating value through to the wlCalculation() procedure in which it will run the if statement to calculate freqVAR. | The program ran the wavelength related procedures and all outputs changed accordingly. | Pass | [C19](#_C19) |
| T2-20 | 10.2 | Normal | This is entering a floating point value into the ‘velEntry’ box when the value of varNonSelect() is ‘Freq’. | The program should successfully run the getVel() procedure and pass the floating value through to the velCalculation() procedure in which it will run the if statement to calculate freqVAR. | The program ran the velocity related procedures and all outputs changed accordingly. | Pass | [C20](#_C20) |
| T2-21 | 10.2 | Normal | This is entering a floating point value into the ‘wlEntry’ box when the value of varNonSelect() is ‘Vel’. | The program should successfully run the getWl() procedure and pass the floating value through to the wlCalculation() procedure in which it will run the if statement to calculate velVAR. | The program ran the wavelength related procedures and all outputs changed accordingly. | Pass | [C21](#_C21) |
| T2-22 | 5.5 | Normal | This is entering a floating point value into the ‘freqEntry’ box when the value of varNonSelect() is ‘Vel’. | The program should successfully run the getFreg() procedure and pass the floating value through to the freqCalculation() procedure in which it will run the if statement to calculate velVAR. | The program ran the frequency related procedures and all outputs changed accordingly. | Pass | [C22](#_C22) |
| T2-23 | 12.4 | Normal | This is entering a floating point value into the ‘velEntry’ box when the value of varNonSelect() is ‘Wl’. | The program should successfully run the getVel() procedure and pass the floating value through to the velCalculation() procedure in which it will run the if statement to calculate wlVAR. | The program ran the velocity related procedures and all outputs changed accordingly. | Pass | [C23](#_C23) |
| T2-24 | 15.5 | Normal | This is entering a floating point value into the ‘freqEntry’ box when the value of varNonSelect() is ‘Wl’. | The program should successfully run the getFreg() procedure and pass the floating value through to the freqCalculation() procedure in which it will run the if statement to calculate wlVAR. | The program ran the frequency related procedures and all outputs changed accordingly. | Pass | [C24](#_C24) |
| T2-25 | 4.7 | Normal | This is entering a floating point value into ‘ampEntry’. | This should pass the value through getAmp() procedure and convert it to the unit designated by ‘ampUnit’ then print these to the console | The program ran the amplitude related procedures and all outputs changed accordingly. | Pass | [C25](#_C25) |
| T2-26 | rayat | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the wlEntry when varNonSelect() = ‘Freq’. | This should attempt to run the getWl() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C26](#_C26) |
| T2-27 | = | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the velEntry when varNonSelect() = ‘Freq’. | This should attempt to run the getVel() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C27](#_C27) |
| T2-28 | Ashley | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the wlEntry when varNonSelect() = ‘Vel’. | This should attempt to run the getWl() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C28](#_C28) |
| T2-29 | True | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the freqEntry when varNonSelect() = ‘Vel’. | This should attempt to run the getFreq() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C29](#_C29) |
| T2-30 | R | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the velEntry when varNonSelect() = ‘Wl’. | This should attempt to run the getVel() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C30](#_C30) |
| T2-31 | - | Erroneous | This is entering an erroneous value (one not able to be casted as a floating point value) into the freqEntry when varNonSelect() = ‘Wl’. | This should attempt to run the getFreq() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C31](#_C31) |
| T2-32 | B | Erroneous | This is entering an erroneous value into the ampEntry. | This should attempt to run the getAmp() procedure yet it will not be able to cast the value and not accept the value into the program. | The program didn’t accept the value and kept running with the last inputted variable | Pass | [C32](#_C32) |
| T2-33 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘wlEntry’ when the varNonSelect() = ‘Freq’ | This should pass the value through the ‘getWl’ procedure and when it gets to the ‘wlCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C33](#_C33) |
| T2-34 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘velEntry’ when the varNonSelect() = ‘Freq’ | This should pass the value through the ‘getVel’ procedure and when it gets to the ‘VelCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C34](#_C34) |
| T2-35 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘wlEntry’ when the varNonSelect() = ‘Vel’ | This should pass the value through the ‘getWl’ procedure and when it gets to the ‘WlCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C35](#_C35) |
| T2-36 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘freqEntry’ when the varNonSelect() = ‘Vel’ | This should pass the value through the ‘getFreq’ procedure and when it gets to the ‘FreqCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C36](#_C36) |
| T2-37 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘velEntry’ when the varNonSelect() = ‘Wl’ | This should pass the value through the ‘getVel’ procedure and when it gets to the ‘VelCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C37](#_C37) |
| T2-38 | 0 | Erroneous/Boundary | This is entering a lone ‘0’ into the ‘freqEntry’ when the varNonSelect() = ‘Wl’ | This should pass the value through the ‘getFreq’ procedure and when it gets to the ‘FreqCalculation()’ it will create a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C38](#_C38) |
| T2-39 | 0 | Erroneous/Boundary | Entering a lone ‘0’ into the ‘ampEntry’ | This should pass the value through the ‘getAmp’ procedure and when it gets to the maths it will produce a math error and not accept the value | The program didn’t accept the value and didn’t halt the program | Pass | [C39](#_C39) |
| T2-40 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘wlEntry’ when VarNonSelect() = ‘Freq’ | The program should successfully run the getWl() procedure and pass the floating value through to the wlCalculation() procedure in which it will run the if statement to calculate freqVAR. | The program ran the wavelength related procedures and all outputs changed accordingly. | Pass | [C40](#_C40) |
| T2-41 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘velEntry’ when VarNonSelect() = ‘Freq’ | The program should successfully run the getVel() procedure and pass the floating value through to the velCalculation() procedure in which it will run the if statement to calculate freqVAR. | The program ran the velocity related procedures and all outputs changed accordingly. | Pass | [C41](#_C41) |
| T2-42 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘wlEntry’ when VarNonSelect() = ‘Vel’ | The program should successfully run the getWl() procedure and pass the floating value through to the wlCalculation() procedure in which it will run the if statement to calculate velVAR. | The program ran the wavelength related procedures and all outputs changed accordingly. | Pass | [C42](#_C42) |
| T2-43 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘freqEntry’ when VarNonSelect() = ‘Vel’ | The program should successfully run the getFreg() procedure and pass the floating value through to the freqCalculation() procedure in which it will run the if statement to calculate velVAR. | The program ran the frequency related procedures and all outputs changed accordingly. | Pass | [C43](#_C43) |
| T2-44 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘velEntry’ when VarNonSelect() = ‘Wl’ | The program should successfully run the getVel() procedure and pass the floating value through to the velCalculation() procedure in which it will run the if statement to calculate wlVAR. | The program ran the velocity related procedures and all outputs changed accordingly. | Pass | [C44](#_C44) |
| T2-45 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘freqEntry’ when VarNonSelect() = ‘Wl’ | The program should successfully run the getFreg() procedure and pass the floating value through to the freqCalculation() procedure in which it will run the if statement to calculate wlVAR. | The program ran the frequency related procedures and all outputs changed accordingly. | Pass | [C45](#_C45) |
| T2-46 | 0.0001 | Boundary | Entering a value that is mathematically extremely close to 0 into the ‘ampEntry’ | This should pass the value through getAmp() procedure and convert it to the unit designated by ‘ampUnit’ then print these to the console | The program ran the amplitude related procedures and all outputs changed accordingly. | Pass | [C46](#_C46) |
| T2-47 | 12.5, 6, 4 | Normal | Entering 3 values for wavelength, velocity, amplitude to see how it affects wave visualisation | This should process the wavelength and velocity variables, calculate frequency. Then plot these variables on the graph and the info box | It processed the independent variables and calculated the dependent variables and outputted them all correctly | Pass | [C47](#_C47) |
| T2-48 | 11.2, 4, 6 | Normal | Entering 3 values for wavelength, frequency, amplitude to see how it affects wave visualisation | This should process the wavelength and frequency variables, calculate velocity. Then plot these variables on the graph and the info box | It processed the independent variables and calculated the dependent variables and outputted them all correctly | Pass | [C48](#_C48) |
| T2-49 | 7, 8, 12 | Normal | Entering 3 values for velocity, frequency, amplitude to see how it affects wave visualisation | This should process the frequency and velocity variables, calculate wavelength. Then plot these variables on the graph and the info box | It processed the independent variables and calculated the dependent variables and outputted them all correctly | Pass | [C49](#_C49) |
| T2-50 | Hover over any element | Normal | Hovering over an element that has a tkinter.tix balloon bound to it | This should show up a tkinter tix balloon and give additional information about the element. | It showed up the additional information balloon and displayed the correct message for the element demonstrated on. | Pass | [C50](#_C50) |

# Evaluation: Iteration 2

## Test Review

In this iteration testing, none of the tests failed. This is good as this iteration is going to be the final product of this development. So this means the final product that will be used by the client is going to work.

## Success Criteria evaluation of Iteration 2

|  |  |  |
| --- | --- | --- |
| Success Criteria | Pass or Fail | Evidence of Pass or Fail |
| Criteria 1 | Pass – The user has the input freqEntry entry box for inputting frequency into. | T2-22, T2-24, T2-29, T2-31, T2-36, T2-38, T2-43, T2-45 |
| Criteria 2 | Pass – The user has the input velEntry entry box for inputting velocity into. | T2-20, T2-23, T2-27, T2-30, T2-34. T2-37, T2-41, T2-44 |
| Criteria 3 | Pass – The user has the input ampEntry entry box for inputting amplitude into. | T2-25, T2-32, T2-39, T2-46 |
| Criteria 4 | Pass – The user has the input wlEntry entry box for inputting wavelength into. | T2-19, T2-21, T2-26, T2-28, T2-33, T2-35, T2-40, T2-42 |
| Criteria 5 | Pass – The program will output to the input entry boxes, output entry boxes and the wave graph. | T2-47, T2-48, T2-49 |
| Criteria 6 | Pass – The program will output to the input entry boxes, output entry boxes and the wave graph. | T2-47, T2-48, T2-49 |
| Criteria 7 | Pass – The program will output to the input entry boxes, output entry boxes and the wave graph. | T2-47, T2-48, T2-49 |
| Criteria 8 | Pass – The program will output to the input entry boxes, output entry boxes and the wave graph. | T2-47, T2-48, T2-49 |
| Criteria 9 | Pass – The program outputs period to the extra info box at the bottom of the GUI | T2-47, T2-48, T2-49 |
| Criteria 10 | Pass – The program outputs photon energy and period via the info box in the GUI. | T2-47, T2-48, T2-49 |
| Criteria 11 | Pass – The program has a GUI | T2-1 all through T2-49 |
| Criteria 12 | Pass – The program using the researched equations in the 3 calculation procedures and the output procedure | T2-19 to T-24, T2-40 to T2-46 |
| Criteria 13 | Pass – All GUI components are placed and most of sized specifically to window height and width so they will fit on most sized screens | T2-1 |
| Criteria 14 | Pass – The program has 4 presets (red light, blue light, gamma wave and microwave) that the user can select from | T2-11 to T2-15 |
| Criteria 15 | Pass – The GUI has all inputs together, all outputs together, all presets together all sorted into boxes. | T2-1 |
| Criteria 16 | Pass – The program has unit choice for wavelength and amplitude, giving extra customisation to inputs over other programs. | T2-5 to T2-10 |
| Criteria 17 | Pass – The program has a moving visualisation of a wave on the GUI | T2-1, T2-47 to T2-49 |
| Criteria 18 | Pass – The program has radio buttons in order for the user to be able to select the dependent variables. | T2-2, T2-3, T2-4 |

## Client Feedback of Iteration 2

My client has said that the program having a moving visualisation of the wave is a pleasant surprise as that development is something I decided part way through this iteration. They appreciate the layout and the additional detailed variables in the bottom output tkinter box. The tix bubbles with the extra information explaining each variable is a good bonus as well and gives my program extra value. A single complaint was due to when it was opened on a smaller form factor laptop some text didn’t scale with the components.

## My feedback of Iteration 2

I think iteration 2 went well as it was my first time using matplotlib and especially learning to imbed it into a tkinter interface was a struggle. Happy it turned out that using a moving graph was actually easier as that was something I was looking to implement in a later iteration so it was good I was able to get it out in this iteration.

## Iteration 2 Final Code

[Appendix D – Iteration 2 Full Code](#_Appendix_D_–)

# Final Evaluation

## Final Success Criteria Evaluation

|  |  |  |
| --- | --- | --- |
| Success Criteria Number | Pass or Fail | Evidence of Pass or Fail |
| Success Criteria 1 | Pass – This program has an entry box for the user to input wavelength | T2-19, T2-21, T2-26, T2-28, T2-33, T2-35, T2-40, T2-42 |
| Success Criteria 2 | Pass – This program has entry boxes for the program to output wavelength | T2-47, T2-48, T2-49 |
| Success Criteria 3 | Pass – This program has an entry box for the user to input velocity | T2-20, T2-23, T2-27, T2-30, T2-34. T2-37, T2-41, T2-44 |
| Success Criteria 4 | Pass – This program has entry boxes for the program to output velocity. | T2-47, T2-48, T2-49 |
| Success Criteria 5 | Pass – This program has an entry box for the user to input frequency. | T2-22, T2-24, T2-29, T2-31, T2-36, T2-38, T2-43, T2-45 |
| Success Criteria 6 | Pass – This program has entry boxes for the program to output frequency. | T2-47, T2-48, T2-49 |
| Success Criteria 7 | Pass – This program has an entry box for the user to input amplitude. | T2-25, T2-32, T2-39, T2-46 |
| Success Criteria 8 | Pass – This program has entry boxes for the program to output amplitude. | T2-47, T2-48, T2-49 |
| Success Criteria 9 | Pass – This program has a selection of entry boxes outputting various dependent and independent variables of the wave. | T2-47, T2-48, T2-49 |
| Success Criteria 10 | Pass – This is solved with radio buttons that allow the user to select their desired dependent variable. | T2-2, T2-3, T2-4 |
| Success Criteria 11 | Pass – This program has a selection of 5 presets the user can pick towards using radio buttons. | T2-11 to T2-15 |
| Success Criteria 12 | Pass – This program uses the equations found from the research to calculate the dependent variables. | T2-19 to T-24, T2-40 to T2-46 |
| Success Criteria 13 | Pass – This program uses a tkinter GUI. | T2-1 |
| Success Criteria 14 | Pass – The GUI’s elements are all placed in accordance to a percentage of the window sized when zoomed to full screen | T2-1 |
| Success Criteria 15 | Pass – The GUI is sorted by shades of grey and elements are sorted inside boxes so users can see related elements. | T2-1 |
| Success Criteria 16 | Pass – The GUI window has an embedded animated graphed sine wave simulating the wave properties | T2-1, T2-47 to T2-49 |
| Success Criteria 17 | Pass – The program uses tkinter.tix balloons to give show extra details about certain variables | T2-50 |

## Final Key Variable Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable name | Data type | Local or global | Description of what stored | Where used | Justification |
| wlVAR | Float | Global | Stores the users input of wavelength. | Main Program, getWl, WlCalculation, VelCalculation, FreqCalculation, animation\_fig, Output | Holds the value for wavelength that the user can input or output. Hence linking to Success Criteria 1 and 2. |
| wlUnit | String | Global | Stores the users input of the wavelength unit. | Main Program, getWl, Output | This holds the customisable unit of measurement unit for the wavelength. |
| velVAR | Float | Global | Stores the users input of the velocity. | Main Program, getVel, WlCalculation, VelCalculation, FreqCalculation, animation\_fig, Output | Holds the value for velocity that the user can input or output. Hence linking to Success Criteria 3 and 4. |
| freqVAR | Float | Global | Stores the users input of the frequency. | Main Program, getFreq, WlCalculation, VelCalculation, FreqCalculation, Output | Holds the value for frequency that the user can input or output. Hence linking to Success Criteria 5 and 6. |
| ampVAR | Float | Global | Stores the users input of the amplitude. | Main Program, getWl, WlCalculation, VelCalculation, FreqCalculation, animation\_fig, Output | Holds the value for amplitude that the user can input or output. Hence linking to Success Criteria 7 and 8. |
| ampUnit | String | Global | Stores the users input of the amplitude unit. | Main Program, GetAmp | This holds the  customisable unit of  measurement unit  for the amplitude. |
| root | Tk Window | Global | Tk window that holds all GUI elements | Main Program | Main window for the  tkinter GUI, this is  apart of Success  Criteria 13. |
| waveFrame | Matplotlib Frame | Global | Frame that will hold matplotlib graph in GUI | Main Program | This is the frame in  the GUI that will  hold the plotted  graph. |
| planck | Float CONST | Global | Value for Planck’s constant | Main Program, Output | Used in the  calculation of  photon energy so  contributes to  Success Criteria 9. |
| mCanvas | Tk Canvas | Global | Canvas for box surrounding wavelength, velocity, frequency and amplitude inputs | Main Program | Canvas holds in the  entry of 4 key  variables in GUI,  contributes to  Success Criteria 15. |
| presetCanvas | Tk Canvas | Global | Canvas for box surrounding preset radio buttons | Main Program | Canvas holds in the  entry of 4 key  variables in GUI,  contributes to  Success Criteria 15. |
| infoCanvas | Tk Canvas | Global | Canvas for box surrounding the total output box | Main Program | Canvas holds in the  output of 6 key  variables in GUI,  contributes to  Success Criteria 15. |
| waveCanvas | Tk Canvas | Global | Canvas that will hold waveFrame on the GUI | Main Program, PlotGraph | Canvas holds the  wave graph in GUI,  contributes to  Success Criteria 15. |
| WaveGraph | matplotlib graph | Local | Holds plotted points from PlotGraph() procedure | PlotGraph | Graph for moving  wave visualisation,  Success Criteria 16. |
| x | numpy linspace | Local | Holds the start and end point of the graphed sine wave | animation\_fig | Used to graph wave,  Success Criteria 16. |
| y | float | Local | Plots sine function co ordinates | animation\_fig | Used to graph wave,  Success Criteria 16. |
| SineWave | matplotlib plot | Local | Holds the points to be plotted on WaveGraph | PlotGraph, animation\_fig, anim\_init | Used to graph wave,  Success Criteria 16. |
| period | float | Global | Holds calculated value of time period | Main Program, WlCalculation, VelCalculation, FreqCalculation, Output | Is one of dependent  variables outputted  by program, Success  Criteria 9. |
| energy | float | Local | Holds calculated value of photon energy | Output | Is one of dependent  variables outputted  by program, Success  Criteria 9. |
| varNonSelect | Tk StringVar | Global | Holds the selected value of the dependent variable radio buttons | Main Program, Output, WlCalculation, VelCalculation, FreqCalculation | Success Criteria 10  needs the user to be  able to select their  desired dependent  variable. |
| radioButtonSelect | Tk StringVar | Global | Holds the selected value of the preset radio buttons | Main Program, SelectWaveInput | This is the variable to  link together the  radio buttons  associated with  determining the  dependent variable. |
| wlSelectRB | Tk radio button | Global | Select wavelength for dependent variables | Main Program | Success Criteria 10  needs the user to be  able to select their  desired dependent  variable. |
| velSelectRB | Tk radio button | Global | Select velocity for dependent variables | Main Program | Success Criteria 10  needs the user to be  able to select their  desired dependent  variable. |
| freqSelectRB | Tk radio button | Global | Select frequency for dependent variables | Main Program | Success Criteria 10  needs the user to be  able to select their  desired dependent  variable. |
| wlEntryVAR | Tk StringVar | Global | Holds the value of the wavelength entry box | Main Program, getWl | Stores value of  wlEntry, used for  inputting and  outputting  wavelength,  therefore Success  Criteria 1 and 2. |
| wlLabel | Tk label | Global | Label for wavelength inputs | Main Program | Label for marking the  wavelength input |
| wlEntry | Tk entry box | Global | Entry box for user input for wavelength | Main Program | Entry box for  wavelength (Success  Criteria 1 and 2). |
| wlMeter | Tk radio button | Global | Radio button for editing the unit of the wavelength | Main Program | Radio button used  for more dynamic  range of inputs in my  program. |
| wlMicro | Tk radio button | Global | Radio button for editing the unit of the wavelength | Main Program | Radio button used  for more dynamic  range of inputs in my  program. |
| wlNano | Tk radio button | Global | Radio button for editing the unit of the wavelength | Main Program | Radio button used  for more dynamic  range of inputs in my  program. |
| velEntryVAR | Tk StringVar | Global | Holds the value of the velocity entry box | Main Program, getVel | Stores value of  velEntry, used for  inputting and  outputting  wavelength,  therefore Success  Criteria 3 and 4. |
| velLabel | Tk label | Global | Label for velocity inputs | Main Program | Label for marking the  velocity input |
| velEntry | Tk entry box | Global | Entry box for user input for velocity | Main Program | Entry box for  velocity (Success  Criteria 3 and 4). |
| freqEntryVAR | Tk StringVar | Global | Holds the value of the frequency entry box | Main Program, getFreq | Stores value of  freqEntry, used for  inputting and  outputting  wavelength,  therefore Success  Criteria 5 and 6. |
| freqLabel | Tk label | Global | Label for frequency inputs | Main Program | Label for marking the  frequency input |
| freqEntry | Tk entry box | Global | Entry box for user input for frequency | Main Program | Entry box for  frequency (Success  Criteria 5 and 6). |
| ampEntryVAR | Tk StringVar | Global | Holds the value of the amplitude entry box | Main Program, getAmp | Stores value of  wlEntry, used for  inputting and  outputting  amplitude,  therefore Success  Criteria 7 and 8. |
| ampLabel | Tk label | Global | Label for amplitude inputs | Main Program | Label for marking the  amplitude input |
| ampEntry | Tk entry box | Global | Entry box for user input for amplitude | Main Program | Entry box for  amplitude (Success  Criteria 7 and 8). |
| ampMeter | Tk radio button | Global | Radio button for editing the unit of the amplitude | Main Program | Radio buttons used  for more dynamic  range of inputs in my  program. |
| ampMilli | Tk radio button | Global | Radio button for editing the unit of the amplitude | Main Program | Radio buttons used  for more dynamic  range of inputs in my  program. |
| ampMicro | Tk radio button | Global | Radio button for editing the unit of the amplitude | Main Program | Radio buttons used  for more dynamic  range of inputs in my  program. |
| preLabel | Tk label | Global | Label for presets canvas box | Main Program | Label for marking the  preset box |
| redLightRadioB | Tk radio button | Global | Radio button for selecting the desired preset | Main Program | This is a preset radio  button that the user  can select to have a  wave preset hence  covering success  criteria 11. |
| blueLightRadioB | Tk radio button | Global | Radio button for selecting the desired preset | Main Program | This is a preset radio  button that the user  can select to have a  wave preset hence  covering success  criteria 11. |
| microWaveRadioB | Tk radio button | Global | Radio button for selecting the desired preset | Main Program | This is a preset radio  button that the user  can select to have a  wave preset hence  covering success  criteria 11. |
| gammaWaveRadioB | Tk radio button | Global | Radio button for selecting the desired preset | Main Program | This is a preset radio  button that the user  can select to have a  wave preset hence  covering success  criteria 11. |
| customWaveRadioB | Tk radio button | Global | Radio button for selecting the desired preset | Main Program | This is a preset radio  button that the user  can select to have a  wave preset hence  covering success  criteria 11. |
| wlOutEntry | Tk entry box | Global | Entry box for outputting wavelength value | Main program, Output | Success Criteria 2  says the program  should have an  output for  wavelength as it is  one of the potential  dependent variables  of the program. |
| ampOutEntry | Tk entry box | Global | Entry box for outputting amplitude value | Main program, Output | Success Criteria 8  says the program  should have an  output for amplitude  as it is one of the  potential dependent  variables of the  program. |
| velOutEntry | Tk entry box | Global | Entry box for outputting velocity value | Main program, Output | Success Criteria 4  says the program  should have an  output for velocity  as it is one of the  potential dependent  variables of the  program. |
| freqOutEntry | Tk entry box | Global | Entry box for outputting frequency value | Main program, Output | Success Criteria 6  says the program  should have an  output for frequency  as it is one of the  potential dependent  variables of the  program. |
| energyOutEntry | Tk entry box | Global | Entry box for outputting energy value | Main program, Output | Success Criteria 9  says the program  should have an  output for energy  as it is one of the  potential dependent  variables of the  program. |
| periodOutEntry | Tk entry box | Global | Entry box for outputting period value | Main program, Output | Success Criteria 9  says the program  should have an  output for time  period as it is one of  the potential  dependent  variables of the  program. |
| wlOutLabel | Tk label | Global | Label for wlOutEntry | Main program only | Marks the entry box  of the output it for  wavelength. |
| ampOutLabel | Tk label | Global | Label for ampOutEntry | Main program only | Marks the entry box  of the output it for  amplitude. |
| velOutLabel | Tk label | Global | Label for velOutEntry | Main program only | Marks the entry box  of the output it for  velocity. |
| freqOutLabel | Tk label | Global | Label for freqOutEntry | Main program only | Marks the entry box  of the output it for  frequency. |
| energyOutLabel | Tk label | Global | Label for energyOutEntry | Main program only | Marks the entry box  of the output it for  photon energy. |
| periodOutLabel | Tk label | Global | Label for periodOutEntry | Main program only | Marks the entry box  of the output it for  time period. |
| tooltips | Tk tix tooltip | Global | Tooltips used for extra information | Main program only | This is used to  give users extra  information on  key variables in  the program  (Success  Criteria 17). |

## Final Usability Features

### Evaluation

In the program we had a range of various entry boxes and radio buttons. Which means my program can use a large range of inputs which was a downside to the programs I noticed during my research. All the usability features successfully are effective as the entry boxes mean users can input a theoretically infinite amount of data as desired. The radio buttons mean that the user can select between units of entry as well as presets effectively as associated radio buttons only allow 1 to be enabled at a time.

### List of Entry Boxes (used for input at some point):

* + wlEntry
    - Input for wavelength.
    - A screenshot of a computer

      Description automatically generated
  + velEntry
    - Input for velocity.
    - A white rectangular object with black text

      Description automatically generated
  + freqEntry
    - Input for frequency.
    - A screenshot of a computer

      Description automatically generated
  + ampEntry
    - Input for amplitude.
    - A screenshot of a computer

      Description automatically generated

### List of Radio Buttons

* + wlSelectRB
    - Select wavelength for dependent variable.
    - A screenshot of a computer

      Description automatically generated
  + velSelectRB
    - Select velocity for dependent variable.
    - A screen shot of a computer

      Description automatically generated
  + freqSelectRB
    - Select frequency for dependent variable.
    - A screenshot of a computer

      Description automatically generated
  + wlMeter
    - Select metre to be wavelength unit.
    - A black and white circle with black text

      Description automatically generated
  + wlMicro
    - Select micrometre to be wavelength unit.
    - A black and white image of a black and white eye

      Description automatically generated
  + wlNano
    - Select nanometre to be wavelength unit.
    - 
  + ampMeter
    - Select metre to be amplitude unit.
    - A white circle with black text

      Description automatically generated
  + ampMicro
    - Select micrometre to be amplitude unit.
    - A white circle with black letters

      Description automatically generated
  + ampMilli
    - Select millimetre to be amplitude unit.
    - A black and white circle with a black circle

      Description automatically generated
  + redLightRadioB
    - Select red light preset.
    - A close up of a logo

      Description automatically generated
  + blueLightRadioB
    - Select blue light preset.
    - A white circle with black text

      Description automatically generated
  + microWaveRadioB
    - Select microwave wave preset.
    - A black and white text

      Description automatically generated
  + gammaWaveRadioB
    - Select gamma wave preset.
    - A black and white text

      Description automatically generated
  + customWaveRadioB
    - Select custom preset.
    - A black and white logo

      Description automatically generated

## Maintenance

A screenshot of a computer program

Description automatically generatedA screenshot of a computer screen

Description automatically generatedBasically all procedures and variables follow the same naming conventions. (e.g. getWl, getVel ; wlVAR, velVAR ; velEntry, freqEntry. Most of the procedures and functions will have code comments every few lines as well as one on the defining line explaining what most of the lines of code will be doing and lines that pass parameters will have a comment explaining which parameter corresponds to what. Variable names follow camel cap rules and the global variables when defined come with a comment explaining what they are designed to hold. This should mean that people who access the code should be able to read it and understand what corresponds to what. Even down to how I’ve organised the code. Procedures and tkinter elements that relate will be grouped with 2 tabbed spaces between them that split up the code into ‘themed’ sections. This makes the code intuitive to something who understands coding and python.

## Limitations and future developments

If I were to continue developing the program I would make the size-ability of the GUI elements more robust and figure out what sort of ratios would make the text and entry boxes fit smaller form factor screens better. I would make the design more pleasing potentially whilst keeping it undistracting and professional. Might of added a few pop ups to tell the user how to use the program and give an options menu to make the program more widely user friendly. Given some of the limitations of the libraries used and of the language python I think the program is a good foundation for what is a good concept of project and works in what it set out to do and is useable for my client in a classroom and educational atmosphere with their students. In my analysis I planned on doing an extra iteration to solve these issues however after the duration of the project my client is happy to use the current produced project for his students in all year groups throughout

## Final client review

The client has said they’re satisfied with the usability of the program and that it has a wider range of utility versus some of the competitors that I researched. They’re happy that the program ticks all the essential success criteria and that students can use it on the schools computers with how it sizes to the screens and that the GUI looks is easy to follow and provides key elements. The additional information in the tooltips bubbles and the additional variables shown in the output box make it useable across a range of year groups and it’s a good beginner friendly tool.

# Bibliography

Matplotlib. (n.d.). *Animations using matplotlib*. Retrieved from Matplotlib: https://matplotlib.org/stable/users/explain/animations/animations.html

NumPy. (n.d.). *numpy.arange* . Retrieved from numpy.org: https://numpy.org/doc/stable/reference/generated/numpy.arange.html

O'Neill, M. (2015). *OCR AS/A Level Physics A.* Pearson Education Limited.

Phet Interactive Simulations. (2022, 10 05). *Wave on a String 1.1.24*. Retrieved from Wave on a String: https://phet.colorado.edu/sims/html/wave-on-a-string/latest/wave-on-a-string\_en.html

the Physics Classroom. (2022, 10 06). *Physics Simulation: Simple Wave Simulator*. Retrieved from Simple Wave Simulator: https://www.physicsclassroom.com/Physics-Interactives/Waves-and-Sound/Simple-Wave-Simulator/Simple-Wave-Simulator-Interactive

w3schools. (n.d.). *Matplotlib Tutorial*. Retrieved from w3schools.com: https://www.w3schools.com/python/matplotlib\_intro.asp

# Appendices

* Appendix A – Iteration 1 Test Screenshots
* Appendix B – Iteration 1 Full Code
* Appendix C – Iteration 2 Test Screenshots
* Appendix D – Iteration 2 Full Code

## Appendix A – Iteration 1 Test Screenshots

### A1

A close up of a white background

Description automatically generated

### A2

A close-up of a white background

Description automatically generated

### A3



### A4

A close up of a white background

Description automatically generated

### A5



### A6

A close-up of a white background

Description automatically generated

### A7



### A8

A close up of a text

Description automatically generated with medium confidence

### A9



### A10



### A11

A close-up of a white background

Description automatically generated

### A12

A close up of blue text

Description automatically generated

### A13

A blue text on a white background

Description automatically generated

### A14

Blue text on a white background

Description automatically generated

### A15

A white background with black text

Description automatically generated

### A16

A white background with black text

Description automatically generated

### A17

A white background with blue text

Description automatically generated

### A18

Blue text on a white background

Description automatically generated

### A19

A close-up of a white background

Description automatically generated

### A20

A close-up of a sign

Description automatically generated

### A21

A close-up of a sign

Description automatically generated

### A22

A close-up of a white background

Description automatically generated

### A23

A close-up of a sign

Description automatically generated

### A24

A close up of words

Description automatically generated

### A25

A close-up of a sign

Description automatically generated

### A26

A black text on a white background

Description automatically generated

### A27

A screenshot of a computer program

Description automatically generated

### A28

A screenshot of a computer

Description automatically generated

### A29

A screenshot of a computer code

Description automatically generated

## Appendix B – Iteration 1 Full Code

wlVAR = float(0.0)#wavelength

velVAR = float(0.0)#velocity

freqVAR = float(0.0)#frequency

ampVAR = float(0.0)#amplitude

period = float(0.0)#period

def inputVARFloat(message): #function to validate data type as float

while True:

try:

varInput = float(input(message)) #attempts to cast user input as float

except ValueError: #if an error message is returned

print("Please enter integer/floating point value") #output to user

continue #re-ask query

else:

if varInput > 0.0:

return varInput #returns result of input to program if it can be casted as float

break

else:

print("Please enter value above 0")

continue #re-ask query

def Calculation(): #procedure for mathematical calculations

global wlVAR, velVAR, freqVAR, period #delcaring the global variables used in the procedure

if wlVAR != 0.0: #if the user has provided an input for wavelength

if freqVAR != 0.0: #if the user has provided an input for frequency

velVAR = float(wlVAR \* freqVAR)

period = float(1 / freqVAR)

printOutput() #will execute the procedure to print the results of the maths

else: # if the user has provided an input for velocity

freqVAR = float(velVAR / wlVAR)

period = float(1 / freqVAR)

printOutput() #will execute the procedure to print the results of the maths

else: #if the user has provided inputs for frequency and velocity

wlVAR = float(velVAR / freqVAR)

period = float(1 / freqVAR)

printOutput() #will execute the procedure to print the results of the maths

def printOutput(): #procedure for printing the result of calculations

global wlVAR, velVAR, freqVAR, ampVAR, period #delcaring the global variables used in the procedure

print("Your wavelength is:", wlVAR, "metres") #wavelength

print("Your velocity is:", velVAR, "metres per second") #velocity

print("Your frequency is:", freqVAR, "hertz") #frequency

print("Your amplitude is:", ampVAR, "metres") #amplitude

print("The period of your wave is:", period, "seconds") #period

wlVAR = 0

velVAR = 0

freqVAR = 0

ampVAR = 0

period = 0

prevOp = " " #previous option

noEnt = 0 #number of entered data

while noEnt < 2: #indefinite iteration

if noEnt == 0:

option = input("Select a variable to input (Wavelength (or w), Frequency (or f), Velocity (or v)): ") #initial input

option = option.lower() #standardise the input to lowercase

if option == "wavelength" or option == "w": #if user inputted wavelength (or variance)

prevOp = "wl" #making wavelength the last picked option

print("you picked wavelength")

wlVAR = inputVARFloat("Please input wavelength in metres: ") #prompting user input using input function

print("wavelength =", wlVAR)

print("PrevOp =", prevOp)

noEnt = 1 #changing the if perameter

continue

elif option == "frequency" or option == "f": #if user inputted frequency (or variance)

prevOp = "freq" #making frequency the last picked option

print("you picked frequency")

freqVAR = inputVARFloat("Please input frequency in hertz: ") #prompting user input using input function

print("frequency =", freqVAR)

print("PrevOp =", prevOp)

noEnt = 1 #changing the if perameter

continue

elif option == "velocity" or option == "v": #if user inputted velocity (or variance)

prevOp = "vel" #making velocity the last picked option

print("you picked velocity")

velVAR = inputVARFloat("Please input velocity in metres/second: ") #prompting user input using input function

print("velocity =", velVAR)

print("PrevOp =", prevOp)

noEnt = 1 #changing the if perameter

continue

else:

print("Please enter valid input") #defensive design

elif noEnt == 1:

if prevOp == "wl": #if the user has already provided a value for the wavelength

option = input("Select a variable to input (Frequency (or f), Velocity (or v)): ")

option = option.lower() #standardise the input to lowercase

if option == "frequency" or option == "f": #if user inputted frequency (or variance)

print("you picked frequency")

freqVAR = inputVARFloat("Please input frequency in hertz: ") #input

print("frequency =", freqVAR)

ampVAR = inputVARFloat("Please input an amplitude in metres: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

elif option == "velocity" or option == "v": #if user inputted velocity (or variance)

print("you picked velocity")

velVAR = inputVARFloat("Please input velocity in metres/second: ") #input

print("velocity =", velVAR)

ampVAR = inputVARFloat("Please input an amplitude in metres: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

else:

print("Please enter valid input") #defensive design

elif prevOp == "freq": #if the user has already provided a value for the frequency

option = input("Select a variable to input (Wavelength (or w), Velocity (or v)): ")

option = option.lower() #standardise the input to lowercase

if option == "wavelength" or option == "w": #if user inputted wavelength (or variance)

print("you picked wavelength")

wlVAR = inputVARFloat("Please input wavelength in metres: ") #input

print("wavelength =", wlVAR)

ampVAR = inputVARFloat("Please input an amplitude in metres: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

elif option == "velocity" or option == "v": #if user inputted velocity (or variance)

print("you picked velocity")

velVAR = inputVARFloat("Please input velocity in meters/second: ") #input

print("velocity =", velVAR)

ampVAR = inputVARFloat("Please input an amplitude in meters: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

else:

print("Please enter valid input") #defensive design

elif prevOp == "vel": #if the user has already provided a value for the velocity

option = input("Select a variable to input (Wavelength (or w), Frequency (or f): ")

option = option.lower() #standardise the input to lowercase

if option == "frequency" or option == "f": #if user inputted frequency (or variance)

print("you picked frequency")

freqVAR = inputVARFloat("Please input wavelength in hertz: ") #input

print("frequency =", freqVAR)

ampVAR = inputVARFloat("Please input an amplitude in meters: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

elif option == "wavelength" or option == "w": #if user inputted wavelength (or variance)

print("you picked wavelength")

wlVAR = inputVARFloat("Please input wavelength in metres: ") #input

print("wavelength =", wlVAR)

ampVAR = inputVARFloat("Please input an amplitude in metres: ") #amplitude input

print("Calculating...")

Calculation() #runs Calculation()

noEnt=0 #keeps program within loop parameters

else:

print("Please enter valid input") #defensive design

else:

print("Program error") #defensive design

noEnt = 0

## Appendix C – Iteration 2 Test Screenshots

### C1

A screenshot of a computer

Description automatically generated

### C2

A screenshot of a computer

Description automatically generated

### C3

A screenshot of a computer

Description automatically generated

### C4

A screenshot of a computer

Description automatically generated

### C5

A screenshot of a computer

Description automatically generated

### C6

A screenshot of a computer

Description automatically generated

### C7

A screenshot of a computer

Description automatically generated

### C8

A screenshot of a computer

Description automatically generated

### C9

A screenshot of a computer

Description automatically generated

### C10

A white rectangle with black text

Description automatically generated

### C11

A screenshot of a computer

Description automatically generated

### C12

A screenshot of a computer

Description automatically generated

### C13

A screenshot of a computer

Description automatically generated

### C14

A screenshot of a computer

Description automatically generated

### C15

A screenshot of a computer

Description automatically generated

### C16

A screenshot of a computer

Description automatically generated

### C17

A screenshot of a computer

Description automatically generated

### C18

A screenshot of a computer

Description automatically generated

### C19

A screenshot of a computer

Description automatically generated

### C20

A screenshot of a computer

Description automatically generated

### C21

A screenshot of a computer

Description automatically generated

### C22

A screenshot of a computer

Description automatically generated

### C23

A screenshot of a computer

Description automatically generated

### C24

A screenshot of a computer

Description automatically generated

### C25

A screenshot of a computer

Description automatically generated

### C26

A screenshot of a computer

Description automatically generated

### C27

A screenshot of a computer

Description automatically generated

### C28

A screenshot of a computer

Description automatically generated

### C29

A screenshot of a computer

Description automatically generated

### C30

A screenshot of a computer

Description automatically generated

### C31

A screenshot of a computer

Description automatically generated

### C32

A screenshot of a computer

Description automatically generated

### C33

A screenshot of a computer

Description automatically generated

### C34

A screenshot of a computer

Description automatically generated

### C35

A screenshot of a computer

Description automatically generated

### C36

A screenshot of a computer

Description automatically generated

### C37

A screenshot of a computer

Description automatically generated

### C38A screenshot of a computer Description automatically generated

### C39

A screenshot of a computer

Description automatically generated

### C40

A screenshot of a computer

Description automatically generated

### C41

A screenshot of a computer

Description automatically generated

### C42

A screenshot of a computer

Description automatically generated

### C43

A screenshot of a computer

Description automatically generated

### C44

A screenshot of a computer

Description automatically generated

### C45

A screenshot of a computer

Description automatically generated

### C46

A screenshot of a computer

Description automatically generated

### C47

A screenshot of a computer

Description automatically generated

### C48

A screenshot of a computer

Description automatically generated

### C49

A screenshot of a computer

Description automatically generated

### C50

A screenshot of a computer

Description automatically generated

## Appendix D – iteration 2 Full Code

from tkinter import \*

import numpy as np

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg)

from tkinter.tix import \*

from matplotlib.animation import FuncAnimation

import matplotlib.pyplot as plt

root = Tk()

root.title("Iteration 2")

root.configure(background="grey")

root.state('zoomed') #fits monitor size

waveFrame = Frame(root)

waveFrame.place(x= root.winfo\_screenwidth() \* (30/100), y= root.winfo\_screenheight() \* (10/100) )

#declaring global variables

wlVAR = float(1.0) #wavelength

velVAR = float(1.0) #velocity

freqVAR = float(0.0) #frequency

ampVAR = float(1.0) #amplitude

period = float(0.0) #period

planck = float((6.62607015)\*(10\*\*-34))

#tkinter canvas'

mCanvas = Canvas(root, height=root.winfo\_screenheight() \* (78/100) , width=root.winfo\_screenwidth() \* (20/100), bg="light grey")

mCanvas.place(x= (root.winfo\_screenwidth() \* (5/100)), y= (root.winfo\_screenheight() \* (10/100)) )

presetCanvas = Canvas(root, height=root.winfo\_screenheight() \* (32/100), width=root.winfo\_screenwidth() \* (20/100), bg="light grey")

presetCanvas.place(x= root.winfo\_screenwidth() \* (70/100), y=root.winfo\_screenheight() \* (56/100) )

infoCanvas = Canvas(root, height=root.winfo\_screenheight() \* (32/100), width=root.winfo\_screenwidth() \* (30/100), bg="light grey")

infoCanvas.place(x= root.winfo\_screenwidth() \* (30/100), y=root.winfo\_screenheight() \* (56/100) )

waveCanvas = None

def PlotGraph():

global waveCanvas

WaveGraph, wGraph = plt.subplots() #creates matplotlib Figure and an array of axes that the program can plot points on

WaveGraph.set\_figheight( (root.winfo\_screenheight()/180) ) #sizes graph to window size

WaveGraph.set\_figwidth( (root.winfo\_screenwidth()/120) ) #sizes graph to window size

wGraph.set\_ylabel("Amplitude (m)") # y axis title

wGraph.set\_xlabel("Time (s)") # x axis title

if waveCanvas: waveCanvas.get\_tk\_widget().pack\_forget() #replaces previous graph when new variables are used and new graph formed

waveCanvas = FigureCanvasTkAgg(WaveGraph, master = waveFrame) #creates graph inside figure that can be embedded in tkinter

waveCanvas.get\_tk\_widget().pack()

wGraph.set\_ylim(-(ampVAR+2), (ampVAR+2)) #sets the limits of the y-axis to be slightly greater than the amplitude of the wave

def animation\_fig(frame):

global wlVAR

global velVAR

global ampVAR

x = np.linspace(frame \* velVAR, (frame +10) \*velVAR, 800) #num1 = start point, num2 = end point, num3 = no. of generated coords

y = ( np.sin(2\*np.pi / wlVAR \* x) \*ampVAR) # y co-ordinates, models wave as sine wave

sineWave.set\_data(x, y) #sineWave is the name given to the line created by plotting the x and y variables

wGraph.set\_xlim(x.min(), x.max()) #sets a limit on the x-axis on either side of the created line to create an illusion of scrolling movement

return sineWave, #the comma returns the full tuple with the sineWave variable included within it

#The init function is the initialisation of the graph for each frame of animation to be placed on top of.

def anim\_init():

sineWave.set\_data([], [])#this sets the x and y values of anything plotted on the graph to blank to clear the graph

return sineWave, #returns these empty values to the FuncAnimation function

sineWave, = wGraph.plot([], []) #clears the plotted points of the graph ready to have the next set of points plotted

animate = FuncAnimation(WaveGraph, animation\_fig, init\_func=anim\_init, blit=True, interval=50 ) #animates the graph

waveCanvas.draw() #draws the FuncAnimation onto the graph in the tkinter window

def getWl(var, index, mode):#whenever edit to wavelength input occurs ("write" trace)

global wlVAR

global wlUnit

wlVAR = float(wlEntryVAR.get()) #value of wlEntry box

if wlUnit.get() == "m":#check radiobutton value

print(wlVAR)

print(wlUnit.get())#prints unit alongside value to console

wlCalculation()

elif wlUnit.get() == "nm":#check radiobutton value

print(wlVAR)

print(wlUnit.get())

wlVAR = float(wlVAR / 1000000000)#convert wl to correct unit for calculation (m)

print(str(wlVAR)+"m")#prints wavelength as seen as in metres to console

wlCalculation()

else:#check radiobutton value

print(wlVAR)

print(wlUnit.get())

wlVAR = float(wlVAR / 1000000)#convert wl to correct unit for calculation (m)

print(str(wlVAR)+"m")#prints wavelength as seen as in metres to console

wlCalculation()

def getVel(var, index, mode):#whenever edit to velocity input occurs ("write" trace)

global velVAR

velVAR = float(velEntryVAR.get())#value of velEntry

print(velVAR)

velCalculation()

def getFreq(var, index, mode):#whenever edit to frequency input occurs ("write" trace)

global freqVAR

freqVAR = float(freqEntryVAR.get())#value of freqEntry

print(freqVAR)

freqCalculation()

def getAmp(var, index, mode):#whenever edit to amplitude input occurs ("write" trace)

global ampVAR

global ampUnit

ampVAR = float(ampEntryVAR.get()) #value of ampEntry

test = 1/ampVAR #Defensive design

if ampUnit.get() == "m":#check radiobutton value

print(ampVAR)

print(ampUnit.get())

Output()

elif ampUnit.get() == "mm":#check radiobutton value

print(ampVAR)

print(ampUnit.get())

ampVAR = float(ampVAR / 1000)#convert amplitude to correct unit for calculation (m)

print(str(ampVAR)+"m")

Output()

else:#check radiobutton value

print(ampVAR)

print(ampUnit.get())

ampVAR = float(ampVAR / 1000000)#convert amplitude to correct unit for calculation (m)

print(str(ampVAR)+"m")

Output()

def wlCalculation():

global wlVAR

global velVAR

global freqVAR

global period

if varNonSelect.get() == "Freq":

freqVAR = velVAR / wlVAR #Use of the wl = vel / freq to find frequency

period = float(1 / freqVAR) #Use of the period = 1 / frequency to find period

Output()

elif varNonSelect.get() == "Vel":

velVAR = wlVAR \* freqVAR #use of the wl = vel / freq to find velocity

period = float(1 / freqVAR) #use of period = 1 / frequency to find period

Output()

else:

print("Dependent Variable Error has occured") #defensive design

def velCalculation():

global wlVAR

global velVAR

global freqVAR

global period

if varNonSelect.get() == "Wl":

wlVAR = velVAR / freqVAR #Use of the wl = vel / freq to find frequency

period = float(1 / freqVAR) #Use of the period = 1 / frequency to find period

Output()

elif varNonSelect.get() == "Freq":

freqVAR = velVAR/wlVAR #Use of the wl = vel / freq to find frequency

period = float(1 / freqVAR) #Use of the period = 1 / frequency to find period

Output()

else:

print("Dependent Variable Error has occured") #defensive design

def freqCalculation():

global wlVAR

global velVAR

global freqVAR

global period

if varNonSelect.get() == "Wl":

wlVAR = velVAR / freqVAR #Use of the wl = vel / freq to find frequency

period = float(1 / freqVAR) #Use of the period = 1 / frequency to find period

Output()

elif varNonSelect.get() == "Vel":

velVAR = wlVAR \* freqVAR #Use of the wl = vel / freq to find frequency

period = float(1 / freqVAR) #Use of the period = 1 / frequency to find period

Output()

else:

print("Dependent Variable Error has occured") #defensive design

def Output():

global wlVAR

global velVAR

global freqVAR

global period

global planck

energy = (planck \* freqVAR)

energy = (energy \* (6.242\*(10\*\*18))) #convert from joules to electronvolts

print(wlVAR)

print(velVAR)

print(freqVAR)

print(period)

wlOutEntry.config(state=NORMAL)

wlOutEntry.delete(0, END)

wlOutEntry.insert(0, wlVAR)

wlOutEntry.config(state=DISABLED)

ampOutEntry.config(state=NORMAL)

ampOutEntry.delete(0, END)

ampOutEntry.insert(0, ampVAR)

ampOutEntry.config(state=DISABLED)

velOutEntry.config(state=NORMAL)

velOutEntry.delete(0, END)

velOutEntry.insert(0, velVAR)

velOutEntry.config(state=DISABLED)

freqOutEntry.config(state=NORMAL)

freqOutEntry.delete(0, END)

freqOutEntry.insert(0, freqVAR)

freqOutEntry.config(state=DISABLED)

energyOutEntry.config(state=NORMAL)

energyOutEntry.delete(0, END)

energyOutEntry.insert(0, energy)

energyOutEntry.config(state=DISABLED)

periodOutEntry.config(state=NORMAL)

periodOutEntry.delete(0, END)

periodOutEntry.insert(0, period)

periodOutEntry.config(state=DISABLED)

if varNonSelect.get() == "Wl": #check if wl is the dependent variable

if wlUnit.get() == "m": #get to see wl unit

wlEntry.config(state=NORMAL)

wlEntry.delete(0, END) #empty wavelength entry box

wlEntry.insert(0, wlVAR) #fill wavelength entry box with calculated variable

wlEntry.config(state=DISABLED)

PlotGraph()

elif wlUnit.get() == "nm": #get to see wl unit

wlEntry.config(state=NORMAL)

wlEntry.delete(0, END) #empty wavelength entry box

wlEntry.insert(0, (wlVAR \*(10\*\*9)) ) #fill wavelength entry box with calculated variable

wlEntry.config(state=DISABLED)

PlotGraph()

else: #get to see wl unit

wlEntry.config(state=NORMAL)

wlEntry.delete(0, END) #empty wavelength entry box

wlEntry.insert(0, (wlVAR \*(10\*\*6)) ) #fill wavelength entry box with calculated variable

wlEntry.config(state=DISABLED)

PlotGraph()

elif varNonSelect.get() == "Vel": #check if vel is the dependent variable

velEntry.config(state=NORMAL)

velEntry.delete(0, END) #empty velocity entry box

velEntry.insert(0, velVAR) #fill velocity entry box with calculated variable

velEntry.config(state=DISABLED)

PlotGraph()

elif varNonSelect.get() == "Freq": #check if freq is the dependent variable

freqEntry.config(state=NORMAL)

freqEntry.delete(0, END) #empty frequency entry box

freqEntry.insert(0, freqVAR) #fill frequency entry box with calculated variable

freqEntry.config(state=DISABLED)

PlotGraph()

else:

print("Output error has occured")#defensive design

PlotGraph()

def SelectGrayedInput():

global varNonSelect

if varNonSelect.get() == "Freq":

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

Output()

elif varNonSelect.get() == "Vel":

velEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

freqEntry.config(state=NORMAL)

Output()

elif varNonSelect.get() == "Wl":

wlEntry.config(state=DISABLED)

freqEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

Output()

def SelectWaveInput():

global radioButtonSelect

if radioButtonSelect.get() == "Red":

freqEntry.config(state=NORMAL)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

ampMeter.select()

freqSelectRB.select()

wlEntry.delete(0, END)

freqEntry.delete(0, END)

velEntry.delete(0, END)

ampEntry.delete(0, END)

wlEntry.insert(0, 700)

velEntry.insert(0, 299792458)

ampEntry.insert(0, 1.0)

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

wlNano.select()

elif radioButtonSelect.get() == "Blue":

freqEntry.config(state=NORMAL)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

ampMeter.select()

freqSelectRB.select()

wlEntry.delete(0, END)

freqEntry.delete(0, END)

velEntry.delete(0, END)

ampEntry.delete(0, END)

wlEntry.insert(0, 460)

velEntry.insert(0, 299792458)

ampEntry.insert(0, 1.0)

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

wlNano.select()

elif radioButtonSelect.get() == "Micro":

freqEntry.config(state=NORMAL)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

ampMeter.select()

freqSelectRB.select()

wlEntry.delete(0, END)

freqEntry.delete(0, END)

velEntry.delete(0, END)

ampEntry.delete(0, END)

wlEntry.insert(0, 1000)

velEntry.insert(0, 299792458)

ampEntry.insert(0, 1.0)

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

wlMicro.select()

elif radioButtonSelect.get() == "Gamma":

print("gamma")

freqEntry.config(state=NORMAL)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

ampMeter.select()

freqSelectRB.select()

wlEntry.delete(0, END)

freqEntry.delete(0, END)

velEntry.delete(0, END)

ampEntry.delete(0, END)

wlEntry.insert(0, 0.005)

velEntry.insert(0, 299792458)

ampEntry.insert(0, 1.0)

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

wlNano.select()

elif radioButtonSelect.get() == "Custom":

freqEntry.config(state=NORMAL)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

ampMeter.select()

freqSelectRB.select()

wlEntry.delete(0, END)

freqEntry.delete(0, END)

velEntry.delete(0, END)

ampEntry.delete(0, END)

wlEntry.insert(0, 10.0)

velEntry.insert(0, 1.0)

ampEntry.insert(0, 1.0)

freqEntry.config(state=DISABLED)

wlEntry.config(state=NORMAL)

velEntry.config(state=NORMAL)

wlMeter.select()

#Radiobutton Select Variable

varNonSelect = StringVar()

wlSelectRB = Radiobutton(root, variable=varNonSelect, value="Wl", bg="light grey", activebackground="light grey", command=SelectGrayedInput)#button for selecting wavelength

velSelectRB = Radiobutton(root, variable=varNonSelect, value="Vel", bg="light grey", activebackground="light grey", command=SelectGrayedInput)#button for selecting velocity

freqSelectRB = Radiobutton(root, variable=varNonSelect, value="Freq", bg="light grey", activebackground="light grey", command=SelectGrayedInput)#button for selecting frequency

wlSelectRB.place(x= root.winfo\_screenwidth() \*(8/100), y= root.winfo\_screenheight() \* (16.25/100) )

velSelectRB.place(x= root.winfo\_screenwidth() \*(8/100), y= root.winfo\_screenheight() \* (28.75/100) )

freqSelectRB.place(x= root.winfo\_screenwidth() \*(8/100), y= root.winfo\_screenheight() \* (41.25/100) )

freqSelectRB.select()

#wavelength

wlEntryVAR = StringVar()

wlLabel = Label(root, text='Wavelength', bg="light grey")#label

wlLabel.place(x= root.winfo\_screenwidth() \* (13/100), y= root.winfo\_screenheight() \* (12.5/100) )

wlEntry = Entry(root, bd="1", relief="solid", textvariable= wlEntryVAR)#entry box

wlEntry.place(x= root.winfo\_screenwidth() \* (11.5/100), y= root.winfo\_screenheight() \* (16.5/100) )

wlEntryVAR.trace\_add("write", callback=getWl)#constantly updating edits to wlVAR (wlEntry input)

#wavelength radiobuttons

wlUnit = StringVar() #designating wlUnit as variable for radio buttons

wlMeter = Radiobutton(root, text="m", variable=wlUnit, value="m", bg="light grey", activebackground="light grey")#button for 'm'

wlMicro = Radiobutton(root, text="μm", variable=wlUnit, value="μm", bg="light grey", activebackground="light grey")#button for 'μm'

wlNano = Radiobutton(root, text="nm", variable=wlUnit, value="nm", bg="light grey", activebackground="light grey") #button for 'nm'

wlMeter.place(x= root.winfo\_screenwidth() \*(10/100), y= root.winfo\_screenheight() \* (20/100) )

wlMicro.place(x= root.winfo\_screenwidth() \*(13.5/100), y= root.winfo\_screenheight() \* (20/100) )

wlNano.place(x= root.winfo\_screenwidth() \*(17/100), y= root.winfo\_screenheight() \* (20/100) )

wlMeter.select()#default select when open program

wlUnit.trace\_add("write", callback=getWl)#constantly updating edits to wlUnit (changes to radiobuttons)

#velocity input

velEntryVAR = StringVar()

velLabel = Label(root, text="Velocity (m/s)", bg="light grey")#label

velLabel.place(x= (root.winfo\_screenwidth() \* (12.75/100)), y= (root.winfo\_screenheight() \* (25/100)) )

velEntry = Entry(root, bd="1", relief="solid", textvariable= velEntryVAR)#entry box

velEntry.place(x= root.winfo\_screenwidth() \* (11.5/100), y= root.winfo\_screenheight() \* (29/100) )

velEntryVAR.trace\_add("write", callback=getVel)#constantly updating edits to velVAR (velEntry input)

#frequency input

freqEntryVAR = StringVar()

freqLabel = Label(root, text="Frequency (Hz)", bg="light grey")#label

freqLabel.place(x= (root.winfo\_screenwidth() \* (12.75/100)), y= (root.winfo\_screenheight() \* (37.5/100)) )#label

freqEntry = Entry(root, bd="1", relief="solid", textvariable= freqEntryVAR)#entry box

freqEntry.place(x= root.winfo\_screenwidth() \* (11.5/100), y= root.winfo\_screenheight() \* (41.5/100) )

freqEntryVAR.trace\_add("write", callback=getFreq)#constantly updating edits to freqVAR (freqEntry input)

#amplitude input

ampEntryVAR = StringVar()

ampLabel = Label(root, text="Amplitude", bg="light grey")#label

ampLabel.place(x= (root.winfo\_screenwidth() \* (13/100)), y= (root.winfo\_screenheight() \* (50/100)) )#label

ampEntry = Entry(root, bd="1", relief="solid", textvariable= ampEntryVAR)#entry box

ampEntry.place(x= root.winfo\_screenwidth() \* (11.5/100), y= root.winfo\_screenheight() \* (54/100) )

ampEntryVAR.trace\_add("write", callback=getAmp)#constantly updating edits to ampVAR (ampEntry input)

#amplitude radiobuttons

ampUnit = StringVar() #designating ampUnit as variable for radio buttons

ampMeter = Radiobutton(root, text="m", variable=ampUnit, value="m", bg="light grey", activebackground="light grey")#button for 'm'

ampMilli = Radiobutton(root, text="mm", variable=ampUnit, value="mm", bg="light grey", activebackground="light grey")#button for 'mm'

ampMicro = Radiobutton(root, text="μm", variable=ampUnit, value="μm", bg="light grey", activebackground="light grey") #button for 'μm'

ampMeter.place(x= root.winfo\_screenwidth() \*(10/100), y= root.winfo\_screenheight() \* (57.5/100) )

ampMilli.place(x= root.winfo\_screenwidth() \*(13.5/100), y= root.winfo\_screenheight() \* (57.5/100) )

ampMicro.place(x= root.winfo\_screenwidth() \*(17/100), y= root.winfo\_screenheight() \* (57.5/100) )

ampMeter.select()#default select when open program

ampUnit.trace\_add("write", callback=getAmp)#constantly updating edits to ampUnit (changes to radiobuttons)

#PRESETS BOX

preLabel = Label(root, text="WAVE PRESETS", bg="light grey", font= ('TkDefaultFont', 20))

preLabel.place(x= root.winfo\_screenwidth() \* (73.5/100), y= root.winfo\_screenheight() \* (58.5/100))

radioButtonSelect = StringVar()

redLightRadioB = Radiobutton(root, variable=radioButtonSelect, text="Red Light", value="Red", bg="light grey", activebackground="light grey", command=SelectWaveInput) #radiobutton

blueLightRadioB = Radiobutton(root, variable=radioButtonSelect, text="Blue Light",value="Blue", bg="light grey", activebackground="light grey", command=SelectWaveInput) #radiobutton

microWaveRadioB = Radiobutton(root, variable=radioButtonSelect, text="Microwave",value="Micro", bg="light grey", activebackground="light grey", command=SelectWaveInput) #radiobutton

gammaWaveRadioB = Radiobutton(root, variable=radioButtonSelect, text="Gamma ray",value="Gamma", bg="light grey", activebackground="light grey", command=SelectWaveInput) #radiobutton

customWaveRadioB = Radiobutton(root, variable=radioButtonSelect, text="Custom",value="Custom", bg="light grey", activebackground="light grey", command=SelectWaveInput) #radiobutton

customWaveRadioB.select() #select a button when program is run

redLightRadioB.place(x= root.winfo\_screenwidth() \* (75/100), y= root.winfo\_screenheight() \* (62.4/100) )

blueLightRadioB.place(x= root.winfo\_screenwidth() \* (75/100), y= root.winfo\_screenheight() \* (68.8/100) )

microWaveRadioB.place(x= root.winfo\_screenwidth() \* (75/100), y= root.winfo\_screenheight() \* (75.2/100) )

gammaWaveRadioB.place(x= root.winfo\_screenwidth() \* (75/100), y= root.winfo\_screenheight() \* (81.6/100) )

customWaveRadioB.place(x= root.winfo\_screenwidth() \* (82.5/100), y= root.winfo\_screenheight() \* (72/100) )

wlOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

wlOutEntry.place(x= root.winfo\_screenwidth() \* (32.5/100), y= root.winfo\_screenheight() \* (61.35/100))

ampOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

ampOutEntry.place(x= root.winfo\_screenwidth() \* (32.5/100), y= root.winfo\_screenheight() \* (72.05/100))

velOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

velOutEntry.place(x= root.winfo\_screenwidth() \* (32.5/100), y= root.winfo\_screenheight() \* (82.75/100))

freqOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

freqOutEntry.place(x= root.winfo\_screenwidth() \* (47.5/100), y= root.winfo\_screenheight() \* (61.35/100))

energyOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

energyOutEntry.place(x= root.winfo\_screenwidth() \* (47.5/100), y= root.winfo\_screenheight() \* (72.05/100))

periodOutEntry = Entry(root, bd="1", relief="solid") #entry box creation

periodOutEntry.place(x= root.winfo\_screenwidth() \* (47.5/100), y= root.winfo\_screenheight() \* (82.75/100))

wlOutLabel= Label(root, text="Wavelength (m)", bg="light grey") #label creation

wlOutLabel.place(x= root.winfo\_screenwidth() \* (39/100), y= root.winfo\_screenheight() \* (61.35/100))

ampOutLabel= Label(root, text="Amplitude (m)", bg="light grey") #label creation

ampOutLabel.place(x= root.winfo\_screenwidth() \* (39/100), y= root.winfo\_screenheight() \* (72.05/100))

velOutLabel= Label(root, text="Velocity (m/s)", bg="light grey") #label creation

velOutLabel.place(x= root.winfo\_screenwidth() \* (39/100), y= root.winfo\_screenheight() \* (82.75/100))

freqOutLabel= Label(root, text="Frequency (Hz)", bg="light grey") #label creation

freqOutLabel.place(x= root.winfo\_screenwidth() \* (54/100), y= root.winfo\_screenheight() \* (61.35/100))

energyOutLabel= Label(root, text="Photon Energy (eV)", bg="light grey") #label creation

energyOutLabel.place(x= root.winfo\_screenwidth() \* (54/100), y= root.winfo\_screenheight() \* (72.05/100))

periodOutLabel= Label(root, text="Period (s)", bg="light grey") #label creation

periodOutLabel.place(x= root.winfo\_screenwidth() \* (54/100), y= root.winfo\_screenheight() \* (82.75/100))

#tooltips

tooltips = Balloon(root)

tooltips.bind\_widget(wlSelectRB, balloonmsg="Set wavelength as the dependant variable")

tooltips.bind\_widget(velSelectRB, balloonmsg="Set velocity as the dependant variable")

tooltips.bind\_widget(freqSelectRB, balloonmsg="Set frequency as the dependant variable")

tooltips.bind\_widget(wlMeter, balloonmsg="This sets the unit of measurement for the wavelength to 'metres'")

tooltips.bind\_widget(wlMicro, balloonmsg="This sets the unit of measurement for the wavelength to 'micrometres'")

tooltips.bind\_widget(wlNano, balloonmsg="This sets the unit of measurement for the wavelength to 'nanometres'")

tooltips.bind\_widget(ampMeter, balloonmsg="This sets the unit of measurement for the amplitude to 'metres'")

tooltips.bind\_widget(ampMilli, balloonmsg="This sets the unit of measurement for the amplitude to 'millimetres'")

tooltips.bind\_widget(ampMicro, balloonmsg="This sets the unit of measurement for the amplitude to 'micrometres'")

tooltips.bind\_widget(wlOutEntry, balloonmsg="The wavelength is the displacement between one point in a wave, to its next adjacent point")

tooltips.bind\_widget(ampOutEntry,balloonmsg="Amplitude is the displacement of the two further particles by a wave from a node to an antinode" )

tooltips.bind\_widget(velOutEntry, balloonmsg="The velocity of the wave is the how many metres it can travel in 1 second")

tooltips.bind\_widget(freqOutEntry, balloonmsg="The frequency of a wave is how many times a wavelength passes through a single point in a second")

tooltips.bind\_widget(energyOutEntry, balloonmsg="The photon energy of a wave is the amount of energy carried per photon of the wave in electronvolts")

tooltips.bind\_widget(periodOutEntry, balloonmsg="The time period of a wave is the amount of time it takes to complete a radian cycle (360 degrees when modelled on a 2d plane circle)")

tooltips.bind\_widget(redLightRadioB, balloonmsg="These are the example measurements of a red light wave from the EM spectrum")

tooltips.bind\_widget(blueLightRadioB, balloonmsg="These are the example measurements of a blue light wave from the EM spectrum")

tooltips.bind\_widget(gammaWaveRadioB, balloonmsg="These are the example measurements of a gamma wave from the EM spectrum")

tooltips.bind\_widget(microWaveRadioB, balloonmsg="These are the example measurements of a microwave from the EM spectrum")

tooltips.bind\_widget(customWaveRadioB, balloonmsg="wavelength = 10, velocity = 1, frequency = 0.1, amplitude = 1")

wlEntry.insert(0, 10.0)

velEntry.insert(0, 1.0)

freqEntry.insert(0, 0.0)

ampEntry.insert(0, 1.0)

SelectGrayedInput()

root.mainloop()