Optics Experimentation Environment|1863, Jamilla Kone

NEA A-Level Computer Science

201863 Kone, Jamilla

Table of Contents

[Analysis 3](#_Toc91918743)

[The Problem Definition 3](#_Toc91918744)

[Background Research 4](#_Toc91918745)

[Required Background Knowledge 4](#_Toc91918746)

[Project Research 6](#_Toc91918747)

[Other Teaching Resources 8](#_Toc91918748)

[End Users 9](#_Toc91918749)

[Interview Anaysis (Students) 9](#_Toc91918750)

[Interview Anaysis (Teacher) 10](#_Toc91918751)

[Objectives 11](#_Toc91918752)

[Modelling 13](#_Toc91918753)

[ON-Screen Display 13](#_Toc91918754)

[Login System Modelling 13](#_Toc91918755)

[Prototype Pseudocode 16](#_Toc91918756)

[Prototype COde 19](#_Toc91918757)

[Documented design 21](#_Toc91918758)

[File Diagram 21](#_Toc91918759)

[Key Functions 22](#_Toc91918760)

[Class Diagram 22](#_Toc91918761)

[Simulation Algorithm and Objects 27](#_Toc91918762)

[Networking (Connection, Display and Chat Box) 41](#_Toc91918763)

[GUI 46](#_Toc91918764)

[Database 54](#_Toc91918765)

[Login System 56](#_Toc91918766)

[Saves 61](#_Toc91918767)

[Unique Code 62](#_Toc91918768)

[Report 63](#_Toc91918769)

[Libaries Used 66](#_Toc91918770)

[Technical Solution 68](#_Toc91918771)

[OSD.py 68](#_Toc91918772)

[Account.py 122](#_Toc91918773)

[Client-Student.py 139](#_Toc91918774)

[ChatboxServer.py 154](#_Toc91918775)

[ChatboxClient.py 159](#_Toc91918776)

[Network.py 161](#_Toc91918777)

[Server-Teacher.py 162](#_Toc91918778)

[Report.py 163](#_Toc91918779)

[Unique\_Code.py 166](#_Toc91918780)

[Algorithms 167](#_Toc91918781)

[Testing 170](#_Toc91918782)

[Test Strategy 170](#_Toc91918783)

[Test Plan 171](#_Toc91918784)

[Video 177](#_Toc91918785)

[Evaluation 177](#_Toc91918786)

[Objective Analysis 177](#_Toc91918787)

[Teacher User Feedback 183](#_Toc91918788)

[Student User Feedback 183](#_Toc91918789)

[Analysis of Feedback and improvements 184](#_Toc91918790)

# Analysis

## The Problem Definition

This project’s purpose is to create an environment in which physics students and teachers can join an experimentation environment where they can carry out physics experiments regarding light rays using physics and mathematical equations. It is a solution for online teaching courses that do not have a physical classroom where they carry out such investigations such as how refraction and reflection work, this project acts as a solution to this problem by allowing the teacher to host the environment and allow students to connect to the environment.

The main problem when teaching online is that either the teacher has already completed the experiments and uploads screenshots or they record the experiment live, the problem with this is that it’s hard to comprehend what’s going on as to see the light rays you must be in a dark room and it hard for cameras to clearly show what going on, another problem teachers of online courses may face is they may not have access to a lab or the lab may have limited equipment so you can’t fully explain and explore the properties of light rays. Another problem with teaching online is the lack of engagement how easily distracted students can get by having an environment where they can engage and actively see.

The project's solution to this is a studio in which a teacher, the host, will be able to control and manipulate light rays using different intensity light sources and objects with different refractive indexes, angles at which the light rays enter the objects, and the density change of two objects and the ability to experiment on the properties of light rays. The students should be able to join the host but cannot access the environment (but can generate their environment) and will only be able to see the relay of what the teacher is doing. A chat box will be implemented to allow them to communicate with the host.

Problems Students encounter is they may not fully understand what is going on during the experiment or need more time to experiment with the different concepts this project will allow students to not feel ashamed to spend more time experimenting as they can open their studio and investigate further privately.

Finally, this project will not only benefit online lessons but also an in-person lesson as the environment will act as a system without any external interference disrupting the results of your experiments.

## Background Research

### Required Background Knowledge

Calendar

Description automatically generatedLight is a form of electromagnetic radiation that the human eye can detect, electromagnetic radiation occurs over a wide range of wavelengths and frequencies from radio waves to gamma rays this spectrum is commonly referred to as the electromagnetic spectrum (figure 1) that houses several different electromagnetic waves, in order of longer wavelength to shorter wavelength, including radio waves, microwaves, infrared, visible light, ultraviolet, x-ray and gamma rays. Electromagnetic radiation is described as coupled electric and magnetic fields perpendicular to each other from the direction of energy transfer propagating through space or a material medium as a traveling wave at the speed of light which is approximately 3.0x108m/s.

Figure 1- Electromagnetic Spectrum

The history of optics (branch of physics that studies the behaviour and properties of light) starts as far back as the 5th to 3rd centuries B.C (Ancient Greece Era) where Epicurus argued that Objects produce light rays, which then travel to the eye and argued that the angle of incidence is equal to the angle of reflection now described as the laws of reflection. Other Greek philosophers particularly Ptolemy experimented and tried to derive an equation for the law of refraction, refraction explains how light rays passing from one transparent medium to another different medium will either slow down or speed and after many experiments discovered that the angle of incidence is proportional to the angle of refraction but was not able to derive a complete equation for the law of refraction. During the life of Ibn AL Haytham (between the 10th and 11th century) debunked the ancient Greece scientist argument, regarding how objects light rays, and discovered what is the correct description of how we can see, he described as light rays bouncing from an object to a person's eye which served as a driving force for modern science. 6 centuries later Sir Isaac Newton proclaimed that “light is a mixture of various colors having different refractivity” (the “Light Particle Theory”) rather than “the pure white” proposed by Aristotle during the ancient Greece era, and demonstrated his theory in the famous prism experiment. In the same century, Christiaan Huygens released a paper advocating his theory that light is a wave. He used this theory of light as a wave to explain the light reflection and refraction phenomenon. After a lot of debates countering Newton’s light particle theory, Huygens' theory that light is a wave became the more accepted scientific concept. During the 18th -19th century Thomas Young showed that when light coming from a point light source is shined onto two pinholes, interference fringes can be observed on a screen an appropriate distance away not long after James Clerk Maxwell theoretically predicted the existence of electromagnetic waves, the fact that electromagnetic waves propagate at the same speed as light, and as horizontal waves which became the foundation for modern electromagnetism.

Light rays are the basic element in geometrical optics a hypothetical construct used to indicate the propagation of light, this concept is useful when explaining the properties of light such as reflection which is defined as light rays changing direction when they reflect off a surface and obeys the laws of reflection states that, on reflection from a smooth surface, the angle of the reflected ray is equal to the angle of the incident ray. Another property of light is refraction which is defined as when light Diagram

Description automatically generatedtraveling in one transparent medium encounters a boundary with a second transparent medium (e.g., air and glass), a portion of the light is reflected (partial reflection), and a portion is transmitted into the second medium (refracted). As the light travels to the second medium, it changes direction due to the wave speeding up or down as the wavelength either decreases or increases (figure 2) we can determine that the wave has slowed down or speed up using the equation wave speed = frequency × wavelength and taking frequency as being constant. The relationship between the angle of incidence and refraction is described using Snell’s law (law of refraction) measured with respect to the normal to the boundary is defined by the equation n1 sin θ1 = n2 sin θ2 in which n1 and n2 are the refractive indexes of the mediums, the refractive index(n) for any medium is equal to the ratio of the speed of light in a vacuum to its speed in that medium, denser mediums have a higher n so there angle of refraction is smaller so it bends towards the normal vice versa is true for mediums that are less dense.

Figure 2- Wave front Image of Refraction

Diagram

Description automatically generated with low confidenceAn interesting consequence of the Snell’s law occurs when light passes into a medium with a lower refractive as stated above the light rays will bend away from the normal. At what is called a critical angle of incidence the refracted rays make a 90 degrees angle with the normal the critical angle can be derived from Snell’s law equation as sin90=1 as the angle therefore n2 sin θ=n1\*1 so the critical angle can be found using the equation sin θ = n2/n1. When the incident angle is larger than the critical angle the rays are completely reflected inside the medium this phenomenon is called total internal reflection (figure 3) this phenomenon is commonly taken advantage with the use of fiber optic cables as it allows for the transmission of light over a long density with little loss of intensity.

Figure 3- Total Internal Refraction Occurring

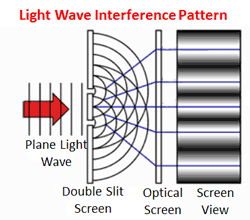
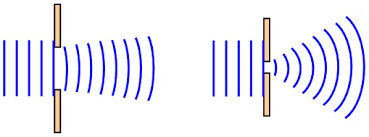
As light is defined as a wave a defining characteristic of all waves is superposition, which describes the behavior of overlapping waves. The superposition principle states that when two or more waves overlap in space, the resultant disturbance is equal to the algebraic sum of the individual disturbances. In constructive interference the crests of two waves coincide, and the waves are said to be in phase with each other. Their superposition results in a reinforcement of the disturbance; the amplitude of the resulting combined wave is the sum of the individual amplitudes. Conversely, in destructive interference the crest of one wave coincides with the valley of a second wave, and they are said to be out of phase. The amplitude of the combined wave equals the difference between the amplitudes of the individual waves. In the special case where those individual amplitudes are equal, the destructive interference is complete, and the net disturbance to the medium is zero. Another crucial property of light is diffraction which is defined as the slight bending of light as it passes around the edge of an object. The amount of bending depends on the relative size of the wavelength of light to the size of the opening. (figure 4) Combining the properties of diffraction and interference was born Young’s Double Slit Experiment which explained that when monochromatic light passing through two narrow slits (diffraction occurs) illuminates a distant screen, a characteristic pattern of bright and dark fringes is observed. This interference pattern is caused by the superposition of overlapping light waves originating from the two slits. (figure 5)

Figure 5- Young's Double Slit Experiment

Figure 4- Diffraction

### Project Research

All these properties and characteristics of light is essential for the project as to predict the behavior of light we must understand the laws, properties, and characteristics of light to be able to carry out experiments. A crucial part of the Project is to be able to tailor to the needs of primarily GCSE and A-Level students and teachers so the project must be able to carry out experiments that are on both curriculums.

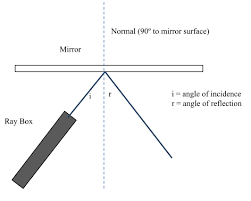
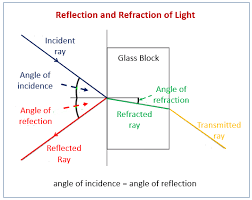
On the GCSE curriculum there are two required practical to investigate the properties of light the first practical is to investigate the reflection of light by a plane mirror. In the classroom the experiment is set up using a ray box (used to project a narrow ray of light) a mirror and paper, usually the mirror and ray box are set on white paper then a normal line perpendicular to the mirror is drawn indicating the entry point for the incident ray, the ray box is then moved to an angle and projects the incident ray at the normal the incident ray is then traced using a pencil and the reflected ray is also traced then from the normal the angle of incidence and angle of reflection are measured. The practical is then repeated at different angles, where a pattern should emerge confirming the law of reflection the angle of incidence and the angle of reflection are equal (figure 7). The possible way the practical could implement this practical is by having an object that acts as mirror (has the same reflective properties) that can be loaded into the environment using object-oriented programing (OOP) and could have an indicator to indicate the angle of incidence and angle of reflection in the form of text. Another practical that is housed within the GCSE curriculum is the investigation of refraction of light through a glass block in this experiment a ray box and a glass block and paper is used. A normal line is drawn perpendicular to the surface of the glass block and the ray box is positioned at an angle and the incident ray is pointed at the intersection of the glass block and the normal line, the incidence ray is then traced and the refracted ray is then also traced then the block is removed and the point of intersection with the normal and the incidence ray is connected with the refracted ray the joined lines should represent the path of the light ray through the glass block (figure 7). The practical is then repeated (note that on some syllabuses the material of the block changes), the main evaluation of the experiment at a GCSE level is that the light rays are not obeying the law of reflection and that at different material densities the angle of refraction will either increase or decrease. A way to incorporate this practical into my project is by having different objects with varying refractive indexes using object-oriented programming and having an indicator to indicate the angle of incidence and the angle of refraction in the form of text.

Figure 7- Refraction Experiment (GCSE)

Figure 2- Set Up of Reflection Practical (GCSE)

On the A-Level curriculum there is a lot more experiments to investigate the properties of light. Young’s Double Slit Experiment is one of the practical within the A-Level syllabus as explained above light is passed through two narrow slits illuminating a distant screen, a characteristic pattern of bright and dark fringes is observed. To implement this experiment within the project there should be an object (slit) that allows diffraction to occur using OOP. The investigating refraction and total internal reflection of light using ray boxes including transparent rectangular and semi-circular blocks, is another practical done at a level physics typically done using ray box, semi-circular glass block, rectangular glass block, paper with a protractor printed on it. The semi-circle glass block is experimented on by placing the semi-circular block on top of the protractor with the centre aligned with 0° on the protractor then at a point, nearly 180 degrees to the block, the ray box and trace the ray entering and leaving the block, repeat process at intervals incrementing in 10 degrees. Record the incidence angle at which the refraction angle is 90°, (the ray leaves along the straight boundary of the block) to find the critical angle. Increasing the incidence angle total internal refraction will occur (figure 4). The implementation of the practical in the project by programming the semi-circular glass block as an object and indicating the incidence angle and the refracted angle. For the rectangle block, place on white paper and mark a point on the longer edge approximately 1-2 cm from the corner. Aim the beam normal to the block and trace the entry and exit ray, the straight part of the protractor should be against the long side of the block incrementing by 10 degrees measuring the incidence angle and refraction angle for each position. To carry out this practical in my project transparent rectangle objects can be added using object-oriented programming.

A similar program that are available all are able to simulate various light sources: ray, beam, and point source, simulate reflection in plane or arc mirror, simulate refraction in plane or arc interfaces, including both refracted and reflected rays, simulate ideal lens/mirror, which obeys lens/mirror equation, view extensions of rays to see if they converge to a virtual image, view real images, virtual images and virtual objects directly view images that can be observed from some given position, distance and angular measurement. Although this all the programs are not tailored around education system but rather more for personal investigating or does not have any networking capabilities, the project I plan to develop will have some of the main features also found in the programs such as the simulation of refraction and reflection with various different objects. However, I plan to make my program networked so that students can connect to their teacher’s environment, so it is more suitable for online lessons and I also plan on adding a login system to allow teachers to save their progress, in the programs researched there seems to be no way to carry to diffraction which I plan on implementing to my program. The problem with implementing diffraction is that is difficult represent as a straight line as when the slit is larger than the wavelength, as the wave front will curve at the ends therefore instead of not including diffraction I plan on adding young’s double slit experiment but the slits are preprogramed to be approximately equal to the rays wavelength then by using the equation wavelength (m) = (slit separation (m) x fringe separation (m))⁄distance to screen (m) in order to find the number of fringes the calculation divide the screen length by fringe separation must be carried out, once the number of fringes are calculated (if there is decimals the whole number is truncated and the decimals left are multiplied by the screen and the first maxima is plotted from that distance from edge of the screen), then the next order of maxima is measured by adding the fringe separation and is continued until off screen, then to trace the rays from the first slit trace a ray to maximas then trace second slit to the same maxima.

### Other Teaching Resources

Kahoot is a common teaching aid for remote learning as students answer questions in a set amount of time competing with their classmates. The competitive element of Kahoot helps to push learners’ skills and memory.

Good Features-

* It allows teachers to create their own questions allowing them to tailor the questions to their particular specification, so all questions are actually useful to the students learning.
* Kahoot also creates a report after each session indicating how many students played, how many questions there were, which questions were the most difficult and might need reteaching, which players need help based on their game results, the correct overall percentage of the question and how long it took for students to answer. All the elements found in Kahoot’s report help the teacher recognise the weaknesses in the class as a whole and on an individual level.

Lacking features-

* Doesn’t have much visual aid for learners, which account for approximately 65 percent of the population
* The report doesn’t highlight does who may need more of a challenge, as if an individual in the class finds the question easy they may need to be challenged in order for them to reach their full potential

IXL is another remote learning resource that allows the teacher to set questions specifically in maths and English.

Good Features-

* It provides real time diagnosis of each student’s overall knowledge, and based on these it generates a targeted action plan to show how to help them grow.
* It also provides recommendation which helps students to learn independently.
* IXL also provides report, indicating who is struggling and allowing you to review individuals progress

Lacking features-

* Doesn’t allow you to set questions so you are restricted to the questions created by the program which become repetitive

Each remote teaching resource has its benefits and disadvantages, after reviewing the features and finding out which ones work the best, I want to employ such as allowing teachers to asked their own questions and also the competitive aspect of Kahoot by adding a timer and indicating how many people have answered the question and as an additional motivator display the top 5 students. The report/analytic features seem to be the most useful so I will be indicating each student correct answer and how long it taken which will be used to identify who is struggling and who is excelling relative to the whole class’ performance. To indicate the progress of the whole class a feature of asking the same questions before and after a session then producing a graph to indicate whether the session has been productive, in the report section.

## End Users

End users for this project will be aimed at students in their GCSE and A-Level year’s corresponding to ages 14-16 and 16-18 respectively, as the project covers concepts that are taught within these years of education. This indicates that users will have an easier time navigating through the program, this age bracket also indicates to me that the GUI does not have to be colorful which is usually used in children younger top attract their attention and keep them engaged. Rather the GUI and layout can be more professional and have more technically correct definitions. To get feedback and suggestion for these end users I created a google form documents housing the following questions:

1. What do you think could improve in the way you carry out practical at school or college?
2. What kind of features for students to improve their experience?
3. How ideally would you like to be able to access the program?
4. Other than the practical on the curriculum what kind of experiments would you like to see able to investigate?

And posted them across public student forums across the internet such as Reddit and The Student Room as they had the biggest student density within their subgroups.

I choose these questions specifically so that end users can express what they feel is lacking with how they do practical in traditional settings or in online settings. The top responses for question were that they wanted “access to relevant data and information” and “more interactivity environment”. For question two students felt that to improve their experience they needed features such as” be able to perform multiple experiments at once” and “visible changes to see what is going on in the practical”. For question three the ideally wanted access to be “Through website or service” or “an app or downloadable software”. For question four most people wanted to be able to carry out experiments that they could not carry out within the classroom.

### Interview Anaysis (Students)

There are some very useful suggestion and insights that was presented in the responses to these questions such as that they wanted a more engaging experience, a way the program could satisfy this need is by recognising the properties that is being investigated and display the theory to that property and links to find out more about the property, another way as mentioned above in the problem definition is by implementing a chat box to allow students to engage with each other and the teacher/host. A suggestion for features of the program is the ability to carry out multiple experiments at once in order for the program to adopt this the program must account for the interaction between light rays (superposition) and how that will display. For access there seems to be a wide variety of suggestion informing me that there is not really a consensus to what form the program is presented in. pupils seemed to want the ability to investigate independently confirming the need for an offline environment which students can access.

Another extremely important end user is a teacher, to understand the needs of the teachers pertaining to the program I asked a physics teacher what features they would like to see from an educational perspective;

*What would you improve in the current way you carry out practical?*

Your program would be really useful when I teach online I find that when I teach online students tend to find understanding practical quite difficult.

*What kind of features would you like to see in the program?*

As a teacher I would like to see the ability to ask the students question based on what is going on in the program, ideally the student will have around 3 tries per question and I also would like to see how long it takes to answer it to make the program more educational. It will be nice to see all the result for each student all in one place like in some kind of file.

*Would excel be fine.*

Excel would be perfect.

### Interview Anaysis (Teacher)

The interview was very useful in learning about what to implement in my program in the perspective of teachers rather than students, it seems that the teacher wanted to make the program more educational by adding an algorithm that takes how many tries and how long it took to answer the question and append it to a file that the teacher can view it.

## Objectives

1. The program must be able to carry out every practical in the GCSE and A-Level curriculum for Physics regarding the investigation of light
2. As the main end users are targeted at GCSE and A-Level students and teachers it is essential that they can use the program for every practical. Experiments include: The investigation of Refraction and Reflection, Young’s Double Slit Experiment, investigating refraction and total internal reflection of light using ray boxes including transparent rectangular and semi-circular blocks.
3. The program must contain all the objects needed to carry out the experiments which include: a light source, mirror, glass block, slit, semicircular glass block, screen, and stock blocks (a block that’s properties can be changed to investigate the behavior of light with material of different properties) using object-oriented programming (classes, overriding, inheritance and composition).
4. Program must be able to simulate the propagation of light and if interaction with other objects recursively
5. Host must have the ability to change the properties of objects
6. Host has the ability to select which properties/elements of the experiment they want to be displayed
7. The program must be networked to allow it to be used remotely
8. The program must be able to allow pupils to join the environment.
9. When a pupil exits the environment, the program must be able to disconnect the pupil from the server.
10. Pupils must be able to connect to an environment with a name using a unique code generated based on the time and the class id of the class that the session will be ran on.
11. Pupils must also be able to communicate with each other and the host using a chat box where the name of the chatter(nickname) is also displayed beside text. When the chat box is opened a new window opens and connected to another server hosted by the host specifically for chat and answering questions.
12. When pupils first open the chatbox window the program should ask them for a nickname they will like to be addressed by and their student id so that when they answer questions the program will be able to identify the person who answered it.
13. When pupils connect to the environment, they must not have the ability to move or change anything within the environment.
14. The program must allow the Host to save and reload environments they have saved within the session
15. A login system to access their account and details on Account, Students, Session, Files, Class and Code should be stored on a relational database
16. Have an Interface where host can add classes, remove classes, view classes, edit their own account and view files from previous sessions.
17. Passwords for each account must be encrypted using a hashing algortihm.
18. The program must be set up as a Client-Server model using TCP.
19. Unique code is sent via email to every student from the class which the session is being ran from
20. Objects must be rendered in 2D environment by using a 2D environment the host will be able to move the objects around on the X and Y axis using a mouse, using pygame.
21. Program able to add and delete objects when specific button clicked
22. Program can be used offline; this function serves to benefit students who want to investigate by themselves.
23. This mode can be used without an account or with an account (no report will be created and program will not connect to be connected to a server).
24. Program must allow teachers to set question depending on the experiment that is being carried out in the environment, the program must automatically suggest question that the teacher can set depending on the elements that are being displayed, elements referring to incident angle, refracted angle, refractive index, critical angle, and elements of the young’s double slit experiment (slit separation, fringe separation, screen distance andsource wavelength).
25. Student must only have a set amount of time to answer the question, the amount of time is set by the host (teacher) as they select the appropriate amount of seconds for each question
26. Host allowed to set custom question where they can set the question and answers.
27. Questions must be answered in chat box but not displayed to other clients using a command.
28. After each session the program must produce a report of the class’ progress
29. The program must be able to report if the answer is correct or incorrect for every student and formatted into an excel file using csv and should be downloadable for teachers.
30. File must identify student who may need extra help and those who needs to be given a challenge. By calculating checking whether there average correct percentage is greater than 95% or lower than 60%
31. File must include a graph showing the classes overall performance per question
32. File will be sent to the host email after each session

Extension Objective:

1. Pupils can connect to the environment over the internet. As this will require more time to do if program can successfully meet all objectives, then extension can be tried to implement.
2. The objects are rendered in a 3D environment, and can be viewed from different angle and along the X, Y and Z axis

## Modelling

### ON-Screen Display

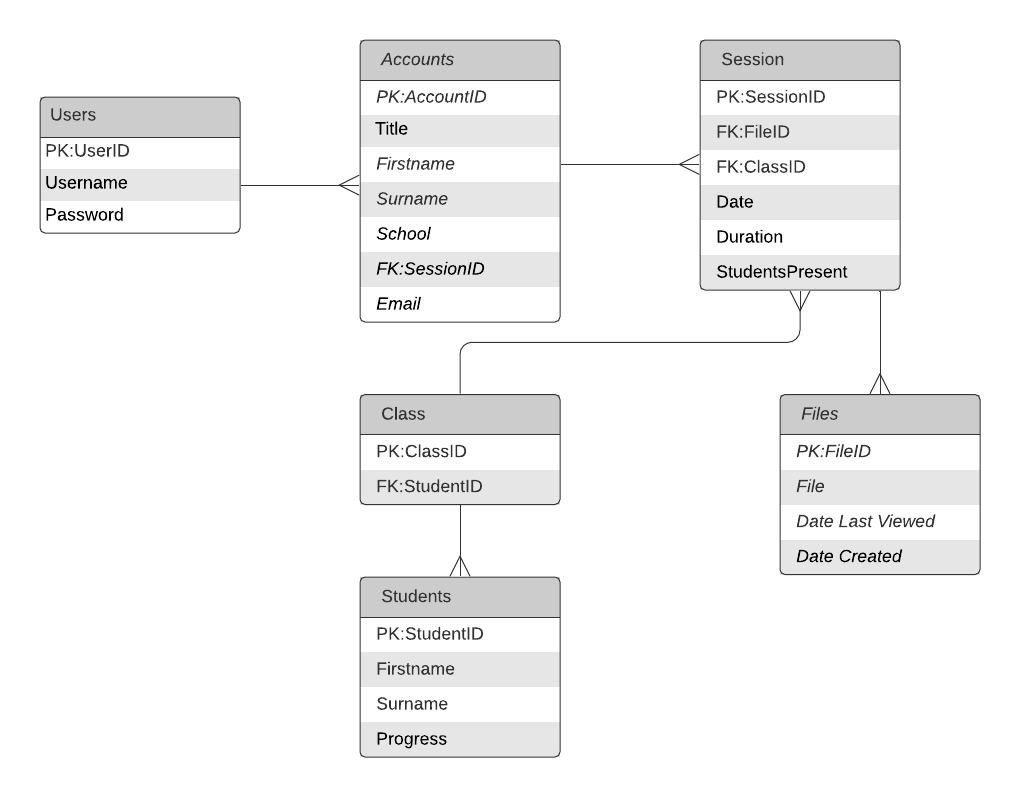
Where experiments will take place

Toolbar

Buttons

### Login System Modelling

As my program will have a login system I have modelled an entity relationship diagram to show what will be stored in the database based on what has been requested by the end users and highlighted within the objectives.



As part of the login system on the student end they must enter their student ID and a unique code they use to be able to enter the environment, using python I have modelled what the algorithm for producing this code will look like, the algorithm will produce a 7 character code based on the time and date and a key.

1 **import** datetime

2 now**=**datetime**.**datetime**.**now**()**

3 now**=**now**.**strftime**(**"%d %H:%M:%S"**)**

4 now**=**now**.**replace**(**"-"**,**""**)**

5 now**=**now**.**replace**(**":"**,**""**)**

6 now**=**now**.**replace**(**" "**,**""**)**

7 key**=str(ord(**"K"**))+str(ord(**"O"**))+str(ord(**"N"**))+str(ord(**"E"**))**

8 group**=**0

9 code**=**""

10 **while** group**<=**6**:**

11 part**=(int(**now**[:**2**])** **+** **int(**key**[:**2**]))**

12 **if** 65**<=**part**<=**90 **or** 97**<=**part**<=**122**:**

13 part**=**part

14 **elif** part**>**122**:**

15 part**=**122**-(**part**-**122**)**

16 **elif** 57**<**part**<**65**:**

17 part**=**part**+**48

18 **elif** 90**<**part**<**97**:**

19 part**=**part**+**8

20 part**=chr(**part**)**

21 code**=**code**+**part

22 now**=**now**[**2**:]**

23 key**=**key**[**2**:]**

24 group**+=**2

25 **print(**code**)**

26

My optics simulation algorithm will consist of propagating rays from a light source find the equation of the straight line which represents the light ray then finding the equation of the object then find the coordinates of where the ray will initially (first interception from light source) intersect the object. If the ray does intercept with the object it will then detect whether the object is a mirror or a block (block refers to a transparent object), if the object is a mirror it will reflect off the surface at the same angle of the incident ray from the normal (which can be calculated using the equation tanθ= | (m2-m1)/(1+m2m1) | where m is equal to the gradients of the normal and ray equation) equation of the reflective ray is y=-m(x-x1)+y1 where (x1,y1) is the coordinates of the interception point, restrict the domain to be greater than or equal to x1 .

For transparent blocks normal and incident angles are calculated the same way however Snell’s law is used to find the change of angle (direction) within the object once angle calculated, the equation tanθ= | (m2-m1)/(1+m2m1) | can be used to find the gradient of the refracted ray equation then using the point of interception to find the equation of the refracted ray. Then the equation of block and equation refracted rays are put equal, and a second point of interception is found then the steps are repeated. If angle of refraction is greater than 90 degrees than total internal refraction occurs than the reflection part of the algorithm is used, if reflected light ray is then treated as the first refracted ray that entered the block.

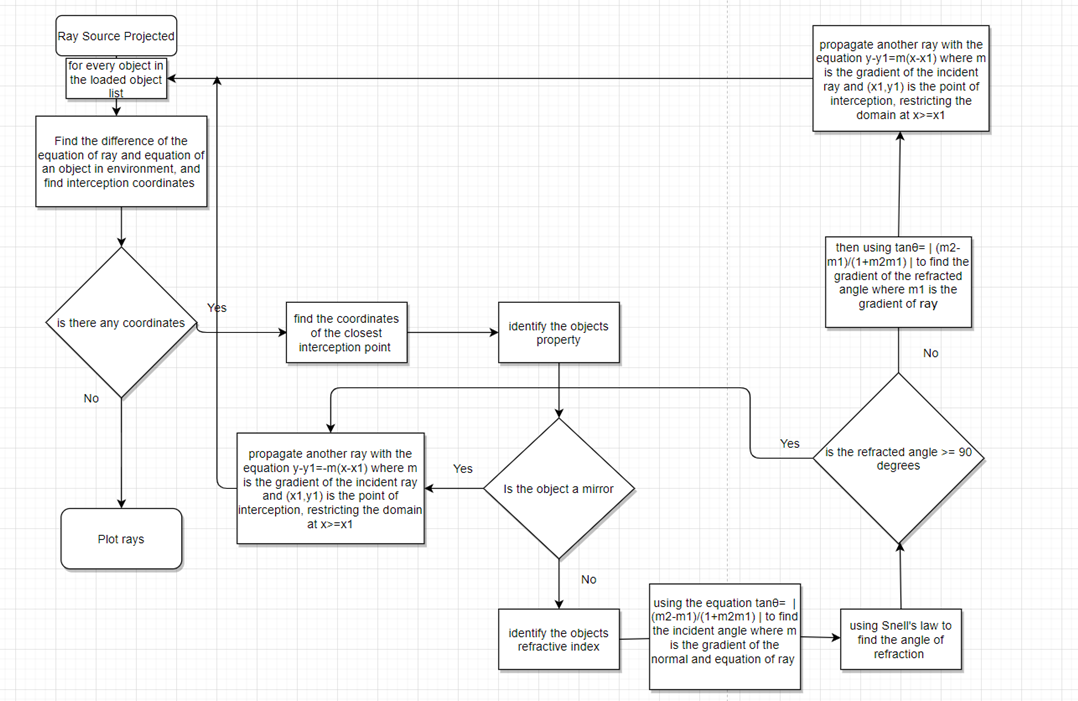


Figure 8 Flowchart of Simulation Algorithm

### Prototype Pseudocode

*Function Simulation (source, loaded\_objects)*

***Check if the light ray has hit an object***

*intercept\_x=[]*

*For every item in loaded\_objects then*

*source.equation[0]-(item.eqaution)=intercept\_eq*

*intercept\_x. append(solve(intercept\_eq))*

*coords=[]*

***If so checks for the nearest collision***

*For every value in intercept\_x then*

*If not intercept\_x then*

*source.plot()*

*return*

*Else then*

*For every element in value then*

*str(source.equation).replace(“x”, str(element))=intercept\_y*

*coords.append(value, int(intercept\_y))*

*distance=[]*

*For every point in coords then*

*d=sqrt((source.x-point[0])^2+(source.y-point[1])^2)*

*distance.append((d,coords)*

*For every length in distance then*

*If length[0]>distance[0][0] then*

*distance.insert(0,length)*

*else then*

*next*

*for every obj in loaded\_objects then*

*if str(obj.eqaution).replace(“x”,str(distance[0][1][0])= distance[0][1][1] then*

*obj=item*

*if source.x>distance[0][1][1] then*

*source.plot**(x>=distance[0][1][1])*

*else then*

*source.plot(x<=distance[0][1][1])*

***Identifies the nearest objects property the simulates the propagation of the light ray accordingly, then repeats process for the reflected or refracted ray***

*If item.properties = mirror then*

*Reflected\_eqaution=-source.gradient(source.x-distance[0][1][0])+distance[0][1][1]*

*reflected\_ray=object(ray,reflected\_eqaution,x>=distance[0][1][0],distance[0][1])*

*Simulation(reflected\_ray, loaded\_objects)*

*else if item.properties = block then*

*item.reflectiveindex=n2*

*m1=source.gradient*

*m2=item.gradient*

*inverse tan(MOD(m2-m1)/(1+m2m1) ) =θ1*

*sin inverse(sin θ1/n2)= θ2*

*if θ2>=90 then*

*reflected\_eqaution=-source.gradient(x-distance[0][1][0])+distance[0][1][1]*

*reflected\_ray=object(ray,reflected\_eqaution,x>=distance[0][1][0],distance[0][1])*

*reflected\_ray.plot*

*Simulation(reflected\_ray, loaded\_objects)*

*else then*

*(-m1-tan θ2)/(m1tan θ2-1))=m3*

*refracted\_eqaution=m3(x-distance[0][1][0])+distance[0][1][1]*

*refracted\_ray=object(ray,refracted\_eqaution,x>=distance[0][1][0],distance[0][1])*

*Simulation(refracted\_ray, loaded\_objects)*

*else then*

*next*

### Prototype COde

1 **def** Simulation**(**self**,**objects\_loaded**):**

2 intercept\_x **=** **[]**

3 **for** i **in** objects\_loaded**:**

4 intercept\_x**.**append**((**i**[**0**],** **(**sympy**.**solve**(eval(**"{0}-{1}"**.format(**self**.**Eqaution**,** i**[**1**].**Eqaution**))))))**

5 coords **=** **[]**

6 **for** n **in** intercept\_x**:**

7 **if** **not** intercept\_x**:**

8 self**.**Plot**()**

9 **return**

10 **else:**

11 **for** m **in** n**:**

12 intercept\_y **=** **eval(str(**n**[**1**]).**replace**(**"x"**,** **str(**m**)))**

13 coords**.**append**((**n**[**0**],** n**[**1**],** **int(**intercept\_y**)))**

14 distance **=** **[]**

15 **for** p **in** coords**:**

16 d **=** math**.**sqrt**((**self**.**xp **-** p**[**1**])** **\*\*** 2 **+** **(**self**.**yp **-** p**[**2**])** **\*\*** 2**)**

17 distance**.**append**((**d**,** p**))**

18 **for** l **in** distance**:**

19 **if** **-**l**[**0**]** **>** **-**distance**[**0**][**0**]:**

20 distance**.**insert**(**0**,** l**)**

21 **else:**

22 **continue**

23 near\_obj **=** **[**item **for** item **in** objects\_loaded **if** item**[**0**]** **==** distance**[**0**][**1**][**0**]]**

24 **if** near\_obj**[**1**].**Properties **==** "Mirror"**:**

25 self**.**Restrict**(**coords**[**0**])**

26 self**.**Plot**()**

27 reflect\_eq **=** **eval(**"-{0}(x-{1})+{2}"**.format(**self**.**Gradient**,** distance**[**0**][**1**][**1**],** distance**[**0**][**1**][**2**]))**

28 reflect\_ray **=** Source**()**

29 reflect\_ray**.**Chg\_Eqaution**(**reflect\_eq**)**

30 reflect\_ray**.**Chg\_Gradient**(**self**.**Gradient**)**

31 **else:**

32 n2 **=** near\_obj**[**1**].**RefractiveIndex

33 m1 **=** self**.**Gradient

34 **if** near\_obj**[**1**].**Gradient **==** 0**:**

35 m2 **=** near\_obj**[**1**].**Gradient

36 **else:**

37 m2 **=** **str(**self**.**Gradient**).**replace**(**"x"**,** distance**[**0**][**1**][**1**])**

38 m2 **=** **int(str(**self**.**Gradient**).**replace**(**"y"**,** distance**[**0**][**1**][**2**]))**

39 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m2 **\*** m1**)))**

40 theta2 **=** asin**(**sin**(**theta1**)** **/** n2**)**

41 **if** theta2 **>=** 90 **and** self**.**Refractive\_Index **>** n2**:**

42 self**.**Restrict**(**coords**[**0**])**

43 self**.**Plot**()**

44 reflect\_eq **=** **eval(**"-{0}(x-{1})+{2}"**.format(**self**.**Gradient**,** distance**[**0**][**1**][**1**],** distance**[**0**][**1**][**2**]))**

45 reflect\_ray **=** Source**()**

46 reflect\_ray**.**Chg\_Eqaution**(**reflect\_eq**)**

47 reflect\_ray**.**Chg\_Gradient**(**self**.**Gradient**)**

48 **elif** theta2 **<** 90 **and** theta2 **!=** 0**:**

49 self**.**Restrict**(**coords**[**0**])**

50 self**.**Plot**()**

51 m3 **=** **(-**m1 **-** tan**(**theta2**)** **/** m1 **\*** tan**(**theta2**))**

52 refracted\_eq **=** **eval(**"-{0}(x-{1})+{2}"**.format(**m3**,** distance**[**0**][**1**][**1**],** distance**[**0**][**1**][**2**]))**

53 refracted\_obj **=** Source**()**

54 refracted\_obj**.**Chg\_Eqaution**(**refracted\_eq**)**

55 refracted\_obj**.**Chg\_Gradient**(**m3**)**

56 refracted\_obj**.**Chg\_Coords**(**coords**[**0**])**

57 refracted\_obj**.**Simulation**()**

58 **else:**

59 self**.**Plot**()**

60

# Documented design

In this documented design section, I will be explaining the classes, functions and libraries I will be using to create my program which will consist of the following files:

* Account.py (The first file that is run when program started. Is where accounts are accessed and can be changed and where simulation can be accessed. Houses most of the SQL queries)
* OSD.py (Where the simulation algorithm and display will be)
* ChatboxClient.py (File used to connect user to chat box)
* ChatoxServer.py (File used to initiate and manage chat box server)
* Client-Student.py (Houses the students display)
* network.py (Where teacher and client connect to server)
* Server-Teacher.py (Initiates server and handles clients)
* Unique\_code.py (generates a unique code used to access the environment from the student side)
* Accounts.db(relational database that houses information about classes, students, teachers, progress, session detail and code used to connect)

## File Diagram

Account.py

Server-Teacher.py

Unique\_code.py

Accounts.db

Client-Student.py

OSD.py

ChatboxClient.py

network.py

ChatboxServer.py

## Key FUnctions

### Class Diagram

A picture containing qr code

Description automatically generatedOSD.py

Account.py

Graphical user interface

Description automatically generated with low confidence

ChatboxServer.py

Text

Description automatically generated with medium confidence

ChatboxClient.py

Text

Description automatically generated with medium confidence

Network.py

Text

Description automatically generated

A picture containing timeline

Description automatically generatedClient-Student.py

### Simulation Algorithm and Objects

The simulation algorithm will simulate the behaviour and properties of light in a closed system, using the pseudocode in the modelling section of the analysis. The use of classes and objects is key when simulating light rays as it allows the ability for multiple object instances for the same object. All object classes will inherit from a super class that has attributes such as refractive index and position and methods including movement, plot, resize. When a new source is instantiated, it will run through the simulation algorithm before being displayed, when new object is added it will run through the algorithm again with every source with the new object in the loaded objects list, to see if there are any new interceptions.

As diffraction is a difficult concept to program, the only way the program will be able to carry out any diffraction is the through the young’s double slit experiment, so to carry out the experiment there is a class (inheriting the object super class) that will instantiate an object with a slit and a screen where there is the ability to configure the slit/screen separation, slit separation and fringe separation, the class will also have its own method for simulating light overriding the super class’. If the source hits a slit it will be absorbed/line of incidence will stop unless it hits the gap in the slit from there the simulating light method will be initiated.

Pseudocode for Interception of Objects

*DEF findintercept(state, previous):*

*[] ←coord #will hold all the coordinates of where the source intercepts another object*

*[] ←intercept\_x #holds the intercepts made on the top and bottom of and object*

*[] ←intercept\_y #holds the intercepts made on the side of an object*

*IF state: #checks if the source is the original source or the product of another*

*None ←previous.re #resets the attribute that identifies whether the source has ajdfbfkfjdsafjkdnsfjdsgnjsjgksjgfnjgjnfg been internally refracted*

*FOR every object IN objects\_loaded:*

*IF “SC” IN defined\_name:*

*IF gradient == 0:*

*CONTINUE*

*ELSE:*

*sqrt((object.x - source.x) \*\* 2 + (object.y – source.y) \*\* 2) ←dist*

*if dist>=object.radius:*

*object.eqaution REPLACE "y" WITH source.eqaution←ep*

*solve(ep,x) ←solutions*

*[]← goodsol*

*FOR every solution IN solutions:*

*TRY:*

*IF solution <= source.x:*

*CONTINUE*

*ELSE:*

*goodsol.insert(solution)*

*CATCH TypeError:*

*CONTINUE*

*IF NOT goodsol:*

*source.draw*

*ELSE:*

*FOR every point IN goodsol:*

*source.equation REPLACE “x” WITH point ←ycoord*

*IF ycoord <= object.y:*

*coord.insert([object,point,ycoord])*

*ELSE:*

*PASS*

*source.eqaution REPLACE “y” WiTH object.y ←xcoord*

*IF object.x - object.radius <= xcoord <= object.x + object.radius:*

*coord.insert([object, xcoord, object.y])*

*ELSE: #evaluating the intercept of rectangular shaped objects*

*IF gradient==0:*

*PASS*

*ELSE:*

*source.eqaution REPLACE “y” WiTH object.y ←xcoord*

*intercept\_x.insert ([xcoord, object.y])*

*source.eqaution REPLACE “y” WiTH object.y+object.length ←xcoord*

*intercept\_x.insert ([xcoord, object.y+object.length])*

*source.eqaution REPLACE “x” WiTH object.x ←ycoord*

*intercept\_x.insert ([object.x,ycoord])*

*source.eqaution REPLACE “x” WiTH object.x+object.width ←ycoord*

*intercept\_x.insert ([object.x+object.width,ycoord])*

*FOR pair IN intercept\_x:*

*IF NOT state:*

*IF object.x <= pair[0] <= object.x + object.width:*

*coord.insert ([object, pair[0], pair[1], "top"])*

*ELSE:*

*PASS*

*ELSE:*

*IF object.x <= pair[0] <= object.x + object.width AND pair[0] != source.x:*

*coord.insert ([object, pair[0], pair[1], "top"])*

*ELSE:*

*PASS*

*ENDLOOP*

*IF NOT coord:*

*source.statechange()*

*ELSE:*

*IF previous IS NOT NONE:*

*IF NOT previous.previousobj:*

*FOR set IN coord:*

*IF set[0] IN previous.previousobj:*

*coord.remove(set)*

*FOR set IN coord:*

*sqrt(((set[1] - source.x)^2) + ((set[2] - source.y)^2)) ←distance*

*coord.index(set)* *← element\_no*

*coord[element].insert(distance)*

*` SORT(coord)*

*IF “SC” in str(coord[0]):*

*IF coord[0][2] != coord[0][0].y and coord[1][2] != coord[0][0].y:*

*source.statechange()*

*ELIF coord[0][2] < coord[0][0].y:*

*[coord[1]] ←coord*

*ELSE:*

*[coord[0]] ←coord*

*ELSE:*

*[coord[0]] ←coord*

Main Object Class-

The main object class purpose is to be the parent class to all the objects that will be loaded in the environment every object will inherit this main object class.

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.defined\_name | Holds the name of the object |
| self.Refractive\_Index | Holds the refractive index of an object which can be changed |
| self.xp | Holds the value of the initial x coordinate placement of an object |
| self.yp | Holds the value of the initial y coordinate placement of an object |
| self.length | Holds value of initial length of rectangular shaped objects (shape of most objects) |
| self.width | Holds value of initial width of rectangular shaped objects (shape of most objects) |
| self.Property | Holds the name of the property of objects, identifies in which way the ray will propagate when it hits the object, for example refract, reflect, stop the ray or diffraction |
| self.graphical | Holds the pygame draw function for an object |
| self.draging | Indicates if an object is being dragged by the mouse, used when moving object around |
| self.intercepters | Holds the sources created by the propagation of the ray |
| self.full\_name | elongates the self.defined\_name of the object used by the show function to specify to the user what each object is |

|  |  |  |
| --- | --- | --- |
| Method | Purpose | Return |
| self.update() | Detects if the show function has enabled the viewing of the refractive index and views it using pygame text function | None |
| self.redraw2() | fills the screen black then redraws every object on the screen, usually used when moving an object around | None |
| self.Simulation(state,previous,obj) | initiates the simulation method for every source object loaded to indicate whether it is in the path of one (how the simulation method works will be explained further below) | self.endx, self.endy |
| self.redraw1() | redraws only the object itself | self.graphical |

Source Class-

Source class is the class in which the rays are instantiated and manipulated. Inherits from the main object class. As the source class inherits from the main object it has some of the same attributes but overrides some of their values, so I will explain the unique attributes of this class

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.wavelength | indicates the wavelength of the source mainly used in the diffraction mode |
| self.graphical2 | this line is attached to the main line created by self.graphical when dragged its used to change the angle of the source object |
| self.endy | indicates the end point of the y coordinate of the source line created using self.graphical |
| self.endx | indicates the end point of the x coordinate of the source line created using self.graphical |
| self.redy | indicates the start point of the y coordinate of the source line created using self.graphical2 |
| self.redx | indicates the start point of the x coordinate of the source line created using self.graphical2 |
| self.rendx | indicates the end point of the x coordinate of the source line created using self.graphical2 |
| self.rendy | indicates the end point of the y coordinate of the source line created using self.graphical2 |
| self.colour | indicates the colour of main source line |
| self.colour2 | indicates colour of self.graphical2 |
| self.thickness | determines the thickness of the line used to create self.graphical and self.graphical2 |
| self.m | holds the value of the gradient of the source object |
| self.findx | using the equation y-y1=m(x-x1), the find x object rearranges the equation to make x1 the subject so that during the simulation stage it can replace the values of y1 and m to find the corresponding x to see it there is an intersection point |
| self.findy | same as self.findx but makes y1 the subject |
| self.state | indicates whether the source is the product of a ray that has interacted with another object so that if the main source object moves it is deleted |
| self.stateproduct | is a list used by the main source object that holds the objects created from interacting with another object |
| self.previousobj | indicates previous objects it been through |
| IncidentRays[self.defined\_name] | adds to dictionary the an empty list with the key being the name of the object, used for transferring objects over a network |
| self.t\_status | indicates the tilting status of the object if self.graphical2 clicked and dragged it will be one until unclicked then it will return to zero |
| self.normals | list of all the normal lines created by source |
| self.angles | list of all the angles created by source |
| self.re | indicates whether or not the source has caused an total internal reflection and indicates the position pos indicates the top and/or bottom sides and neg indicates the side of object |

|  |  |  |
| --- | --- | --- |
| Methods | Purpose | Return |
| self.findupdate(x,y) | when position or source manipulated it updates the attributes self,findx and self,findy in accordance with the change that has occurred | self.findy, self.findx |
| self.blackout() | when source objects are created because of another source it loses its ability to rotate manually so the self.graphical2 is blacked out | return self.colour |
| self.InsideSimulation(source,intialsource,obj) | simulates the products of the initial source if they are found inside another object and manipulated accordingly, parameter source refers to the object is in, instead of finding it again, initial source is the original source object that caused the change, the parameter obj takes the Boolean values true or false, if true then the propagation of the ray was caused by the movement of an object and products of original source will be stored accordingly | self.endx, self.endy |
| self.Simulation(state,previous, obj) | simulates what would happen to a ray when it interacts with another object depending on its property. Parameter state refers to whether the ray was the product of another ray, the parameter previous refers to the original source, and the parameter obj is the same as in the inside simulation method. First it finds if its ray intersects with any points on all the objects loaded, then for each intersection calculates the distance from start point to find closest, then depending on the property of the object manipulates the ray | self.endx, self.endy |
| self.statechange() | returns original end point of a source after it has left an object | self.endy, self.endx, self.graphical |
| self.redraw(x,y) | redraws source in accordance with where it has been moved by a mouse | self.graphical, self.xp, self.yp, self.redy, self.redx, self.rendy, self.rendx, self.endy, self.m |
| self.rotate(mousex,mousey) | rotates the source in accordance with where the mouse stops the dragging | self.graphical, self.graphical2, self.endy, self.redy, self.rendy, self.m, self.stateproduct |
| self.anglesfunc(theta1, theta2, orientation, coord, nor, nor2, sign, source) | draws in the arcs that represent the angle of refraction, incidence, and reflection against the normal line. The parameter theta1 represents the incidence angle, theta2 represents the refraction angle, orientation refers to which orientation the normal line is, coord refers to the coordinates of interaction. Nor refers to the degrees at which normal line is relative to incidence angle, Nor2 refers to the degrees at which normal line is relative to refraction angle, sign refers to the arc positions to the left or right of the normal or on top or under the normal, source refers to the initial source | None |
| self.normalfunc(self, orientation, source, coord) | draws in normal line when a source interacts with an object, orientation determines whether it is horizontal or vertical and source refers to initial source and coord is the coordinates of the intersection point | None |

Block Class-

Block and Glass Block class have the exact structure so I will only be explaining the structure of the block. They don’t hold one unique attribute but just override some from the inherited class (self.graphical, self.full\_name and self.refractive\_index) and all methods are the same as in their parent class

Mirror Class-

Has the same attributes and method as in parent class main objects, but notably has a different property so objects interact with light differently which will be addressed when designing simulation algorithm

Screen Class-

Stops any source object that intercepts it, similar case as for mirror class has same attributes and methods as parent class and interaction with light will be addressed in simulation algorithm

Semi-Circle Class-

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.Radius | Indicates the radius for the semi-circle |
| self.graphical2 | As pygame doesn’t have a draw function I will have to draw the semi-circle using a composite of the rectangle and circle function, self.graphical is the circle function with radius self.Radius and this attributes represents the rectangle that covers the bottom half of the circle |
| self.Eqaution | Identifies the equation of self.graphical (which is a circle) |
| self.Gradient | Is a attribute that calls a function from the sympy library that finds derivative of self.Equation |

|  |  |
| --- | --- |
| Method | Purpose |
| self.equa\_update | As the semi circle is moved around its equation changes but as the attribute self.Eqaution has already been assigned it doesn’t change as the center of the circle changes, so this method updates the self.Eqaution attribute when called |

Diffraction Class-

Due to the nature of representing diffraction as mentioned in the analysis section, I have opted to create a mode where only diffraction can occur activated using a button and deactivated using the same button while in this mode only objects from the diffraction class will be displayed but no progress will be lost when deactivated. The experiment that will be carried out is young’s double slit experiment, using another class(changeGUI) you will be able to manipulate the experiment.

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.slitdis | the slit distance |
| self.scrlength | the length of the screen used to stop the light rays |
| self.screendis | distance between slits and the screen used to stop the rays |
| self.sourcexp | the x coordinate of the source |
| self.sourceyp | the y coordinate of the source |
| self.source\_draging | indicates whether source is moving |
| self.wavelength | states the wavelength of the source |
| self.stateproduct | used to store the source objects created due to the diffraction of the main source |
| self.graphical | represents the top slit |
| self.graphical2 | represents the middle slit |
| self.graphical3 | represents the bottom slit |
| self.graphical4 | represents the screen |
| self.source | represents the source, instead of a line, a circle is used to represent the source to give a more radial property to the source rather than a direct one |
| self.a | represents the distance between slits |
| self.fringe | the separation between adjacent maximas |

|  |  |  |
| --- | --- | --- |
| Method | Purpose | Return |
| self.update() | checks if it is supposed to show the following: fringe separation, wavelength of source, screen distance, or slit separation. If so uses pygame to create a text that is displayed on screen | None |
| self.diffraction() | is the function responsible for simulating young’s double slit experiment | None |
| self.redraw(x) | redraws the composite objects when mouse is dragging one of the objects but only allows movement in the x axis hence only the parameter x | return self.source, self.sourcexp, self.endx, self.stateproduct |
| self.sourceredraw(x) | redraws the source when moving along the x axis | None |

Angles class-

Angles class does not inherit from the main object class.

|  |  |
| --- | --- |
| Attribute | Purpose |
| self.x | represent the x coordinate of where the angle will be drawn from |
| self.y | represent the y coordinate of where the angle will be drawn from |
| self.radius | represents radius of arc |
| self.theta1 | represents starting angle relative to the line of 0 degrees |
| self.theta2 | represents ending angle relative to the line of 0 degrees |
| self.colour | sets colour for arc outline |
| self.show\_theta | is the actual angle of the arc and the value that is shown |
| self.type | indicates whether it is an incident or refraction angle |
| self.orientation | indicates orientation of normal line so text can be drawn next to correct angle |

|  |  |  |
| --- | --- | --- |
| Method | Purpose | Return |
| self.redraw1() | redraws the arc | None |
| self.update() | checks if it is supposed to show the following: incidence angle and refraction angle. If so uses pygame to create a text that is displayed on screen | None |

Young’s Double Experiment Pseudocode

*DEF diffraction():*

*[] ← self.stateproduct*

*arcsin((source.wavelength / source.a)) ←theta1*

*source.y ← mid*

*INTEGER((source.scrlength - mid) / source.fringeseperation) ←fringe\_no*

*1 ←max*

*FOR point IN RANGE(fringe\_no):*

*max \* source.fringe ←height*

*tag\_generator(tag, "S")←Tag*

*Source(Tag)←diff\_ray*

*source.stateproduct.insert (diff\_ray)*

*source.xp + source.screendis←diff\_ray.endx*

*mid - height ←diff\_ray.endy*

*source.x←diff\_ray.x*

*mid ←diff\_ray.y*

*diff\_ray.blackout()*

*max += 1*

*1 ←min*

*FOR point IN RANGE(fringe\_no):*

*min \* source.fringe ←height*

*tag\_generator(tag, "S")←Tag*

*Source(Tag)←diff\_ray*

*source.stateproduct.insert (diff\_ray)*

*source.xp + source.screendis←diff\_ray.endx*

*mid - height ←diff\_ray.endy*

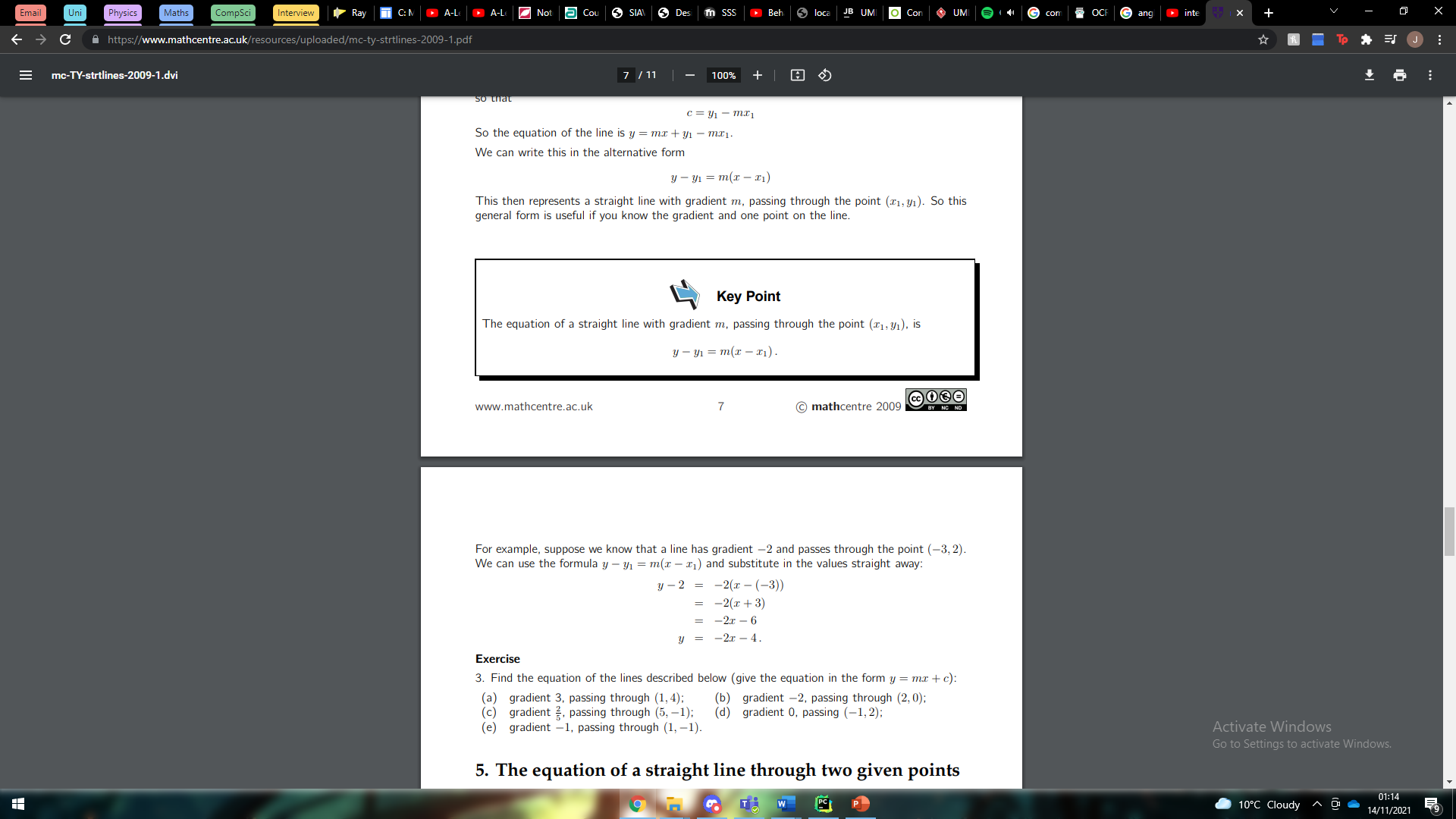
*source.x←diff\_ray.x*

*mid ←diff\_ray.y*

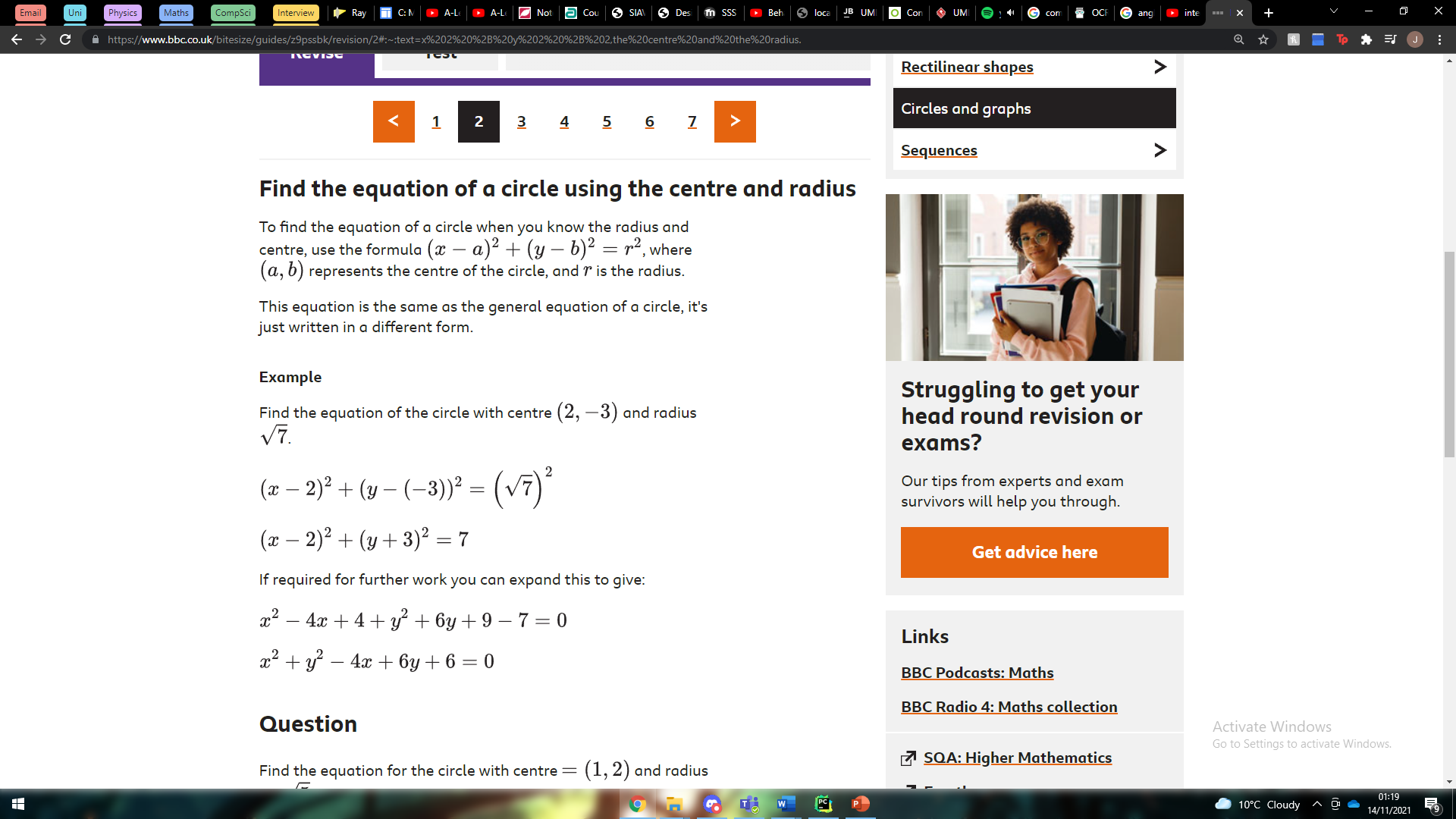
*diff\_ray.blackout()*

*min +=*

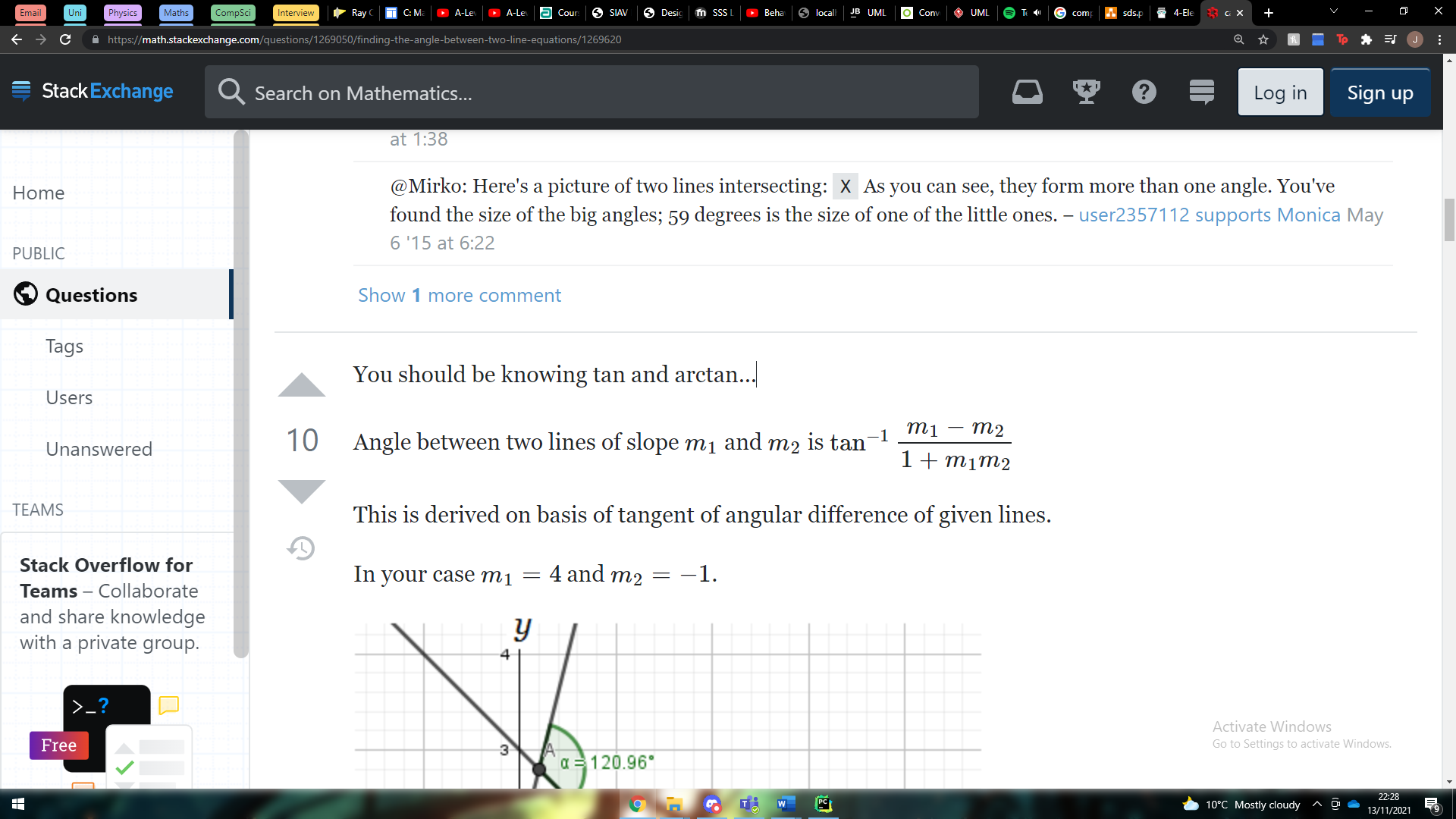
#### Eqautions

As my project surrounds the interaction of light rays with objects in a cartesian coordinate plane, I will be using a variety of different physics and mathematical.

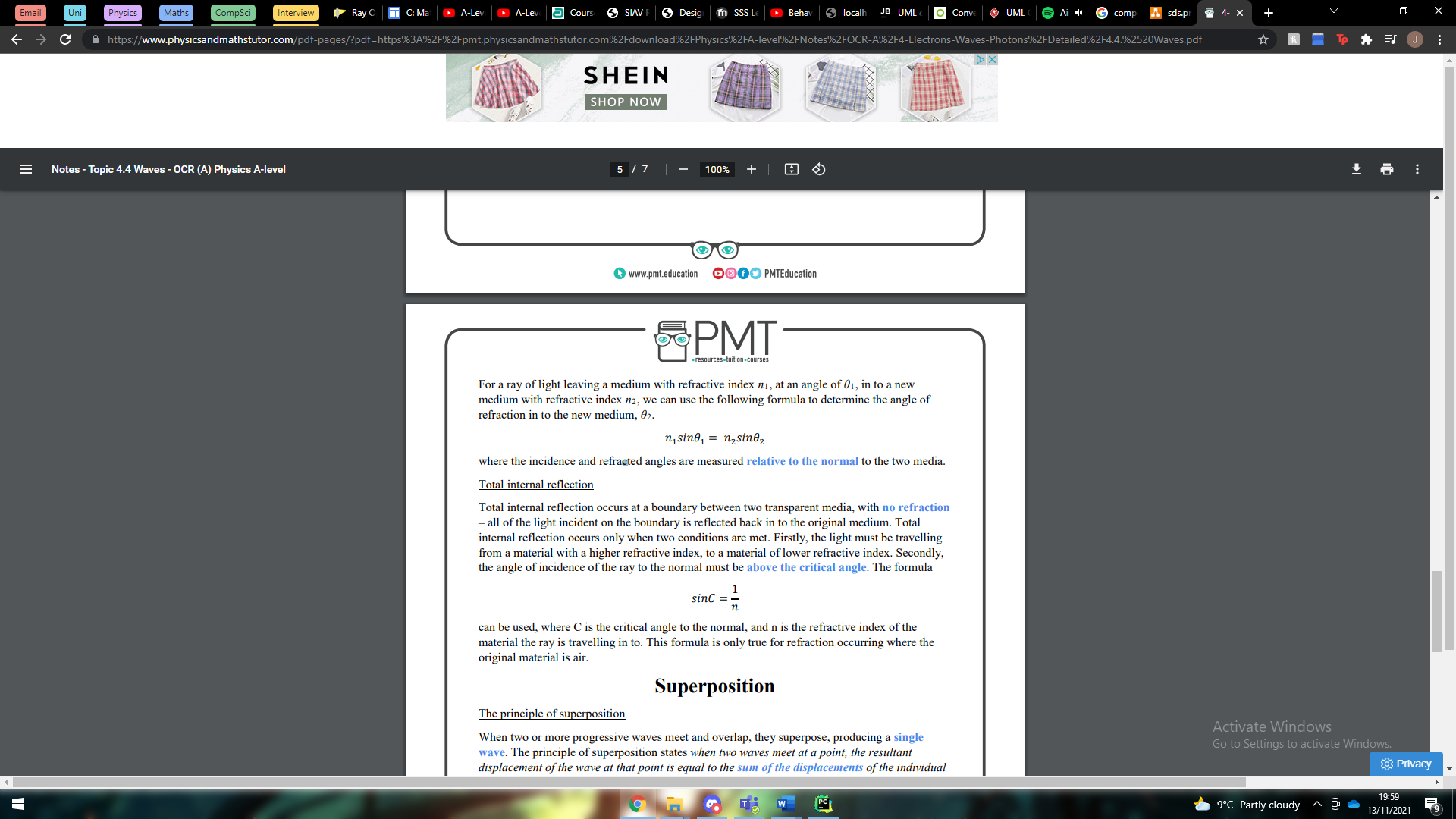
This version of the line equation was used to represent the equation of the source, by rearranging the equation to make x or y the subject the equation can be used to find the corresponding y or x values.



Used to represent the equation of the semi-circle, with a and b representing the x and y coordinate of the centre respectively and r representing the radius of circle. By making the equation equal to zero and replacing x or y with the equation of the source, it can be solved to find the points of interception in the circle. As we are working with a semicircle, we can check if the y coordinate is below or above the centre, then depending on how the semi-circle is drawn, then determine if the coordinate of interception is valid

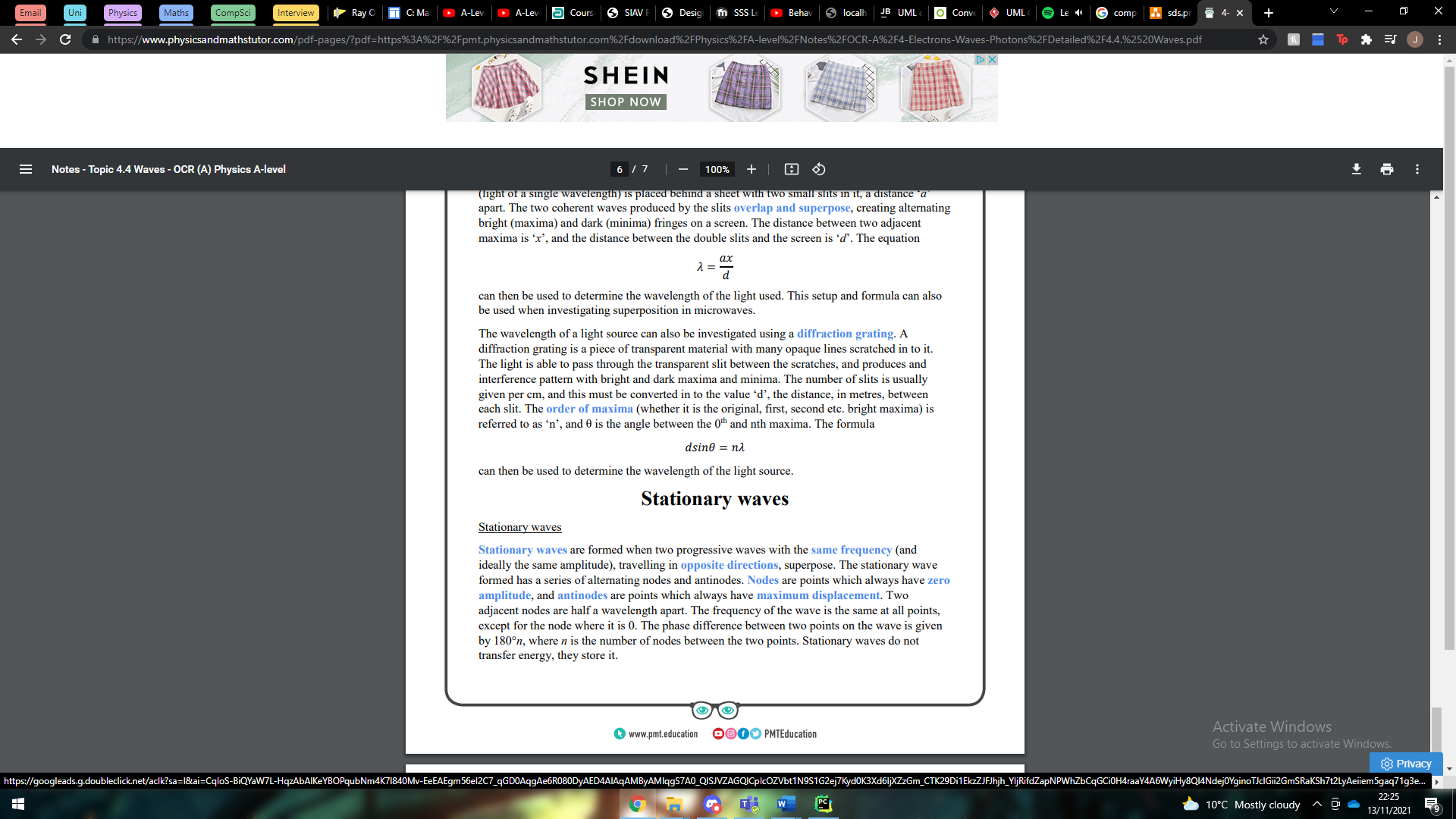


As we are working in a cartesian coordinate plane we can find the angle between two lines using the equation above, where m1 is the gradient of the one line and m2 is the gradient of the other. This equation is especially useful when finding the angle of incidence in relation to the normal.

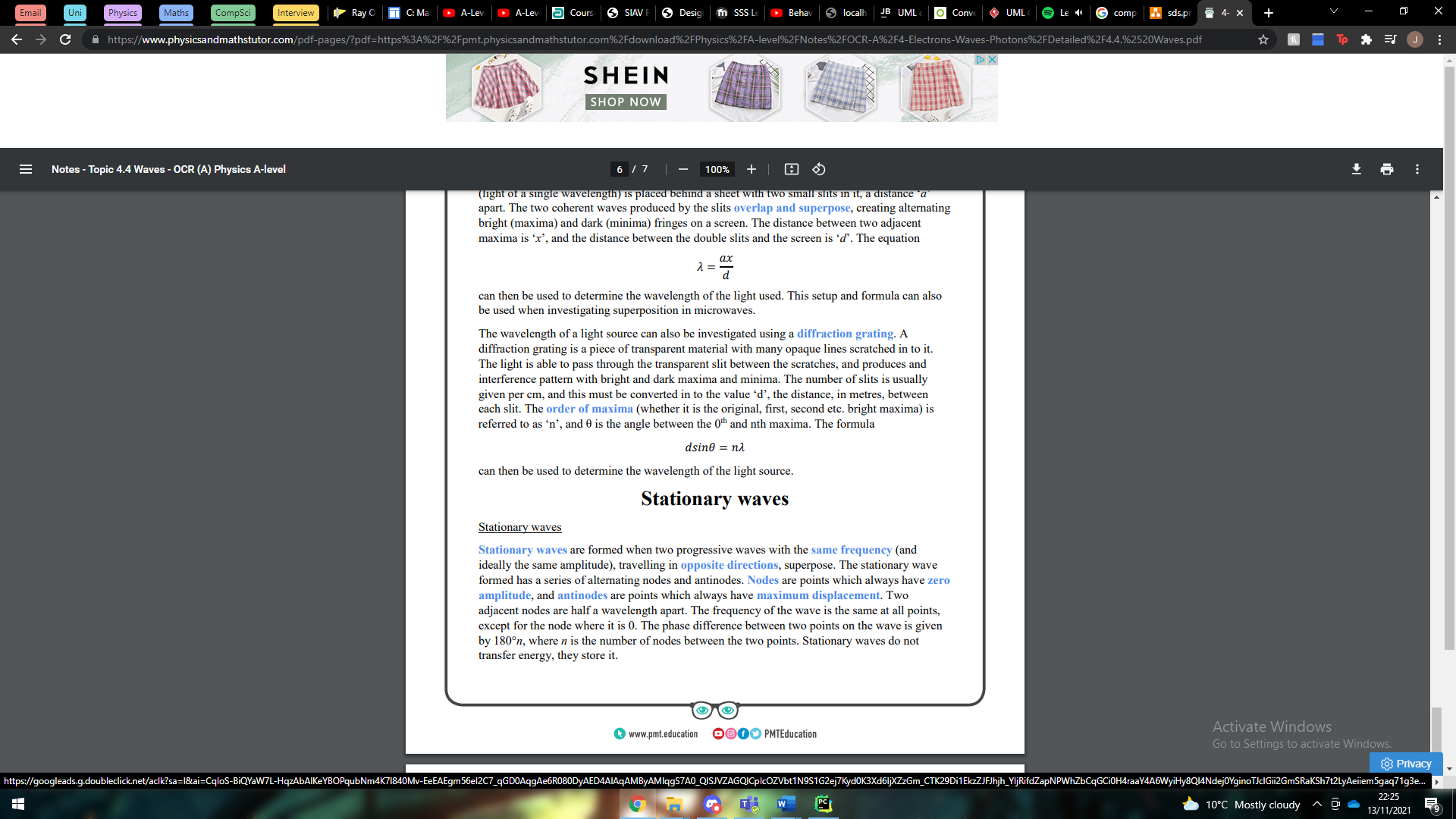


Using the equation above by using the angle of incidence and refractive index of the source is initially in, and the refractive index of object about to enter the angle of refraction can be found, and by rearranging the previous equation the gradient of the refracted line can the found. Also used for creating question answers.

Using the equation above the critical angle can be found, by determining the critical angle and comparing it with the angle of incidence (only when going from an object with a higher refractive index to a lower one) it can be determined whether total internal reflection has occurred. Also used for creating question answers.



This equation used when creating the young’s double slit experiment, used to find the fringe separation of the experiment. Also used for creating question answers.



Used for creating question answers.



This formula is used to determine the closet interception point by finding the distance between source and point using the formula above.

### Networking (Connection, Display and Chat Box)

The network structure that I will be using is not conventional, I will be using Server-Teacher.py to initiate server, however on the client side to connect to the server another file is run called network.py this structure is used as it allows the main file to continue running. Once connected to server only the first client (Top Client/Teacher) to connect can send data over the server and all other clients (Low Clients/Students) can only receive and not send. Teachers are not the server here as the sending and receiving data will interfere with the loop. To allow for multiple clients to connect to server threading is used, every time a client connects an instance of the threading function is initiated.

network.py

Network Class-

|  |  |
| --- | --- |
| Attributes | Purpose |
| Self.client | A socket instance is made. The first parameter is AF\_INET and the second one is SOCK\_STREAM. AF\_INET refers to the address-family ipv4. The SOCK\_STREAM means connection-oriented TCP protocol. |
| Self.server | IP address of the local host. |
| Self.port | Indicates port in which data is sent and retrieved from |
| Self.addr | Tuple consisting of self.server and self.port |
| Self.p | Runs the method connect |

|  |  |
| --- | --- |
| Methods | Purpose |
| Self.getP() | Retrieves the attribute self.p |
| Self.connect() | Connects user to server and returns what has been sent to client |
| Self.rec() | Returns what has been sent to clients |
| Self.send(data) | Sends data across to server and returns what has been sent to clients |

Server File-

|  |  |
| --- | --- |
| Variables | Purpose |
| server | Get IP of the local host |
| port | Indicates port in which data is sent and retrieved from |
| clients | Lists of clients that have connected to server |
| s | A socket instance is made. The first parameter is AF\_INET and the second one is SOCK\_STREAM. AF\_INET refers to the address-family ipv4. The SOCK\_STREAM means connection-oriented TCP protocol. |
| d | Dictionary used to complete connection |

|  |  |
| --- | --- |
| Function | Purpose |
| threaded\_client(conn, clients) | Completes connection to server then a while loop which retrieves pickled data sent by clients, if client is the first client to connect it will send the pickled data to every client, if it isn’t the first client then nothing occurs and loop continues.  If data is none then server will be disconnected as the first client has left and connection is closed. |

Network Hierarchy (Game Display)-

Server

Top Client





Low Clients

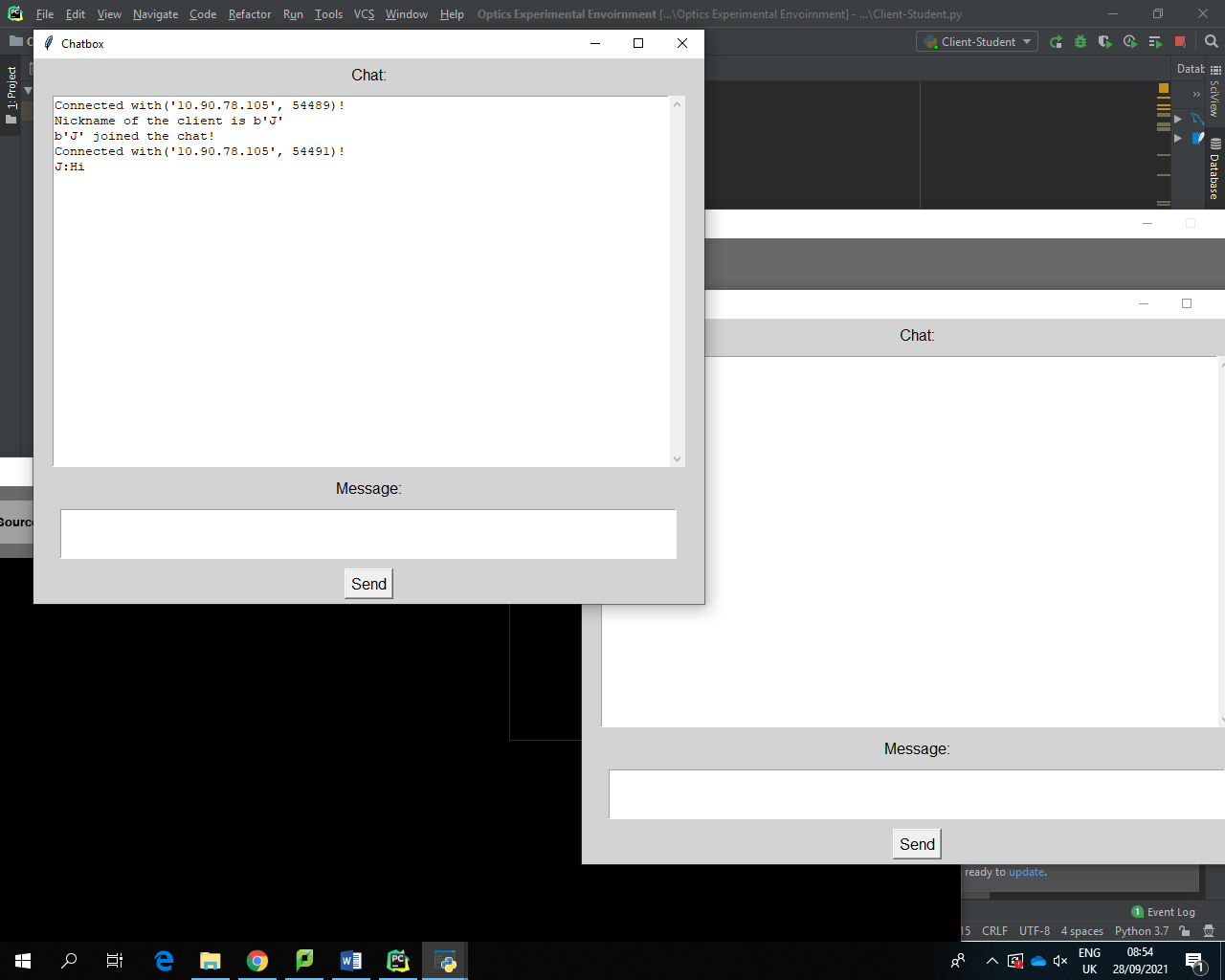
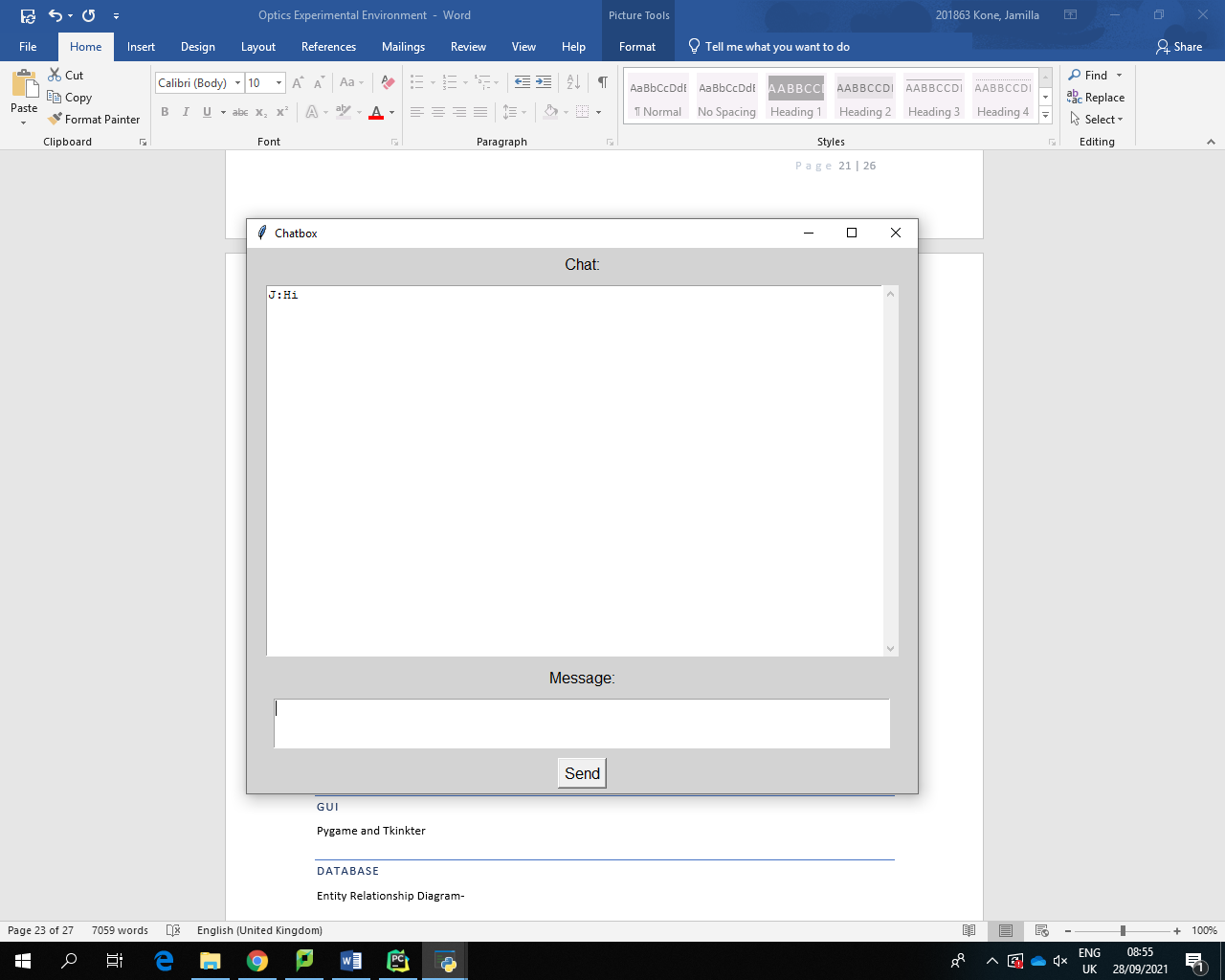
For the chat box the teacher is the server and the students are the clients the chat box can be accessed from the GUI by clicking the “Chat” function on both the teachers and students display. Clients chat box can only be assessed if server chat box is open. The GUI of the chat box will use tkinkter and clients will be asked for a display nickname. Teacher can also chat through the server chat box.

Network Hierarchy (Chat box)-

Server



Clients



Client Chat Box

Server Chat Box

ChatboxServer.py

This file is responsible for being the server and host chatbox for the chatbox network.

Server class-

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.host | returns the host name of the current system under which the Python interpreter is executed. |
| self.port | assigns a specific port which data will be received and transferred |
| self.clients | list that will hold clients that have joined the chat box |
| self.nicknames | holds the names the client will be addressed as on chat box |
| self.answered | using a dictionary with the key being the student’s id and the value being a list of right or wrong answers the question is are represented by the index for example index 1 represents question 2 |
| self.answers | holds the correct answer for the current question being asked reset after times up |
| self.question | holds the string of every question asked during the session in a list |
| self.sock | A socket instance is made. The first parameter is AF\_INET and the second one is SOCK\_STREAM. AF\_INET refers to the address-family ipv4. The SOCK\_STREAM means connection-oriented TCP protocol |
| self.nickname | holds nickname for the server/teacher set on default as “Server” |
| self.gui\_done | indicates whether the GUI has been created and is being displayed |
| self.running | used for the main while loop |
| self.timer | holds the amount of time that is set for a question |
| self.asked | indicates if a question has been asked |
| self.t0 | used when calculating how much time has passed |

|  |  |
| --- | --- |
| Method | Purpose |
| self.questions(timeset, question, inputs) | leads on from the QuestionGUI’s method run, for every item in question identifies which question is being asked and broadcasts the question from the inputs calculates, answer which is then assigned to self.answer, adds the question to self.question list and starts timer. |
| self.gui\_loop() | using tkinkter draws out a chatbox |
| self.broadcast(message) | sends parameter message to every client |
| self.receive() | when a new client joins server-side states the address that the client joined with sends string NICK to trigger a popup screen that will ask client to enter their nickname and same for ID but asks for the client’s student ID that connects to the database. It then creates a dictionary key with student id and values of an empty list for the dictionary self.answered. it then starts thread for that client to allow connection of multiple clients |
| self.stop() | set self.running as false and destroys window and returns self.questions and self.answered used in making a report |
| self.handle(client) | if question is asked and then checks that time hasn’t ran out if it has resets self.answers, self.timer and self.asked. Then tries if self.asked true and if command ‘/answer’ in message, which indicates following characters are the clients answer for the asked question, it then check their answer against the correct one and if correct 1 else 0 is added to value list in the dictionary self.answered. if /answer not in message the message is broadcasted, and if self.asked false it broadcast the message. If try runs into an error it removes client from list and closes the connection |
| self.write() | writes messages onto the server’s chat box |

ChatboxClients.py

File responsible for handling the chat box of clients.

Clients class-

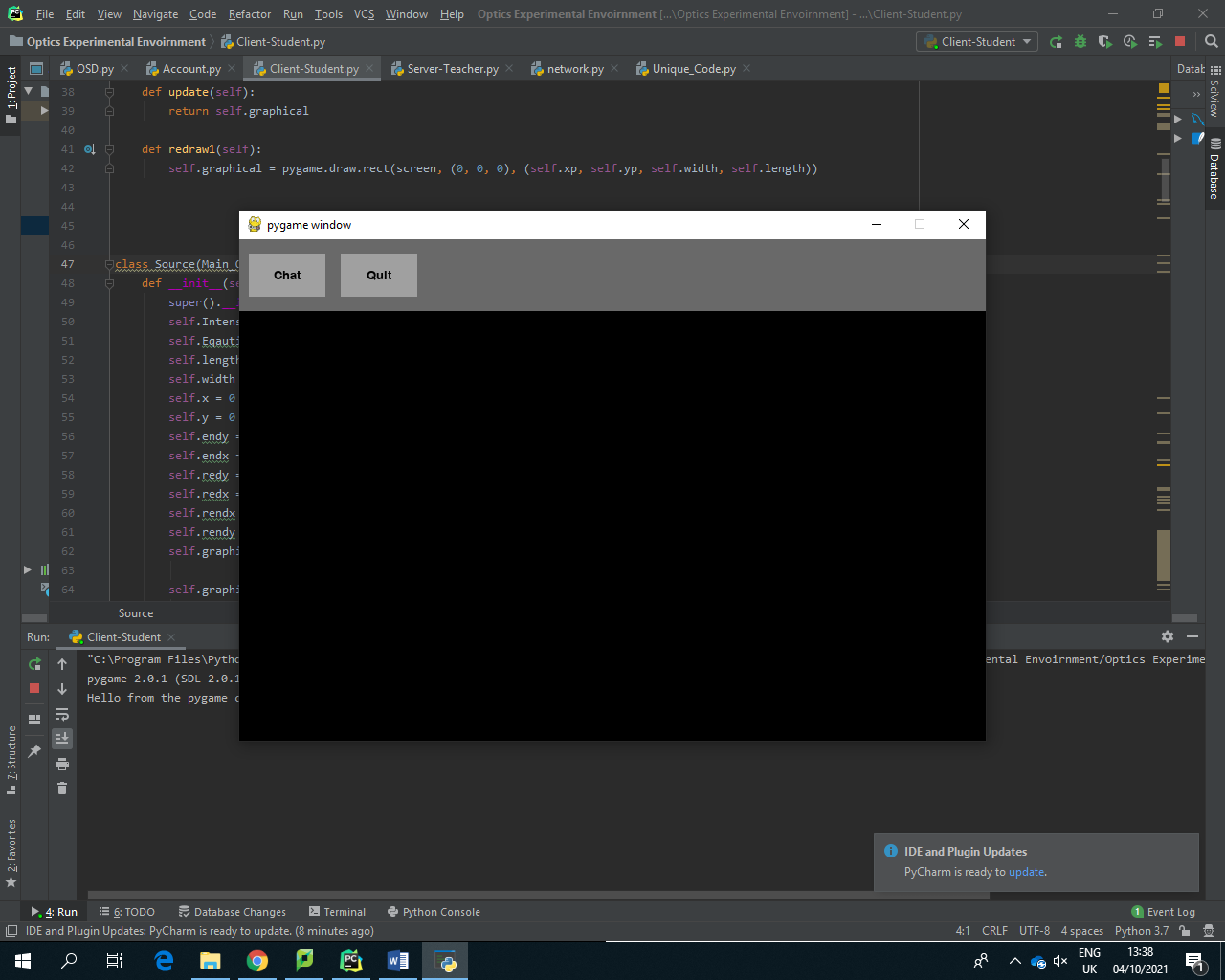
The following attributes that are the same as in the Server class in the file ChatboxServer.py: self.host, self.port, self.sock, self.gui\_done and self.running. The method self.gui\_loop and self.stop is also the same as in Server class.

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.nickname | holds the nickname of client |
| self.stuId | holds the student id of client |

|  |  |
| --- | --- |
| Method | Purpose |
| self.write() | when send button clicked sends the message in the input box to the server to be handled |
| self.receive() | tries to check if the sent message contains triggers such as NICK and ID if so sends their corresponding attribute over to the server, if no triggers triggered then it writes the message on screen. If errors happen during try it closes connection |

### GUI

I will be using pygame and tkinkter as my GUI with pygame being used for the simulation window and tkinkter being used for the admin windows. Pygame is being used for my simulation windows as it allows for easier manipulation of objects and a straight forward coordinate system.

Model of Simulation Window-

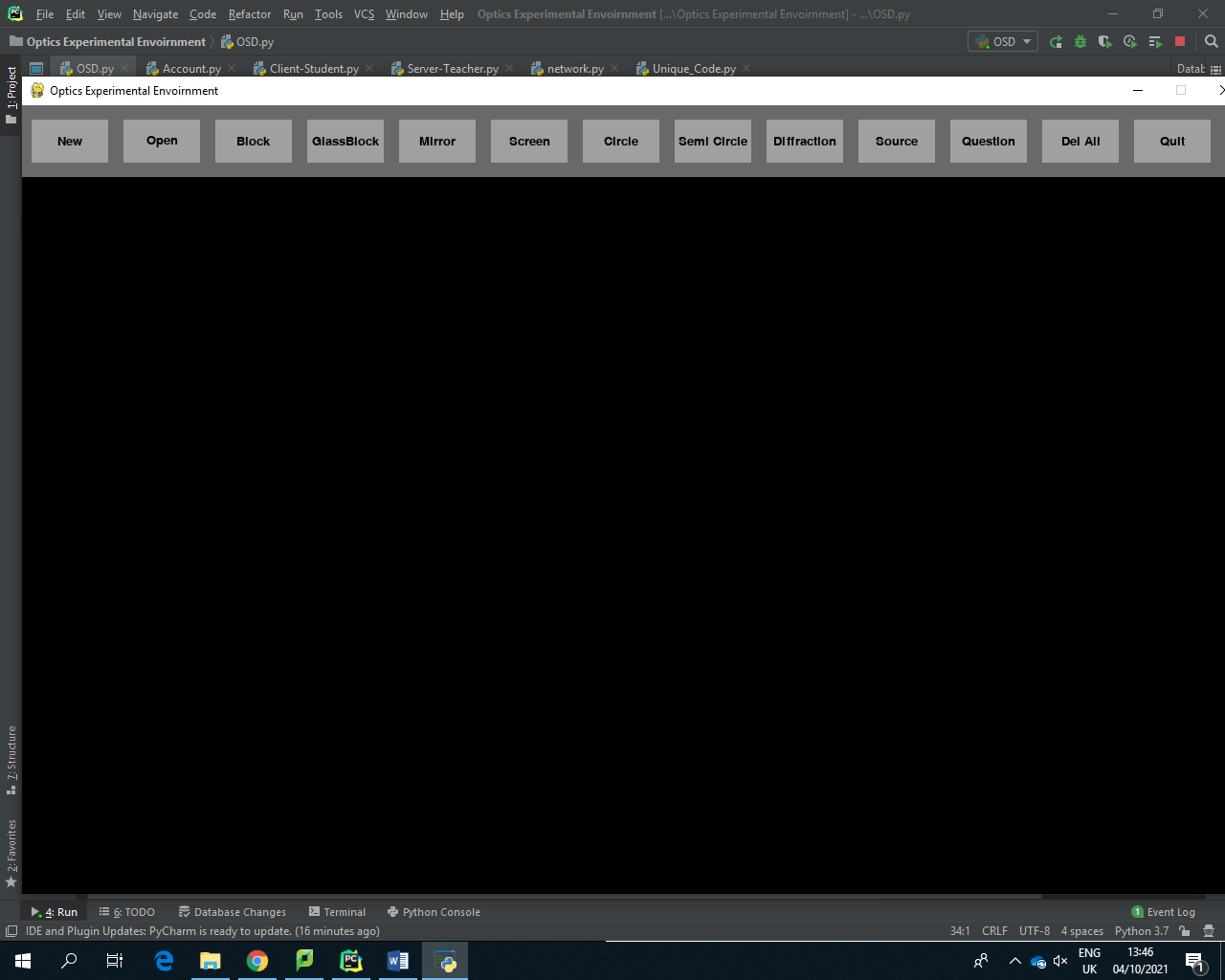
Client Side-

Buttons:

Chat- runs file ChatboxClient.py

Quit- Disconnects from server and stops running code

Where objects will be displayed

Server Side-

Where objects will be displayed and can be moved around

|  |  |
| --- | --- |
| Button Name | Purpose |
| New | Creates new file that can be opened |
| Screen | Instantiates the screen object and displays it |
| Open | Allows user to open previously saved environments |
| Circle | Instantiates the Circle object and displays it |
| Open | Allows user to open previously saved environments |
| Block | Instantiates the Block object and displays it |
| Semi-Circle | Instantiates the Semi-Circle object and displays it |
| Glass Block | Instantiates the Glass Block object and displays it |
| Diffraction | Instantiates the Diffraction object and displays it |
| Mirror | Instantiates the Mirror object and displays it |
| Source | Instantiates the Source object and displays it |
| Question | Allows user to set questions to clients and also suggests possible questions |
| Del All | Deletes every object that is being displayed |
| Quit | Quits the code |
| Show | Allows users to select which properties they want to display |
| Change | Allows users to change properties of individual objects |

Button class-

Is responsible for positioning and drawing the button and also indicates whether the button has been clicked

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.defined\_name | Similar to like in the main object parent class it holds the name for button |
| self.rect | is the attribute responsible for holding the pygame function to draw the rectangular shape of the button |
| self.x | Indicates the x coordinate location of the top left corner of the left most button. |
| self.y | Indicates the y coordinate location of the top left corner of the left most button. |
| self.id | Holds the number of how far right to place the top left corner of the bottom relative to self.x and self.y |

|  |  |
| --- | --- |
| Method | Purpose |
| self.draw(text) | writes the name of the button, onto the button, identified by the parameter text |
| self.click(event) | using the event parameter checks if there has been any collisions with self.rect and returns the name of the button if true and returns false when false |

ChangeGUI class-

Used to change attributes of objects being displayed. Not every attribute from these classes can be changed as they are protected.

|  |  |
| --- | --- |
| Attributes | Purpose |
| self.main | Intialises main window tkinter |
| self.Tab | uses the method in tkinkter called Notebook so that I can manage a collection of windows and displays a single one at a time in the form of tabs. |

|  |  |
| --- | --- |
| Method | Purpose |
| self.run() | for every object being displayed, when the Change button is clicked, it will use the unique self.full\_name attribute given to objects as the title for the tab of every object using tkinkter notebook. Within a tab of a different object there is are different attributes that can be changed, with an input box and a confirmation button |
| self.diffraction\_change(i, slitdis, screendis)  self.block\_change (i, newRI, newlength, newwidth)  self.glass\_change (i, newlength, newwidth)  self.source\_change (i, wavelength, angle)  self.sm\_change (i, newlength)  self.semi\_change(i, newradius) | the parameter i represents the self for a particular object, the parameter other are results from the input of the tabs. The method checks for an input then changes the corresponding attribute |
| self.stop() | destroys window after a selection has been made |

ShowGUI class-

A tick box selection GUI that allows you to pick the aspects of an experiment that the host wants displayed.

|  |  |
| --- | --- |
| Attribute | Purpose |
| self.IA | holds a Boolean value for whether the incidence angles can be displayed |
| self.RA | holds a Boolean value for whether the refraction/reflection angles can be displayed |
| self.RI | holds a Boolean value for whether the refractive index of an object can be displayed |
| self.SlitS | holds a Boolean value for whether the narrow-slit separation can be displayed |
| self.FS | holds a Boolean value for whether the fringe separation can be displayed |
| self.SourceS | holds a Boolean value for whether the distance from the screen can be displayed |
| self.W | holds a Boolean value for whether the wavelength can be displayed |

|  |  |
| --- | --- |
| Method | Purpose |
| self.gui\_loop() | using tkinkter a window with tick boxes is displayed beside each one is labelled the names of every attribute, if clicked it will return a value of 1 if not 0 is returned |
| self.showing(IA, RA, RI, SlitS, FS, SourceS, W) | determines whether for each attribute the user has selected it to be shown or not |
| self.stop() | destroys the window |

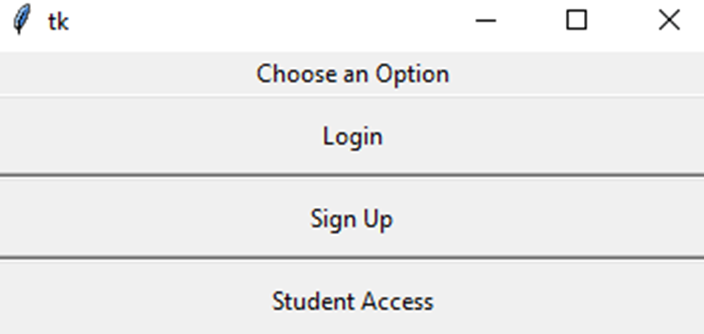
QuestionGUI class-

GUI and algorithm that combines using the ShowGUI class to automatically generate questions that can be asked and sent over a network to students.

|  |  |
| --- | --- |
| Attribute | Purpose |
| self.objects | holds the value of currently objects loaded |
| self.sources | holds the value of currently sources loaded |

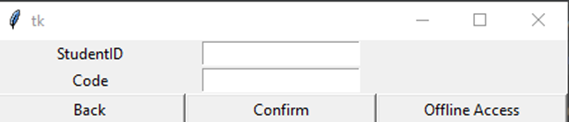
|  |  |
| --- | --- |
| Method | Purpose |
| self.AutoGen() | using the show class checks for every possible combination of show attribute that will can be used to calculate another value and also ensures if value being calculated is not showing it then appends key words of the specific question into a list and return it if question is suitable to be asked. |
| self.run(c) | similarly, to the changeGUI it uses tabs to separate each question, but first it runs the method AutoGen to see what questions can be asked, then in a for loop checks if a key word is in an element then creates a unique tab for the question. The tab contains the question and how much time the clients have to answer the question and as multiple experiments can be conducted on the environment, other variables needed to answer the question for a specific experiment (which are already being shown) are asked to be inputted so that the right answer is calculated, once the button set is clicked it sends the inputs to a method questions in the server class in the file ChatboxServer.py |
| self.stop() | destroys the window |

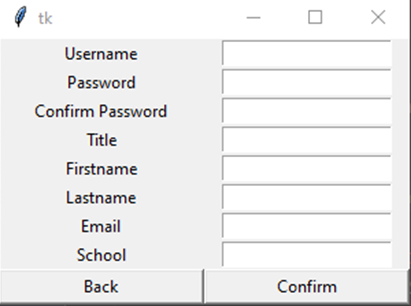
Model of Admin Windows-

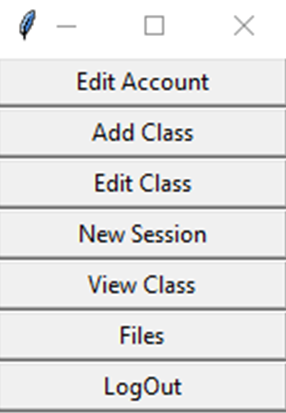
Main Menu

Login

Student Access



Sign-up

Logged in Window

Flow Chart for GUIDiagram

Description automatically generated

### Database

Database is fully normalised and is relational, built with a mixture of composite, foreign and private keys. Most queries are found in Account.py

|  |  |
| --- | --- |
| Table Standard Notation | Table Purpose |
| Users(UsersID, Username, Password) | Stores the username and passwords of all Accounts |
| Account(AccountID, Title, Firstname, Surname, School, Email, UsersID) | Stores all the essential details for each Account |
| Class(ClassID, StudentID, AccountID, Progress) | Stores the student that in the class the teacher and their progress in that particular class |
| Code(MasterCode, ClientIP, HostIP) | Stores the users that have connected to the server all deleted when session is done |
| Files(FileID, File, AccountID) | Stores information on the file created in each session |
| Session(SessionID, FileID, ClassID, Duration, StudentsPresent) | Stores information on each session |
| Students(StudentID, Firstname, Surname, Email) | Stores information on each student |

Key:

**Bold** and \_\_\_\_\_\_ indicates private key

**Bold** indicates foreign key

Entity Relationship Diagram-

Key:

One to one

One to many

Users

Students

Code

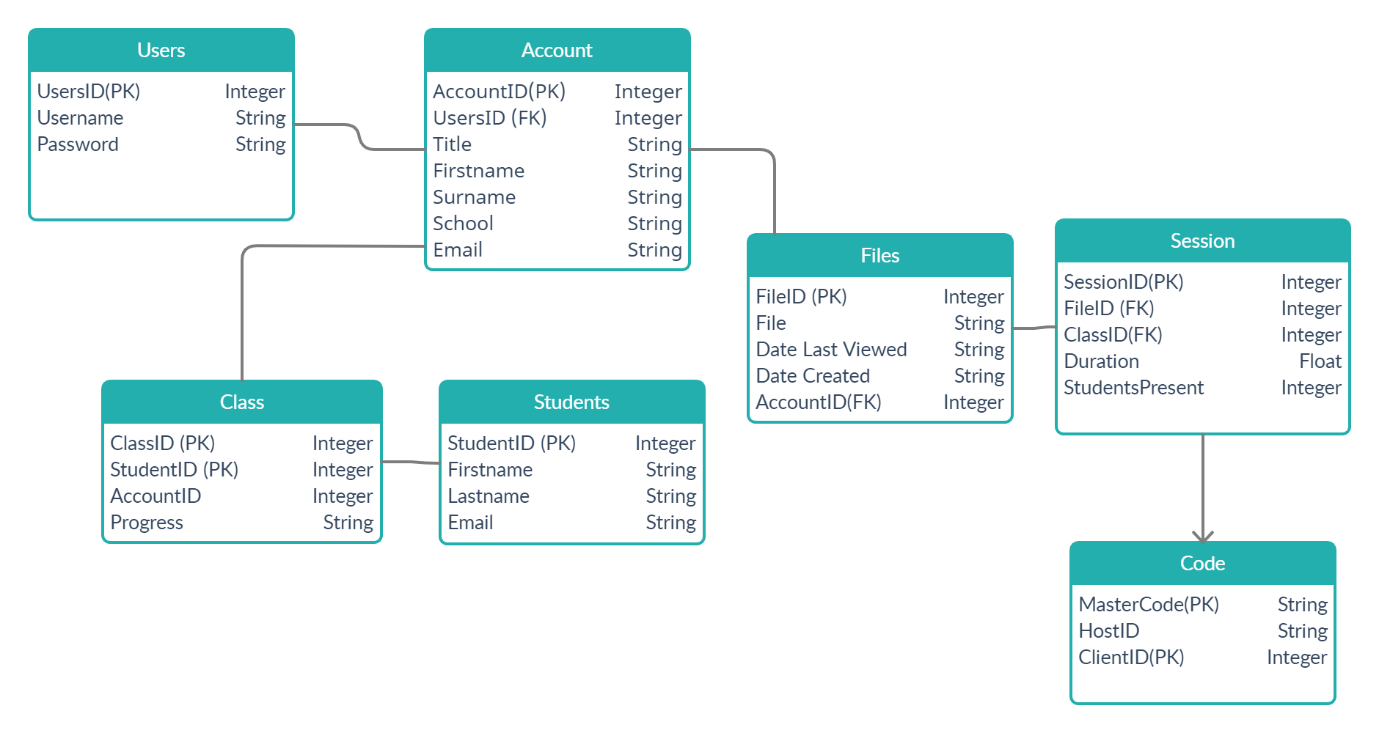
Account

Class

Files

Session

Database Linked Diagram-



One to Many

One to Many

One to Many

SQL Queries-

"SELECT StudentsID,Email FROM Students INNER JOIN Class USING(StudentsID) WHERE ClassID=? AND AccountID=?"

In this query I have requested the following data: StudentsID, Email and AccountID, from both the table Students and Class using inner join to link the tables so that it retrieves the email of the students where the StudentsID from the class table matches the one in the Students table where classID and AccountID are equal to a set of values.

"SELECT StudentsID, Firstname, Surname,Progress,Email FROM Students INNER JOIN Class USING(StudentsID) WHERE ClassID=?"

In this query I have requested the following data: StudentsID, Firstname, Surname, Progress and Email from both the table Students and Class using inner join to link the tables so that it retrieves the Firstname, Surname and Email of the students where the StudentsID from the class table matches the one in the Students table and the Progress from the Class table where classID are equal to a value.

### Login System

The login system will consist of using tkinter as the GUI and the back end is made-up object-oriented programming and database access. Login System code is all contained within the file Account.py

Account Class-

When the account class is first intialised the from the parameters sets; the Users ID from login, the username entered by the user during login and account ID from the login as well as attributes in the class. Also, as the class is called it also runs a method that is within the class.

|  |  |
| --- | --- |
| Attributes | Purpose |
| Self.ID | To hold the user ID of the logged in user |
| Self.Username | To hold the username of the logged user |
| Self.main | Intialises main window tkinter |
| Self.main.geometry | Sets the geometry (sets the window size of the main window) |
| Self.accID | Holds the account ID of the username |
| Self.hostIP | Stores the IP of the user that has logged in |
| self.db | holds the access to the database |

|  |  |
| --- | --- |
| Methods | Purpose |
| Self.start() | Creates the first screen after you have logged in which displays the options; edit account, add class, edit class, new session, view class, files, logout, all of the options are tkinter buttons and when pressed sends you to another method within the class |
| Self.code(win) | When a new session is created, the code method is running which creates a pop-up display(indicating nature of error which can be closed and classID can be reentered) using the win parameter as the previous window so that it can be displayed above it that allows you to input the class you want to create the session for, once the confirm tkinter button is pressed it runs the method teach\_class |
| Self.teach\_class (CID, win) | This method takes the class id inputted in the method code and check if you are the teacher of that class, if not it displays a pop-up error (indicating nature of error which can be closed and classID can be reentered) using the win parameter which is the previous display, if the user is it creates a unique code that is created in an imported file called Unique\_Code.py and inserts the hosts id and code into a database called Code, it then runs the session from the imported file OSD.py. Once the session is complete it selects from the file all the clients that joined the session and removes duplicates then counts how many students are present using len then inserts into the session table class ID, duration and number of students present. |
| Self.log\_out () | Destroys the main tkinter window and calls the function main which takes the user back to the display that lets you select if you want to login sign up or student access |
| Self.edit\_account () | Allows you to edit your account details each option is a tkinter button, once selected it runs the method edit\_sql |
| Self.edit\_sql (table, element, win) | Displays a window that asks for the edited version twice as a verification procedure, then takes the parameter table and element to know what table and which field will be edited then using self.ID changes the user’s field, if verification not correct creates error popup until matching or if back is clicked |
| self.edit\_sql2() | verifies that inputs are the same and adds to database |
| Self.add\_class () | First adds a dummy entity into the table class the retrieves the id from the dummy entity, it then displays buttons with the options; add new student, add existing student and done. It allows you to either add students already in the database to a new class or create a new student which is then added to class. |
| Self.add\_student\_new(win,classid) | Displays input boxes that ask for students first and surname and email then add to student table then adds to class. |
| self.student\_add(first,surname,email,win,classid) | verifies that inputs where correctly inputted and adds student to database and adds them to class specified |
| Self.add\_student\_existing(classid,win) | Displays input boxes that asks for the student’s ID then adds it to the to the class. |
| Self.edit\_class() | Display option button to delete a student from the class or delete class if delete student selected it runs the method del\_student, if delete class selected del\_student is run. |
| Self.del\_class(win) | Displays an input box asking for class id, then if okay selected runs method del\_class2. |
| Self.del\_class2(edit,classid) | Using SQL, it checks if the user is the teacher of that class or if the class exists if not a pop-up error is displayed (indicating nature of error which can be closed and classID can be reentered) else it deletes every entity in class with the class ID of the selected classID. |
| Self.del\_student(win) | Asks for studentID and classID then when okay is selected it runs the method del\_sudent2. |
| Self.del\_student2(edit, studentid, classid) | Checks using the parameters checks if student is in class and if student exists and also if student in the same class as the teacher, if so deletes student from class else it displays a pop-up error indicating nature of error which can be closed and classID and studentID can be reentered. |
| Self.edit\_student () | Displays a window in which there are the following buttons Firstname, Lastname, Email and Okay. When Firstname Lastname and email are clicked it runs the edit\_sql method with uniquely appropriate parameters. |
| Self.view\_class1() | Displays a window asking for classID and a button okay, when okay pressed method view\_class ran. |
| Self.view\_class (topwin, id) | Selects "StudentID", "Firstname", "Surname", "Progress", "Email" from database table Students and Class from all students with from the classID |

First Window Class-

Displays the first window when program is first run and handles the purpose of running the program for either logging in or student access or signing up.

Class Diagram-

|  |  |
| --- | --- |
| Attributes | Purpose |
| Self.cursor | Enables traversal over the records and tables in a database |
| Self.db | holds the access to the database |
| Self.create | Initiates another class Create\_SQL |
| Self.hostIP | Retrieves the IP of the person trying to access account or student access |

|  |  |
| --- | --- |
| Methods | Purpose |
| Self.verify(username, password, win) | verifies the user’s login and logins them into the main page. Verifies using the parameters given hashes the password and searches for accounts with that password and the username. If password and username correct class Account is initiate, if not pop up error is displayed indicating nature of error which can be closed and parameters can be reentered. |
| Self.Create\_Users (username, password, passcon, title, firstname, lastname, school, email, win): | Creates new users using the parameters to insert the new user into the database. First checks if username is taken if it is, it displays a pop-up indicating nature of area which can be closed and data can be reentered, else then verifies the first password matches the second entered the hashes the password and adds it to database. |
| Self.login(window) | Displays a window asking for the input of username and password. Also has the buttons confirm which then the method verify is ran and the button back which when clicked method back\_reg is ran |
| Self.sign\_up(window) | Displays a window asking for username, password, confirm password, title, firstname, lastname, email, school. When button confirm is clicked, method Create\_Users is ran and when button back is clicked goes back to first window and destroys current |
| Self.back\_reg(win) | Destroys window that’s in parameter and initiates first window. |
| Self.student\_access(window) | Ask user for studentID and Code. When button confirm clicked method studentaccess is ran or when back is clicked method back\_reg is ran |
| Self.studentaccess(stu\_id, code, win) | Checks if student is in class and checks if code is correct then takes IP address and adds it to the code table along with code and hosts IP. If student not in class or code incorrect a pop-up error is displayed indicating nature of error. |
| Self.Main | Initiates first window with a choice of options Login, sign up and Student access each button will run a different method. For login method login is ran, method sign\_up is ran for Sign Up and method student\_access for Student access |

Create\_SQL class-

Holds the methods of SQL queries that are frequently used throughout the code.

Class Diagram-

|  |  |
| --- | --- |
| Attributes | Purpose |
| Self.dbname | Holds the name of the database |
| Self.cursor | Holds method that enables traversal over the records and tables in a database |
| Self.db | holds the access to the database |

|  |  |
| --- | --- |
| Method | Purpose |
| Self. Create\_ClassStu(self,classid,studentid,accid) | Using the parameters adds new student to class |
| Self. Create\_Accounts(self, Title, Firstname, Surname, School, Email, userID) | Using the parameters creates new account |
| Self. Session(self, FileID, ClassID, Date, Duration, StudentsPresent) | Using the parameters logs a session |
| Self. Create\_Users(self, Username, Password) | Creates the user |
| Self.Create\_Class(self, StudentID, AccountID) | Using parameters adds a new class |
| Self. Create\_Students(self,Title,Firstname,Surname,School,Email,UsersID) | Using parameters adds new student |
| Self. Create\_Files(self, File, DateLV, DateC) | Using parameter stores information on the file |
| Self. Edit(self, Table, ID, Field, Editted) | Using parameter edits a specific field of a record on any table |
| Self. Delete(self, Table, ID) | Using parameters deletes a record on a table |

### Saves

To save the user’s environment the program objects will be sterilized using pickle and stored in a separate file which can be reassessed on request of the user, once selected it will desterilize and objects will be redraw onto the display where they were last saved. Can only save and reopen saved progress during the same session.

Prototype Code For Saving Objects  
**with** **open(**"savegame"**,** "wb"**)** **as** f**:**

**if** objects\_loaded**:**

pickle**.**dump**(**objects\_loaded**,** f**)**

**else:**

**pass**

**if** sources\_loaded**:**

pickle**.**dump**(**sources\_loaded**,** f**)**

**else:**

**pass**

f**.**close**()**

Protype Code For Loading Objects Saved

objects\_loaded **=** **[]**

sources\_loaded **=** **[]**

screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

**with** **open(**"savegame"**,** "rb"**)** **as** f**:**

loaded **=** pickle**.**load**(**f**)**

f**.**close**()**

**for** m **in** loaded**:**

**if** "Source" **in** **str(**m**):**

sources\_loaded**.**append**(**m**)**

**else:**

objects\_loaded**.**append**(**m**)**

**if** objects\_loaded**:**

**for** n **in** objects\_loaded**:**

n**.**redraw1**()**

**else:**

**pass**

**if** sources\_loaded

**for** n **in** sources\_loaded**:**

n**.**redraw1**()**

**else:**

**pass**

### Unique Code

Uses time and the class ID to create a unique code by generating time in the format year+month+date+hour+minute+second all in their numerical forms, then adds the adjacent integers together to form a string of unrecognisable integers then adds class ID at the front of string.

Unique\_Code.py  
*DEF unique\_code(classID):  
  
 “”←code  
  
 datetime.now ← now  
  
 FOR digit IN range(7):  
  
 INTEGER(now[0]) + INTEGER(now[1]) ←no  
  
 code + STRING(no) ←no  
  
 now[2:] ←now  
  
 STRING(classID) + "-" + code ←code  
  
 RETURN code*

### Report

Report.py

Responsible for creating report of session using the variables from the ChatboxServer.py and Account.py files. Creates a csv report of each session, 2 files are made one of a bar chart for question averages and the other one goes into detail of which question each student struggled with and how needs an extra challenge.

Prototype code for report

**def** CSV\_Report**(**code**,** answered**,** question**):**

header **=** **[**"StudentID"**,** "Name"**]**

f **=** **open(**"{0}.csv"**.format(**code**),** 'w'**)**

data **=** **[]**

**for** qs **in** question**:**

data**.**append**(**0**)**

**for** i **in** answered**.**values**():**

**print(**i**)**

x **=** 0

**for** p **in** i**:**

**print(**p**)**

data**[**x**]** **=** data**[**x**]** **+** **int(**p**)**

x **+=** 1

**print(**data**)**

**for** d **in** **range(**0**,** **len(**data**)):**

data**[**d**]** **=** data**[**d**]** **/** **len(**answered**.**keys**())** **\*** 100

f**.**close**()**

**print(**data**)**

workbook **=** xlsxwriter**.**Workbook**(**"{0}.xlsx"**.format(**code**))**

worksheet **=** workbook**.**add\_worksheet**()**

chart **=** workbook**.**add\_chart**({**'type'**:** 'bar'**})**

worksheet**.**write**(**"A1"**,**"Question Average"**)**

worksheet**.**write\_column**(**"A2"**,** data**)**

chart**.**add\_series**({**'name'**:**'=Sheet1!$A$1'**,**'values'**:**'=Sheet1!$A$2:$A${}'**.format(len(**data**)+**1**)})**

chart**.**set\_title**({**'name'**:** 'Class Average Per Question'**})**

chart**.**set\_x\_axis**({**'name'**:** 'Percentage(%)'**})**

chart**.**set\_y\_axis**({**'name'**:**'Question No.'**})**

chart**.**set\_style**(**13**)**

worksheet**.**insert\_chart**(**'B1'**,** chart**,{**'x\_offset'**:** 25**,** 'y\_offset'**:** 10**})**

workbook**.**close**()**

f **=** **open(**"{0}.csv"**.format(**code**),** 'a'**)**

writer **=** csv**.**writer**(**f**,** delimiter**=**','**)**

header **=** header **+** question

writer**.**writerow**(**i **for** i **in** header**)**

ID **=** answered**.**keys**()**

rankings **=** **{}**

**for** student **in** ID**:**

total **=** 0

**for** i **in** answered**[**student**]:**

total **=** total **+** **int(**i**)**

total **=** total **/** **len(**question**)** **\*** 100

rankings**[**student**]** **=** total

**with** sqlite3**.**connect**(**"Accounts.db"**)** **as** db**:**

cursor **=** db**.**cursor**()**

classId **=** code**.**split**(**"-"**)[**0**]**

**for** student **in** ID**:**

details **=** **[]**

sql **=** "SELECT Firstname Surname FROM Students WHERE StudentsID=?"

cursor**.**execute**(**sql**,** **(**student**,))**

name **=** cursor**.**fetchall**()**

details**.**append**(**student**)**

details**.**append**(**"{} {}"**.format(**name**[**0**],**name**[**1**]))**

**for** i **in** **(**""**.**join**(repr(**e**)** **for** e **in** answered**[**student**])):**

details**.**append**(**i**)**

writer**.**writerow**(**i **for** i **in** details**)**

sql **=** "UPDATE Class SET Progress=? WHERE ClassID=? AND StudentsID=?"

cursor**.**execute**(**sql**,** **(**rankings**[**student**],** classId**,** student**))**

top **=** **[]**

**help** **=** **[]**

**for** i **in** ID**:**

**if** rankings**[**i**]** **>=** 95**:**

top**.**append**(**i**)**

**elif** rankings**[**i**]** **<=** 60**:**

**help.**append**(**i**)**

writer**.**writerow**(**i **for** i **in** **[**"StudentID of Those That Need a Challenge"**])**

writer**.**writerow**(**i **for** i **in** top**)**

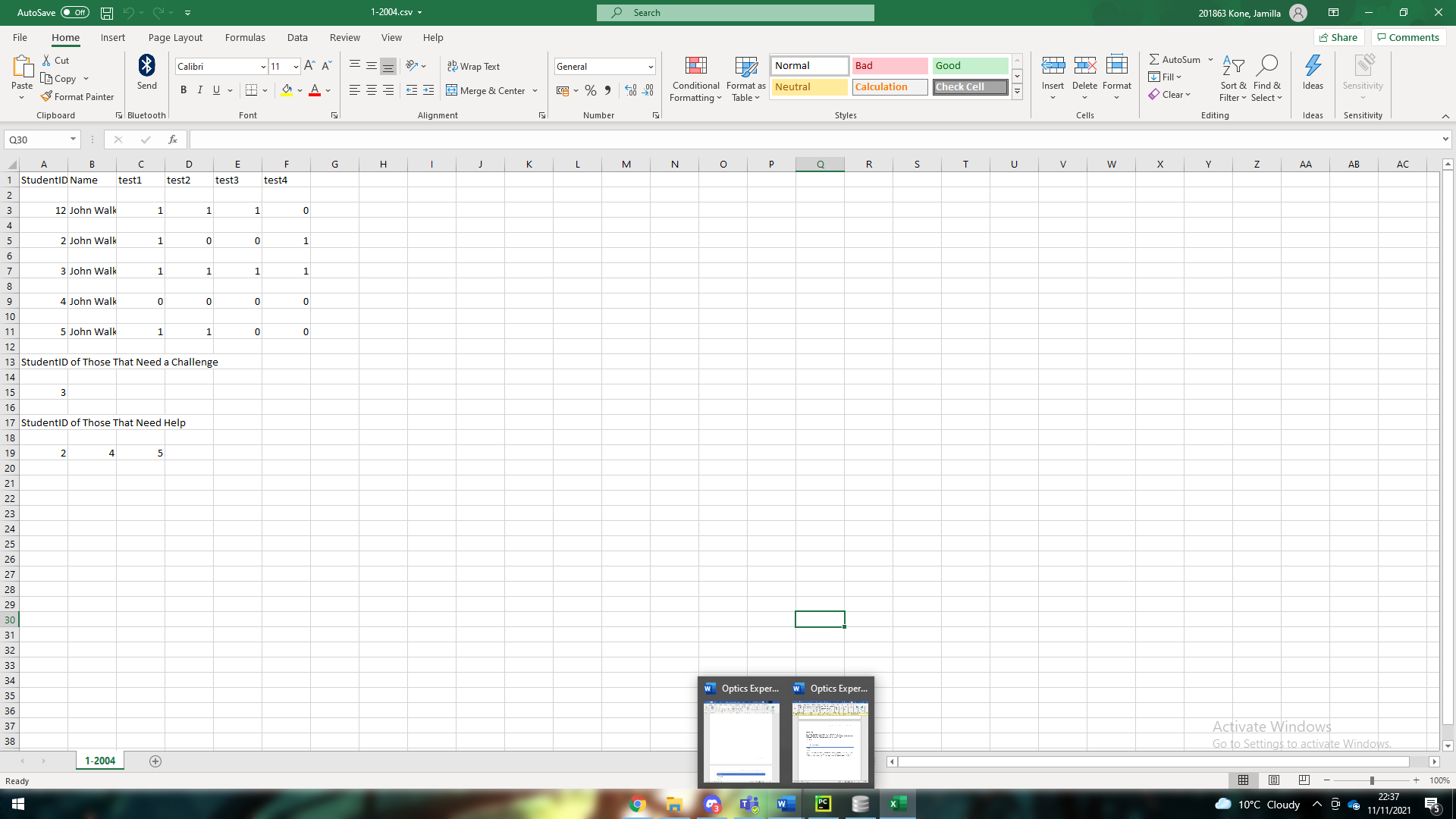
writer**.**writerow**(**i **for** i **in** **[**"StudentID of Those That Need Help"**])**

writer**.**writerow**(**i **for** i **in** **help)**

f**.**close**()**

Report Samples-

Questions that where asked during session

File1:

Student ID of students that appear to be need more help

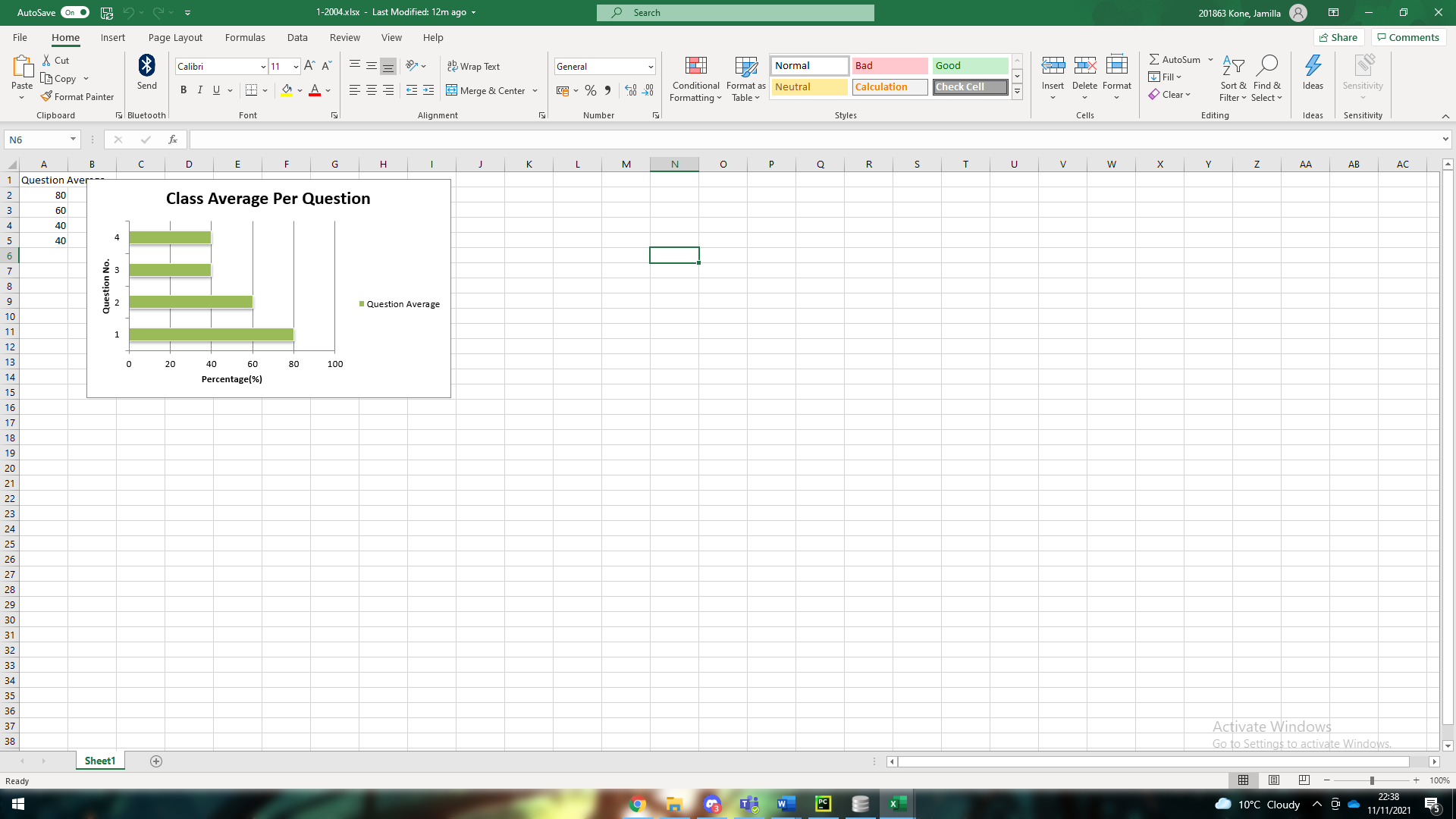
Student ID of students that appear to be need more of a challenge

Results: 1 is correct, 0 is incorrect

Name of Students who’s results are in that row

Student ID of student who’s results are in that row

File2:



Bar chart showing class average for each question

### Libaries Used

CSV-used to access and edit excel files to produce a report

sqlite3- used to connect to and edit database tables

xlsxwriter- used to create a bar chart for one of the files for the report

hashlib-used to hash password, to keep it secure inside the database

smtplib, ssl, email.mime.text - used to send emails

OS- used to interact with the underlying operating system

subprocess- used to run other files within a file

socket- used for networking

math-used to have access to trigonometry functions and pi

sys-used to exit file

tkinkter- used for GUI

pygame- used for GUI

threading- used to make a multithreaded Server

sympy- used for math function such as differentiation, solve and defining symbols

pickle-used to sterilize object to send them over the network and save environment

time-used to time how long program is going for

ctypes-used to get dimension of screen

# Technical Solution

## OSD.py

1 **import** math

2 **import** sys

3 **import** tkinter

4 **from** tkinter **import** ttk

5 **import** ChatboxServer

6 **import** pygame

7 **import** sympy

8 **from** sympy **import** **\***

9 **import** pickle

10 **import** time

11 **from** network **import** Network

12 **import** ctypes

13 **from** pygame **import** gfxdraw

14 **import** Report

15

16 user32 **=** ctypes**.**windll**.**user32

17 width **=** user32**.**GetSystemMetrics**(**0**)** # holds value of the width of the monitor

18 length **=** user32**.**GetSystemMetrics**(**1**)** # holds value of the length of the monitor

19 start\_time **=** time**.**time**()** # starts the timer for the session

20 pygame**.**display**.**set\_caption**(**'Optics Experimental Envoirnment'**)** # sets the title of window

21 tag **=** **[**0**]**

22 Id **=** 0

23 pygame**.**init**()**

24 screen **=** pygame**.**display**.**set\_mode**((**width**,** length **-** 200**))** # sets dimension of window

25 screen**.**fill**((**0**,** 0**,** 0**))**

26 objects\_loaded **=** **[]**

27 sources\_loaded **=** **[]**

28 toolbar\_button **=** **[]**

29 IncidentRays **=** **{}**

30 DiffRays **=** **{}**

31

32

33 ##### tag\_generator #######

34 # Parameters :- tag:List, obj\_name:String

35 # Return Type :- String

36 # Purpose :- This is to generate a unique tag whenever an instance of

37 # a class is loaded it has an attribute that holds it name as there can be

38 # multiple of instances of one class having this unique name will help identify

39 # which instance needs to be manipulated

40 ###########################

41 **def** tag\_generator**(**tag**,** obj\_name**):**

42 x **=** tag**[-**1**]** **+** 1

43 tag**.**append**(**x**)**

44 Tag **=** obj\_name **+** **str(**tag**[-**1**])**

45 **return** Tag

46

47

48 ##### Angle #######

49 # Parameters :- None

50 # Purpose :- This class creates the pie shape that represents the angle

51 # between two lines and has the ability to show what angle the pie represents

52 ###########################

53 **class** **Angle:**

54 **def** \_\_init\_\_**(**self**):**

55 self**.**x **=** 0

56 self**.**y **=** 0

57 self**.**radius **=** 30

58 self**.**theta1 **=** 0

59 self**.**theta2 **=** 0

60 self**.**colour **=** **(**255**,** 0**,** 0**)**

61 self**.**show\_theta **=** 0

62 self**.type** **=** ""

63 self**.**orientation **=** ""

64 self**.**update**()**

65

66 ##### redraw1 #######

67 # Parameters :- None

68 # Return Type :- None

69 # Purpose :- Purpose is to redraw instance of class on the screen

70 ###########################

71

72 **def** redraw1**(**self**):**

73 pygame**.**gfxdraw**.**pie**(**screen**,** **int(**self**.**x**),** **int(**self**.**y**),** self**.**radius**,** **int(**self**.**theta1**),**

74 **int(**self**.**theta2**),** self**.**colour**)**

75 self**.**update**()**

76

77 ##### update #######

78 # Parameters :- None

79 # Return Type :- None

80 # Purpose :- To check what can be displayed on screens in terms of angles and

81 # updates the screen to display them

82 ###########################

83 **def** update**(**self**):**

84 **if** m**.**IA **and** self**.type** **==** "Incident" **and** self**.**orientation **==** "vertical"**:**

85 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

86 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

87 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 10**,** self**.**y **+** 10**))**

88 screen**.**blit**(**text**,** textrect**)**

89 **elif** m**.**IA **and** self**.type** **==** "Incident" **and** self**.**orientation **==** "horizontal"**:**

90 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

91 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

92 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 20**,** self**.**y **+** 20**))**

93 screen**.**blit**(**text**,** textrect**)**

94

95 **elif** m**.**RA **and** self**.type** **==** "Response" **and** self**.**orientation **==** "veritical"**:**

96 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

97 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

98 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 10**,** self**.**y **+** 10**))**

99 screen**.**blit**(**text**,** textrect**)**

100 **elif** m**.**RA **and** self**.type** **==** "Response" **and** self**.**orientation **==** "horizontal"**:**

101 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

102 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

103 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **+** 20**,** self**.**y **+** 20**))**

104 screen**.**blit**(**text**,** textrect**)**

105

106

107 ##### Main\_Objects #######

108 # Parameters :- name:String

109 # Purpose :- To act as a parent class to all manipulatable objects

110 ###########################

111

112 **class** **Main\_Objects:**

113 **def** \_\_init\_\_**(**self**,** name**):**

114 self**.**defined\_name **=** name

115 self**.**Refractive\_Index **=** 1.00

116 self**.**xp **=** **(**width **/** 2**)**

117 self**.**yp **=** **(**length **/** 2**)**

118 self**.**length **=** 200

119 self**.**width **=** 250

120 self**.**colour **=** **(**0**,** 0**,** 0**)**

121 self**.**Property **=** "Transparent"

122 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

123 self**.**draging **=** **False**

124 self**.**interceptors **=** **[]**

125

126 ##### redraw #######

127 # Parameters :- x:Int, y:Int

128 # Return Type :- Int, Object, List

129 # Purpose :- To move an objects to its new position dependant on the parameter x and y by

130 # filling the screen and redraw all loaded objects and instances new position and whether it can be moved

131 ###########################

132

133 **def** redraw**(**self**,** x**,** y**):**

134 **if** self**.**interceptors**:**

135 **for** i **in** sources\_loaded**:**

136 **for** obj **in** i**.**previousobj**:**

137 **if** obj**.**defined\_name **==** self**.**defined\_name**:**

138 **return** **None**

139 **else:**

140 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

141 **if** self**.**interceptors**:**

142 first **=** self**.**interceptors**[**0**]**

143 **for** i **in** objects\_loaded**:**

144 **if** i**.**defined\_name **!=** self**.**defined\_name **and** "D" **not** **in** i**.**defined\_name**:**

145 i**.**redraw1**()**

146 **for** i **in** sources\_loaded**:**

147 **if** i **!=** first**:**

148 i**.**redraw1**()**

149

150 first**.**normals **=** **[]**

151 first**.**angles **=** **[]**

152 IncidentRays**[**first**.**defined\_name**]** **=** **[]**

153 first**.**re **=** **None**

154

155 **else:**

156 **for** i **in** objects\_loaded**:**

157 **if** i**.**defined\_name **!=** self**.**defined\_name **and** "D" **not** **in** i**.**defined\_name**:**

158 i**.**redraw1**()**

159 **for** i **in** sources\_loaded**:**

160 i**.**redraw1**()**

161 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**x**,** y**,** self**.**width**,** self**.**length**))**

162 self**.**xp **=** x

163 self**.**yp **=** y

164 self**.**interceptors **=** **[]**

165 self**.**redraw1**()**

166

167 **return** self**.**yp**,** self**.**xp**,** self**.**interceptors

168

169 ##### update #######

170 # Parameters :- None

171 # Return Type :- None

172 # Purpose :- To check what can be displayed on screens in terms of angles and

173 # updates the screen to display them

174 ###########################

175 **def** update**(**self**):**

176 **if** m**.**RI**:**

177 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

178 text **=** font**.**render**(str(**self**.**Refractive\_Index**),** **True,** **(**105**,** 105**,** 105**))**

179 textrect **=** text**.**get\_rect**(**center**=(**self**.**xp **+** 20**,** self**.**yp **+** 20**))**

180 screen**.**blit**(**text**,** textrect**)**

181

182 ##### redraw2 #######

183 # Parameters :- None

184 # Return Type :- None

185 # Purpose :- To redraw every object on screen (updating screen contents)

186 ###########################

187

188 **def** redraw2**(**self**):**

189 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

190 **for** i **in** objects\_loaded**:**

191 **if** "D" **not** **in** i**.**defined\_name**:**

192 i**.**redraw1**()**

193 **for** i **in** sources\_loaded**:**

194 i**.**redraw1**()**

195

196 ##### Simulation #######

197 # Parameters :- state:Boolean, previous:Memory Address of Object, obj:Boolean

198 # Return Type :- None

199 # Purpose :- Checks if any sources loaded interacts with the object and manipulates

200 # the source dependant on the objects properties

201 ###########################

202

203 **def** Simulation**(**self**,** state**,** previous**,** obj**):**

204 **if** sources\_loaded**:**

205 **for** i **in** sources\_loaded**:**

206 i**.**Simulation**(False,** **None,** **True)**

207

208 ##### redraw1 #######

209 # Parameters :- None

210 # Return Type :- None

211 # Purpose :- updates instance of object on screen

212 ###########################

213 **def** redraw1**(**self**):**

214 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

215 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

216 self**.**update**()**

217

218

219 ##### Sources #######

220 # Inheirts :- Main\_Objects

221 # Parameters :- name:String

222 # Purpose :- Holds every attribute and operation that can be done on a source object

223 ###########################

224 **class** **Source(**Main\_Objects**):**

225 **def** \_\_init\_\_**(**self**,** name**):**

226 **super().**\_\_init\_\_**(**name**)**

227 self**.**length **=** 2

228 self**.**width **=** 10000

229 self**.**endy **=** self**.**yp

230 self**.**endx **=** self**.**width

231 self**.**redy **=** self**.**yp

232 self**.**redx **=** self**.**xp **+** 10

233 self**.**rendx **=** self**.**xp **+** 20

234 self**.**rendy **=** self**.**endy

235 self**.**colour **=** **(**255**,** 0**,** 0**)**

236 self**.**colour2 **=** **(**255**,** 193**,** 110**)**

237 self**.**thickness **=** 4

238 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,**

239 **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** self**.**endy**),** self**.**thickness**)**

240 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,**

241 **(**self**.**redx**,** self**.**redy**),** **(**self**.**rendx**,** self**.**rendy**),** self**.**thickness**)**

242 self**.**m **=** 0

243 self**.**findx **=** "((y-{0})/m)+{1}"**.format(**self**.**yp**,** self**.**xp**)**

244 self**.**findy **=** "m\*(x-{0})+{1}"**.format(**self**.**xp**,** self**.**yp**)**

245 self**.**state **=** **False**

246 self**.**previousobj **=** **[]**

247 IncidentRays**[**self**.**defined\_name**]** **=** **[]**

248 self**.**normals **=** **[]**

249 self**.**angles **=** **[]**

250 self**.**full\_name **=** "Source" **+** **str(**self**.**defined\_name**[-**1**])**

251 self**.**t\_status **=** 0

252 self**.**re **=** **None**

253

254 ##### findupdate #######

255 # Parameters :- x:float, y:float

256 # Return Type :- self.findy:float, self.findx:float

257 # Purpose :- updates equations used to find y or x coordinates of a point

258 # usually called after object is moved

259 ###########################

260 **def** findupdate**(**self**,** x**,** y**):**

261 self**.**findx **=** "((y-{0})/m)+{1}"**.format(**y**,** x**)**

262 self**.**findy **=** "m\*(x-{0})+{1}"**.format(**x**,** y**)**

263 **return** self**.**findy**,** self**.**findx

264

265 ##### blackout #######

266 # Parameters :- None

267 # Return Type :- self.colour: triplet

268 # Purpose :- isolates the red rotation button from source

269 # usually done to sources created due to an interaction with an object

270 ###########################

271 **def** blackout**(**self**):**

272 self**.**colour **=** **(**0**,** 0**,** 0**)**

273 self**.**redraw2**()**

274 **return** self**.**colour

275

276 ##### anglesfunc #######

277 # Parameters :- theta1:float, theta2:float, orientation:string, coord:list, nor:int,

278 # nor2:int, sign:string, source:Memory Address of Object

279 # Return Type :- None

280 # Purpose :- creates and displays (if allowed to be shown) the angles dependant

281 # on the parameters using the Angle class

282 ###########################

283 **def** anglesfunc**(**self**,** theta1**,** theta2**,** orientation**,** coord**,** nor**,** nor2**,** sign**,** source**):**

284 angle **=** Angle**()**

285 IncidentRays**[**source**].**append**(**angle**)**

286 self**.**angles**.**append**(**angle**)**

287 angle2 **=** Angle**()**

288 IncidentRays**[**source**].**append**(**angle2**)**

289 self**.**angles**.**append**(**angle2**)**

290 **if** sign **==** "reflecting"**:**

291 angle**.**show\_theta **=** theta1

292 angle**.type** **=** "Incident"

293 angle2**.**show\_theta **=** theta2

294 angle2**.type** **=** "Response"

295 angle**.**orientation **=** orientation

296 angle2**.**orientation **=** orientation

297 angle**.**theta1 **=** **-((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

298 angle**.**theta2 **=** nor

299 angle**.**x **=** coord**[**1**]**

300 angle**.**y **=** coord**[**2**]**

301 angle2**.**theta1 **=** nor2

302 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

303 angle2**.**x **=** coord**[**1**]**

304 angle2**.**y **=** coord**[**2**]**

305 **elif** orientation **==** "reverse"**:**

306 angle**.**show\_theta **=** theta1

307 angle**.type** **=** "Incident"

308 angle2**.**show\_theta **=** theta2

309 angle2**.type** **=** "Response"

310 angle**.**orientation **=** "vertical"

311 angle2**.**orientation **=** "vertical"

312 angle**.**theta2 **=** **((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

313 angle**.**theta1 **=** nor

314 angle**.**x **=** coord**[**1**]**

315 angle**.**y **=** coord**[**2**]**

316 angle2**.**theta1 **=** nor2

317 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

318 angle2**.**x **=** coord**[**1**]**

319 angle2**.**y **=** coord**[**2**]**

320 **elif** orientation **==** "reverse2"**:**

321 angle**.**show\_theta **=** theta1

322 angle**.type** **=** "Incident"

323 angle2**.**show\_theta **=** theta2

324 angle2**.type** **=** "Response"

325 angle**.**orientation **=** "horizontal"

326 angle2**.**orientation **=** "horizontal"

327 angle**.**theta2 **=** **((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

328 angle**.**theta1 **=** nor

329 angle**.**x **=** coord**[**1**]**

330 angle**.**y **=** coord**[**2**]**

331 angle2**.**theta1 **=** nor2

332 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

333 angle2**.**x **=** coord**[**1**]**

334 angle2**.**y **=** coord**[**2**]**

335 **else:**

336 angle**.**show\_theta **=** theta1

337 angle**.type** **=** "Incident"

338 angle2**.**show\_theta **=** theta2

339 angle2**.type** **=** "Response"

340 angle**.**orientation **=** orientation

341 angle2**.**orientation **=** orientation

342 angle**.**theta1 **=** **eval(**sign **+** **str((**theta1 **\*** 180**)** **/** math**.**pi**))** **+** nor

343 angle**.**theta2 **=** nor

344 angle**.**x **=** coord**[**1**]**

345 angle**.**y **=** coord**[**2**]**

346 angle2**.**theta2 **=** nor2

347 angle2**.**theta1 **=** **eval(**sign **+** **str((**theta2 **\*** 180**)** **/** math**.**pi**))** **+** nor2

348 angle2**.**x **=** coord**[**1**]**

349 angle2**.**y **=** coord**[**2**]**

350

351 ##### normalfunc #######

352 # Parameters :- orientation:string, source:Memory Address of Object, coord:list

353 # Return Type :- None

354 # Purpose :- creates a normal line dependant on the propagation position

355 ###########################

356 **def** normalfunc**(**self**,** orientation**,** source**,** coord**):**

357 Tag **=** tag\_generator**(**tag**,** "N"**)**

358 normal **=** Source**(**Tag**)**

359 IncidentRays**[**source**.**defined\_name**].**append**(**normal**)**

360 source**.**normals**.**append**(**normal**)**

361 normal**.**blackout**()**

362 normal**.**thickness **=** 2

363 **if** orientation **==** "top"**:**

364 normal**.**xp **=** coord**[**1**]**

365 normal**.**yp **=** coord**[**2**]** **-** 50

366 normal**.**endx **=** coord**[**1**]**

367 normal**.**endy **=** coord**[**2**]** **+** 50

368 **else:**

369 normal**.**xp **=** coord**[**1**]** **-** 50

370 normal**.**yp **=** coord**[**2**]**

371 normal**.**endx **=** coord**[**1**]** **+** 50

372 normal**.**endy **=** coord**[**2**]**

373 normal**.**colour2 **=** **(**255**,** 0**,** 0**)**

374

375 ##### InsideSimulation #######

376 # Parameters :- source:Memory Address of Object, initialsource:Memory Address of initial source, obj:boolean

377 # Return Type :- self.endx:float, self.endy:float

378 # Purpose :- method responsible for handling the interaction of the rays within an object

379 ###########################

380 **def** InsideSimulation**(**self**,** source**,** initialsource**,** obj**):**

381 coord **=** **[]**

382 intercept\_x **=** **[]**

383 intercept\_y **=** **[]**

384 **if** self**.**m **==** 0**:**

385 self**.**redraw1**()**

386 **else:**

387

388 **if** initialsource**.**m **<** 0**:**

389 eq **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(-**self**.**m**))**

390 xcoord **=** **eval(**eq**.**replace**(**"y"**,** "{0}"**.format(**source**.**yp**)))**

391 intercept\_x**.**append**((**xcoord**,** source**.**yp**))**

392 eq3 **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(-**self**.**m**))**

393

394 xcoord **=** **eval(**eq3**.**replace**(**"y"**,** "source.yp+source.length"**))**

395 intercept\_x**.**append**((**xcoord**,** source**.**yp **+** source**.**length**))**

396 eq4 **=** self**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**self**.**m**))**

397 ycoord **=** **eval(**eq4**.**replace**(**"x"**,** "source.xp+source.width"**))**

398 intercept\_y**.**append**((**source**.**xp **+** source**.**width**,** ycoord**))**

399 intercept\_x **=** **list(dict.**fromkeys**(**intercept\_x**))**

400 intercept\_y **=** **list(dict.**fromkeys**(**intercept\_y**))**

401 **else:**

402 eq **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

403 xcoord **=** **eval(**eq**.**replace**(**"y"**,** "{0}"**.format(**source**.**yp**)))**

404 intercept\_x**.**append**((**xcoord**,** source**.**yp**))**

405 eq3 **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

406 xcoord **=** **eval(**eq3**.**replace**(**"y"**,** "source.yp+source.length"**))**

407 intercept\_x**.**append**((**xcoord**,** source**.**yp **+** source**.**length**))**

408 eq4 **=** self**.**findy**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

409 ycoord **=** **eval(**eq4**.**replace**(**"x"**,** "source.xp+source.width"**))**

410 intercept\_y**.**append**((**source**.**xp **+** source**.**width**,** ycoord**))**

411 intercept\_x **=** **list(dict.**fromkeys**(**intercept\_x**))**

412 intercept\_y **=** **list(dict.**fromkeys**(**intercept\_y**))**

413

414 **if** **(**self**.**yp **==** source**.**yp **+** source**.**width **or** self**.**yp **==** source**.**yp**)** **and** self**.**m **>** 0**:**

415 **for** co **in** intercept\_x**:**

416 **if** source**.**xp **+** source**.**width **>=** co**[**0**]** **>=** source**.**xp **!=** co**[**0**]:**

417 coord**.**append**([**source**,** co**[**0**],** co**[**1**],** "top"**])**

418 **else:**

419 **pass**

420 **for** co **in** intercept\_y**:**

421 **if** source**.**yp **<=** co**[**1**]** **<=** source**.**yp **+** source**.**length **and** co**[**1**]** **!=** self**.**yp**:**

422

423 coord**.**append**([**source**,** co**[**0**],** co**[**1**],** "side"**])**

424 **else:**

425

426 **pass**

427 **else:**

428 **for** co **in** intercept\_x**:**

429 **if** source**.**xp **+** source**.**width **>=** co**[**0**]** **>=** source**.**xp **!=** co**[**0**]** **and** co**[**0**]** **!=** self**.**xp**:**

430

431 coord**.**append**([**source**,** co**[**0**],** co**[**1**],** "top"**])**

432

433 **else:**

434 **pass**

435 **for** co **in** intercept\_y**:**

436 **if** source**.**yp **<=** co**[**1**]** **<=** source**.**yp **+** source**.**length **and** co**[**1**]** **!=** self**.**yp**:**

437

438 coord**.**append**([**source**,** co**[**0**],** co**[**1**],** "side"**])**

439 **else:**

440

441 **pass**

442 **if** **not** coord**:**

443 self**.**redraw1**()**

444

445 **else:**

446 **for** i **in** coord**:**

447 distance **=** math**.**sqrt**(((**i**[**1**]** **-** self**.**xp**)** **\*\*** 2**)** **+** **((**i**[**2**]** **-** self**.**yp**)** **\*\*** 2**))**

448

449 in1 **=** coord**.**index**(**i**)**

450

451 coord**[**in1**].**append**(**distance**)**

452

453 coord **=** **sorted(**coord**,** key**=lambda** x**:** x**[-**1**],** reverse**=True)**

454

455 coord **=** coord**[**0**]**

456

457 **if** coord**[**0**].**Property **!=** "Reflective" **or** coord**[**0**].**Property **!=** "Stop" **or** "D" **not** **in** coord**[**

458 0**].**defined\_name**:**

459 **if** self**.**m **==** 0**:**

460 self**.**redraw1**()**

461 **else:**

462 **if** coord**[**3**]** **==** "top"**:**

463 self**.**normalfunc**(**"top"**,** initialsource**,** coord**)**

464 **else:**

465 self**.**normalfunc**(**"side"**,** initialsource**,** coord**)**

466

467 self**.**endx **=** coord**[**1**]**

468 self**.**endy **=** coord**[**2**]**

469 self**.**findupdate**(**coord**[**1**],** coord**[**2**])**

470 self**.**redraw1**()**

471 n2 **=** source**.**Refractive\_Index

472 critical **=** asin**(**1 **/** n2**)**

473 m1 **=** self**.**m

474 m2 **=** 0

475

476 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m1 **\*** m2**)))**

477 **if** coord**[**3**]** **==** "top"**:**

478 theta1 **=** **(**math**.**pi **/** 2**)** **-** theta1

479 **if** theta1 **>** critical**:**

480 Tag **=** tag\_generator**(**tag**,** "S"**)**

481 reflected\_ray **=** Source**(**Tag**)**

482 source**.**interceptors**.**append**(**reflected\_ray**)**

483 IncidentRays**[**initialsource**.**defined\_name**].**append**(**reflected\_ray**)**

484 reflected\_ray**.**m **=** **-**self**.**m

485 **if** self**.**m **<** 0**:**

486 self**.**anglesfunc**(**theta1**,** theta1**,** "vertical"**,** coord**,** 270**,** 270**,** "reflecting"**,**

487 initialsource**.**defined\_name**)**

488

489 **elif** self**.**m **>** 0 **and** coord**[**2**]** **==** source**.**yp**:**

490 self**.**anglesfunc**(**theta1**,** theta1**,** "vertical"**,** coord**,** 90**,** 90**,** "reflecting"**,**

491 initialsource**.**defined\_name**)**

492 initialsource**.**re **=** "pos"

493

494 **elif** self**.**m **>** 0 **and** coord**[**2**]** **==** source**.**yp **+** source**.**length**:**

495 self**.**anglesfunc**(**theta1**,** theta1**,** "vertical"**,** coord**,** **-**90**,** 270**,** "reflecting"**,**

496 initialsource**.**defined\_name**)**

497 initialsource**.**re **=** "neg"

498

499 **if** self**.**m **<** 0 **or** **(**self**.**m **>** 0 **and** coord**[**2**]** **==** source**.**yp **+** source**.**length**):**

500 eq **=** reflected\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**reflected\_ray**.**m**))**

501 reflected\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

502 **else:**

503 eq **=** reflected\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**reflected\_ray**.**m**))**

504 reflected\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

505 reflected\_ray**.**state **=** **True**

506 reflected\_ray**.**xp **=** coord**[**1**]**

507 reflected\_ray**.**yp **=** coord**[**2**]**

508 reflected\_ray**.**findupdate**(**coord**[**1**],** coord**[**2**])**

509 reflected\_ray**.**blackout**()**

510

511 reflected\_ray**.**InsideSimulation**(**source**,** initialsource**,** obj**)**

512

513 **elif** theta1 **<** critical**:**

514

515 **if** theta1 **<** critical **and** theta1 **!=** 0**:**

516 self**.**endx **=** coord**[**1**]**

517 self**.**endy **=** coord**[**2**]**

518 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m1 **\*** m2**)))**

519 t2 **=** asin**((**sin**(**theta1**)** **/** n2**))**

520 t2 **=** **(**math**.**pi **/** 2**)** **-** t2

521 theta1 **=** **(**math**.**pi **/** 2**)** **-** theta1

522 theta2 **=** asin**((**sin**(**theta1**)** **\*** n2**))**

523 newm1 **=** tan**(**t2**)**

524 **if** initialsource**.**re **==** "side"**:**

525 **if** self**.**m **>** 0**:**

526 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 90**,** 270**,** "-"**,**

527 initialsource**.**defined\_name**)**

528

529 **elif** self**.**m **<** 0 **and** coord**[**2**]** **==** source**.**yp**:**

530 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 90**,** 270**,** "-"**,**

531 initialsource**.**defined\_name**)**

532 initialsource**.**re **=** "side"

533

534 **elif** self**.**m **<** 0 **and** coord**[**2**]** **==** source**.**yp **+** source**.**length**:**

535 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 270**,** 90**,** "+"**,**

536 initialsource**.**defined\_name**)**

537 initialsource**.**re **=** "side2"

538

539

540 **elif** self**.**m **<** 0**:**

541 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** coord**,** 90**,** 270**,** "+"**,**

542 initialsource**.**defined\_name**)**

543

544 **elif** self**.**m **>** 0**:**

545 theta2 **=** **(**math**.**pi **/** 2**)** **-** theta2

546 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** coord**,** 90**,** 270**,** "+"**,**

547 initialsource**.**defined\_name**)**

548 Tag **=** tag\_generator**(**tag**,** "S"**)**

549 refracted\_ray **=** Source**(**Tag**)**

550 source**.**interceptors**.**append**(**refracted\_ray**)**

551 IncidentRays**[**initialsource**.**defined\_name**].**append**(**refracted\_ray**)**

552

553 refracted\_ray**.**state **=** **True**

554 refracted\_ray**.**xp **=** coord**[**1**]**

555 refracted\_ray**.**yp **=** coord**[**2**]**

556 refracted\_ray**.**findupdate**(**coord**[**1**],** coord**[**2**])**

557 refracted\_ray**.**blackout**()**

558 **if** initialsource**.**re **==** "pos"**:**

559 refracted\_ray**.**m **=** newm1

560 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

561 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

562 **elif** initialsource**.**re **==** "neg"**:**

563 refracted\_ray**.**m **=** newm1

564 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

565 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

566 **elif** initialsource**.**re **==** "side"**:**

567 refracted\_ray**.**m **=** newm1

568 refracted\_ray**.**width **=** **-**10000

569 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

570 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

571

572 **elif** initialsource**.**re **==** "side2"**:**

573 refracted\_ray**.**m **=** newm1

574 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

575 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(-**self**.**width**)))**

576 refracted\_ray**.**endx **=** **-**10000

577

578 **elif** initialsource**.**re **is** **None:**

579 **if** self**.**m **>** 0**:**

580 refracted\_ray**.**m **=** newm1

581 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

582 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

583 **else:**

584 refracted\_ray**.**m **=** **-**newm1

585 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

586 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

587 initialsource**.**previousobj**.**append**(**source**)**

588 refracted\_ray**.**Simulation**(**self**.**state**,** initialsource**,** obj**)**

589 **else:**

590 source**.**redraw1**(**m**)**

591 **else:**

592 **if** theta1 **>** critical**:**

593 theta1 **=** theta1

594 theta2 **=** theta1

595

596 **if** self**.**m **>** 0 **and** coord**[**3**]** **==** "side"**:**

597 self**.**anglesfunc**(**theta1**,** theta1**,** "horizontal"**,** coord**,** 180**,** 180**,** "reflecting"**,**

598 initialsource**.**defined\_name**)**

599 initialsource**.**re **=** "side"

600

601 Tag **=** tag\_generator**(**tag**,** "S"**)**

602 reflected\_ray **=** Source**(**Tag**)**

603 source**.**interceptors**.**append**(**reflected\_ray**)**

604 IncidentRays**[**initialsource**.**defined\_name**].**append**(**reflected\_ray**)**

605 reflected\_ray**.**m **=** **-**self**.**m

606 reflected\_ray**.**state **=** **True**

607 reflected\_ray**.**xp **=** coord**[**1**]**

608 reflected\_ray**.**yp **=** coord**[**2**]**

609 reflected\_ray**.**findupdate**(**coord**[**1**],** coord**[**2**])**

610 reflected\_ray**.**blackout**()**

611 eq **=** reflected\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**reflected\_ray**.**m**))**

612 reflected\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

613 reflected\_ray**.**InsideSimulation**(**source**,** initialsource**,** obj**)**

614 **elif** theta1 **<** critical **and** theta1 **!=** 0**:**

615 self**.**endx **=** coord**[**1**]**

616 self**.**endy **=** coord**[**2**]**

617 theta2 **=** asin**((**sin**(**theta1**)** **\*** n2**))**

618 newm1 **=** tan**(**theta2**)**

619 **if** initialsource**.**re **==** "pos"**:**

620 **if** self**.**m **<** 0**:**

621 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse2"**,** coord**,** 180**,** 0**,** "+"**,**

622 initialsource**.**defined\_name**)**

623 **else:**

624 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 180**,** 0**,** "+"**,**

625 initialsource**.**defined\_name**)**

626 **elif** initialsource**.**re **==** "neg"**:**

627 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 180**,** 0**,** "-"**,**

628 initialsource**.**defined\_name**)**

629

630 **elif** self**.**m **<** 0 **and** coord**[**3**]** **==** "side"**:**

631 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 180**,** 0**,** "-"**,**

632 initialsource**.**defined\_name**)**

633

634 **elif** self**.**m **>** 0 **and** coord**[**3**]** **==** "side"**:**

635 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** coord**,** 180**,** 0**,** "-"**,**

636 initialsource**.**defined\_name**)**

637

638 Tag **=** tag\_generator**(**tag**,** "S"**)**

639 refracted\_ray **=** Source**(**Tag**)**

640 source**.**interceptors**.**append**(**refracted\_ray**)**

641 IncidentRays**[**initialsource**.**defined\_name**].**append**(**refracted\_ray**)**

642

643 refracted\_ray**.**state **=** **True**

644 refracted\_ray**.**xp **=** coord**[**1**]**

645 refracted\_ray**.**yp **=** coord**[**2**]**

646 refracted\_ray**.**findupdate**(**coord**[**1**],** coord**[**2**])**

647 refracted\_ray**.**blackout**()**

648 **if** initialsource**.**re **==** "pos"**:**

649 **if** self**.**m **>** 0**:**

650 refracted\_ray**.**m **=** newm1

651 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

652 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

653 **else:**

654 refracted\_ray**.**m **=** newm1

655 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

656 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

657 **elif** initialsource**.**re **==** "neg"**:**

658 refracted\_ray**.**m **=** **-**newm1

659 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

660 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

661 **elif** initialsource**.**re **is** **None:**

662 **if** self**.**m **>** 0**:**

663 refracted\_ray**.**m **=** **-**newm1

664 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

665 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

666 **else:**

667 refracted\_ray**.**m **=** **-**newm1

668 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

669 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

670 initialsource**.**previousobj**.**append**(**source**)**

671 refracted\_ray**.**Simulation**(**self**.**state**,** initialsource**,** obj**)**

672 **else:**

673 source**.**redraw1**(**m**)**

674

675 **return** self**.**endx**,** self**.**endy

676

677 ##### Simulation #######

678 # Parameters :- state:Boolean, previous:Memory Address of Object, obj:Boolean

679 # Return Type :- self.endx:float, self.endy:float

680 # Purpose :- Checks if any sources loaded interacts with the object and manipulates

681 # the source dependant on the objects properties

682 ###########################

683 **def** Simulation**(**self**,** state**,** previous**,** obj**):**

684 coord **=** **[]**

685 intercept\_x **=** **[]**

686 intercept\_y **=** **[]**

687 **if** state**:**

688 previous**.**re **=** **None**

689 **for** i **in** objects\_loaded**:**

690 **if** state**:**

691 **if** i **in** previous**.**previousobj**:**

692 **continue**

693 **if** "SC" **in** i**.**defined\_name**:**

694 **if** self**.**m **==** 0**:**

695 **continue**

696 **else:**

697 dist **=** math**.**sqrt**((**i**.**xp **-** self**.**xp**)** **\*\*** 2 **+** **(**i**.**yp **-** self**.**yp**)** **\*\*** 2**)**

698 x**,** y **=** sympy**.**symbols**(**"x,y"**)**

699 ep **=** self**.**findy**.**replace**(**"m"**,** **str(**self**.**m**))**

700 ep **=** i**.**Equation**.**replace**(**"y"**,** "{0}"**.format(**ep**))**

701 sol1 **=** sympy**.**solve**(eval(**ep**),** x**)**

702 sol **=** **[]**

703 intery **=** **[]**

704 **for** p **in** sol1**:**

705 **try:**

706 **if** p **<=** self**.**xp**:**

707 **continue**

708 **else:**

709 sol**.**append**(**p**)**

710 **except** **TypeError:**

711 **continue**

712

713 **if** **not** sol**:**

714 self**.**redraw1**()**

715 **else:**

716 **for** point **in** sol**:**

717 ep **=** self**.**findy**.**replace**(**"m"**,** **str(**self**.**m**))**

718 ycoord **=** **eval(**ep**.**replace**(**"x"**,** **str(**point**)))**

719 intery**.**append**([**point**,** ycoord**])**

720 interyp **=** **[]**

721 **for** pair **in** intery**:**

722 **if** pair**[**1**]** **<=** i**.**yp**:**

723 interyp**.**append**(**pair**)**

724 **else:**

725 intery**.**remove**(**pair**)**

726 **for** pair **in** interyp**:**

727 coord**.**append**([**i**,** pair**[**0**],** pair**[**1**]])**

728 ep **=** self**.**findx**.**replace**(**"m"**,** **str(**self**.**m**))**

729 xcoord **=** **eval(**ep**.**replace**(**"y"**,** **str(**i**.**yp**)))**

730 **if** i**.**xp **-** i**.**Radius **<=** xcoord **<=** i**.**xp **+** i**.**Radius**:**

731 coord**.**append**([**i**,** xcoord**,** i**.**yp**])**

732

733

734 **else:**

735 **if** self**.**m **==** 0**:**

736 **pass**

737 **else:**

738

739 eq **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

740 xcoord **=** **eval(**eq**.**replace**(**"y"**,** "{0}"**.format(**i**.**yp**)))**

741 intercept\_x**.**append**((**xcoord**,** i**.**yp**))**

742 eq3 **=** self**.**findx**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

743 xcoord **=** **eval(**eq3**.**replace**(**"y"**,** "{0}"**.format(**i**.**yp **+** i**.**length**)))**

744 intercept\_x**.**append**((**xcoord**,** i**.**yp **+** i**.**length**))**

745 eq2 **=** self**.**findy**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

746 ycoord **=** **eval(**eq2**.**replace**(**"x"**,** "{0}"**.format(**i**.**xp**)))**

747 intercept\_y**.**append**((**i**.**xp**,** ycoord**))**

748 eq4 **=** self**.**findy**.**replace**(**"m"**,** "{0}"**.format(**self**.**m**))**

749 ycoord **=** **eval(**eq4**.**replace**(**"x"**,** "{0}"**.format(**i**.**xp **+** i**.**width**)))**

750 intercept\_y**.**append**((**i**.**xp **+** i**.**width**,** ycoord**))**

751 intercept\_x **=** **list(dict.**fromkeys**(**intercept\_x**))**

752 intercept\_y **=** **list(dict.**fromkeys**(**intercept\_y**))**

753

754 **for** co **in** intercept\_x**:**

755 **if** **not** state**:**

756 **if** i**.**xp **<=** co**[**0**]** **<=** i**.**xp **+** i**.**width **and** co**[**0**]** **>** self**.**xp**:**

757 coord**.**append**([**i**,** co**[**0**],** co**[**1**],** "top"**])**

758 **else:**

759 **pass**

760 **else:**

761 **if** i**.**xp **<=** co**[**0**]** **<=** i**.**xp **+** i**.**width **and** co**[**0**]** **!=** self**.**xp **and** co**[**0**]** **>** self**.**xp**:**

762 coord**.**append**([**i**,** co**[**0**],** co**[**1**],** "top"**])**

763 **else:**

764 **pass**

765 **for** co **in** intercept\_y**:**

766 **if** i**.**yp **<=** co**[**1**]** **<=** i**.**yp **+** i**.**length **and** co**[**1**]** **!=** self**.**yp**:**

767 coord**.**append**([**i**,** co**[**0**],** co**[**1**],** "side"**])**

768 **else:**

769 **pass**

770

771 **if** **not** coord**:**

772 self**.**statechange**()**

773 **else:**

774 **if** previous **is** **not** **None:**

775 **if** **not** previous**.**previousobj**:**

776 **for** i **in** coord**:**

777 **if** i**[**0**]** **in** previous**.**previousobj**:**

778 coord**.**remove**(**i**)**

779

780 **for** i **in** coord**:**

781 distance **=** math**.**sqrt**(((**i**[**1**]** **-** self**.**xp**)** **\*\*** 2**)** **+** **((**i**[**2**]** **-** self**.**yp**)** **\*\*** 2**))**

782

783 in1 **=** coord**.**index**(**i**)**

784

785 coord**[**in1**].**append**(**distance**)**

786

787 coord **=** **sorted(**coord**,** key**=lambda** x**:** x**[-**1**])**

788 condition **=** **True**

789 **if** "SC" **in** coord**[**0**][**0**].**defined\_name**:**

790 **if** coord**[**0**][**2**]** **!=** coord**[**0**][**0**].**yp **and** coord**[**1**][**2**]** **!=** coord**[**0**][**0**].**yp**:**

791 condition **=** **False**

792 self**.**statechange**()**

793 **elif** coord**[**0**][**2**]** **<** coord**[**0**][**0**].**yp**:**

794 coord **=** **[**coord**[**1**]]**

795

796 **else:**

797 coord **=** **[**coord**[**0**]]**

798 **else:**

799 coord **=** **[**coord**[**0**]]**

800 origin **=** "temp"

801 **if** condition**:**

802 **for** sources **in** sources\_loaded**:**

803 **if** sources**.**defined\_name **==** self**.**defined\_name**:**

804 origin **=** sources

805

806 **for** source **in** coord**:**

807 **if** previous **is** **not** **None:**

808 previous**.**previousobj**.**append**(**source**[**0**])**

809 self**.**endx **=** source**[**1**]**

810 self**.**endy **=** source**[**2**]**

811

812 **if** source**[**0**].**Property **==** "Stop"**:**

813 self**.**endx **=** source**[**1**]**

814 self**.**endy **=** source**[**2**]**

815 self**.**redraw1**()**

816 **elif** source**[**0**].**Property **==** "Reflective"**:**

817 Tag **=** tag\_generator**(**tag**,** "S"**)**

818 reflected\_ray **=** Source**(**Tag**)**

819 **if** **not** state**:**

820 IncidentRays**[**self**.**defined\_name**].**append**(**reflected\_ray**)**

821 **else:**

822 IncidentRays**[**previous**.**defined\_name**].**append**(**reflected\_ray**)**

823 **if** obj**:**

824 source**[**0**].**interceptors**.**append**(**origin**)**

825 source**[**0**].**interceptors**.**append**(**reflected\_ray**)**

826 **else:**

827 source**[**0**].**interceptors**.**append**(**reflected\_ray**)**

828 IncidentRays**[**self**.**defined\_name**].**append**(**reflected\_ray**)**

829 self**.**endx **=** source**[**1**]**

830 self**.**endy **=** source**[**2**]**

831 reflected\_ray**.**m **=** **-**self**.**m

832 reflected\_ray**.**state **=** **True**

833 reflected\_ray**.**xp **=** source**[**1**]**

834 reflected\_ray**.**yp **=** source**[**2**]**

835 reflected\_ray**.**blackout**()**

836 reflected\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

837 eq **=** reflected\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**reflected\_ray**.**m**))**

838 reflected\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

839 reflected\_ray**.**endx **=** self**.**width

840

841 **if** state**:**

842 reflected\_ray**.**Simulation**(True,** previous**,** obj**)**

843 **else:**

844 reflected\_ray**.**Simulation**(True,** origin**,** obj**)**

845 **elif** source**[**0**].**Property **!=** "Reflective" **or** source**[**0**].**Property **!=** "Stop" **or** "D" **not** **in** source**[**

846 0**].**defined\_name**:**

847 n2 **=** source**[**0**].**Refractive\_Index

848 m1 **=** self**.**m

849 **if** self**.**m **==** 0**:**

850 self**.**redraw1**()**

851 **else:**

852 **if** "SC" **in** source**[**0**].**defined\_name**:**

853 **if** state**:**

854 previous**.**previousobj**.**append**(**source**[**0**])**

855 **else:**

856 self**.**previousobj**.**append**(**source**[**0**])**

857 m2 **=** 0

858 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m2 **\*** m1**)))**

859 theta2 **=** asin**(**sin**(**theta1**)** **/** n2**)**

860 critical **=** asin**(**1 **/** n2**)**

861 **if** state**:**

862 self**.**normalfunc**(**"top"**,** previous**,** source**)**

863 **else:**

864 self**.**normalfunc**(**"top"**,** origin**,** source**)**

865 theta2 **=** **(**math**.**pi **/** 2**)** **-** theta2

866

867 **if** theta1 **!=** 0 **and** theta1 **<** critical**:**

868 self**.**endx **=** source**[**1**]**

869 self**.**endy **=** source**[**2**]**

870 m3 **=** tan**(**theta2**)**

871

872 Tag **=** tag\_generator**(**tag**,** "S"**)**

873 refracted\_ray **=** Source**(**Tag**)**

874 **if** state **==** 0**:**

875 **if** obj**:**

876 source**[**0**].**interceptors**.**append**(**origin**)**

877 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

878 **else:**

879 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

880 IncidentRays**[**self**.**defined\_name**].**append**(**refracted\_ray**)**

881 **else:**

882 IncidentRays**[**previous**.**defined\_name**].**append**(**refracted\_ray**)**

883

884 **if** self**.**m **<** 0**:**

885 theta2 **=** math**.**pi **/** 2 **-** theta2

886 theta1 **=** math**.**pi **/** 2 **-** theta1

887 **if** state**:**

888 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "+"**,**

889 previous**.**defined\_name**)**

890 **else:**

891 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "-"**,**

892 self**.**defined\_name**)**

893 refracted\_ray**.**m **=** **-**m3

894 refracted\_ray**.**state **=** **True**

895 refracted\_ray**.**xp **=** source**[**1**]**

896 refracted\_ray**.**yp **=** source**[**2**]**

897 refracted\_ray**.**blackout**()**

898 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

899 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

900 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

901 **if** state**:**

902 refracted\_ray**.**Simulation**(True,** previous**,** obj**)**

903 **else:**

904 refracted\_ray**.**Simulation**(True,** origin**,** obj**)**

905 **else:**

906 theta2 **=** theta2 **+** math**.**pi **/** 2

907 m3 **=** tan**(**theta2**)**

908 theta1 **=** theta1 **-** math**.**pi **/** 2

909 theta2 **=** theta2 **-** math**.**pi **/** 2

910 **if** state**:**

911 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 270**,** 90**,** "+"**,**

912 previous**.**defined\_name**)**

913 **else:**

914 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 270**,** 90**,** "+"**,**

915 self**.**defined\_name**)**

916 refracted\_ray**.**m **=** **-**m3

917 refracted\_ray**.**state **=** **True**

918 refracted\_ray**.**xp **=** source**[**1**]**

919 refracted\_ray**.**yp **=** source**[**2**]**

920 refracted\_ray**.**blackout**()**

921 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

922 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

923 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

924 **if** state**:**

925 refracted\_ray**.**Simulation**(True,** previous**,** obj**)**

926 **else:**

927 refracted\_ray**.**Simulation**(True,** origin**,** obj**)**

928 **elif** theta1 **!=** 0 **and** theta1 **>** critical **and** self**.**m **<** 0**:**

929 theta1 **=** math**.**pi **/** 2 **-** theta1

930 theta2 **=** **(**math**.**pi **/** 2**)** **-** theta2

931 **if** state**:**

932 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "+"**,**

933 previous**.**defined\_name**)**

934 **else:**

935 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "+"**,**

936 self**.**defined\_name**)**

937 theta2 **=** **(**math**.**pi **/** 2 **-** theta2**)** **-** **(**math**.**pi **/** 2**)**

938 m3 **=** tan**(**theta2 **+** math**.**pi **/** 2**)**

939 Tag **=** tag\_generator**(**tag**,** "S"**)**

940 refracted\_ray **=** Source**(**Tag**)**

941 **if** **not** state**:**

942 **if** obj**:**

943 source**[**0**].**interceptors**.**append**(**origin**)**

944 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

945 **else:**

946 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

947 IncidentRays**[**self**.**defined\_name**].**append**(**refracted\_ray**)**

948 **else:**

949 IncidentRays**[**previous**.**defined\_name**].**append**(**refracted\_ray**)**

950

951 refracted\_ray**.**m **=** **-**m3

952 refracted\_ray**.**state **=** **True**

953 refracted\_ray**.**xp **=** source**[**1**]**

954 refracted\_ray**.**yp **=** source**[**2**]**

955 refracted\_ray**.**blackout**()**

956 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

957 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

958 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

959 **if** state**:**

960 refracted\_ray**.**Simulation**(True,** previous**,** obj**)**

961 **else:**

962 refracted\_ray**.**Simulation**(True,** origin**,** obj**)**

963 **elif** theta1 **!=** 0 **and** theta1 **>** critical **and** self**.**m **>** 0**:**

964 **if** state**:**

965

966 self**.**anglesfunc**(**theta1**,** theta1**,** "vertical"**,** source**,** 270**,** 270**,** "reflecting"**,**

967 previous**.**defined\_name**)**

968 **else:**

969 self**.**anglesfunc**(**theta1**,** theta1**,** "vertical"**,** source**,** 270**,** 270**,** "reflecting"**,**

970 self**.**defined\_name**)**

971

972 Tag **=** tag\_generator**(**tag**,** "S"**)**

973 reflected\_ray **=** Source**(**Tag**)**

974 **if** **not** state**:**

975 **if** obj**:**

976 source**[**0**].**interceptors**.**append**(**origin**)**

977 source**[**0**].**interceptors**.**append**(**reflected\_ray**)**

978 **else:**

979 source**[**0**].**interceptors**.**append**(**reflected\_ray**)**

980 IncidentRays**[**self**.**defined\_name**].**append**(**reflected\_ray**)**

981 **else:**

982 IncidentRays**[**previous**.**defined\_name**].**append**(**reflected\_ray**)**

983

984 reflected\_ray**.**m **=** **-**self**.**m

985 reflected\_ray**.**state **=** **True**

986 reflected\_ray**.**xp **=** source**[**1**]**

987 reflected\_ray**.**yp **=** source**[**2**]**

988 reflected\_ray**.**blackout**()**

989 eq **=** reflected\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**reflected\_ray**.**m**))**

990 reflected\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

991 **if** state**:**

992 reflected\_ray**.**Simulation**(True,** previous**,** obj**)**

993 **else:**

994 reflected\_ray**.**Simulation**(True,** origin**,** obj**)**

995 **elif** theta1 **!=** 0 **and** self**.**m **==** 0**:**

996 **if** state**:**

997 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 90**,** 270**,** "+"**,**

998 previous**.**defined\_name**)**

999 **else:**

1000 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 90**,** 270**,** "+"**,**

1001 self**.**defined\_name**)**

1002

1003 self**.**endx **=** source**[**1**]**

1004 self**.**endy **=** source**[**2**]**

1005 m3 **=** tan**(**theta2**)**

1006 Tag **=** tag\_generator**(**tag**,** "S"**)**

1007 refracted\_ray **=** Source**(**Tag**)**

1008 **if** **not** state**:**

1009 **if** obj**:**

1010 source**[**0**].**interceptors**.**append**(**origin**)**

1011 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1012 **else:**

1013 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1014 IncidentRays**[**self**.**defined\_name**].**append**(**refracted\_ray**)**

1015 **else:**

1016 IncidentRays**[**previous**.**defined\_name**].**append**(**refracted\_ray**)**

1017

1018 refracted\_ray**.**m **=** **-**m3

1019 refracted\_ray**.**state **=** **True**

1020 refracted\_ray**.**xp **=** source**[**1**]**

1021 refracted\_ray**.**yp **=** source**[**2**]**

1022 refracted\_ray**.**blackout**()**

1023 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

1024 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

1025 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1026 **if** state**:**

1027 refracted\_ray**.**Simulation**(True,** previous**,** obj**)**

1028 **else:**

1029 refracted\_ray**.**Simulation**(True,** origin**,** obj**)**

1030 **elif** theta1 **!=** 0 **and** theta1 **==** critical**:**

1031 **if** state**:**

1032 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 90**,** 270**,** "+"**,**

1033 previous**.**defined\_name**)**

1034 **else:**

1035 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 90**,** 270**,** "+"**,**

1036 self**.**defined\_name**)**

1037 self**.**endx **=** source**[**1**]**

1038 self**.**endy **=** source**[**2**]**

1039 Tag **=** tag\_generator**(**tag**,** "S"**)**

1040 refracted\_ray **=** Source**(**Tag**)**

1041 **if** **not** state**:**

1042 **if** obj**:**

1043 source**[**0**].**interceptors**.**append**(**origin**)**

1044 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1045 **else:**

1046 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1047 IncidentRays**[**self**.**defined\_name**].**append**(**refracted\_ray**)**

1048 **else:**

1049 IncidentRays**[**previous**.**defined\_name**].**append**(**refracted\_ray**)**

1050

1051 refracted\_ray**.**m **=** 0

1052 refracted\_ray**.**state **=** **True**

1053 refracted\_ray**.**xp **=** source**[**1**]**

1054 refracted\_ray**.**yp **=** source**[**2**]**

1055 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

1056 refracted\_ray**.**blackout**()**

1057 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

1058 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1059 **if** state**:**

1060 refracted\_ray**.**Simulation**(True,** previous**,** obj**)**

1061 **else:**

1062 refracted\_ray**.**Simulation**(True,** origin**,** obj**)**

1063 **else:**

1064

1065 self**.**redraw1**()**

1066

1067 **else:**

1068

1069 **if** source**[**3**]** **==** "top"**:**

1070 **if** state**:**

1071 self**.**normalfunc**(**"top"**,** previous**,** source**)**

1072 **else:**

1073 self**.**normalfunc**(**"top"**,** origin**,** source**)**

1074 **else:**

1075 **if** state**:**

1076 self**.**normalfunc**(**"side"**,** previous**,** source**)**

1077 **else:**

1078 self**.**normalfunc**(**"side"**,** origin**,** source**)**

1079 m2 **=** self**.**m

1080 m1 **=** 0

1081 **if** source**[**3**]** **==** "top" **and** self**.**m **>** 0**:**

1082 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m1 **\*** m2**)))**

1083 theta1 **=** **(**math**.**pi **/** 2**)** **-** theta1

1084 theta2 **=** asin**((**sin**(**theta1**)** **/** n2**))**

1085 newm1 **=** tan**((**math**.**pi **/** 2**)** **-** theta2**)**

1086

1087 **elif** source**[**3**]** **==** "top" **and** self**.**m **<** 0**:**

1088 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m1 **\*** m2**)))**

1089 theta2 **=** asin**((**sin**(**theta1**)** **/** n2**))**

1090 theta1 **=** **(**math**.**pi **/** 2**)** **-** theta1

1091 theta2 **=** **(**math**.**pi **/** 2**)** **-** theta2

1092 newm1 **=** tan**(**theta2**)**

1093

1094 **else:**

1095 theta1 **=** atan**(abs((**m2 **-** m1**)** **/** **(**1 **+** m1 **\*** m2**)))**

1096 theta2 **=** asin**((**sin**(**theta1**)** **/** n2**))**

1097 newm1 **=** tan**(**theta2**)**

1098

1099 **if** theta1 **!=** 0**:**

1100 **if** self**.**m **>** 0 **and** source**[**3**]** **==** "side"**:**

1101 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse2"**,** source**,** 180**,** 0**,** "+"**,**

1102 self**.**defined\_name**)**

1103

1104 **elif** self**.**m **<** 0 **and** source**[**3**]** **==** "side"**:**

1105 **if** state**:**

1106 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** source**,** 180**,** 0**,** "-"**,**

1107 previous**.**defined\_name**)**

1108 **else:**

1109 self**.**anglesfunc**(**theta1**,** theta2**,** "horizontal"**,** source**,** 180**,** 0**,** "-"**,**

1110 self**.**defined\_name**)**

1111

1112 **elif** self**.**m **>** 0 **and** source**[**3**]** **==** "top"**:**

1113 **if** state**:**

1114 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 270**,** 90**,** "-"**,**

1115 previous**.**defined\_name**)**

1116 **else:**

1117 self**.**anglesfunc**(**theta1**,** theta2**,** "vertical"**,** source**,** 270**,** 90**,** "-"**,**

1118 self**.**defined\_name**)**

1119

1120 **elif** self**.**m **<** 0 **and** source**[**3**]** **==** "top"**:**

1121 theta2 **=** math**.**pi **/** 2 **-** theta2

1122 **if** state**:**

1123 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "+"**,**

1124 previous**.**defined\_name**)**

1125 **else:**

1126 self**.**anglesfunc**(**theta1**,** theta2**,** "reverse"**,** source**,** 90**,** 270**,** "+"**,**

1127 self**.**defined\_name**)**

1128

1129 Tag **=** tag\_generator**(**tag**,** "S"**)**

1130 refracted\_ray **=** Source**(**Tag**)**

1131

1132 **if** **not** state**:**

1133 **if** obj**:**

1134 source**[**0**].**interceptors**.**append**(**origin**)**

1135 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1136 **else:**

1137 source**[**0**].**interceptors**.**append**(**refracted\_ray**)**

1138

1139 IncidentRays**[**self**.**defined\_name**].**append**(**refracted\_ray**)**

1140

1141 **else:**

1142 IncidentRays**[**previous**.**defined\_name**].**append**(**refracted\_ray**)**

1143

1144 refracted\_ray**.**xp **=** source**[**1**]**

1145 refracted\_ray**.**yp **=** source**[**2**]**

1146

1147 refracted\_ray**.**findupdate**(**source**[**1**],** source**[**2**])**

1148

1149 refracted\_ray**.**state **=** **True**

1150 **if** self**.**m **<** 0**:**

1151 **if** **not** state**:**

1152 refracted\_ray**.**m **=** newm1

1153 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

1154 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1155 **else:**

1156 refracted\_ray**.**m **=** **-**newm1

1157 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(**refracted\_ray**.**m**))**

1158 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1159

1160 **else:**

1161 **if** state**:**

1162 refracted\_ray**.**m **=** **-**newm1

1163 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

1164 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1165 **else:**

1166 refracted\_ray**.**m **=** newm1

1167 eq **=** refracted\_ray**.**findy**.**replace**(**"m"**,** "{0}"**.format(-**refracted\_ray**.**m**))**

1168 refracted\_ray**.**endy **=** **eval(**eq**.**replace**(**"x"**,** "{0}"**.format(**self**.**width**)))**

1169 self**.**findupdate**(**source**[**1**],** source**[**2**])**

1170 **if** state**:**

1171 refracted\_ray**.**InsideSimulation**(**source**[**0**],** previous**,** obj**)**

1172 **else:**

1173 refracted\_ray**.**InsideSimulation**(**source**[**0**],** origin**,** obj**)**

1174

1175 **else:**

1176 self**.**redraw1**()**

1177 self**.**redraw2**()**

1178 **if** previous **is** **not** **None:**

1179 previous**.**redraw2**()**

1180 **return** self**.**endx**,** self**.**endy

1181

1182 ##### statechange #######

1183 # Parameters :- None

1184 # Return Type :- self.endy:float, self.endx:float, self.graphical:function

1185 # Purpose :- changes state and increases the end of line when the source is no longer interacting with any object

1186 ###########################

1187 **def** statechange**(**self**):**

1188 self**.**state **=** **False**

1189 self**.**endx **=** self**.**width

1190 new\_y **=** self**.**m **\*** **(**self**.**width **-** self**.**xp**)** **+** self**.**yp

1191 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,** **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** new\_y**),** self**.**thickness**)**

1192 self**.**endy **=** **int(**new\_y**)**

1193 **return** self**.**endy**,** self**.**endx**,** self**.**graphical

1194

1195 ##### redraw #######

1196 # Parameters :- x:Int, y:Int

1197 # Return Type :- Int, Object, List

1198 # Purpose :- To move an objects to its new position dependant on the parameter x and y by

1199 # filling the screen and redraw all loaded objects and instances new postion and whether it can be moved

1200 ###########################

1201 **def** redraw**(**self**,** x**,** y**):**

1202 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1203 **for** i **in** objects\_loaded**:**

1204 **if** "D" **not** **in** i**.**defined\_name**:**

1205 i**.**redraw1**()**

1206 **for** i **in** sources\_loaded**:**

1207 **if** i**.**defined\_name **!=** self**.**defined\_name**:**

1208 i**.**redraw1**()**

1209 new\_y **=** self**.**m **\*** **(**self**.**endx **-** x**)** **+** y

1210 start\_y **=** self**.**m **\*** **((**x **+** 10**)** **-** x**)** **+** y

1211 end\_y **=** self**.**m **\*** **((**x **+** 20**)** **-** x**)** **+** y

1212 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,** **(**x**,** y**),** **(**self**.**endx**,** new\_y**),** self**.**thickness**)**

1213 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,** **(**x **+** 10**,** start\_y**),** **(**x **+** 20**,** end\_y**),** self**.**thickness**)**

1214 self**.**redx **=** x **+** 10

1215 self**.**rendx **=** x **+** 20

1216 self**.**rendy **=** end\_y

1217 self**.**redy **=** start\_y

1218 self**.**xp **=** x

1219 self**.**yp **=** y

1220 self**.**endy **=** new\_y

1221 self**.**m **=** **((**self**.**yp **-** self**.**endy**)** **/** **(**self**.**xp **-** self**.**endx**))**

1222 self**.**findupdate**(**x**,** y**)**

1223 IncidentRays**[**self**.**defined\_name**]** **=** **[]**

1224 self**.**normals **=** **[]**

1225 self**.**angles **=** **[]**

1226 self**.**previousobj **=** **[]**

1227 self**.**re **=** **None**

1228 **return** self**.**graphical**,** self**.**xp**,** self**.**yp**,** self**.**redy**,** self**.**redx**,** self**.**rendy**,** self**.**rendx**,** self**.**endy**,** self**.**m

1229

1230 ##### redraw1 #######

1231 # Parameters :- None

1232 # Return Type :- None

1233 # Purpose :- updates instance of object on screen

1234 ###########################

1235 **def** redraw1**(**self**):**

1236 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,** **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** self**.**endy**),**

1237 self**.**thickness**)**

1238 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,** **(**self**.**redx**,** self**.**redy**),** **(**self**.**rendx**,** self**.**rendy**),**

1239 self**.**thickness**)**

1240 **for** i **in** IncidentRays**[**self**.**defined\_name**]:**

1241 i**.**redraw1**()**

1242 **for** i **in** self**.**normals**:**

1243 i**.**redraw1**()**

1244 **for** i **in** self**.**angles**:**

1245 i**.**redraw1**()**

1246

1247 ##### rotate #######

1248 # Parameters :- mousex:float, mousey:float

1249 # Return Type :- self.graphical:function, self.graphical2:function, self.endy:float,

1250 # self.redy:float, self.rendy:float, self.m:float, self.stateproduct:list

1251 # Purpose :- rotates the source when red button drag

1252 ###########################

1253 **def** rotate**(**self**,** mousex**,** mousey**):**

1254 angle\_to\_pointer **=** math**.**degrees**(**math**.**atan2**(**self**.**yp **-** mousey**,** self**.**xp **-** mousex**))**

1255 delta\_x **=** mousex **-** self**.**redx

1256 delta\_y **=** mousey **-** self**.**redy

1257 angle **=** math**.**atan2**(**delta\_y**,** delta\_x**)**

1258 angle **=** **(**angle **\*** 180**)** **/** math**.**pi

1259 new **=** pygame**.**math**.**Vector2**(**self**.**redx**,** self**.**redy**)** **+** pygame**.**math**.**Vector2**(**self**.**endx**,** self**.**endy**).**rotate**(**angle**)**

1260 null**,** new\_y **=** new

1261 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1262 **for** i **in** objects\_loaded**:**

1263 i**.**redraw1**()**

1264 **for** i **in** sources\_loaded**:**

1265 **if** i**.**defined\_name **!=** self**.**defined\_name**:**

1266 i**.**redraw1**()**

1267 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,** **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** new\_y**),** self**.**thickness**)**

1268 self**.**m **=** **((**self**.**yp **-** new\_y**)** **/** **(**self**.**xp **-** self**.**endx**))**

1269 start\_y **=** self**.**m **\*** **((**self**.**xp **+** 10**)** **-** self**.**xp**)** **+** self**.**yp

1270 end\_y **=** self**.**m **\*** **((**self**.**xp **+** 20**)** **-** self**.**xp**)** **+** self**.**yp

1271 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,** **(**self**.**xp **+** 10**,** start\_y**),** **(**self**.**xp **+** 20**,** end\_y**),**

1272 self**.**thickness**)**

1273 self**.**redy **=** start\_y

1274 self**.**rendy **=** end\_y

1275 self**.**endy **=** new\_y

1276 self**.**findupdate**(**self**.**xp**,** self**.**yp**)**

1277 IncidentRays**[**self**.**defined\_name**]** **=** **[]**

1278 self**.**normals **=** **[]**

1279 self**.**angles **=** **[]**

1280 self**.**previousobj **=** **[]**

1281 self**.**re **=** **None**

1282 **return** self**.**graphical**,** self**.**graphical2**,** self**.**endy**,** self**.**redy**,** self**.**rendy**,** self**.**m

1283

1284

1285##### Block #######

1286# Inheirts :- Main\_Objects

1287# Parameters :- name:String

1288# Purpose :- Holds every attribute and operation that can be done on a block object

1289###########################

1290class Block**(**Main\_Objects**):**

1291 **def** \_\_init\_\_**(**self**,** name**):**

1292 **super().**\_\_init\_\_**(**name**)**

1293 self**.**colour **=** **(**255**,** 105**,** 105**)**

1294 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1295 self**.**full\_name **=** "Block" **+** **str(**self**.**defined\_name**[-**1**])**

1296 self**.**Refractive\_Index **=** 1.7

1297

1298 ##### redraw1 #######

1299 # Parameters :- None

1300 # Return Type :- None

1301 # Purpose :- updates instance of object on screen

1302 ###########################

1303 **def** redraw1**(**self**):**

1304 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

1305 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1306 self**.**update**()**

1307

1308

1309##### GlassBlock #######

1310# Inheirts :- Main\_Objects

1311# Parameters :- name:String

1312# Purpose :- Holds every attribute and operation that can be done on a glassblock object

1313###########################

1314class GlassBlock**(**Main\_Objects**):**

1315 **def** \_\_init\_\_**(**self**,** name**):**

1316 **super().**\_\_init\_\_**(**name**)**

1317 self**.**Refractive\_Index **=** 1.52

1318 self**.**colour **=** **(**168**,** 204**,** 215**)**

1319 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1320 self**.**full\_name **=** "GlassBlock" **+** **str(**self**.**defined\_name**[-**1**])**

1321

1322 ##### redraw1 #######

1323 # Parameters :- None

1324 # Return Type :- None

1325 # Purpose :- updates instance of object on screen

1326 ###########################

1327 **def** redraw1**(**self**):**

1328 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

1329 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1330 self**.**update**()**

1331

1332

1333##### Mirror #######

1334# Inheirts :- Main\_Objects

1335# Parameters :- name:String

1336# Purpose :- Holds every attribute and operation that can be done on a mirror object

1337###########################

1338class Mirror**(**Main\_Objects**):**

1339 **def** \_\_init\_\_**(**self**,** name**):**

1340 **super().**\_\_init\_\_**(**name**)**

1341 self**.**Property **=** "Reflective"

1342 self**.**width **=** 10

1343 self**.**colour **=** **(**192**,** 192**,** 192**)**

1344 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

1345 pygame**.**Rect**(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1346 self**.**full\_name **=** "Mirror" **+** **str(**self**.**defined\_name**[-**1**])**

1347

1348 ##### redraw1 #######

1349 # Parameters :- None

1350 # Return Type :- None

1351 # Purpose :- updates instance of object on screen

1352 ###########################

1353 **def** redraw1**(**self**):**

1354 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

1355 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1356 self**.**update**()**

1357

1358

1359##### Screen #######

1360# Inheirts :- Main\_Objects

1361# Parameters :- name:String

1362# Purpose :- Holds every attribute and operation that can be done on a screen object

1363###########################

1364class Screen**(**Main\_Objects**):**

1365 **def** \_\_init\_\_**(**self**,** name**):**

1366 **super().**\_\_init\_\_**(**name**)**

1367 self**.**width **=** 10

1368 self**.**colour **=** **(**169**,** 169**,** 169**)**

1369 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1370 self**.**Property **=** "Stop"

1371

1372 self**.**full\_name **=** "Screen" **+** **str(**self**.**defined\_name**[-**1**])**

1373

1374 ##### redraw1 #######

1375 # Parameters :- None

1376 # Return Type :- None

1377 # Purpose :- updates instance of object on screen

1378 ###########################

1379 **def** redraw1**(**self**):**

1380 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

1381 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1382 self**.**update**()**

1383

1384

1385##### Diffraction #######

1386# Inheirts :- Main\_Objects

1387# Parameters :- name:String

1388# Purpose :- Holds every attribute and operation that can be done on diffraction objects

1389###########################

1390class Diffraction**(**Main\_Objects**):**

1391 **def** \_\_init\_\_**(**self**,** name**):**

1392 **super().**\_\_init\_\_**(**name**)**

1393 self**.**width **=** 5

1394 self**.**length **=** 50

1395 self**.**slitdis **=** 0.0002

1396 self**.**scrlength **=** 1500

1397 self**.**screendis **=** 500

1398 self**.**sourcexp **=** self**.**xp **-** 100

1399 self**.**sourceyp **=** self**.**yp **-** self**.**length **-** self**.**slitdis **+** self**.**length **/** 2

1400 self**.**source\_draging **=** **False**

1401 self**.**wavelength **=** 600 **\*** 10 **\*\*** **-**9

1402 DiffRays**[**self**.**defined\_name**]** **=** **[]**

1403 self**.**yp **=** **(**length **/** 2**)** **-** **(**self**.**slitdis **+** self**.**length**)**

1404 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1405 **(**self**.**xp**,** self**.**yp **-** self**.**length **-** self**.**slitdis**,** self**.**width**,**

1406 self**.**length**))**

1407 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1408 self**.**graphical3 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1409 **(**self**.**xp**,** self**.**yp **+** self**.**length **+** self**.**slitdis**,** self**.**width**,**

1410 self**.**length**))**

1411 self**.**graphical4 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1412 **(**self**.**xp **+** self**.**screendis**,** 0**,** self**.**width**,** self**.**scrlength**))**

1413 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**self**.**sourcexp**,** self**.**sourceyp**),** 7**)**

1414 self**.**Property **=** "D"

1415 self**.**a **=** 2 **\*** self**.**slitdis **\*** 3 **\*** 10 **\*\*** **-**3 **+** self**.**length **\*** 3 **\*** 10 **\*\*** **-**5

1416 self**.**full\_name **=** "Diffraction" **+** **str(**self**.**defined\_name**[-**1**])**

1417 self**.**fringe **=** **(**self**.**wavelength **\*** self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **/** self**.**a

1418 self**.**update**()**

1419

1420 ##### update #######

1421 # Parameters :- None

1422 # Return Type :- None

1423 # Purpose :- checks if properties of diffraction experiment is allowed to be

1424 # displayed and displays them if so

1425 ###########################

1426 **def** update**(**self**):**

1427 self**.**a **=** 2 **\*** self**.**slitdis **\*** 3 **\*** 10 **\*\*** **-**3 **+** self**.**length **\*** 3 **\*** 10 **\*\*** **-**5

1428 self**.**fringe **=** **(**self**.**wavelength **\*** self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **/** self**.**a

1429 **if** m**.**SlitS**:**

1430 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

1431 text **=** font**.**render**(**"Slit Seperation:" **+** **str(**self**.**a**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

1432 textrect **=** text**.**get\_rect**(**center**=(**150**,** 80**))**

1433 screen**.**blit**(**text**,** textrect**)**

1434 **if** m**.**FS**:**

1435 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

1436 text **=** font**.**render**(**"Fringe seperation:" **+** **str(**self**.**fringe**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

1437 textrect **=** text**.**get\_rect**(**center**=(**150**,** 95**))**

1438 screen**.**blit**(**text**,** textrect**)**

1439 **if** m**.**W**:**

1440 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

1441 text **=** font**.**render**(**"Wavelength:" **+** **str(**self**.**wavelength**)** **+** "nm"**,** **True,** **(**105**,** 105**,** 105**))**

1442 textrect **=** text**.**get\_rect**(**center**=(**150**,** 110**))**

1443 screen**.**blit**(**text**,** textrect**)**

1444 **if** m**.**SourceS**:**

1445 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

1446 text **=** font**.**render**(**"Screen distance:" **+** **str(**self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

1447 textrect **=** text**.**get\_rect**(**center**=(**150**,** 125**))**

1448 screen**.**blit**(**text**,** textrect**)**

1449

1450 ##### diffraction #######

1451 # Parameters :- None

1452 # Return Type :- None

1453 # Purpose :- simualtes the young's double slit experiment

1454 ###########################

1455 **def** diffraction**(**self**):**

1456 DiffRays**[**self**.**defined\_name**]** **=** **[]**

1457 mid **=** self**.**sourceyp

1458 fringe\_no **=** **((**1500 **\*** 3 **\*** 10 **\*\*** **-**3**)** **-** **(**mid **\*** 3 **\*** 10 **\*\*** **-**3**))** **/** self**.**fringe

1459 **max** **=** 1

1460 **for** i **in** **range(int(**fringe\_no**)):**

1461 height **=** **max** **\*** **(**0.03 **/** self**.**fringe**)**

1462 **if** mid **-** height **<** 0**:**

1463 **pass**

1464 **else:**

1465 Tag **=** tag\_generator**(**tag**,** "S"**)**

1466 diff\_ray **=** Source**(**Tag**)**

1467 DiffRays**[**self**.**defined\_name**].**append**(**diff\_ray**)**

1468 diff\_ray**.**endx **=** self**.**xp **+** self**.**screendis

1469 diff\_ray**.**endy **=** mid **-** height

1470 diff\_ray**.**xp **=** self**.**xp

1471 diff\_ray**.**yp **=** mid

1472 diff\_ray**.**blackout**()**

1473 **max** **+=** 1

1474 **min** **=** 1

1475 **for** i **in** **range(int(**fringe\_no**)):**

1476 height **=** **min** **\*** **(**0.03 **/** self**.**fringe**)**

1477 **if** mid **+** height **>** 1000**:**

1478 **pass**

1479 **else:**

1480 Tag **=** tag\_generator**(**tag**,** "S"**)**

1481 diff\_ray **=** Source**(**Tag**)**

1482 DiffRays**[**self**.**defined\_name**].**append**(**diff\_ray**)**

1483 diff\_ray**.**endx **=** self**.**xp **+** self**.**screendis

1484 diff\_ray**.**endy **=** mid **+** height

1485 diff\_ray**.**xp **=** self**.**xp

1486 diff\_ray**.**yp **=** mid

1487 diff\_ray**.**blackout**()**

1488 **min** **+=** 1

1489

1490 ##### sourceredraw #######

1491 # Parameters :- x:float

1492 # Return Type :- self.source:function, self.sourcexp:float, self.stateproduct:list

1493 # Purpose :- updates instance of object on screen

1494 ###########################

1495 **def** sourceredraw**(**self**,** x**):**

1496 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1497 **if** x **<** self**.**xp**:**

1498 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**x**,** self**.**sourceyp**),** 5**)**

1499 self**.**sourcexp **=** x

1500 DiffRays**[**self**.**defined\_name**]** **=** **[]**

1501 self**.**update**()**

1502

1503 **return** self**.**source**,** self**.**sourcexp

1504

1505 **def** redraw**(**self**,** x**,** y**):**

1506 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1507 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**

1508 x**,** self**.**yp **-** self**.**length **-** self**.**slitdis**,** self**.**width**,** self**.**length**))**

1509 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**x**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1510 self**.**graphical3 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1511 **(**x**,** self**.**yp **+** self**.**length **+** self**.**slitdis**,** self**.**width**,** self**.**length**))**

1512 self**.**graphical4 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1513 **(**x **+** self**.**screendis**,** 0**,** self**.**width**,** self**.**scrlength**))**

1514 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**self**.**sourcexp**,** self**.**sourceyp**),** 7**)**

1515 self**.**xp **=** x

1516 self**.**update**()**

1517 **return** self**.**xp

1518

1519 ##### redraw1 #######

1520 # Parameters :- None

1521 # Return Type :- None

1522 # Purpose :- updates instance of object on screen

1523 ###########################

1524 **def** redraw1**(**self**):**

1525 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1526 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1527 **(**self**.**xp**,** self**.**yp **-** self**.**length **-** self**.**slitdis**,** self**.**width**,**

1528 self**.**length**))**

1529 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

1530 self**.**graphical3 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1531 **(**self**.**xp**,** self**.**yp **+** self**.**length **+** self**.**slitdis**,** self**.**width**,**

1532 self**.**length**))**

1533 self**.**graphical4 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

1534 **(**self**.**xp **+** self**.**screendis**,** 0**,** self**.**width**,** self**.**scrlength**))**

1535 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**self**.**sourcexp**,** self**.**sourceyp**),** 7**)**

1536 **for** i **in** DiffRays**[**self**.**defined\_name**]:**

1537 i**.**redraw1**()**

1538 self**.**update**()**

1539

1540

1541##### Semi\_Circle #######

1542# Inherits :- Main\_Objects

1543# Parameters :- name:String

1544# Purpose :- Holds every attribute and operation that can be done on a semi circle object

1545###########################

1546class Semi\_Circle**(**Main\_Objects**):**

1547 **def** \_\_init\_\_**(**self**,** name**):**

1548 **super().**\_\_init\_\_**(**name**)**

1549 x**,** y **=** sympy**.**symbols**(**"x,y"**)**

1550 self**.**Radius **=** 100

1551 self**.**graphical **=** pygame**.**draw**.**circle**(**screen**,** **(**168**,** 204**,** 215**),** **(**self**.**xp**,** self**.**yp**),** self**.**Radius**)**

1552 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

1553 **(**self**.**xp **-** self**.**Radius**,** self**.**yp**,** self**.**Radius **\*** 2**,**

1554 self**.**Radius**))**

1555 self**.**Refractive\_Index **=** 1.52

1556 self**.**Equation **=** **(**"((x-{0})\*\*2 + (y-{1})\*\*2)-{2}"**.format(int(**self**.**xp**),** **int(**self**.**yp**),** **int(**self**.**Radius **\*\*** 2**)))**

1557 self**.**Gradient **=** sympy**.**idiff**(eval(**self**.**Equation**),** y**,** x**)**

1558 self**.**full\_name **=** "SemiCircle" **+** **str(**self**.**defined\_name**[-**1**])**

1559

1560 ##### redraw #######

1561 # Parameters :- x:float, y:float

1562 # Return Type :- self.xp:float, self.yp:float, self.interceptors:list, self.Equation:string

1563 # Purpose :- updates instance of object on screen when moved

1564 ###########################

1565 **def** redraw**(**self**,** x**,** y**):**

1566 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

1567 **if** self**.**interceptors**:**

1568 first **=** self**.**interceptors**[**0**]**

1569 **for** i **in** objects\_loaded**:**

1570 **if** i**.**defined\_name **!=** self**.**defined\_name **and** "D" **not** **in** i**.**defined\_name**:**

1571 i**.**redraw1**()**

1572 **for** i **in** sources\_loaded**:**

1573 **if** i **!=** first**:**

1574 i**.**redraw1**()**

1575

1576 first**.**normals **=** **[]**

1577 first**.**angles **=** **[]**

1578 IncidentRays**[**first**.**defined\_name**]** **=** **[]**

1579 first**.**re **=** **None**

1580 **else:**

1581 **for** i **in** objects\_loaded**:**

1582 **if** i**.**defined\_name **!=** self**.**defined\_name **and** "D" **not** **in** i**.**defined\_name**:**

1583 i**.**redraw1**()**

1584 **for** i **in** sources\_loaded**:**

1585 i**.**redraw1**()**

1586

1587 self**.**graphical **=** pygame**.**draw**.**circle**(**screen**,** **(**168**,** 204**,** 215**),** **(**x**,** y**),** self**.**Radius**)**

1588 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

1589 **(**x **-** self**.**Radius**,** y**,** self**.**Radius **\*** 2**,**

1590 self**.**Radius**))**

1591 self**.**xp **=** x

1592 self**.**yp **=** y

1593 self**.**Equation **=** **(**"(x-{0})\*\*2 + (y-{1})\*\*2-{2}"**.format(**x**,** y**,** self**.**Radius **\*\*** 2**))**

1594 self**.**interceptors **=** **[]**

1595 self**.**redraw1**()**

1596

1597 **return** self**.**xp**,** self**.**yp**,** self**.**interceptors**,** self**.**Equation

1598

1599 ##### redraw1 #######

1600 # Parameters :- None

1601 # Return Type :- None

1602 # Purpose :- updates instance of object on screen

1603 ###########################

1604 **def** redraw1**(**self**):**

1605 self**.**graphical **=** pygame**.**draw**.**circle**(**screen**,** **(**168**,** 204**,** 215**),** **(**self**.**xp**,** self**.**yp**),** self**.**Radius**)**

1606 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

1607 **(**self**.**xp **-** self**.**Radius**,** self**.**yp**,** self**.**Radius **\*** 2**,**

1608 self**.**Radius**))**

1609 self**.**update**()**

1610

1611

1612##### Button #######

1613# Parameters :- id:int, def\_name:string

1614# Purpose :- creates button and displays it on screen

1615###########################

1616class Button**:**

1617 **def** \_\_init\_\_**(**self**,** **id,** def\_name**):**

1618 self**.**defined\_name **=** def\_name

1619 self**.**x**,** self**.**y **=** **(**80**,** 45**)**

1620 self**.id** **=** **id**

1621 toolbar\_button**.**append**(**def\_name**)**

1622 self**.**rect **=** pygame**.**draw**.**rect**(**screen**,** **(**160**,** 160**,** 160**),** **(**10 **+** **(**12 **\*** self**.id),** 15**,** self**.**x**,** self**.**y**))**

1623

1624 ##### draw #######

1625 # Parameters :- text:string

1626 # Return Type :- None

1627 # Purpose :- draws text on button

1628 ###########################

1629 **def** draw**(**self**,** text**):**

1630 **if** text **!=** ''**:**

1631 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

1632 text **=** font**.**render**(**text**,** **True,** **(**0**,** 0**,** 0**))**

1633 textrect **=** text**.**get\_rect**(**center**=((**self**.**x **/** 2**)** **+** 10 **+** **(**12 **\*** self**.id),** **(**self**.**y **/** 2**)** **+** 15**))**

1634 screen**.**blit**(**text**,** textrect**)**

1635

1636 ##### click #######

1637 # Parameters :- event:function

1638 # Return Type :- False:boolean, self.defined\_name:string

1639 # Purpose :- checks if the button has been pressed and if so returns its name

1640 ###########################

1641 **def** click**(**self**,** event**):**

1642 x**,** y **=** pygame**.**mouse**.**get\_pos**()**

1643 **if** event**.type** **==** pygame**.**MOUSEBUTTONDOWN**:**

1644 **if** self**.**rect**.**collidepoint**(**x**,** y**):**

1645 **return** self**.**defined\_name

1646 **else:**

1647 **return** **False**

1648

1649

1650##### ChangeGUI #######

1651# Parameters :- None

1652# Return Type :- None

1653# Purpose :- changes the properties of objects displayed

1654###########################

1655class ChangeGUI**():**

1656 **def** \_\_init\_\_**(**self**):**

1657 self**.**main **=** tkinter**.**Tk**()**

1658 self**.**main**.**geometry**(**"500x500"**)**

1659 self**.**Tab **=** ttk**.**Notebook**(**self**.**main**)**

1660 self**.**run**()**

1661

1662 ##### run #######

1663 # Parameters :- None

1664 # Return Type :- None

1665 # Purpose :- creates a a gui for changing properties

1666 ###########################

1667 **def** run**(**self**):**

1668 **for** i **in** objects\_loaded**:**

1669 **if** "D" **in** i**.**defined\_name**:**

1670 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1671 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1672 ttk**.**Label**(**tab1**,**

1673 text**=**"Slit Distance"**).**grid**(**column**=**0**,** row**=**0**)**

1674 ttk**.**Label**(**tab1**,**

1675 text**=**"Screen Distance"**).**grid**(**column**=**0**,** row**=**1**)**

1676 ttk**.**Label**(**tab1**,**

1677 text**=**"Wavelength(400-700)\*10^-9"**).**grid**(**column**=**0**,** row**=**2**)**

1678 enter **=** tkinter**.**StringVar**()**

1679 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1680 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1681 enter1 **=** tkinter**.**StringVar**()**

1682 e2 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter1**)**

1683 e2**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1684 enter2 **=** tkinter**.**StringVar**()**

1685 e3 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter2**)**

1686 e3**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1687

1688 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,**

1689 command**=lambda:** self**.**diffraction\_change**(**i**,** **str(**e1**.**get**()),** **str(**e2**.**get**()),**

1690 **str(**e3**.**get**())))**

1691 B1**.**grid**(**row**=**3**,** column**=**1**)**

1692 **elif** "M" **in** i**.**defined\_name**:**

1693 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1694 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1695 ttk**.**Label**(**tab1**,**

1696 text**=**"Length"**).**grid**(**column**=**0**,** row**=**0**)**

1697 enter **=** tkinter**.**StringVar**()**

1698 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1699 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1700 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,** command**=lambda:** self**.**sm\_change**(**i**,** **str(**e1**.**get**())))**

1701 B1**.**grid**(**row**=**0**,** column**=**2**,** sticky**=**tkinter**.**E**)**

1702 **elif** "SC" **in** i**.**defined\_name**:**

1703 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1704 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1705 ttk**.**Label**(**tab1**,**

1706 text**=**"Radius"**).**grid**(**column**=**0**,** row**=**0**)**

1707 enter **=** tkinter**.**StringVar**()**

1708 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1709 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1710 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,** command**=lambda:** self**.**semi\_change**(**i**,** **str(**e1**.**get**())))**

1711 B1**.**grid**(**row**=**0**,** column**=**2**,** sticky**=**tkinter**.**E**)**

1712 **elif** "S" **in** i**.**defined\_name**:**

1713 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1714 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1715 ttk**.**Label**(**tab1**,**

1716 text**=**"Length"**).**grid**(**column**=**0**,** row**=**0**)**

1717 enter **=** tkinter**.**StringVar**()**

1718 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1719 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1720 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,** command**=lambda:** self**.**sm\_change**(**i**,** **str(**e1**.**get**())))**

1721 B1**.**grid**(**row**=**0**,** column**=**2**,** sticky**=**tkinter**.**E**)**

1722 **elif** "G" **in** i**.**defined\_name**:**

1723 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1724 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1725 ttk**.**Label**(**tab1**,**

1726 text**=**"Length"**).**grid**(**column**=**0**,** row**=**0**)**

1727 ttk**.**Label**(**tab1**,**

1728 text**=**"Width"**).**grid**(**column**=**0**,** row**=**1**)**

1729 enter **=** tkinter**.**StringVar**()**

1730 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1731 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1732 enter1 **=** tkinter**.**StringVar**()**

1733 e2 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter1**)**

1734 e2**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1735 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,**

1736 command**=lambda:** self**.**glass\_change**(**i**,** **str(**e1**.**get**()),** **str(**e2**.**get**())))**

1737 B1**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1738

1739 **elif** "B" **in** i**.**defined\_name**:**

1740 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1741 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1742 ttk**.**Label**(**tab1**,**

1743 text**=**"Refractive Index"**).**grid**(**column**=**0**,** row**=**0**)**

1744 ttk**.**Label**(**tab1**,**

1745 text**=**"Length"**).**grid**(**column**=**0**,** row**=**1**)**

1746 ttk**.**Label**(**tab1**,**

1747 text**=**"Width"**).**grid**(**column**=**0**,** row**=**2**)**

1748 enter **=** tkinter**.**StringVar**()**

1749 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1750 e1**.**grid**(**row**=**0**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1751 enter1 **=** tkinter**.**StringVar**()**

1752 e2 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter1**)**

1753 e2**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1754 enter2 **=** tkinter**.**StringVar**()**

1755 e3 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter2**)**

1756 e3**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1757 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,**

1758 command**=lambda:** self**.**block\_change**(**i**,** **str(**e1**.**get**()),** **str(**e2**.**get**()),** **str(**e3**.**get**())))**

1759 B1**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1760

1761 **for** i **in** sources\_loaded**:**

1762 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1763 self**.**Tab**.**add**(**tab1**,** text**=eval(**'i.full\_name'**))**

1764 ttk**.**Label**(**tab1**,**

1765 text**=**"Wavelength(400-700)\*10^-9"**).**grid**(**column**=**0**,** row**=**0**)**

1766 enter **=** tkinter**.**StringVar**()**

1767 e1 **=** tkinter**.**Entry**(**tab1**,** textvariable**=**enter**)**

1768 e1**.**grid**(**row**=**1**,** column**=**0**,** sticky**=**tkinter**.**E**)**

1769 B1 **=** tkinter**.**Button**(**tab1**,** text**=**"Okay"**,**

1770 command**=lambda:** self**.**source\_change**(**i**,** **str(**e1**.**get**())))**

1771 B1**.**grid**(**row**=**2**,** column**=**0**,** sticky**=**tkinter**.**E**)**

1772 self**.**Tab**.**pack**(**expand**=**1**,** fill**=**"both"**)**

1773 self**.**main**.**protocol**(**"WM\_DELETE\_WINDOW"**,** self**.**stop**)**

1774 self**.**main**.**mainloop**()**

1775

1776 ##### diffraction\_change #######

1777 # Parameters :- i:Memory Address of object, slitdis:float, screendis:float

1778 # Return Type :- None

1779 # Purpose :- changes properties specifically for the diffraction class

1780 ###########################

1781 **def** diffraction\_change**(**self**,** i**,** slitdis**,** screendis**,** wavelength**):**

1782 **if** slitdis **!=** ""**:**

1783 i**.**slitdis **=** **float(**slitdis**)** **/** 3 **\*** 10 **\*\*** **-**3

1784 **if** screendis **!=** ""**:**

1785 i**.**screendis **=** **float(**screendis**)** **/** 3 **\*** 10 **\*\*** **-**3

1786 **if** wavelength **!=** "" **and** 400 **\*** 10 **\*\*** **-**9 **<=** **float(**wavelength**)** **<=** 700 **\*** 10 **\*\*** **-**9**:**

1787 i**.**wavelength **=** **float(**wavelength**)**

1788 i**.**update**()**

1789 i**.**diffraction**()**

1790 i**.**redraw1**()**

1791 pygame**.**display**.**flip**()**

1792 self**.**stop**()**

1793

1794 ##### block\_change #######

1795 # Parameters :- i:Memory Address of object, newRI:float, newlength:float, newwidth:float

1796 # Return Type :- None

1797 # Purpose :- changes properties specifically for the block class

1798 ###########################

1799 **def** block\_change**(**self**,** i**,** newRI**,** newlength**,** newwidth**):**

1800 **if** newRI **!=** ""**:**

1801 i**.**Refractive\_Index **=** **float(**newRI**)**

1802 **if** newlength **!=** ""**:**

1803 i**.**length **=** **int(**newlength**)**

1804 **if** newwidth **!=** ""**:**

1805 i**.**width **=** **int(**newwidth**)**

1806 i**.**redraw1**()**

1807 pygame**.**display**.**flip**()**

1808 self**.**stop**()**

1809

1810 ##### glass\_change #######

1811 # Parameters :- i:Memory Address of object, newlength:float, newwidth:float

1812 # Return Type :- None

1813 # Purpose :- changes properties specifically for the glassblock class

1814 ###########################

1815 **def** glass\_change**(**self**,** i**,** newlength**,** newwidth**):**

1816 **if** newlength **!=** ""**:**

1817 i**.**length **=** **int(**newlength**)**

1818 **if** newwidth **!=** ""**:**

1819 i**.**width **=** **int(**newwidth**)**

1820 i**.**redraw1**()**

1821 pygame**.**display**.**flip**()**

1822 self**.**stop**()**

1823

1824 ##### semi\_change #######

1825 # Parameters :- i:Memory Address of object, newradius:float

1826 # Return Type :- None

1827 # Purpose :- changes properties specifically for the semi circle class

1828 ###########################

1829 **def** semi\_change**(**self**,** i**,** newradius**):**

1830 **if** newradius **!=** ""**:**

1831 i**.**Radius **=** **int(**newradius**)**

1832 i**.**redraw1**()**

1833 pygame**.**display**.**flip**()**

1834 self**.**stop**()**

1835

1836 ##### sm\_change #######

1837 # Parameters :- i:Memory Address of object, newlength:float

1838 # Return Type :- None

1839 # Purpose :- changes properties specifically for the screen class

1840 ###########################

1841 **def** sm\_change**(**self**,** i**,** newlength**):**

1842 **if** newlength **!=** ""**:**

1843 i**.**length **=** **int(**newlength**)**

1844 i**.**redraw1**()**

1845 pygame**.**display**.**flip**()**

1846 self**.**stop**()**

1847

1848 ##### source\_change #######

1849 # Parameters :- i:Memory Address of object, wavelength:float

1850 # Return Type :- None

1851 # Purpose :- changes properties specifically for the source class

1852 ###########################

1853 **def** source\_change**(**self**,** i**,** wavelength**):**

1854 **if** wavelength **!=** "" **and** 400 **\*** 10 **\*\*** **-**9 **<=** **float(**wavelength**)** **<=** 700 **\*** 10 **\*\*** **-**9**:**

1855 i**.**wavelength **=** **float(**wavelength**)**

1856 i**.**redraw1**()**

1857 pygame**.**display**.**flip**()**

1858 self**.**stop**()**

1859

1860 ##### stop #######

1861 # Parameters :- None

1862 # Return Type :- None

1863 # Purpose :- destroys gui

1864 ###########################

1865 **def** stop**(**self**):**

1866 self**.**main**.**destroy**()**

1867

1868

1869##### ShowGUI #######

1870# Parameters :- None

1871# Return Type :- None

1872# Purpose :- dictates which properties of an experiment can be displayed

1873###########################

1874class ShowGUI**:**

1875 **def** \_\_init\_\_**(**self**):**

1876 self**.**name **=** "ShowGUI"

1877 self**.**IA **=** **False**

1878 self**.**RA **=** **False**

1879 self**.**RI **=** **False**

1880 self**.**SlitS **=** **False**

1881 self**.**FS **=** **False**

1882 self**.**SourceS **=** **False**

1883 self**.**W **=** **False**

1884

1885 ##### gui\_loop #######

1886 # Parameters :- None

1887 # Return Type :- None

1888 # Purpose :- creates a a gui for showing properties

1889 ###########################

1890 **def** gui\_loop**(**self**):**

1891 self**.**main **=** tkinter**.**Tk**()**

1892 self**.**main**.**geometry**(**"50x200"**)**

1893 var1 **=** tkinter**.**IntVar**()**

1894 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Incident Angle"**,** variable**=**var1**).**grid**(**row**=**0**,** sticky**=**tkinter**.**W**)**

1895 var2 **=** tkinter**.**IntVar**()**

1896 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Refraction Angle"**,** variable**=**var2**).**grid**(**row**=**1**,** sticky**=**tkinter**.**W**)**

1897 var3 **=** tkinter**.**IntVar**()**

1898 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Refractive Index"**,** variable**=**var3**).**grid**(**row**=**2**,** sticky**=**tkinter**.**W**)**

1899 var4 **=** tkinter**.**IntVar**()**

1900 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Slit Seperation"**,** variable**=**var4**).**grid**(**row**=**3**,** sticky**=**tkinter**.**W**)**

1901 var5 **=** tkinter**.**IntVar**()**

1902 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Fringe Seperation"**,** variable**=**var5**).**grid**(**row**=**4**,** sticky**=**tkinter**.**W**)**

1903 var6 **=** tkinter**.**IntVar**()**

1904 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Source Seperation"**,** variable**=**var6**).**grid**(**row**=**5**,** sticky**=**tkinter**.**W**)**

1905 var7 **=** tkinter**.**IntVar**()**

1906 tkinter**.**Checkbutton**(**self**.**main**,** text**=**"Wavelength"**,** variable**=**var7**).**grid**(**row**=**6**,** sticky**=**tkinter**.**W**)**

1907 B1 **=** tkinter**.**Button**(**self**.**main**,** text**=**"Okay"**,**

1908 command**=lambda:** self**.**showing**(**var1**.**get**(),** var2**.**get**(),** var3**.**get**(),** var4**.**get**(),** var5**.**get**(),**

1909 var6**.**get**(),** var7**.**get**()))**

1910 B1**.**grid**(**row**=**7**,** column**=**0**,** sticky**=**tkinter**.**E**)**

1911 self**.**main**.**protocol**(**"WM\_DELETE\_WINDOW"**,** self**.**stop**)**

1912 self**.**main**.**mainloop**()**

1913

1914 ##### showing #######

1915 # Parameters :- IA:boolean, RA:boolean, RI:boolean, SlitS:boolean, FS:boolean, SourceS:boolean, W:boolean

1916 # Return Type :- None

1917 # Purpose :- checks whether a property can be displayed or not

1918 ###########################

1919 **def** showing**(**self**,** IA**,** RA**,** RI**,** SlitS**,** FS**,** SourceS**,** W**):**

1920 **if** IA **==** 1**:**

1921 self**.**IA **=** **True**

1922 **if** RA **==** 1**:**

1923 self**.**RA **=** **True**

1924 **if** RI **==** 1**:**

1925 self**.**RI **=** **True**

1926 **if** SlitS **==** 1**:**

1927 self**.**SlitS **=** **True**

1928 **if** FS **==** 1**:**

1929 self**.**FS **=** **True**

1930 **if** SourceS **==** 1**:**

1931 self**.**SourceS **=** **True**

1932 **if** W **==** 1**:**

1933 self**.**W **=** **True**

1934 **if** IA **==** 0**:**

1935 self**.**IA **=** **False**

1936 **if** RA **==** 0**:**

1937 self**.**RA **=** **False**

1938 **if** RI **==** 0**:**

1939 self**.**RI **=** **False**

1940 **if** SlitS **==** 0**:**

1941 self**.**SlitS **=** **False**

1942 **if** FS **==** 0**:**

1943 self**.**FS **=** **False**

1944 **if** SourceS **==** 0**:**

1945 self**.**SourceS **=** **False**

1946 **if** W **==** 0**:**

1947 self**.**W **=** **False**

1948 self**.**main**.**destroy**()**

1949 **for** i **in** sources\_loaded**:**

1950 i**.**redraw1**()**

1951 #return self.IA, self.RA, self.RI, self.SlitS, self.W, self.IA, self.FS

1952

1953

1954 ##### stop #######

1955 # Parameters :- None

1956 # Return Type :- None

1957 # Purpose :- destroys the gui

1958 ###########################

1959 **def** stop**(**self**):**

1960 self**.**main**.**destroy**()**

1961

1962

1963##### QuestionsGUI #######

1964# Parameters :- c:object

1965# Return Type :- None

1966# Purpose :- creates questions depending what is being displayed

1967###########################

1968class QuestionsGUI**:**

1969 **def** \_\_init\_\_**(**self**,** c**):**

1970 self**.**objects **=** objects\_loaded

1971 self**.**sources **=** sources\_loaded

1972 self**.**run**(**c**)**

1973

1974 ##### run #######

1975 # Parameters :- c:object

1976 # Return Type :- None

1977 # Purpose :- creates a gui for creating or setting question

1978 ###########################

1979 **def** run**(**self**,** c**):**

1980 possible **=** self**.**AutoGen**()**

1981 self**.**main **=** tkinter**.**Tk**()**

1982 self**.**main**.**geometry**(**"500x500"**)**

1983 self**.**Tab **=** ttk**.**Notebook**(**self**.**main**)**

1984 tab1 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**1000**,** height**=**280**)**

1985 self**.**Tab**.**add**(**tab1**,** text**=**'Auto Generated'**)**

1986 tab2 **=** ttk**.**Frame**(**self**.**Tab**,** width**=**400**,** height**=**280**)**

1987 self**.**Tab**.**add**(**tab2**,** text**=**'Create'**)**

1988 ttk**.**Label**(**tab2**,**

1989 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

1990 ttk**.**Label**(**tab2**,**

1991 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

1992 ttk**.**Label**(**tab2**,**

1993 text**=**"Answer"**).**grid**(**column**=**0**,** row**=**3**)**

1994 enter **=** tkinter**.**StringVar**()**

1995 c1 **=** tkinter**.**Entry**(**tab2**,** textvariable**=**enter**)**

1996 c1**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

1997 enter1 **=** tkinter**.**StringVar**()**

1998 c2 **=** tkinter**.**Entry**(**tab2**,** textvariable**=**enter1**)**

1999 c2**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2000 enter2 **=** tkinter**.**StringVar**()**

2001 c3 **=** tkinter**.**Entry**(**tab2**,** textvariable**=**enter2**)**

2002 c3**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2003 B1 **=** tkinter**.**Button**(**tab2**,** text**=**"Set"**,**

2004 command**=lambda:** **[**c**.**questions**(**c2**.**get**(),** c1**.**get**(),** **[**c3**.**get**(),** "created"**]),**

2005 self**.**main**.**destroy**()])**

2006

2007 B1**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2008 self**.**Tab**.**pack**(**expand**=**1**,** fill**=**"both"**)**

2009 intertab1 **=** ttk**.**Notebook**(**tab1**)**

2010

2011 **for** i **in** possible**:**

2012 **if** "index" **in** i**:**

2013 ttab1 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2014 intertab1**.**add**(**ttab1**,** text**=**'Refractive Index'**)**

2015 ttk**.**Label**(**ttab1**,**

2016 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2017 ttk**.**Label**(**ttab1**,**

2018 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2019 ttk**.**Label**(**ttab1**,**

2020 text**=**"Enter Refraction Angle"**).**grid**(**column**=**0**,** row**=**3**)**

2021 ttk**.**Label**(**ttab1**,**

2022 text**=**"Enter Incident Angle"**).**grid**(**column**=**0**,** row**=**4**)**

2023 enter **=** tkinter**.**StringVar**()**

2024 e11 **=** tkinter**.**Entry**(**ttab1**,** textvariable**=**enter**)**

2025 e11**.**insert**(**0**,** 'Find the refractive index of the object'**)**

2026 e11**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2027 enter1 **=** tkinter**.**StringVar**()**

2028 e21 **=** tkinter**.**Entry**(**ttab1**,** textvariable**=**enter1**)**

2029 e21**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2030 enter2 **=** tkinter**.**StringVar**()**

2031 e31 **=** tkinter**.**Entry**(**ttab1**,** textvariable**=**enter2**)**

2032 e31**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2033 enter3 **=** tkinter**.**StringVar**()**

2034 e41 **=** tkinter**.**Entry**(**ttab1**,** textvariable**=**enter3**)**

2035 e41**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2036 B1 **=** tkinter**.**Button**(**ttab1**,** text**=**"Set"**,**

2037 command**=lambda:** **[**c**.**questions**(**e21**.**get**(),** e11**.**get**(),** **[float(**e31**.**get**()),** **float(**e41**.**get**())]),**

2038 self**.**main**.**destroy**()])**

2039 B1**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2040 **elif** "critical" **and** "type1" **in** i**:**

2041 ttab2 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2042 intertab1**.**add**(**ttab2**,** text**=**'Crtical Angle Type1'**)**

2043 ttk**.**Label**(**ttab2**,**

2044 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2045 ttk**.**Label**(**ttab2**,**

2046 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2047 ttk**.**Label**(**ttab2**,**

2048 text**=**"Enter Refractive Index"**).**grid**(**column**=**0**,** row**=**3**)**

2049 enter **=** tkinter**.**StringVar**()**

2050 e12 **=** tkinter**.**Entry**(**ttab2**,** textvariable**=**enter**)**

2051 e12**.**insert**(**0**,** 'Find the critical angle of the object'**)**

2052 e12**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2053 enter1 **=** tkinter**.**StringVar**()**

2054 e22 **=** tkinter**.**Entry**(**ttab2**,** textvariable**=**enter1**)**

2055 e22**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2056 enter2 **=** tkinter**.**StringVar**()**

2057 e32 **=** tkinter**.**Entry**(**ttab2**,** textvariable**=**enter2**)**

2058 e32**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2059 B1 **=** tkinter**.**Button**(**ttab2**,** text**=**"Set"**,**

2060 command**=lambda:** **[**c**.**questions**(**e22**.**get**(),** e12**.**get**(),** **[float(**e32**.**get**())]),** self**.**main**.**destroy**()])**

2061 B1**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2062 **elif** "fringes" **in** i**:**

2063 ttab3 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2064 intertab1**.**add**(**ttab3**,** text**=**'Fringe'**)**

2065 ttk**.**Label**(**ttab3**,**

2066 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2067 ttk**.**Label**(**ttab3**,**

2068 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2069 ttk**.**Label**(**ttab3**,**

2070 text**=**"Enter Slit Seperation"**).**grid**(**column**=**0**,** row**=**3**)**

2071 ttk**.**Label**(**ttab3**,**

2072 text**=**"Enter Source Seperartion"**).**grid**(**column**=**0**,** row**=**4**)**

2073 ttk**.**Label**(**ttab3**,**

2074 text**=**"Enter Wavelength"**).**grid**(**column**=**0**,** row**=**5**)**

2075 enter **=** tkinter**.**StringVar**()**

2076 e13 **=** tkinter**.**Entry**(**ttab3**,** textvariable**=**enter**)**

2077 e13**.**insert**(**0**,** 'Find the fringe seperation between adjacent maximas'**)**

2078 e13**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2079 enter1 **=** tkinter**.**StringVar**()**

2080 e23 **=** tkinter**.**Entry**(**ttab3**,** textvariable**=**enter1**)**

2081 e23**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2082 enter2 **=** tkinter**.**StringVar**()**

2083 e33 **=** tkinter**.**Entry**(**ttab3**,** textvariable**=**enter2**)**

2084 e33**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2085 enter3 **=** tkinter**.**StringVar**()**

2086 e43 **=** tkinter**.**Entry**(**ttab3**,** textvariable**=**enter3**)**

2087 e43**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2088 enter4 **=** tkinter**.**StringVar**()**

2089 e53 **=** tkinter**.**Entry**(**ttab3**,** textvariable**=**enter4**)**

2090 e53**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2091 B1 **=** tkinter**.**Button**(**ttab3**,** text**=**"Set"**,**

2092 command**=lambda:** **[**c**.**questions**(**e23**.**get**(),** e13**.**get**(),** **[float(**e33**.**get**()),** **float(**e43**.**get**()),** **float(**e53**.**get**())]),**

2093 self**.**main**.**destroy**()])**

2094 B1**.**grid**(**row**=**7**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2095 **elif** "slits and screen" **in** i**:**

2096 ttab4 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2097 intertab1**.**add**(**ttab4**,** text**=**'Screen Distance'**)**

2098 ttk**.**Label**(**ttab4**,**

2099 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2100 ttk**.**Label**(**ttab4**,**

2101 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2102 ttk**.**Label**(**ttab4**,**

2103 text**=**"Enter Slit Seperation"**).**grid**(**column**=**0**,** row**=**3**)**

2104 ttk**.**Label**(**ttab4**,**

2105 text**=**"Enter Fringe"**).**grid**(**column**=**0**,** row**=**4**)**

2106 ttk**.**Label**(**ttab4**,**

2107 text**=**"Enter Wavelength"**).**grid**(**column**=**0**,** row**=**5**)**

2108 enter **=** tkinter**.**StringVar**()**

2109 e14 **=** tkinter**.**Entry**(**ttab4**,** textvariable**=**enter**)**

2110 e14**.**insert**(**0**,** 'Find the distance between the screen and slits'**)**

2111 e14**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2112 enter1 **=** tkinter**.**StringVar**()**

2113 e24 **=** tkinter**.**Entry**(**ttab4**,** textvariable**=**enter1**)**

2114 e24**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2115 enter2 **=** tkinter**.**StringVar**()**

2116 e34 **=** tkinter**.**Entry**(**ttab4**,** textvariable**=**enter2**)**

2117 e34**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2118 enter3 **=** tkinter**.**StringVar**()**

2119 e44 **=** tkinter**.**Entry**(**ttab4**,** textvariable**=**enter3**)**

2120 e44**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2121 enter4 **=** tkinter**.**StringVar**()**

2122 e54 **=** tkinter**.**Entry**(**ttab4**,** textvariable**=**enter4**)**

2123 e54**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2124 B1 **=** tkinter**.**Button**(**ttab4**,** text**=**"Set"**,**

2125 command**=lambda:** **[**c**.**questions**(**e24**.**get**(),** e14**.**get**(),** **[float(**e34**.**get**()),** **float(**e44**.**get**()),** **float(**e54**.**get**())]),**

2126 self**.**main**.**destroy**()])**

2127 B1**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2128 **elif** "slits" **in** i**:**

2129 ttab5**=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2130 intertab1**.**add**(**ttab5**,** text**=**'Slit Seperation'**)**

2131 ttk**.**Label**(**ttab5**,**

2132 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2133 ttk**.**Label**(**ttab5**,**

2134 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2135 ttk**.**Label**(**ttab5**,**

2136 text**=**"Enter Fringe"**).**grid**(**column**=**0**,** row**=**3**)**

2137 ttk**.**Label**(**ttab5**,**

2138 text**=**"Enter Source Seperartion"**).**grid**(**column**=**0**,** row**=**4**)**

2139 ttk**.**Label**(**ttab5**,**

2140 text**=**"Enter Wavelength"**).**grid**(**column**=**0**,** row**=**5**)**

2141 enter **=** tkinter**.**StringVar**()**

2142 e15 **=** tkinter**.**Entry**(**ttab5**,** textvariable**=**enter**)**

2143 e15**.**insert**(**0**,** 'Find the seperation between the narrow slits'**)**

2144 e15**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2145 enter1 **=** tkinter**.**StringVar**()**

2146 e25**=** tkinter**.**Entry**(**ttab5**,** textvariable**=**enter1**)**

2147 e25**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2148 enter2 **=** tkinter**.**StringVar**()**

2149 e35 **=** tkinter**.**Entry**(**ttab5**,** textvariable**=**enter2**)**

2150 e35**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2151 enter3 **=** tkinter**.**StringVar**()**

2152 e45 **=** tkinter**.**Entry**(**ttab5**,** textvariable**=**enter3**)**

2153 e45**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2154 enter4 **=** tkinter**.**StringVar**()**

2155 e55 **=** tkinter**.**Entry**(**ttab5**,** textvariable**=**enter4**)**

2156 e55**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2157 B1 **=** tkinter**.**Button**(**ttab5**,** text**=**"Set"**,**

2158 command**=lambda:** **[**c**.**questions**(**e25**.**get**(),** e15**.**get**(),** **[float(**e35**.**get**()),** **float(**e45**.**get**()),** **float(**e55**.**get**())]),**

2159 self**.**main**.**destroy**()])**

2160 B1**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2161 **elif** "wavelength" **in** i**:**

2162 ttab6 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2163 intertab1**.**add**(**ttab6**,** text**=**'Wavelength'**)**

2164 ttk**.**Label**(**ttab6**,**

2165 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2166 ttk**.**Label**(**ttab6**,**

2167 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2168 ttk**.**Label**(**ttab6**,**

2169 text**=**"Enter Fringe"**).**grid**(**column**=**0**,** row**=**3**)**

2170 ttk**.**Label**(**ttab6**,**

2171 text**=**"Enter Source Seperartion"**).**grid**(**column**=**0**,** row**=**4**)**

2172 ttk**.**Label**(**ttab6**,**

2173 text**=**"Enter Slit Seperation"**).**grid**(**column**=**0**,** row**=**5**)**

2174 enter **=** tkinter**.**StringVar**()**

2175 e16 **=** tkinter**.**Entry**(**ttab6**,** textvariable**=**enter**)**

2176 e16**.**insert**(**0**,** 'Find the wavelength of the source'**)**

2177 e16**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2178 enter1 **=** tkinter**.**StringVar**()**

2179 e26 **=** tkinter**.**Entry**(**ttab6**,** textvariable**=**enter1**)**

2180 e26**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2181 enter2 **=** tkinter**.**StringVar**()**

2182 e36 **=** tkinter**.**Entry**(**ttab6**,** textvariable**=**enter2**)**

2183 e36**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2184 enter3 **=** tkinter**.**StringVar**()**

2185 e46 **=** tkinter**.**Entry**(**ttab6**,** textvariable**=**enter3**)**

2186 e46**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2187 enter4 **=** tkinter**.**StringVar**()**

2188 e56 **=** tkinter**.**Entry**(**ttab6**,** textvariable**=**enter4**)**

2189 e56**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2190 B1 **=** tkinter**.**Button**(**ttab6**,** text**=**"Set"**,**

2191 command**=lambda:** **[**c**.**questions**(**e26**.**get**(),** e16**.**get**(),** **[float(**e36**.**get**()),** **float(**e46**.**get**()),** **float(**e56**.**get**())]),**

2192 self**.**main**.**destroy**()])**

2193 B1**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2194 **elif** "order" **and** "type1" **in** i**:**

2195 ttab7 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2196 intertab1**.**add**(**ttab7 **,** text**=**'Diffraction Order Type1'**)**

2197 ttk**.**Label**(**ttab7 **,**

2198 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2199 ttk**.**Label**(**ttab7 **,**

2200 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2201 ttk**.**Label**(**ttab7 **,**

2202 text**=**"Enter Fringe"**).**grid**(**column**=**0**,** row**=**3**)**

2203 ttk**.**Label**(**ttab7 **,**

2204 text**=**"Enter Source Seperartion"**).**grid**(**column**=**0**,** row**=**4**)**

2205 ttk**.**Label**(**ttab7 **,**

2206 text**=**"Enter Slit Seperation"**).**grid**(**column**=**0**,** row**=**5**)**

2207 ttk**.**Label**(**ttab7 **,**

2208 text**=**"Enter Fringe Number"**).**grid**(**column**=**0**,** row**=**6**)**

2209 enter **=** tkinter**.**StringVar**()**

2210 e17 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter**)**

2211 e17**.**insert**(**0**,** 'Find the diffrtaction angle of the maxima order n'**)**

2212 e17**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2213 enter1 **=** tkinter**.**StringVar**()**

2214 e27 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter1**)**

2215 e27**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2216 enter2 **=** tkinter**.**StringVar**()**

2217 e37 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter2**)**

2218 e37**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2219 enter3 **=** tkinter**.**StringVar**()**

2220 e47 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter3**)**

2221 e47**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2222 enter4 **=** tkinter**.**StringVar**()**

2223 e57 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter4**)**

2224 e57**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2225 enter5 **=** tkinter**.**StringVar**()**

2226 e67 **=** tkinter**.**Entry**(**ttab7 **,** textvariable**=**enter5**)**

2227 e67**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2228 B1 **=** tkinter**.**Button**(**ttab7 **,** text**=**"Set"**,**

2229 command**=lambda:** **[**c**.**questions**(**e27**.**get**(),** e17**.**get**(),**

2230 **[float(**e37**.**get**()),** **float(**e47**.**get**()),** **float(**e57**.**get**()),** **float(**e67**.**get**())]),**

2231 self**.**main**.**destroy**()])**

2232 B1**.**grid**(**row**=**7**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2233 **elif** "refraction" **in** i**:**

2234 ttab8 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2235 intertab1**.**add**(**ttab8**,** text**=**'Refraction Angle'**)**

2236 ttk**.**Label**(**ttab8**,**

2237 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2238 ttk**.**Label**(**ttab8**,**

2239 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2240 ttk**.**Label**(**ttab8**,**

2241 text**=**"Enter Incidence Angle"**).**grid**(**column**=**0**,** row**=**3**)**

2242 ttk**.**Label**(**ttab8**,**

2243 text**=**"Enter Refractive Index"**).**grid**(**column**=**0**,** row**=**4**)**

2244 enter **=** tkinter**.**StringVar**()**

2245 e19 **=** tkinter**.**Entry**(**ttab8**,** textvariable**=**enter**)**

2246 e19**.**insert**(**0**,** 'Find the refraction angle of object'**)**

2247 e19**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2248 enter1 **=** tkinter**.**StringVar**()**

2249 e29**=** tkinter**.**Entry**(**ttab8**,** textvariable**=**enter1**)**

2250 e29**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2251 enter2 **=** tkinter**.**StringVar**()**

2252 e39 **=** tkinter**.**Entry**(**ttab8**,** textvariable**=**enter2**)**

2253 e39**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2254 enter3 **=** tkinter**.**StringVar**()**

2255 e49 **=** tkinter**.**Entry**(**ttab8**,** textvariable**=**enter3**)**

2256 e49**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2257 B1 **=** tkinter**.**Button**(**ttab8**,** text**=**"Set"**,**

2258 command**=lambda:** **[**c**.**questions**(**e29**.**get**(),** e19**.**get**(),** **[float(**e39**.**get**()),** **float(**e49**.**get**())]),**

2259 self**.**main**.**destroy**()])**

2260 B1**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2261

2262 **elif** "incidence" **in** i**:**

2263 ttab9 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2264 intertab1**.**add**(**ttab9**,** text**=**'Incidence Angle'**)**

2265 ttk**.**Label**(**ttab9**,**

2266 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2267 ttk**.**Label**(**ttab9**,**

2268 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2269 ttk**.**Label**(**ttab9**,**

2270 text**=**"Enter Refraction Angle"**).**grid**(**column**=**0**,** row**=**3**)**

2271 ttk**.**Label**(**ttab9**,**

2272 text**=**"Enter Refractive Index"**).**grid**(**column**=**0**,** row**=**4**)**

2273 enter **=** tkinter**.**StringVar**()**

2274 e10 **=** tkinter**.**Entry**(**ttab9**,** textvariable**=**enter**)**

2275 e10**.**insert**(**0**,** 'Find the incidence angle of object'**)**

2276 e10**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2277 enter1 **=** tkinter**.**StringVar**()**

2278 e20 **=** tkinter**.**Entry**(**ttab9**,** textvariable**=**enter1**)**

2279 e20**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2280 enter2 **=** tkinter**.**StringVar**()**

2281 e30**=** tkinter**.**Entry**(**ttab9**,** textvariable**=**enter2**)**

2282 e30**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2283 enter3 **=** tkinter**.**StringVar**()**

2284 e40 **=** tkinter**.**Entry**(**ttab9**,** textvariable**=**enter3**)**

2285 e40**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2286 B1 **=** tkinter**.**Button**(**ttab9**,** text**=**"Set"**,**

2287 command**=lambda:** **[**c**.**questions**(**e20**.**get**(),** e10**.**get**(),** **[float(**e30**.**get**()),** **float(**e40**.**get**())]),**

2288 self**.**main**.**destroy**()])**

2289 B1**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2290 **elif** "order" **and** "type2" **in** i**:**

2291 ttab10 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2292 intertab1**.**add**(**ttab10**,** text**=**'Diffraction Order Type2'**)**

2293 ttk**.**Label**(**ttab10**,**

2294 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2295 ttk**.**Label**(**ttab10**,**

2296 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2297 ttk**.**Label**(**ttab10**,**

2298 text**=**"Enter Wavelength"**).**grid**(**column**=**0**,** row**=**3**)**

2299 ttk**.**Label**(**ttab10**,**

2300 text**=**"Enter Slit Seperation"**).**grid**(**column**=**0**,** row**=**4**)**

2301 ttk**.**Label**(**ttab10**,**

2302 text**=**"Enter Fringe Number"**).**grid**(**column**=**0**,** row**=**5**)**

2303 enter **=** tkinter**.**StringVar**()**

2304 e111 **=** tkinter**.**Entry**(**ttab10**,** textvariable**=**enter**)**

2305 e111**.**insert**(**0**,** 'Find the diffrtaction angle of the maxima order n'**)**

2306 e111**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2307 enter1 **=** tkinter**.**StringVar**()**

2308 e211**=** tkinter**.**Entry**(**ttab10**,** textvariable**=**enter1**)**

2309 e211**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2310 enter2 **=** tkinter**.**StringVar**()**

2311 e311 **=** tkinter**.**Entry**(**ttab10**,** textvariable**=**enter2**)**

2312 e311**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2313 enter3 **=** tkinter**.**StringVar**()**

2314 e411 **=** tkinter**.**Entry**(**ttab10**,** textvariable**=**enter3**)**

2315 e411**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2316 enter4 **=** tkinter**.**StringVar**()**

2317 e511 **=** tkinter**.**Entry**(**ttab10**,** textvariable**=**enter4**)**

2318 e511**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2319 B1 **=** tkinter**.**Button**(**ttab10**,** text**=**"Set"**,**

2320 command**=lambda:** **[**c**.**questions**(**e211**.**get**(),** e111**.**get**(),** **[float(**e311**.**get**()),** **float(**e411**.**get**()),** **float(**e511**.**get**())]),**

2321 self**.**main**.**destroy**()])**

2322 B1**.**grid**(**row**=**6**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2323

2324 **elif** "critcal" **and** "type2" **in** i**:**

2325 ttab11 **=** ttk**.**Frame**(**intertab1**,** width**=**400**,** height**=**280**)**

2326 intertab1**.**add**(**ttab11**,** text**=**'Critical Angle'**)**

2327 ttk**.**Label**(**ttab11**,**

2328 text**=**"Question"**).**grid**(**column**=**0**,** row**=**1**)**

2329 ttk**.**Label**(**ttab11**,**

2330 text**=**"Time Set"**).**grid**(**column**=**0**,** row**=**2**)**

2331 ttk**.**Label**(**ttab11**,**

2332 text**=**"Enter Incidence Angle"**).**grid**(**column**=**0**,** row**=**3**)**

2333 ttk**.**Label**(**ttab11**,**

2334 text**=**"Enter Refraction Angle"**).**grid**(**column**=**0**,** row**=**4**)**

2335 enter **=** tkinter**.**StringVar**()**

2336 e121 **=** tkinter**.**Entry**(**ttab11**,** textvariable**=**enter**)**

2337 e121**.**insert**(**0**,** 'Find the critical angle of object'**)**

2338 e121**.**grid**(**row**=**1**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2339 enter1 **=** tkinter**.**StringVar**()**

2340 e221 **=** tkinter**.**Entry**(**ttab11**,** textvariable**=**enter1**)**

2341 e221**.**grid**(**row**=**2**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2342 enter2 **=** tkinter**.**StringVar**()**

2343 e321 **=** tkinter**.**Entry**(**ttab11**,** textvariable**=**enter2**)**

2344 e321**.**grid**(**row**=**3**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2345 enter3 **=** tkinter**.**StringVar**()**

2346 e421 **=** tkinter**.**Entry**(**ttab11**,** textvariable**=**enter3**)**

2347 e421**.**grid**(**row**=**4**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2348 B1 **=** tkinter**.**Button**(**ttab11**,** text**=**"Set"**,**

2349 command**=lambda:** **[**c**.**questions**(**e221**.**get**(),** e121**.**get**(),** **[float(**e321**.**get**()),** **float(**e421**.**get**())]),**

2350 self**.**main**.**destroy**()])**

2351 B1**.**grid**(**row**=**5**,** column**=**1**,** sticky**=**tkinter**.**E**)**

2352

2353 intertab1**.**pack**(**expand**=**1**,** fill**=**"both"**)**

2354 self**.**main**.**mainloop**()**

2355

2356 ##### stop #######

2357 # Parameters :- None

2358 # Return Type :- None

2359 # Purpose :- destroys the gui

2360 ###########################

2361 **def** stop**(**self**):**

2362 self**.**main**.**destroy**()**

2363

2364 ##### AutoGen #######

2365 # Parameters :- None

2366 # Return Type :- qs:list

2367 # Purpose :- depending on what is being displayed (in terms of properties) it creates questions

2368 ###########################

2369 **def** AutoGen**(**self**):**

2370 qs **=** **[]**

2371 **if** m**.**RI **and** **(**"Block" **or** "GlassBlock" **or** "Semi"**)** **in** **str(**self**.**objects**):**

2372 qs**.**append**(**"critical angle type1"**)**

2373 **if** m**.**IA **and** m**.**RA **and** **(not** m**.**RI**)** **and** **(**"Block" **or** "GlassBlock" **or** "Semi"**)** **in** **str(**self**.**objects**):**

2374 qs**.**append**(**"refractive index"**)**

2375 qs**.**append**(**"critical angle type2"**)**

2376

2377 **if** m**.**SlitS **and** m**.**SourceS **and** m**.**W **and** **(not** m**.**FS**)** **and** diff\_mode**:**

2378 qs**.**append**(**"fringes"**)**

2379 **if** m**.**FS **and** m**.**SourceS **and** m**.**W **and** **(not** m**.**SlitS**)** **and** diff\_mode**:**

2380 qs**.**append**(**"slits"**)**

2381

2382 **if** m**.**SlitS **and** m**.**FS **and** m**.**W **and** **(not** m**.**SourceS**)** **and** diff\_mode**:**

2383 qs**.**append**(**"slits and screen"**)**

2384

2385 **if** m**.**SlitS **and** m**.**SourceS **and** m**.**FS **and** **(not** m**.**W**)** **and** diff\_mode**:**

2386 qs**.**append**(**"wavelength"**)**

2387 qs**.**append**(**"order type1"**)**

2388

2389 **if** m**.**IA **and** m**.**RI **and** **(not** m**.**RA**)** **and** **(**"Block" **or** "GlassBlock" **or** "Semi"**)** **in** **str(**self**.**objects**):**

2390 qs**.**append**(**"refraction"**)**

2391

2392 **if** m**.**RA **and** m**.**RI **and** **(not** m**.**IA**)** **and** **(**"Block" **or** "GlassBlock" **or** "Semi"**)** **in** **str(**self**.**objects**):**

2393 qs**.**append**(**"incidence"**)**

2394

2395 **if** m**.**W **and** m**.**SlitS **and** diff\_mode**:**

2396 qs**.**append**(**"order type2"**)**

2397 **return** qs

2398

2399

2400##### quit #######

2401# Parameters :- None

2402# Return Type :- info:list

2403# Purpose :- when exiting gui it returns crucial information for analysis

2404###########################

2405def **quit(**c**):**

2406 pygame**.quit()**

2407 n**.**disconnect**()**

2408 n**.**disconnect2**()**

2409 end\_time **=** time**.**time**()**

2410 dur **=** **(**end\_time **-** start\_time**)** **/** 60

2411 dur **=** **round(**dur**,** 2**)**

2412 **if** c **is** **not** **None:**

2413 c **=** c**.**stop**()**

2414 **else:**

2415 c **=** **[None,** **None]**

2416 **with** **open(**"return.txt"**,** "r"**)** **as** f**:**

2417 qu **=** f**.**read**()**

2418 f**.**close**()**

2419 qu **=** **eval(**qu**)**

2420 Report**.**Reported**(**qu**)**

2421 **exit()**

2422

2423

2424m **=** ShowGUI**()**

2425c **=** **None**

2426diff\_mode **=** **False**

2427n **=** Network**()**

2428n.connect**()**

2429t **=** 0

2430

2431while **True:**

2432 # Tool bar

2433 change **=** **False**

2434 toolbar\_button **=** **[]**

2435 objects\_loaded **=** **list(dict.**fromkeys**(**objects\_loaded**))**

2436 sources\_loaded **=** **list(dict.**fromkeys**(**sources\_loaded**))**

2437 pygame**.**draw**.**rect**(**screen**,** **(**105**,** 105**,** 105**),** pygame**.**Rect**(**0**,** 0**,** width**,** 75**))**

2438 width **=** screen**.**get\_width**()**

2439 height **=** screen**.**get\_height**()**

2440 Open **=** Button**(**8**,** "Open"**)**

2441 Open**.**draw**(**"Open"**)**

2442 New **=** Button**(**0**,** "New"**)**

2443 New**.**draw**(**"New"**)**

2444 Block\_btn **=** Button**(**16**,** "Block\_btn"**)**

2445 Block\_btn**.**draw**(**"Block"**)**

2446 GlassBlock\_btn **=** Button**(**24**,** "GlassBlock\_btn"**)**

2447 GlassBlock\_btn**.**draw**(**"GlassBlock"**)**

2448 Mirror\_btn **=** Button**(**32**,** "Mirror\_btn"**)**

2449 Mirror\_btn**.**draw**(**"Mirror"**)**

2450 Screen\_btn **=** Button**(**40**,** "Screen\_btn"**)**

2451 Screen\_btn**.**draw**(**"Screen"**)**

2452 Semi\_Circle\_btn **=** Button**(**48**,** "Semi\_Circle\_btn"**)**

2453 Semi\_Circle\_btn**.**draw**(**"Semi Circle"**)**

2454 Diffraction\_btn **=** Button**(**56**,** "Diffraction\_btn"**)**

2455 Diffraction\_btn**.**draw**(**"Diffraction"**)**

2456 Source\_btn **=** Button**(**64**,** "Source\_btn"**)**

2457 Source\_btn**.**draw**(**"Source"**)**

2458 Question **=** Button**(**72**,** "Question"**)**

2459 Question**.**draw**(**"Question"**)**

2460 Del\_All **=** Button**(**80**,** "Del\_All"**)**

2461 Del\_All**.**draw**(**"Del All"**)**

2462 Quit **=** Button**(**88**,** "Quit"**)**

2463 Quit**.**draw**(**"Quit"**)**

2464 Chat **=** Button**(**96**,** "Chat"**)**

2465 Chat**.**draw**(**"Chat"**)**

2466 Save **=** Button**(**104**,** "Save"**)**

2467 Save**.**draw**(**"Save"**)**

2468 Show **=** Button**(**112**,** "Show"**)**

2469 Show**.**draw**(**"Show"**)**

2470 Change **=** Button**(**120**,** "Change"**)**

2471 Change**.**draw**(**"Change"**)**

2472 obj **=** **None**

2473 **if** diff\_mode**:**

2474 **if** "Diffraction" **in** **str(**objects\_loaded**):**

2475 **for** i **in** objects\_loaded**:**

2476 **if** "D" **in** i**.**defined\_name**:**

2477 obj **=** i

2478 **else:**

2479 Tag **=** tag\_generator**(**tag**,** "D"**)**

2480 obj **=** Diffraction**(**Tag**)**

2481 objects\_loaded**.**append**(**obj**)**

2482 obj**.**redraw1**()**

2483 **for** event **in** pygame**.**event**.**get**():**

2484 pos **=** pygame**.**mouse**.**get\_pos**()**

2485 x\_axis **=** pos**[**0**]**

2486 y\_axis **=** pos**[**1**]**

2487 **if** event**.type** **==** pygame**.**QUIT**:**

2488 pygame**.quit()**

2489 # n.send(bytes("STOP", encoding="utf-8"))

2490 **quit(**c**)**

2491 sys**.exit()**

2492

2493 **elif** event**.type** **==** pygame**.**MOUSEBUTTONDOWN **and** y\_axis **>=** 100**:**

2494 **if** event**.**button **==** 1**:**

2495 **if** obj**.**graphical**.**collidepoint**(**event**.**pos**)** **or** obj**.**graphical2**.**collidepoint**(**

2496 event**.**pos**)** **or** obj**.**graphical3**.**collidepoint**(**event**.**pos**)** **or** obj**.**graphical4**.**collidepoint**(**

2497 event**.**pos**):**

2498 obj**.**draging **=** **True**

2499 obj**.**diffraction**()**

2500 obj**.**redraw1**()**

2501 change **=** **True**

2502 **elif** obj**.**source**.**collidepoint**(**event**.**pos**):**

2503 obj**.**source\_draging **=** **True**

2504 obj**.**diffraction**()**

2505 obj**.**redraw1**()**

2506 change **=** **True**

2507

2508 **elif** event**.type** **==** pygame**.**MOUSEBUTTONUP **and** y\_axis **>=** 100**:**

2509 **if** event**.**button **==** 1 **and** obj**.**draging**:**

2510 obj**.**diffraction**()**

2511 obj**.**redraw1**()**

2512 obj**.**draging **=** **False**

2513 change **=** **True**

2514 **elif** event**.**button **==** 1 **and** obj**.**source\_draging**:**

2515 obj**.**source\_draging **=** **False**

2516 obj**.**diffraction**()**

2517 obj**.**redraw1**()**

2518 change **=** **True**

2519

2520 **elif** event**.type** **==** pygame**.**MOUSEMOTION **and** y\_axis **>=** 100**:**

2521 **if** obj**.**draging**:**

2522 mouse\_x**,** mouse\_y **=** event**.**pos

2523 obj**.**graphical**.**x **=** mouse\_x

2524 obj**.**graphical**.**y **=** mouse\_y

2525 obj**.**redraw**(**mouse\_x**,** **None)**

2526 obj**.**diffraction**()**

2527 obj**.**redraw1**()**

2528 change **=** **True**

2529 **elif** obj**.**source\_draging**:**

2530 mouse\_x**,** mouse\_y **=** event**.**pos

2531 obj**.**sourceredraw**(**mouse\_x**)**

2532 obj**.**diffraction**()**

2533 obj**.**redraw1**()**

2534 change **=** **True**

2535

2536 **elif** event**.type** **==** pygame**.**MOUSEBUTTONDOWN **and** y\_axis **<=** 75**:**

2537 **for** i **in** toolbar\_button**:**

2538 clicktest **=** **eval(**i**).**click**(**event**)**

2539 **if** clicktest **==** "Diffraction\_btn"**:**

2540 diff\_mode **=** **False**

2541 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

2542 **for** i **in** objects\_loaded**:**

2543 **if** "D" **not** **in** i**.**defined\_name**:**

2544 i**.**redraw1**()**

2545 **for** i **in** sources\_loaded**:**

2546 i**.**redraw1**()**

2547 **elif** clicktest **==** "Change"**:**

2548 n **=** ChangeGUI**()**

2549 **elif** clicktest **==** "Show"**:**

2550 m**.**gui\_loop**()**

2551

2552 **if** change **and** diff\_mode**:**

2553 so **=** **[vars(**m**)]**

2554 **if** "main" **in** **vars(**m**):**

2555 temp1 **=** so**[**0**][**"main"**]**

2556 so**[**0**][**"main"**]** **=** **str(**temp1**)**

2557 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2558 so**[**0**][**"main"**]** **=** temp1

2559 **else:**

2560 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2561

2562 **for** item **in** objects\_loaded**:**

2563 **if** "Diffraction" **in** **str(**item**):**

2564 so **=** **[vars(**item**)]**

2565 temp1 **=** so**[**0**][**"graphical"**]**

2566 temp2 **=** so**[**0**][**"graphical2"**]**

2567 temp3 **=** so**[**0**][**"graphical3"**]**

2568 temp4 **=** so**[**0**][**"graphical4"**]**

2569 temp5 **=** so**[**0**][**"source"**]**

2570 so**[**0**][**"interceptors"**]** **=** **[]**

2571 so**[**0**][**"graphical"**]** **=** ""

2572 so**[**0**][**"graphical2"**]** **=** ""

2573 so**[**0**][**"graphical3"**]** **=** ""

2574 so**[**0**][**"graphical4"**]** **=** ""

2575 so**[**0**][**"source"**]** **=** ""

2576 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2577 so**[**0**][**"graphical"**]** **=** temp1

2578 so**[**0**][**"graphical2"**]** **=** temp2

2579 so**[**0**][**"graphical3"**]** **=** temp3

2580 so**[**0**][**"graphical4"**]** **=** temp4

2581 so**[**0**][**"source"**]** **=** temp5

2582 **for** item2 **in** DiffRays**[**item**.**defined\_name**]:**

2583 so2 **=** **[vars(**item2**)]**

2584 temp1 **=** so2**[**0**][**"graphical"**]**

2585 temp2 **=** so2**[**0**][**"graphical2"**]**

2586 so2**[**0**][**"interceptors"**]** **=** **[]**

2587 so2**[**0**][**"previousobj"**]** **=** **[]**

2588 so2**[**0**][**"normals"**]** **=** **[]**

2589 so2**[**0**][**"angles"**]** **=** **[]**

2590 so2**[**0**][**"graphical"**]** **=** **str(**so2**[**0**][**"graphical"**])**

2591 so2**[**0**][**"graphical2"**]** **=** **str(**so2**[**0**][**"graphical2"**])**

2592 n**.**send**(bytes(str(**so2**),** encoding**=**"utf-8"**))**

2593 so2**[**0**][**"graphical"**]** **=** temp1

2594 so2**[**0**][**"graphical2"**]** **=** temp2

2595 n**.**send**(bytes(**"PAUSE"**,** encoding**=**"utf-8"**))**

2596

2597 **else:**

2598 # Events for when a mouse movements are made

2599 **for** event **in** pygame**.**event**.**get**():**

2600 pos **=** pygame**.**mouse**.**get\_pos**()**

2601 x\_axis **=** pos**[**0**]**

2602 y\_axis **=** pos**[**1**]**

2603 **if** event**.type** **==** pygame**.**QUIT**:**

2604 n**.**send**(bytes(**"STOP"**,** encoding**=**"utf-8"**))**

2605 **quit(**c**)**

2606 sys**.exit()**

2607 **elif** event**.type** **==** pygame**.**MOUSEBUTTONDOWN **and** y\_axis **>=** 100**:**

2608 **for** i **in** objects\_loaded**:**

2609 **if** event**.**button **==** 1**:**

2610 **if** i**.**graphical**.**collidepoint**(**event**.**pos**):**

2611 **if** "SC" **in** i**.**defined\_name**:**

2612 **if** event**.**pos**[**1**]** **>** i**.**yp**:**

2613 **pass**

2614 **else:**

2615 i**.**draging **=** **True**

2616 i**.**Simulation**(False,** **None,** **True)**

2617 change **=** **True**

2618 **else:**

2619 i**.**draging **=** **True**

2620 i**.**Simulation**(False,** **None,** **True)**

2621 change **=** **True**

2622 **for** i **in** sources\_loaded**:**

2623 **if** event**.**button **==** 1**:**

2624 **if** i**.**graphical**.**collidepoint**(**event**.**pos**):**

2625 **if** i**.**graphical2**.**collidepoint**(**event**.**pos**):**

2626 i**.**t\_status **=** 1

2627 i**.**Simulation**(False,** **None,** **False)**

2628 i**.**redraw1**()**

2629 i**.**draging **=** **True**

2630 change **=** **True**

2631

2632 **else:**

2633 i**.**Simulation**(False,** **None,** **False)**

2634 i**.**redraw1**()**

2635 i**.**draging **=** **True**

2636 change **=** **True**

2637 **elif** event**.type** **==** pygame**.**MOUSEBUTTONUP **and** y\_axis **>=** 100**:**

2638 **for** i **in** objects\_loaded**:**

2639 **if** event**.**button **==** 1 **and** i**.**draging**:**

2640 i**.**Simulation**(False,** **None,** **True)**

2641 **for** item **in** i**.**interceptors**:**

2642 item**.**redraw1**()**

2643 i**.**draging **=** **False**

2644 change **=** **True**

2645 **for** i **in** sources\_loaded**:**

2646 **if** event**.**button **==** 1 **and** i**.**draging**:**

2647 **if** i**.**t\_status **==** 1**:**

2648 mousex**,** mousey **=** event**.**pos

2649 i**.**rotate**(**mousex**,** mousey**)**

2650 i**.**Simulation**(False,** **None,** **False)**

2651 i**.**redraw1**()**

2652 i**.**draging **=** **False**

2653 i**.**t\_status **=** 0

2654 change **=** **True**

2655 **else:**

2656 i**.**Simulation**(False,** **None,** **False)**

2657 i**.**redraw1**()**

2658 i**.**draging **=** **False**

2659 change **=** **True**

2660 **elif** event**.type** **==** pygame**.**MOUSEMOTION **and** y\_axis **>=** 100**:**

2661 **for** i **in** objects\_loaded**:**

2662 **if** i**.**draging**:**

2663 mouse\_x**,** mouse\_y **=** event**.**pos

2664 i**.**redraw**(**mouse\_x**,** mouse\_y**)**

2665 i**.**Simulation**(False,** **None,** **True)**

2666 change **=** **True**

2667

2668 **for** i **in** sources\_loaded**:**

2669 **if** i**.**draging**:**

2670 mouse\_x**,** mouse\_y **=** event**.**pos

2671 **if** i**.**t\_status **==** 1**:**

2672 i**.**Simulation**(False,** **None,** **False)**

2673 i**.**redraw1**()**

2674 change **=** **True**

2675

2676 **else:**

2677 i**.**redraw**(**mouse\_x**,** mouse\_y**)**

2678 i**.**Simulation**(False,** **None,** **False)**

2679 i**.**redraw1**()**

2680 change **=** **True**

2681 # dictates what happens when a button is pressed

2682 **elif** event**.type** **==** pygame**.**MOUSEBUTTONDOWN **and** y\_axis **<=** 75**:**

2683 **for** i **in** toolbar\_button**:**

2684 clicktest **=** **eval(**i**).**click**(**event**)**

2685 **if** clicktest **==** "Block\_btn"**:**

2686 Tag **=** tag\_generator**(**tag**,** "B"**)**

2687 objects\_loaded**.**append**(**Block**(**Tag**))**

2688 **elif** clicktest **==** "GlassBlock\_btn"**:**

2689 Tag **=** tag\_generator**(**tag**,** "GB"**)**

2690 objects\_loaded**.**append**(**GlassBlock**(**Tag**))**

2691 **elif** clicktest **==** "Mirror\_btn"**:**

2692 Tag **=** tag\_generator**(**tag**,** "M"**)**

2693 objects\_loaded**.**append**(**Mirror**(**Tag**))**

2694 **elif** clicktest **==** "Screen\_btn"**:**

2695 Tag **=** tag\_generator**(**tag**,** "S"**)**

2696 objects\_loaded**.**append**(**Screen**(**Tag**))**

2697 **elif** clicktest **==** "Semi\_Circle\_btn"**:**

2698 Tag **=** tag\_generator**(**tag**,** "SC"**)**

2699 objects\_loaded**.**append**(**Semi\_Circle**(**Tag**))**

2700 **elif** clicktest **==** "Diffraction\_btn"**:**

2701 diff\_mode **=** **True**

2702 **elif** clicktest **==** "Source\_btn"**:**

2703 Tag **=** tag\_generator**(**tag**,** "R"**)**

2704 sources\_loaded**.**append**(**Source**(**Tag**))**

2705 **elif** clicktest **==** "Quit"**:**

2706 # n.send(bytes("STOP", encoding="utf-8"))

2707 **quit(**c**)**

2708 sys**.exit()**

2709 **elif** clicktest **==** "Chat"**:**

2710 c **=** ChatboxServer**.**Server**()**

2711 **elif** clicktest **==** "Del\_All"**:**

2712 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

2713 objects\_loaded **=** **[]**

2714 sources\_loaded **=** **[]**

2715 tag **=** **[**0**]**

2716 **elif** clicktest **==** "Save"**:**

2717 **with** **open(**"savegame"**,** "wb"**)** **as** f**:**

2718 data **=** **[]**

2719 **if** objects\_loaded**:**

2720 data **+=** objects\_loaded

2721 **else:**

2722 **pass**

2723 **if** sources\_loaded**:**

2724 data **+=** sources\_loaded

2725 **else:**

2726 **pass**

2727 pickle**.**dump**(**data**,** f**)**

2728 f**.**close**()**

2729

2730 **elif** clicktest **==** "Open"**:**

2731 objects\_loaded **=** **[]**

2732 sources\_loaded **=** **[]**

2733 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

2734 **with** **open(**"savegame"**,** "rb"**)** **as** f**:**

2735 loaded **=** pickle**.**load**(**f**)**

2736 f**.**close**()**

2737 **for** pos **in** loaded**:**

2738 **if** "Source" **in** **str(**pos**):**

2739 sources\_loaded**.**append**(**pos**)**

2740 **else:**

2741 objects\_loaded**.**append**(**pos**)**

2742 **if** objects\_loaded**:**

2743 **for** items **in** objects\_loaded**:**

2744 items**.**redraw1**()**

2745 **else:**

2746 **pass**

2747 **if** sources\_loaded**:**

2748 **for** items **in** sources\_loaded**:**

2749 items**.**redraw1**()**

2750 **else:**

2751 **pass**

2752 **elif** clicktest **==** "Show"**:**

2753 m**.**gui\_loop**()**

2754 **elif** clicktest **==** "Change"**:**

2755 cng **=** ChangeGUI**()**

2756 **elif** clicktest **==** "Question"**:**

2757 **if** c **is** **None:**

2758 **pass**

2759 **else:**

2760 QuestionsGUI**(**c**)**

2761 # detects if a change has been made and sends the changes to the students/ networking part

2762 **if** change **and** **not** diff\_mode**:**

2763 incidentsbefore **=** IncidentRays**.**values**()**

2764 incidentsafter **=** **[]**

2765 **for** lis **in** incidentsbefore**:**

2766 **if** lis**:**

2767 incidentsafter **=** incidentsafter **+** lis

2768 incidentsafter **=** **list(dict.**fromkeys**(**incidentsafter**))**

2769 so **=** **[vars(**m**)]**

2770 **if** "main" **in** **vars(**m**):**

2771 temp1 **=** so**[**0**][**"main"**]**

2772 so**[**0**][**"main"**]** **=** **str(**temp1**)**

2773 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2774 so**[**0**][**"main"**]** **=** temp1

2775 **else:**

2776 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2777 **if** sources\_loaded**:**

2778 **for** item **in** sources\_loaded**:**

2779 so **=** **[vars(**item**)]**

2780 temp1 **=** so**[**0**][**"graphical"**]**

2781 temp2 **=** so**[**0**][**"graphical2"**]**

2782 so**[**0**][**"interceptors"**]** **=** **[]**

2783 so**[**0**][**"previousobj"**]** **=** **[]**

2784 so**[**0**][**"normals"**]** **=** **[]**

2785 so**[**0**][**"angles"**]** **=** **[]**

2786 so**[**0**][**"graphical"**]** **=** **str(**so**[**0**][**"graphical"**])**

2787 so**[**0**][**"graphical2"**]** **=** **str(**so**[**0**][**"graphical2"**])**

2788 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2789 so**[**0**][**"graphical"**]** **=** temp1

2790 so**[**0**][**"graphical2"**]** **=** temp2

2791 **if** objects\_loaded**:**

2792 **for** item **in** objects\_loaded**:**

2793 **if** "Semi" **in** **str(**item**):**

2794 so **=** **[vars(**item**)]**

2795 temp1 **=** so**[**0**][**"graphical"**]**

2796 temp2 **=** so**[**0**][**"graphical2"**]**

2797 so**[**0**][**"interceptors"**]** **=** **[]**

2798 so**[**0**][**"graphical"**]** **=** **str(**so**[**0**][**"graphical"**])**

2799 so**[**0**][**"graphical2"**]** **=** **str(**so**[**0**][**"graphical2"**])**

2800 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2801 so**[**0**][**"graphical"**]** **=** temp1

2802 so**[**0**][**"graphical2"**]** **=** temp2

2803 **elif** "Diffraction" **in** **str(**item**):**

2804 **pass**

2805 **else:**

2806 so **=** **[vars(**item**)]**

2807 temp1 **=** so**[**0**][**"graphical"**]**

2808 so**[**0**][**"interceptors"**]** **=** **[]**

2809 so**[**0**][**"graphical"**]** **=** **str(**so**[**0**][**"graphical"**])**

2810 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2811 so**[**0**][**"graphical"**]** **=** temp1

2812 **if** incidentsafter**:**

2813 **for** item **in** incidentsafter**:**

2814 **if** "Source" **in** **str(**item**):**

2815 so **=** **[vars(**item**)]**

2816 temp1 **=** so**[**0**][**"graphical"**]**

2817 temp2 **=** so**[**0**][**"graphical2"**]**

2818 so**[**0**][**"interceptors"**]** **=** **[]**

2819 so**[**0**][**"previousobj"**]** **=** **[]**

2820 so**[**0**][**"normals"**]** **=** **[]**

2821 so**[**0**][**"angles"**]** **=** **[]**

2822 so**[**0**][**"graphical"**]** **=** **str(**so**[**0**][**"graphical"**])**

2823 so**[**0**][**"graphical2"**]** **=** **str(**so**[**0**][**"graphical2"**])**

2824 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2825 so**[**0**][**"graphical"**]** **=** temp1

2826 so**[**0**][**"graphical2"**]** **=** temp2

2827 **else:**

2828 so **=** **[vars(**item**)]**

2829 n**.**send**(bytes(str(**so**),** encoding**=**"utf-8"**))**

2830 n**.**send**(bytes(**"PAUSE"**,** encoding**=**"utf-8"**))**

2831 pygame**.**display**.**flip**()**

## Account.py

1 **import** sqlite3

2 **import** os

3 **import** hashlib

4 **import** pickle

5 **from** tkinter **import** **\***

6 **from** tkinter **import** ttk

7 **import** smtplib**,** ssl

8 **from** email**.**mime**.**text **import** **\***

9 **from** email**.**mime**.**multipart **import** MIMEMultipart

10 **import** os

11 **import** re

12 **import** Unique\_Code

13 **import** subprocess

14 **import** socket

15 **import** urllib**.**request

16 **import** csv

17 **import** Report

18 ##### Create\_SQL #######

19 # Parameters :- None

20 # Purpose :- This class holds methods for frequently used sql queries

21 ###########################

22 **class** **Create\_SQL:**

23 **def** \_\_init\_\_**(**self**):**

24 self**.**dbname **=** "Account.db"

25 self**.**db**=**sqlite3**.**connect**(**self**.**dbname**)**

26 self**.**cursor **=** self**.**db**.**cursor**()**

27

28 ##### Create\_ClassStu #######

29 # Parameters :- classid:int, studentid:int, accid:int

30 # Return Type :- None

31 # Purpose :- adds student into a particular class

32 #######################

33 **def** Create\_ClassStu**(**self**,**classid**,**studentid**,**accid**):**

34 sql **=** "INSERT INTO Class(ClassID, StudentsID, AccountID) VALUES(?,?,?)"

35 self**.**cursor**.**execute**(**sql**,** **(**classid**,** studentid**,** accid**))**

36 self**.**db**.**commit**()**

37

38 ##### Create\_Accounts #######

39 # Parameters :- Title:string, Firstname:string, Surname:string, School:string, Email:string, userID:int

40 # Return Type :- None

41 # Purpose :- creates an account for teachers

42 #######################

43 **def** Create\_Accounts**(**self**,** Title**,** Firstname**,** Surname**,** School**,** Email**,** userID**):**

44 values **=** **(**Title**,** Firstname**,** Surname**,** School**,** Email**,**userID**)**

45 sql **=** "insert into Account(Title,Firstname,Surname,School,Email,UsersID) values (?,?,?,?,?,?)"

46 self**.**cursor**.**execute**(**sql**,** values**)**

47 self**.**db**.**commit**()**

48

49 ##### Session #######

50 # Parameters :- FileID:int, ClassID:int, Date:string, Duration:float, StudentsPresent:int

51 # Return Type :- None

52 # Purpose :- creates session entity for recently complete session

53 #######################

54 **def** Session**(**self**,** FileID**,** ClassID**,** Date**,** Duration**,** StudentsPresent**):**

55 values **=** **(**FileID**,** ClassID**,** Date**,** Duration**,** StudentsPresent**)**

56 sql **=** "insert into Session(FileID,ClassID, Date,Duration,StudentsPresent) values (?,?,?,?,?)"

57 self**.**cursor**.**execute**(**sql**,** values**)**

58 self**.**db**.**commit**()**

59

60 ##### Create\_Users #######

61 # Parameters :- Username:string, Password:string

62 # Return Type :- None

63 # Purpose :- adds account login requirements to seperate table

64 #######################

65 **def** Create\_Users**(**self**,** Username**,** Password**):**

66 values **=** **(**Username**,**Password**)**

67 sql **=** "insert into Users(Username, Password) values (?,?)"

68 self**.**cursor**.**execute**(**sql**,** values**)**

69 self**.**db**.**commit**()**

70 sql **=** "SELECT UsersID FROM Users"

71 self**.**cursor**.**execute**(**sql**)**

72 ids **=** self**.**cursor**.**fetchall**()**

73 sorting **=** **[]**

74 **for** i **in** ids**:**

75 **if** **list(**i**)[**0**]** **==** **None:**

76 sorting**.**append**(**0**)**

77 **else:**

78 sorting**.**append**(list(**i**)[**0**])**

79 sorting **=** **sorted(**sorting**)**

80 **return** sorting**[-**1**]**

81

82 ##### Create\_Class #######

83 # Parameters :- StudentID:int, AccountID:int

84 # Return Type :- None

85 # Purpose :- creates a new class

86 #######################

87 **def** Create\_Class**(**self**,** StudentID**,** AccountID**):**

88 values **=** **(**StudentID**,**AccountID**)**

89 sql **=** "insert into Class(StudentsID, AccountID) values (?,?)"

90 self**.**cursor**.**execute**(**sql**,** values**)**

91 self**.**db**.**commit**()**

92

93 ##### Create\_Students #######

94 # Parameters :- Firstname:string, Surname:string, Email:string

95 # Return Type :- None

96 # Purpose :- creates new student

97 #######################

98 **def** Create\_Students**(**self**,**Firstname**,**Surname**,**Email**):**

99 values **=** **(**Firstname**,** Surname**,** Email**)**

100 sql **=** "insert into Students(Firstname,Surname,Email) values (?,?,?)"

101 self**.**cursor**.**execute**(**sql**,** values**)**

102 self**.**db**.**commit**()**

103

104 ##### Edit #######

105 # Parameters :- Table:string, ID:int, Field:string, Editted:string

106 # Return Type :- None

107 # Purpose :- edits a specific field in a specific table

108 #######################

109 **def** Edit**(**self**,** Table**,** ID**,** Field**,** Editted**):**

110 sql **=** "update {0} set {1}=? where {0}ID=?"**.format(**Table**,** Field**)**

111 self**.**cursor**.**execute**(**sql**,** **(**Editted**,** ID**))**

112 self**.**db**.**commit**()**

113

114 ##### Delete #######

115 # Parameters :- Table:string, ID:int

116 # Return Type :- None

117 # Purpose :- deletes any entity in any table

118 #######################

119 **def** Delete**(**self**,** Table**,** ID**):**

120 sql **=** "delete from {0} where {0}ID=?"**.format(**Table**)**

121 self**.**cursor**.**execute**(**sql**,** ID**)**

122 self**.**db**.**commit**()**

123

124##### Account #######

125# Parameters :- ID:int, username:string, accID:int

126# Return Type :- None

127# Purpose :- class for main account window and the windows it branches out to

128#######################

129class Account**:**

130 **def** \_\_init\_\_**(**self**,** ID**,** username**,** accID**):**

131 self**.**ID **=** ID

132 self**.**username **=** username

133 self**.**main **=** Tk**()**

134 self**.**main**.**geometry**(**"150x205"**)**

135 self**.**accId **=** accID**[**0**]**

136 self**.**create **=** Create\_SQL**()**

137 self**.**hostIP **=**socket**.**gethostbyname**(**socket**.**gethostname**())**

138 self**.**start**()**

139 self**.**db**=**sqlite3**.**connect**(**"Account.db"**)**

140 self**.**cursor **=** self**.**db**.**cursor**()**

141

142 ##### back\_reg #######

143 # Parameters :- win:object

144 # Return Type :- None

145 # Purpose :- allows users to go back to first window

146 #######################

147 **def** back\_reg**(**self**,** win**):**

148 win**.**destroy**()**

149 n**.**Main**()**

150

151 ##### start #######

152 # Parameters :- None

153 # Return Type :- None

154 # Purpose :- generates main menu within the account

155 #######################

156 **def** start**(**self**):**

157 btn **=** Button**(**self**.**main**,** text**=**"Edit Account"**,** width**=**20**,**

158 command**=lambda:** self**.**edit\_account**())**

159 btn**.**pack**()**

160 btn1 **=** Button**(**self**.**main**,** text**=**"Add Class"**,** width**=**20**,**

161 command**=lambda:** self**.**add\_class**())**

162 btn1**.**pack**()**

163 btn2 **=** Button**(**self**.**main**,** text**=**"Edit Class"**,** width**=**20**,**

164 command**=lambda:** self**.**edit\_class**())**

165 btn2**.**pack**()**

166 btn3 **=** Button**(**self**.**main**,** text**=**"New Session"**,** width**=**20**,**

167 command**=lambda:** self**.**code**(**self**.**main**))**

168 btn3**.**pack**()**

169 btn4 **=** Button**(**self**.**main**,** text**=**"View Class"**,** width**=**20**,**

170 command**=lambda:** self**.**view\_class1**())**

171 btn4**.**pack**()**

172 btn5 **=** Button**(**self**.**main**,** text**=**"Offline Mode"**,** width**=**20**,**

173 command**=lambda:** self**.**offline**(**self**.**main**))**

174 btn5**.**pack**()**

175 btn6 **=** Button**(**self**.**main**,** text**=**"LogOut"**,** width**=**20**,**

176 command**=**self**.**log\_out**)**

177 btn6**.**pack**()**

178

179 ##### offline #######

180 # Parameters :- None

181 # Return Type :- None

182 # Purpose :- runs the envoirnment without connecting it to a network

183 #######################

184 **def** offline**(**self**,**win**):**

185 win**.**destroy**()**

186 off **=** subprocess**.**call**((**'python OSD.py'**))**

187 n**.**Main**()**

188

189 ##### code #######

190 # Parameters :- win:object

191 # Return Type :- None

192 # Purpose :- window asks for the class id of the class you want to run the session with

193 #######################

194 **def** code**(**self**,** win**):**

195 stu **=** Toplevel**(**win**)**

196 classId **=** Label**(**stu**,** text**=**"ClassID"**)**

197 classId**.**grid**(**row**=**1**,** column**=**1**)**

198

199 c\_id\_text **=** StringVar**()**

200 e1 **=** Entry**(**stu**,** textvariable**=**c\_id\_text**)**

201 e1**.**grid**(**row**=**1**,** column**=**2**)**

202

203 btn **=** Button**(**stu**,** text**=**"Confirm"**,** width**=**20**,**

204 command**=lambda:** self**.**teach\_class**(**e1**.**get**(),** stu**))**

205 btn**.**grid**(**row**=**9**,** column**=**2**)**

206

207

208 ##### teach\_class #######

209 # Parameters :- cID:int ,win:object

210 # Return Type :- None

211 # Purpose :- sends email to everyone in class with code, loads the envoirnment, and intitialieses the report making process

212 #######################

213 **def** teach\_class**(**self**,** cID**,** win**):**

214 sql **=** "SELECT StudentsID,Email FROM Students INNER JOIN Class USING(StudentsID) WHERE ClassID=? AND AccountID=?"

215 self**.**cursor**.**execute**(**sql**,** **(int(**cID**),** **int(**self**.**accId**[**0**])))**

216 stu**=**self**.**cursor**.**fetchall**()**

217 **if** **len(**stu**):**

218 MasterCode **=** Unique\_Code**.**uni\_code**(**cID**)**

219 **for** i **in** stu**:**

220 sql **=** "SELECT Email FROM Account WHERE AccountID=?"

221 self**.**cursor**.**execute**(**sql**,** **(**self**.**accId**[**0**],))**

222 context **=** ssl**.**create\_default\_context**()**

223 msg **=** MIMEMultipart**()**

224 MESSAGE\_BODY **=** "Hello \n A session on the Optics Experimentation Environment has started with a class you are in the Code is:{0}"**.format(**

225 MasterCode**)**

226 body\_part **=** MIMEText**(**MESSAGE\_BODY**,** 'plain'**)**

227 msg**[**'Subject'**]** **=** "Session Starting"

228 msg**[**'From'**]** **=** "christmasshopmail@gmail.com"

229 msg**[**'To'**]** **=** i**[**1**]**

230 # Add body to email

231 msg**.**attach**(**body\_part**)**

232 # Create SMTP object

233 smtp\_obj **=** smtplib**.**SMTP\_SSL**(**"smtp.gmail.com"**,** 465**,** context**=**context**)**

234 # Login to the server

235 smtp\_obj**.**login**(**"christmasshopmail@gmail.com"**,** "testpassword123+"**)**

236 # Convert the message to a string and send it

237 smtp\_obj**.**sendmail**(**msg**[**'From'**],** msg**[**'To'**],** msg**.**as\_string**())**

238 smtp\_obj**.quit()**

239 sql **=** "INSERT INTO Code(MasterCode,HostIP,ClassID) VALUES(?,?,?)"

240 self**.**cursor**.**execute**(**sql**,** **(**MasterCode**,** self**.**hostIP**,** cID**))**

241 self**.**db**.**commit**()**

242 **print(**"before"**)**

243 **with** **open(**"return.txt"**,** "w"**)** **as** f**:**

244 f**.**write**(str([**MasterCode**,**self**.**accId**[**0**]]))**

245 f**.**close**()**

246 self**.**osd**(**self**.**main**,**win**)**

247 n**.**Main**()**

248 **else:**

249 popup **=** Toplevel**(**win**)**

250 popup**.**wm\_title**(**"Error!"**)**

251 label **=** Label**(**popup**,** text**=**"Class Error"**)**

252 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

253 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

254 B1**.**pack**()**

255 **def** osd**(**self**,**main**,**win**):**

256 win**.**destroy**()**

257 main**.**destroy**()**

258 subprocess**.**call**((**'python OSD.py'**))**

259 n**.**Main**()**

260

261 ##### log\_out #######

262 # Parameters :- None

263 # Return Type :- None

264 # Purpose :- logs the users out and brings them back to first window

265 #######################

266 **def** log\_out**(**self**):**

267 self**.**main**.**destroy**()**

268 n**.**Main**()**

269

270 ##### edit\_account #######

271 # Parameters :- None

272 # Return Type :- None

273 # Purpose :- allows you to edit a certain area of your account

274 #######################

275 **def** edit\_account**(**self**):**

276 editor **=** Toplevel**(**self**.**main**)**

277 editor**.**wm\_title**(**"Choose"**)**

278 label **=** Label**(**editor**,** text**=**"Which Element Do You Want To Edit"**)**

279 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

280 B1 **=** Button**(**editor**,** text**=**"Firstname"**,** command**=lambda:** self**.**edit\_sql**(**"Account"**,** "Firstname"**,** editor**))**

281 B1**.**pack**()**

282 B5 **=** Button**(**editor**,** text**=**"Lastname"**,** command**=lambda:** self**.**edit\_sql**(**"Account"**,** "Surname"**,** editor**))**

283 B5**.**pack**()**

284 B2 **=** Button**(**editor**,** text**=**"Password"**,** command**=lambda:** self**.**edit\_sql**(**"Users"**,** "Password"**,** editor**))**

285 B2**.**pack**()**

286 B3 **=** Button**(**editor**,** text**=**"Email"**,** command**=lambda:** self**.**edit\_sql**(**"Account"**,** "Email"**,** editor**))**

287 B3**.**pack**()**

288 B4 **=** Button**(**editor**,** text**=**"School"**,** command**=lambda:** self**.**edit\_sql**(**"Account"**,** "School"**,** editor**))**

289 B4**.**pack**()**

290 B6 **=** Button**(**editor**,** text**=**"Okay"**,** command**=**editor**.**destroy**)**

291 B6**.**pack**()**

292

293

294 ##### edit\_sql #######

295 # Parameters :- table:string, element:string, win:object

296 # Return Type :- None

297 # Purpose :- updates any field and any table of database

298 #######################

299 **def** edit\_sql**(**self**,** table**,** element**,** win**):**

300 win**.**destroy**()**

301 edit **=** Tk**()**

302 label **=** Label**(**edit**,** text**=**"New:"**)**

303 label**.**grid**(**row**=**1**,** column**=**1**)**

304 label **=** Label**(**edit**,** text**=**"Confirm New:"**)**

305 label**.**grid**(**row**=**2**,** column**=**1**)**

306 new\_text **=** StringVar**()**

307 e1 **=** Entry**(**edit**,** textvariable**=**new\_text**)**

308 e1**.**grid**(**row**=**1**,** column**=**2**)**

309 connew\_text **=** StringVar**()**

310 e2 **=** Entry**(**edit**,** textvariable**=**connew\_text**)**

311 e2**.**grid**(**row**=**2**,** column**=**2**)**

312 B1 **=** Button**(**edit**,** text**=**"Okay"**,** command**=lambda:**self**.**edit\_sql2**(str(**e1**.**get**()),str(**e2**.**get**()),**edit**,**table**,**element**))**

313 B1**.**grid**(**row**=**3**,** column**=**2**)**

314

315 ##### edit\_sql2 #######

316 # Parameters :- new:string,con\_new:string,win:object,table:string,element:string

317 # Return Type :- None

318 # Purpose :- verifies that inputs are the same and adds to database

319 #######################

320 **def** edit\_sql2**(**self**,**new**,**con\_new**,**win**,**table**,**element**):**

321 **if** new **==** con\_new**:**

322 win**.**destroy**()**

323 sql **=** "update {0} set {1}=? where {0}ID=?"**.format(**table**,** element**)**

324 self**.**cursor**.**execute**(**sql**,** **(**new**,** self**.**ID**))**

325 self**.**db**.**commit**()**

326 self**.**edit\_account**()**

327 **else:**

328 popup **=** Toplevel**(**win**)**

329 popup**.**wm\_title**(**"Error!"**)**

330 label **=** Label**(**popup**,** text**=**"Inputs Don't Match"**)**

331 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

332 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

333 B1**.**pack**()**

334

335 ##### add\_class #######

336 # Parameters :- None

337 # Return Type :- None

338 # Purpose :- adds a new class to database

339 #######################

340 **def** add\_class**(**self**):**

341 addclass **=** Tk**()**

342 addclass**.**geometry **=** "400x250"

343 sql **=** "SELECT ClassID FROM Class"

344 self**.**cursor**.**execute**(**sql**)**

345 ids**=**self**.**cursor**.**fetchall**()**

346 sorting**=[]**

347 **for** i **in** ids**:**

348 **if** **list(**i**)[**0**]==None:**

349 sorting**.**append**(**0**)**

350 **else:**

351 sorting**.**append**(list(**i**)[**0**])**

352 sorting**=sorted(**sorting**)**

353 q**=**1**+**sorting**[-**1**]**

354 self**.**db**.**commit**()**

355 btn **=** Button**(**addclass**,** text**=**"Add New Student"**,**

356 command**=lambda:** self**.**add\_student\_new**(**addclass**,** q**))**

357 btn**.**pack**()**

358 btn1 **=** Button**(**addclass**,** text**=**"Add Existing Student"**,**

359 command**=lambda:** self**.**add\_student\_existing**(**q**,** addclass**))**

360 btn1**.**pack**()**

361

362 ##### add\_student\_new #######

363 # Parameters :- win:object, classid:int

364 # Return Type :- None

365 # Purpose :- ask for information to create a student

366 #######################

367 **def** add\_student\_new**(**self**,** win**,** classid**):**

368 add **=** Toplevel**(**win**)**

369 fname **=** Label**(**add**,** text**=**"New ClassID:{0}"**.format(**classid**))**

370 fname**.**grid**(**row**=**1**,** column**=**1**)**

371 fname **=** Label**(**add**,** text**=**"Student Firstname"**)**

372 fname**.**grid**(**row**=**2**,** column**=**1**)**

373 sname **=** Label**(**add**,** text**=**"Student Surname"**)**

374 sname**.**grid**(**row**=**3**,** column**=**1**)**

375 email **=** Label**(**add**,** text**=**"Student Email"**)**

376 email**.**grid**(**row**=**4**,** column**=**1**)**

377 fname\_text **=** StringVar**()**

378 e1 **=** Entry**(**add**,** textvariable**=**fname\_text**)**

379 e1**.**grid**(**row**=**2**,** column**=**2**)**

380 sname\_text **=** StringVar**()**

381 e2 **=** Entry**(**add**,** textvariable**=**sname\_text**)**

382 e2**.**grid**(**row**=**3**,** column**=**2**)**

383 email\_text **=** StringVar**()**

384 e3 **=** Entry**(**add**,** textvariable**=**email\_text**)**

385 e3**.**grid**(**row**=**4**,** column**=**2**)**

386 btn**=** Button**(**add**,** text**=**"Done"**,**

387 command**=** **lambda:** self**.**student\_add**(str(**e1**.**get**()),** **str(**e2**.**get**()),** **str(**e3**.**get**()),**add**,**classid**))**

388 btn**.**grid**(**row**=**5**,**column**=**2**)**

389

390 ##### student\_add #######

391 # Parameters :- first:string,surname:string,email:string,win:object,classid:int

392 # Return Type :- None

393 # Purpose :-verifies that inputs where correctly inputted and adds student to database

394 # and adds them to class specified

395 #######################

396 **def** student\_add**(**self**,**first**,**surname**,**email**,**win**,**classid**):**

397 **if** first**!=**"" **and** surname**!=**"" **and** email**!=**""**:**

398 self**.**create**.**Create\_Students**(**first**,** surname**,** email**)**

399 sql **=** "SELECT StudentsID FROM Students"

400 self**.**cursor**.**execute**(**sql**)**

401 ids **=** self**.**cursor**.**fetchall**()**

402 sorting **=** **[]**

403 **for** i **in** ids**:**

404 **if** **list(**i**)[**0**]** **==** **None:**

405 sorting**.**append**(**0**)**

406 **else:**

407 sorting**.**append**(list(**i**)[**0**])**

408 sorting **=** **sorted(**sorting**)**

409 self**.**create**.**Create\_ClassStu**(**classid**,** sorting**[-**1**],** self**.**ID**)**

410 win**.**destroy**()**

411 **else:**

412 popup **=** Toplevel**(**win**)**

413 popup**.**wm\_title**(**"Error!"**)**

414 label **=** Label**(**popup**,** text**=**"Parameters not filled completely"**)**

415 label**.**pack**()**

416 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

417 B1**.**pack**()**

418

419

420 ##### add\_student\_exisiting #######

421 # Parameters :- win:object, classid:int

422 # Return Type :- None

423 # Purpose :- asks for exisiting students id so that it can add it to the database

424 #######################

425 **def** add\_student\_existing**(**self**,** classid**,** win**):**

426 edit **=** Toplevel**(**win**)**

427 label **=** Label**(**edit**,** text**=**"StudentID:"**)**

428 label**.**grid**(**row**=**1**,** column**=**1**)**

429 ID\_text **=** StringVar**()**

430 e1 **=** Entry**(**edit**,** textvariable**=**ID\_text**)**

431 e1**.**grid**(**row**=**1**,** column**=**2**)**

432 B1 **=** Button**(**edit**,** text**=**"Okay"**,** command**=lambda:** self**.**existing\_student**(**classid**,** edit**,**e1**.**get**()))**

433 B1**.**grid**(**row**=**2**,** column**=**2**)**

434

435

436 ##### existing\_student #######

437 # Parameters :- win:object, classid:int, st\_id:int

438 # Return Type :- None

439 # Purpose :- adds existing students to class

440 #######################

441 **def** existing\_student**(**self**,** classid**,** win**,**st\_id**):**

442 self**.**cursor**.**execute**(**"SELECT ClassID,StudentsID FROM Class WHERE ClassID=? AND StudentsID=?"**,**

443 **(**classid**,** st\_id**))**

444 exist **=** self**.**cursor**.**fetchone**()**

445 **if** exist **is** **not** **None:**

446 popup **=** Toplevel**(**win**)**

447 popup**.**wm\_title**(**"Error!"**)**

448 label **=** Label**(**popup**,** text**=**"Student Already in Class"**)**

449 label**.**pack**()**

450 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

451 B1**.**pack**()**

452 **else:**

453 self**.**cursor**.**execute**(**"SELECT StudentsID FROM Students WHERE StudentsID=?"**,**

454 **(**st\_id**,))**

455 exist **=** self**.**cursor**.**fetchone**()**

456 **if** exist**:**

457 self**.**create**.**Create\_ClassStu**(**classid**,**st\_id**,** self**.**ID**)**

458 self**.**db**.**commit**()**

459 win**.**destroy**()**

460 **else:**

461 popup **=** Toplevel**(**win**)**

462 popup**.**wm\_title**(**"Error!"**)**

463 label **=** Label**(**popup**,** text**=**"Invalid ID"**)**

464 label**.**pack**()**

465 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

466 B1**.**pack**()**

467

468 ##### edit\_class #######

469 # Parameters :- None

470 # Return Type :- None

471 # Purpose :- presents the option to delete class or deletes student from class

472 #######################

473 **def** edit\_class**(**self**):**

474 edit **=** Toplevel**(**self**.**main**)**

475 B1 **=** Button**(**edit**,** text**=**"Delete Class"**,** command**=lambda:** self**.**del\_class**(**edit**))**

476 B1**.**pack**()**

477 B2 **=** Button**(**edit**,** text**=**"Delete Student"**,** command**=lambda:** self**.**del\_student**(**edit**))**

478 B2**.**pack**()**

479

480

481 ##### del\_class #######

482 # Parameters :- win:object

483 # Return Type :- None

484 # Purpose :- asks for class id of class that needs to be deleted

485 #######################

486 **def** del\_class**(**self**,** win**):**

487 edit **=** Toplevel**(**win**)**

488 label **=** Label**(**edit**,** text**=**"ClassID:"**)**

489 label**.**grid**(**row**=**1**,** column**=**1**)**

490 ID\_text **=** StringVar**()**

491 e1 **=** Entry**(**edit**,** textvariable**=**ID\_text**)**

492 e1**.**grid**(**row**=**1**,** column**=**2**)**

493 B1 **=** Button**(**edit**,** text**=**"Okay"**,** command**=lambda:** self**.**del\_class2**(**edit**,** **str(**e1**.**get**())))**

494 B1**.**grid**(**row**=**2**,** column**=**2**)**

495

496

497 ##### del\_class2 #######

498 # Parameters :- edit:object, classid:int

499 # Return Type :- None

500 # Purpose :- deletes seleected class from database

501 #######################

502 **def** del\_class2**(**self**,** edit**,** classid**):**

503 self**.**cursor**.**execute**(**"SELECT ClassID FROM Class WHERE ClassID=? AND AccountID=?"**,** **(**classid**,** self**.**ID**))**

504 exist **=** self**.**cursor**.**fetchone**()**

505 **if** **len(**exist**)** **>** 0**:**

506 self**.**cursor**.**execute**(**"DELETE FROM Class WHERE ClassID=?"**,** **(**classid**,))**

507 self**.**db**.**commit**()**

508 edit**.**destroy**()**

509 **else:**

510 popup **=** Toplevel**(**edit**)**

511 popup**.**wm\_title**(**"Error!"**)**

512 label **=** Label**(**popup**,** text**=**"ID Incorrect"**)**

513 label**.**pack**()**

514 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

515 B1**.**pack**()**

516

517 ##### del\_student #######

518 # Parameters :- win:object

519 # Return Type :- None

520 # Purpose :- asks for student id of student that is being deleted

521 #######################

522 **def** del\_student**(**self**,** win**):**

523 edit **=** Toplevel**(**win**)**

524 label **=** Label**(**edit**,** text**=**"StudentID:"**)**

525 label**.**grid**(**row**=**1**,** column**=**1**)**

526 ID\_text **=** StringVar**()**

527 e1 **=** Entry**(**edit**,** textvariable**=**ID\_text**)**

528 e1**.**grid**(**row**=**1**,** column**=**2**)**

529 label1 **=** Label**(**edit**,** text**=**"ClassID:"**)**

530 label1**.**grid**(**row**=**2**,** column**=**1**)**

531 classid\_text **=** StringVar**()**

532 e2 **=** Entry**(**edit**,** textvariable**=**classid\_text**)**

533 e2**.**grid**(**row**=**2**,** column**=**2**)**

534 B1 **=** Button**(**edit**,** text**=**"Okay"**,** command**=lambda:** self**.**del\_student2**(**edit**,** **str(**e1**.**get**()),** **str(**e2**.**get**())))**

535 B1**.**grid**(**row**=**3**,**column**=**2**)**

536

537

538 ##### del\_student2 #######

539 # Parameters :- edit:object, studentid:int, classid:int

540 # Return Type :- None

541 # Purpose :- deletes student from that class in the class table

542 #######################

543 **def** del\_student2**(**self**,** edit**,** studentid**,** classid**):**

544 self**.**cursor**.**execute**(**"SELECT StudentsID FROM Class WHERE StudentsID=? AND ClassID=? AND AccountID=?"**,**

545 **(**studentid**,** classid**,** self**.**ID**))**

546 exist **=** self**.**cursor**.**fetchone**()**

547 **if** exist **is** **not** **None:**

548 self**.**cursor**.**execute**(**"DELETE FROM Class WHERE StudentsID=? AND ClassID=?"**,** **(**studentid**,** classid**))**

549 self**.**db**.**commit**()**

550 edit**.**destroy**()**

551 **else:**

552 popup **=** Toplevel**(**edit**)**

553 popup**.**wm\_title**(**"Error!"**)**

554 label **=** Label**(**popup**,** text**=**"ID Incorrect"**)**

555 label**.**pack**()**

556 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

557 B1**.**pack**()**

558

559 ##### edit\_student #######

560 # Parameters :- None

561 # Return Type :- None

562 # Purpose :- edits students information

563 #######################

564 **def** edit\_student**(**self**):**

565 editor **=** Toplevel**(**self**.**main**)**

566 editor**.**wm\_title**(**"Choose"**)**

567 label **=** Label**(**editor**,** text**=**"Which Element Do You Want To Edit"**)**

568 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

569 B1 **=** Button**(**editor**,** text**=**"Firstname"**,** command**=lambda:** self**.**edit\_sql**(**"Students"**,** "Firstname"**,** editor**))**

570 B1**.**pack**()**

571 B5 **=** Button**(**editor**,** text**=**"Lastname"**,** command**=lambda:** self**.**edit\_sql**(**"Students"**,** "Surname"**,** editor**))**

572 B5**.**pack**()**

573 B3 **=** Button**(**editor**,** text**=**"Email"**,** command**=lambda:** self**.**edit\_sql**(**"Students"**,** "Email"**,** editor**))**

574 B3**.**pack**()**

575 B6 **=** Button**(**editor**,** text**=**"Okay"**,** command**=**editor**.**destroy**)**

576 B6**.**pack**()**

577

578

579 ##### view\_class1 #######

580 # Parameters :- None

581 # Return Type :- None

582 # Purpose :- asks which class that wants to be viewed

583 #######################

584 **def** view\_class1**(**self**):**

585 edit **=** Tk**()**

586 label **=** Label**(**edit**,** text**=**"ClassID:"**)**

587 label**.**grid**(**row**=**1**,** column**=**1**)**

588 ID\_text **=** StringVar**()**

589 e1 **=** Entry**(**edit**,** textvariable**=**ID\_text**)**

590 e1**.**grid**(**row**=**1**,** column**=**2**)**

591 B1 **=** Button**(**edit**,** text**=**"Okay"**,** command**=lambda:** self**.**view\_class**(**edit**,** **str(**e1**.**get**())))**

592 B1**.**grid**(**row**=**2**,** column**=**2**)**

593

594

595 ##### view\_class #######

596 # Parameters :- topwin:object, id:int

597 # Return Type :- None

598 # Purpose :- creates a window displaying the details of every member of the class and there progress in that class

599 #######################

600 **def** view\_class**(**self**,** topwin**,** **id):**

601 win **=** Toplevel**(**topwin**)**

602 self**.**cursor**.**execute**(**"SELECT StudentsID, Firstname, Surname,Progress,Email FROM "

603 "Students INNER JOIN Class USING(StudentsID) WHERE ClassID=?"**,** **(id,))**

604 info **=** self**.**cursor**.**fetchall**()**

605 **if** **len(**info**)** **>** 0**:**

606 style **=** ttk**.**Style**()**

607 style**.**theme\_use**(**'clam'**)**

608 tree **=** ttk**.**Treeview**(**win**,** column**=(**"StudentID"**,** "Firstname"**,** "Surname"**,** "Progress"**,** "Email"**),**

609 show**=**'headings'**,** height**=**5**)**

610 tree**.**column**(**"# 1"**,** anchor**=**CENTER**)**

611 tree**.**heading**(**"# 1"**,** text**=**"StudentID"**)**

612 tree**.**column**(**"# 2"**,** anchor**=**CENTER**)**

613 tree**.**heading**(**"# 2"**,** text**=**"Firstname"**)**

614 tree**.**column**(**"# 3"**,** anchor**=**CENTER**)**

615 tree**.**heading**(**"# 3"**,** text**=**"Surname"**)**

616 tree**.**column**(**"# 4"**,** anchor**=**CENTER**)**

617 tree**.**heading**(**"# 4"**,** text**=**"Progress"**)**

618 tree**.**column**(**"# 5"**,** anchor**=**CENTER**)**

619 tree**.**heading**(**"# 5"**,** text**=**"Email"**)**

620 # Insert the data in Treeview widget

621 **for** i **in** info**:**

622 tree**.**insert**(**''**,** 'end'**,** text**=**"1"**,** values**=**i**)**

623 tree**.**pack**()**

624 btn **=** Button**(**win**,** text**=**"Back"**,** command**=lambda:** win**.**destroy**())**

625 btn**.**pack**()**

626 **else:**

627 popup **=** Toplevel**(**win**)**

628 popup**.**wm\_title**(**"Error!"**)**

629 label **=** Label**(**popup**,** text**=**"ID Incorrect"**)**

630 label**.**pack**()**

631 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

632 B1**.**pack**()**

633

634

635##### First\_Window #######

636# Parameters :- None

637# Return Type :- None

638# Purpose :- class holds the windows and option given when not logged in yet

639#######################

640class First\_Window**:**

641 **def** \_\_init\_\_**(**self**):**

642 self**.**db**=**sqlite3**.**connect**(**"Account.db"**)**

643 self**.**cursor **=** self**.**db**.**cursor**()**

644 self**.**create**=**Create\_SQL**()**

645 self**.**hostIP **=** socket**.**gethostname**()**

646

647 ##### verify #######

648 # Parameters :- username:string, password:string, win:object

649 # Return Type :- None

650 # Purpose :- verifies the users login and logins them into the main page

651 #######################

652 **def** verify**(**self**,** username**,** passw**,** win**):**

653 password **=** self**.**encrypt**(**passw**)**

654 self**.**cursor**.**execute**(**"select UsersID,Username from Users where Password=?"**,** **(**password**,))**

655 possible **=** self**.**cursor**.**fetchall**()**

656 l1 **=** **[]**

657 **print(**possible**)**

658 **for** user **in** possible**:**

659 **if** user**[**1**]** **==** **str(**username**):**

660 l1**.**append**(**user**[**0**])**

661 l1**.**append**(**user**[**1**])**

662

663 **else:**

664 **pass**

665 **if** **len(**l1**)** **>** 0**:**

666 **print(**l1**)**

667 win**.**destroy**()**

668 **print(**l1**[**0**])**

669 self**.**cursor**.**execute**(**"SELECT AccountID FROM Account WHERE UsersID=?"**,** **(**l1**[**0**],))**

670 t **=** self**.**cursor**.**fetchall**()**

671 **print(**t**)**

672 Logged **=** Account**(**l1**[**0**],** l1**[**1**],** t**)**

673 **else:**

674 popup **=** Toplevel**(**win**)**

675 popup**.**wm\_title**(**"Error!"**)**

676 label **=** Label**(**popup**,** text**=**"Username or Password Incorrect"**)**

677 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

678 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

679 B1**.**pack**()**

680

681 ##### Create\_Users #######

682 # Parameters :- username:string, password:string, passcon:string, title:string, firstname:string,

683 # lastname:string, school:string, email:string, win:object

684 # Return Type :- None

685 # Purpose :- creates users who sign up and add to database, checks that username is unique

686 #######################

687 **def** Create\_Users**(**self**,** username**,** password**,** passcon**,** title**,** firstname**,** lastname**,** school**,** email**,** win**):**

688 self**.**cursor**.**execute**(**"SELECT Username FROM Users WHERE Username =?"**,** **(str(**username**),))**

689 # checks if username already exist within the Accounts database

690 existence **=** self**.**cursor**.**fetchone**()**

691 **if** existence**:**

692 popup **=** Toplevel**(**win**)**

693 popup**.**wm\_title**(**"Error!"**)**

694 label **=** Label**(**popup**,** text**=**"Username Already Being Used"**)**

695 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

696 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

697 B1**.**pack**()**

698 **elif** password **!=** passcon**:**

699 popup **=** Toplevel**(**win**)**

700 popup**.**wm\_title**(**"Error!"**)**

701 label **=** Label**(**popup**,** text**=**"Password Don't Match"**)**

702 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

703 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

704 B1**.**pack**()**

705

706

707 **elif** password **==** passcon**:**

708 # hashes the password

709 encryptpass**=**self**.**encrypt**(**password**)**

710 cur**=**self**.**create**.**Create\_Users**(**username**,** encryptpass**)**

711 self**.**create**.**Create\_Accounts**(**title**,** firstname**,** lastname**,** school**,** email**,** cur**)**

712 self**.**db**.**commit**()**

713 win**.**destroy**()**

714 n**.**Main**()**

715

716 ##### encrypt #######

717 # Parameters :- password:string

718 # Return Type :- string

719 # Purpose :- using vernier cipher encrypts passwords

720 #######################

721 **def** encrypt**(**self**,** password**):**

722 key **=** **[]**

723 key**[:**0**]** **=** "KONE"

724 **if** **len(**password**)** **!=** **len(**key**):**

725 x **=** 0

726 **for** i **in** **range(len(**password**)** **-** **len(**key**)):**

727 **if** i **==** 4**:**

728 x **=** 0

729 key**.**append**(**key**[**x**])**

730 **else:**

731 key**.**append**(**key**[**x**])**

732 x **+=** 1

733 encrypt\_pass **=** **[]**

734 **for** i **in** **range(len(**password**)):**

735 x **=** **(ord(**password**[**i**])** **+** **ord(**key**[**i**]))** **%** 26

736 x **+=** **ord(**'A'**)**

737 encrypt\_pass**.**append**(chr(**x**))**

738 **return** ""**.**join**(**encrypt\_pass**)**

739

740 ##### login #######

741 # Parameters :- window:object

742 # Return Type :- None

743 # Purpose :- creates login window

744 #######################

745 **def** login**(**self**,** window**):**

746 window**.**destroy**()**

747 login\_window **=** Tk**()**

748 login\_window**.**geometry **=** "300x150"

749 user **=** Label**(**login\_window**,** text**=**"Username"**)**

750 user**.**grid**(**row**=**1**,** column**=**1**)**

751 password **=** Label**(**login\_window**,** text**=**"Password"**)**

752 password**.**grid**(**row**=**2**,** column**=**1**)**

753 user\_text **=** StringVar**()**

754 e1 **=** Entry**(**login\_window**,** textvariable**=**user\_text**)**

755 e1**.**grid**(**row**=**1**,** column**=**2**)**

756

757 pass\_text **=** StringVar**()**

758 e2 **=** Entry**(**login\_window**,** textvariable**=**pass\_text**)**

759 e2**.**config**(**show**=**"\*"**)**

760 e2**.**grid**(**row**=**2**,** column**=**2**)**

761

762 btn **=** Button**(**login\_window**,** text**=**"Confirm"**,** width**=**20**,**

763 command**=lambda:** self**.**verify**(**e1**.**get**(),** e2**.**get**(),** login\_window**))**

764 btn**.**grid**(**row**=**3**,** column**=**2**)**

765 btn2 **=** Button**(**login\_window**,** text**=**"Back"**,** width**=**20**,**

766 command**=lambda:** self**.**back\_reg**(**login\_window**))**

767 btn2**.**grid**(**row**=**3**,** column**=**1**)**

768

769 ##### sign\_up #######

770 # Parameters :- window:object

771 # Return Type :- None

772 # Purpose :- creates the window for the sign up

773 #######################

774 **def** sign\_up**(**self**,** window**):**

775 window**.**destroy**()**

776 signup\_window **=** Tk**()**

777 signup\_window**.**geometry **=** "400x250"

778 user **=** Label**(**signup\_window**,** text**=**"Username"**)**

779 user**.**grid**(**row**=**1**,** column**=**1**)**

780 password **=** Label**(**signup\_window**,** text**=**"Password"**)**

781 password**.**grid**(**row**=**2**,** column**=**1**)**

782 confirm\_pass **=** Label**(**signup\_window**,** text**=**"Confirm Password"**)**

783 confirm\_pass**.**grid**(**row**=**3**,** column**=**1**)**

784 title **=** Label**(**signup\_window**,** text**=**"Title"**)**

785 title**.**grid**(**row**=**4**,** column**=**1**)**

786 firstname **=** Label**(**signup\_window**,** text**=**"Firstname"**)**

787 firstname**.**grid**(**row**=**5**,** column**=**1**)**

788 lastname **=** Label**(**signup\_window**,** text**=**"Lastname"**)**

789 lastname**.**grid**(**row**=**6**,** column**=**1**)**

790 email **=** Label**(**signup\_window**,** text**=**"Email"**)**

791 email**.**grid**(**row**=**7**,** column**=**1**)**

792 school **=** Label**(**signup\_window**,** text**=**"School"**)**

793 school**.**grid**(**row**=**8**,** column**=**1**)**

794

795 user\_text **=** StringVar**()**

796 e1 **=** Entry**(**signup\_window**,** textvariable**=**user\_text**)**

797 e1**.**grid**(**row**=**1**,** column**=**2**)**

798

799 pass\_text **=** StringVar**()**

800 e2 **=** Entry**(**signup\_window**,** textvariable**=**pass\_text**)**

801 e2**.**grid**(**row**=**2**,** column**=**2**)**

802 e2**.**config**(**show**=**"\*"**)**

803 passcon\_text **=** StringVar**()**

804 e3 **=** Entry**(**signup\_window**,** textvariable**=**passcon\_text**)**

805 e3**.**grid**(**row**=**3**,** column**=**2**)**

806 e3**.**config**(**show**=**"\*"**)**

807

808 title\_text **=** StringVar**()**

809 e4 **=** Entry**(**signup\_window**,** textvariable**=**title\_text**)**

810 e4**.**grid**(**row**=**4**,** column**=**2**)**

811

812 firstname\_text **=** StringVar**()**

813 e5 **=** Entry**(**signup\_window**,** textvariable**=**firstname\_text**)**

814 e5**.**grid**(**row**=**5**,** column**=**2**)**

815

816 lastname\_text **=** StringVar**()**

817 e6 **=** Entry**(**signup\_window**,** textvariable**=**lastname\_text**)**

818 e6**.**grid**(**row**=**6**,** column**=**2**)**

819

820 email\_text **=** StringVar**()**

821 e7 **=** Entry**(**signup\_window**,** textvariable**=**email\_text**)**

822 e7**.**grid**(**row**=**7**,** column**=**2**)**

823

824 school\_text **=** StringVar**()**

825 e8 **=** Entry**(**signup\_window**,** textvariable**=**school\_text**)**

826 e8**.**grid**(**row**=**8**,** column**=**2**)**

827

828 btn **=** Button**(**signup\_window**,** text**=**"Confirm"**,** width**=**20**,**

829 command**=lambda:** self**.**Create\_Users**(**e1**.**get**(),** e2**.**get**(),** e3**.**get**(),** e4**.**get**(),** e5**.**get**(),** e6**.**get**(),** e8**.**get**(),**

830 e7**.**get**(),** signup\_window**))**

831 btn**.**grid**(**row**=**9**,** column**=**2**)**

832

833 btn2 **=** Button**(**signup\_window**,** text**=**"Back"**,** width**=**20**,**

834 command**=lambda:** self**.**back\_reg**(**signup\_window**))**

835 btn2**.**grid**(**row**=**9**,** column**=**1**)**

836

837 ##### back\_reg #######

838 # Parameters :- win:object

839 # Return Type :- None

840 # Purpose :- allows users to go back to first window

841 #######################

842 **def** back\_reg**(**self**,** win**):**

843 win**.**destroy**()**

844 n**.**Main**()**

845

846 **def** offline**(**self**,**win**):**

847 win**.**destroy**()**

848 off **=** subprocess**.**call**((**'python OSD.py'**))**

849 n**.**Main**()**

850

851 ##### student\_access #######

852 # Parameters :- window:object

853 # Return Type :- None

854 # Purpose :- allows student to enter a code to enter an envoirnment and allows students to access offline mode

855 #######################

856 **def** student\_access**(**self**,** window**):**

857 window**.**destroy**()**

858 access\_window **=** Tk**()**

859 access\_window**.**geometry **=** "400x250"

860 student **=** Label**(**access\_window**,** text**=**"StudentID"**)**

861 student**.**grid**(**row**=**1**,** column**=**1**)**

862 code **=** Label**(**access\_window**,** text**=**"Code"**)**

863 code**.**grid**(**row**=**2**,** column**=**1**)**

864

865 s\_id\_text **=** StringVar**()**

866 e1 **=** Entry**(**access\_window**,** textvariable**=**s\_id\_text**)**

867 e1**.**grid**(**row**=**1**,** column**=**2**)**

868

869 code\_text **=** StringVar**()**

870 e2 **=** Entry**(**access\_window**,** textvariable**=**code\_text**)**

871 e2**.**grid**(**row**=**2**,** column**=**2**)**

872 btn **=** Button**(**access\_window**,** text**=**"Offline Access"**,** width**=**20**,**

873 command**=lambda:**self**.**offline**(**access\_window**)** **)**

874 btn**.**grid**(**row**=**9**,** column**=**3**)**

875

876 btn **=** Button**(**access\_window**,** text**=**"Confirm"**,** width**=**20**,**

877 command**=lambda:** self**.**studentaccess**(**e1**.**get**(),** e2**.**get**(),**access\_window**))**

878 btn**.**grid**(**row**=**9**,** column**=**2**)**

879

880 btn2 **=** Button**(**access\_window**,** text**=**"Back"**,** width**=**20**,**

881 command**=lambda:** self**.**back\_reg**(**access\_window**))**

882 btn2**.**grid**(**row**=**9**,** column**=**1**)**

883

884 ##### studentaccess #######

885 # Parameters :- win:object, code:string, stu\_id:int

886 # Return Type :- None

887 # Purpose :- verifies that code entered is valid and that they are in the class in which

888 # the session is being hosted for

889 #######################

890 **def** studentaccess**(**self**,** stu\_id**,** code**,**win**):**

891 classID **=** code**.**partition**(**"-"**)[**0**]**

892 sql **=** "SELECT \* FROM Class WHERE StudentsID=? AND ClassID=? "

893 self**.**cursor**.**execute**(**sql**,** **(**stu\_id**,** classID**))**

894 **if** **len(**self**.**cursor**.**fetchall**())** **!=**0**:**

895 self**.**cursor**.**execute**(**"SELECT MasterCode FROM Code WHERE ClassID={0}"**.format(str(**classID**)))**

896 co**=**self**.**cursor**.**fetchone**()**

897 **print(**classID**)**

898 truecode**=**co**+(**"%"**,)**

899 **if** **len(**truecode**)** **!=** 0**:**

900 **if** truecode**[**0**]==**code**:**

901 clientip **=** socket**.**gethostbyname**(**socket**.**gethostname**())**

902 sql **=** "SELECT MasterCode,HostIP,ClientIP FROM Code WHERE MasterCode=? AND HostIP=? AND ClientIP=?"

903 self**.**cursor**.**execute**(**sql**,** **(**code**,** self**.**hostIP**,** clientip**))**

904 **print(**self**.**cursor**.**fetchall**())**

905 **if** **not** **len(**self**.**cursor**.**fetchall**()):**

906 sql **=** "INSERT INTO Code(MasterCode,HostIP,ClientIP,ClassID) VALUES(?,?,?,?)"

907 self**.**cursor**.**execute**(**sql**,** **(**code**,** self**.**hostIP**,**clientip**,**classID**))**

908 self**.**db**.**commit**()**

909 **else:**

910 **pass**

911 subprocess**.**call**((**'python Client-Student.py'**))**

912 **else:**

913 popup **=** Toplevel**(**win**)**

914 popup**.**wm\_title**(**"Error!"**)**

915 label **=** Label**(**popup**,** text**=**"Code Incorrect"**)**

916 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

917 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

918 B1**.**pack**()**

919

920 **else:**

921 popup **=** Toplevel**(**win**)**

922 popup**.**wm\_title**(**"Error!"**)**

923 label **=** Label**(**popup**,** text**=**"Not In Class"**)**

924 label**.**pack**(**side**=**"top"**,** fill**=**"x"**,** pady**=**10**)**

925 B1 **=** Button**(**popup**,** text**=**"Okay"**,** command**=**popup**.**destroy**)**

926 B1**.**pack**()**

927

928 ##### Main #######

929 # Parameters :- None

930 # Return Type :- None

931 # Purpose :- first window to pop up when you run the program

932 #######################

933 **def** Main**(**self**):**

934 window **=** Tk**()**

935 window**.**geometry**(**"360x144"**)**

936

937 op **=** Label**(**window**,** text**=**"Choose an Option"**)**

938 op**.**grid**(**row**=**1**,** column**=**2**,** columnspan**=**2**)**

939

940 log\_btn **=** Button**(**window**,** text**=**"Login"**,** width**=**50**,** height**=**2**,** command**=lambda:** self**.**login**(**window**))**

941 log\_btn**.**grid**(**row**=**3**,** column**=**2**)**

942

943 reg\_btn **=** Button**(**window**,** text**=**"Sign Up"**,** width**=**50**,** height**=**2**,** command**=lambda:** self**.**sign\_up**(**window**))**

944 reg\_btn**.**grid**(**row**=**5**,** column**=**2**)**

945

946 stu\_btn **=** Button**(**window**,** text**=**"Student Access"**,** width**=**50**,** height**=**2**,** command**=lambda:** self**.**student\_access**(**window**))**

947 stu\_btn**.**grid**(**row**=**7**,** column**=**2**)**

948 window**.**mainloop**()**

949

950n**=**First\_Window**()**

951n.Main**()**

## Client-Student.py

1 **import** pygame

2 **import** sys

3 **import** ChatboxClient

4 **from** network **import** Network

5 **import** ctypes

6 **import** math

7 **import** sympy

8 **from** pygame **import** gfxdraw

9

10 user32 **=** ctypes**.**windll**.**user32

11 width **=** user32**.**GetSystemMetrics**(**0**)**

12 length **=** user32**.**GetSystemMetrics**(**1**)**

13 pygame**.**init**()**

14 screen **=** pygame**.**display**.**set\_mode**((**width**,** length **-** 200**))**

15 screen**.**fill**((**0**,** 0**,** 0**))**

16 toolbar\_button **=** **[]**

17 objects\_loaded **=** **[]**

18 ang **=** **[]**

19 sources\_loaded **=** **[]**

20 IncidentRays **=** **{}**

21 clientNumber **=** 0

22 tag **=** **[**0**]**

23

24

25 ##### tag\_generator #######

26 # Parameters :- tag:List, obj\_name:String

27 # Return Type :- String

28 # Purpose :- This is to generate a unique tag whenever an instance of

29 # a class is loaded it has an attribute that holds it name as there can be

30 # multiple of instances of one class having this unique name will help identify

31 # which instance needs to be manipulated

32 ###########################

33 **def** tag\_generator**(**tag**,** obj\_name**):**

34 x **=** tag**[-**1**]** **+** 1

35 tag**.**append**(**x**)**

36 Tag **=** obj\_name **+** **str(**tag**[-**1**])**

37 **return** Tag

38

39

40 ##### Angle #######

41 # Parameters :- None

42 # Purpose :- This class creates the pie shape that represents the angle

43 # between two lines and has the ability to show what angle the pie represents

44 ###########################

45 **class** **Angle:**

46 **def** \_\_init\_\_**(**self**):**

47 self**.**x **=** 0

48 self**.**y **=** 0

49 self**.**radius **=** 30

50 self**.**theta1 **=** 0

51 self**.**theta2 **=** 0

52 self**.**colour **=** **(**255**,** 0**,** 0**)**

53 self**.**show\_theta **=** 0

54 self**.type** **=** ""

55 self**.**orientation **=** ""

56 pygame**.**gfxdraw**.**pie**(**screen**,** **int(**self**.**x**),** **int(**self**.**y**),** self**.**radius**,** **int(**self**.**theta1**),**

57 **int(**self**.**theta2**),** self**.**colour**)**

58 self**.**update**()**

59

60 ##### redraw1 #######

61 # Parameters :- None

62 # Return Type :- None

63 # Purpose :- Purpose is to redraw instance of class on the screen

64 ###########################

65

66 **def** redraw1**(**self**):**

67 pygame**.**gfxdraw**.**pie**(**screen**,** **int(**self**.**x**),** **int(**self**.**y**),** self**.**radius**,** **int(**self**.**theta1**),**

68 **int(**self**.**theta2**),** self**.**colour**)**

69 self**.**update**()**

70

71 ##### update #######

72 # Parameters :- None

73 # Return Type :- None

74 # Purpose :- To check what can be displayed on screens in terms of angles and

75 # updates the screen to display them

76 ###########################

77 **def** update**(**self**):**

78 **if** m**.**IA **and** self**.type** **==** "Incident" **and** self**.**orientation **==** "vertical"**:**

79 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

80 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

81 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 10**,** self**.**y **+** 10**))**

82 screen**.**blit**(**text**,** textrect**)**

83 **elif** m**.**IA **and** self**.type** **==** "Incident" **and** self**.**orientation **==** "horizontal"**:**

84 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

85 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

86 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 20**,** self**.**y **+** 20**))**

87 screen**.**blit**(**text**,** textrect**)**

88

89 **elif** m**.**RA **and** self**.type** **==** "Response" **and** self**.**orientation **==** "veritical"**:**

90 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

91 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

92 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **-** 10**,** self**.**y **+** 10**))**

93 screen**.**blit**(**text**,** textrect**)**

94 **elif** m**.**RA **and** self**.type** **==** "Response" **and** self**.**orientation **==** "horizontal"**:**

95 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

96 text **=** font**.**render**(str(int((**self**.**show\_theta **\*** 180**)** **/** math**.**pi**))** **+** "°"**,** **True,** **(**105**,** 105**,** 105**))**

97 textrect **=** text**.**get\_rect**(**center**=(**self**.**x **+** 20**,** self**.**y **+** 20**))**

98 screen**.**blit**(**text**,** textrect**)**

99

100

101##### Main\_Objects #######

102# Parameters :- name:String

103# Purpose :- To act as a parent class to all manipulatable objects

104###########################

105

106class Main\_Objects**:**

107 **def** \_\_init\_\_**(**self**,** name**):**

108 self**.**defined\_name **=** name

109 self**.**Refractive\_Index **=** 1.00

110 self**.**xp **=** **float(**width **/** 2**)**

111 self**.**yp **=** **float(**length **/** 2**)**

112 self**.**length **=** 200

113 self**.**width **=** 250

114 self**.**colour **=** **(**0**,** 0**,** 0**)**

115 self**.**Property **=** "Transparent"

116 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

117 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

118 self**.**draging **=** **False**

119 self**.**interceptors **=** **[]**

120

121 ##### update #######

122 # Parameters :- None

123 # Return Type :- None

124 # Purpose :- To check what can be displayed on screens in terms of angles and

125 # updates the screen to display them

126 ###########################

127

128 **def** update**(**self**):**

129 **if** m**.**RI**:**

130 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

131 text **=** font**.**render**(str(**self**.**Refractive\_Index**),** **True,** **(**105**,** 105**,** 105**))**

132 textrect **=** text**.**get\_rect**(**center**=(**self**.**xp **+** 20**,** self**.**yp **+** 20**))**

133 screen**.**blit**(**text**,** textrect**)**

134

135 ##### redraw2 #######

136 # Parameters :- None

137 # Return Type :- None

138 # Purpose :- To redraw every object on screen (updating screen contents)

139 ###########################

140

141 **def** redraw2**(**self**):**

142 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

143 **for** i **in** objects\_loaded**:**

144 **if** "D" **not** **in** i**.**defined\_name**:**

145 i**.**redraw1**()**

146 **for** i **in** sources\_loaded**:**

147 i**.**redraw1**()**

148

149 ##### Simulation #######

150 # Parameters :- state:Boolean, previous:Memory Address of Object, obj:Boolean

151 # Return Type :- None

152 # Purpose :- Checks if any sources loaded interacts with the object and manipulates

153 # the source dependant on the objects properties

154 ###########################

155

156 **def** Simulation**(**self**,** state**,** previous**,** obj**):**

157 **if** sources\_loaded**:**

158 **for** i **in** sources\_loaded**:**

159 i**.**Simulation**(False,** **None,** **True)**

160

161 ##### redraw1 #######

162 # Parameters :- None

163 # Return Type :- None

164 # Purpose :- updates instance of object on screen

165 ###########################

166 **def** redraw1**(**self**):**

167 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

168 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

169 self**.**update**()**

170

171

172##### Sources #######

173# Inheirts :- Main\_Objects

174# Parameters :- name:String

175# Purpose :- Holds every attribute and operation that can be done on a source object

176###########################

177class Source**(**Main\_Objects**):**

178 **def** \_\_init\_\_**(**self**,** name**):**

179 **super().**\_\_init\_\_**(**name**)**

180 self**.**length **=** 2

181 self**.**width **=** 10000

182 self**.**endy **=** self**.**yp

183 self**.**endx **=** self**.**width

184 self**.**redy **=** self**.**yp

185 self**.**redx **=** self**.**xp **+** 10

186 self**.**rendx **=** self**.**xp **+** 20

187 self**.**rendy **=** self**.**endy

188 self**.**colour **=** **(**255**,** 0**,** 0**)**

189 self**.**colour2 **=** **(**255**,** 193**,** 110**)**

190 self**.**thickness **=** 4

191 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,**

192 **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** self**.**endy**),** self**.**thickness**)**

193 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,**

194 **(**self**.**redx**,** self**.**redy**),** **(**self**.**rendx**,** self**.**rendy**),** self**.**thickness**)**

195 self**.**m **=** 0

196 self**.**findx **=** "((y-{0})/m)+{1}"**.format(int(**self**.**yp**),** **int(**self**.**xp**))**

197 self**.**findy **=** "m\*(x-{0})+{1}"**.format(**self**.**xp**,** self**.**yp**)**

198 self**.**state **=** **False**

199 self**.**previousobj **=** **[]**

200 IncidentRays**[**self**.**defined\_name**]** **=** **[]**

201 self**.**full\_name **=** "Source" **+** **str(**self**.**defined\_name**[-**1**])**

202 self**.**t\_status **=** 0

203 self**.**normals **=** **[]**

204 self**.**angles **=** **[]**

205 self**.**re **=** **None**

206

207 ##### blackout #######

208 # Parameters :- None

209 # Return Type :- self.colour: triplet

210 # Purpose :- isolates the red rotation button from source

211 # usually done to sources created due to an interaction with an object

212 ###########################

213 **def** blackout**(**self**):**

214 self**.**colour **=** **(**0**,** 0**,** 0**)**

215 self**.**redraw2**()**

216 **return** self**.**colour

217

218 ##### anglesfunc #######

219 # Parameters :- theta1:float, theta2:float, orientation:string, coord:list, nor:int,

220 # nor2:int, sign:string, source:Memory Address of Object

221 # Return Type :- None

222 # Purpose :- creates and displays (if allowed to be shown) the angles dependant

223 # on the parameters using the Angle class

224 ###########################

225 **def** anglesfunc**(**self**,** theta1**,** theta2**,** orientation**,** coord**,** nor**,** nor2**,** sign**,** source**):**

226 angle **=** Angle**()**

227 IncidentRays**[**source**].**append**(**angle**)**

228 self**.**angles**.**append**(**angle**)**

229 angle2 **=** Angle**()**

230 IncidentRays**[**source**].**append**(**angle2**)**

231 self**.**angles**.**append**(**angle2**)**

232 **if** sign **==** "reflecting"**:**

233 angle**.**show\_theta **=** theta1

234 angle**.type** **=** "Incident"

235 angle2**.**show\_theta **=** theta2

236 angle2**.type** **=** "Response"

237 angle**.**orientation **=** orientation

238 angle2**.**orientation **=** orientation

239 angle**.**theta1 **=** **-((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

240 angle**.**theta2 **=** nor

241 angle**.**x **=** coord**[**1**]**

242 angle**.**y **=** coord**[**2**]**

243 angle2**.**theta1 **=** nor2

244 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

245 angle2**.**x **=** coord**[**1**]**

246 angle2**.**y **=** coord**[**2**]**

247 **elif** orientation **==** "reverse"**:**

248 angle**.**show\_theta **=** theta1

249 angle**.type** **=** "Incident"

250 angle2**.**show\_theta **=** theta2

251 angle2**.type** **=** "Response"

252 angle**.**orientation **=** "vertical"

253 angle2**.**orientation **=** "vertical"

254 angle**.**theta2 **=** **((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

255 angle**.**theta1 **=** nor

256 angle**.**x **=** coord**[**1**]**

257 angle**.**y **=** coord**[**2**]**

258 angle2**.**theta1 **=** nor2

259 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

260 angle2**.**x **=** coord**[**1**]**

261 angle2**.**y **=** coord**[**2**]**

262 **elif** orientation **==** "reverse2"**:**

263 angle**.**show\_theta **=** theta1

264 angle**.type** **=** "Incident"

265 angle2**.**show\_theta **=** theta2

266 angle2**.type** **=** "Response"

267 angle**.**orientation **=** "horizontal"

268 angle2**.**orientation **=** "horizontal"

269 angle**.**theta2 **=** **((**theta1 **\*** 180**)** **/** math**.**pi**)** **+** nor

270 angle**.**theta1 **=** nor

271 angle**.**x **=** coord**[**1**]**

272 angle**.**y **=** coord**[**2**]**

273 angle2**.**theta1 **=** nor2

274 angle2**.**theta2 **=** **(**theta2 **\*** 180**)** **/** math**.**pi **+** nor2

275 angle2**.**x **=** coord**[**1**]**

276 angle2**.**y **=** coord**[**2**]**

277 **else:**

278 angle**.**show\_theta **=** theta1

279 angle**.type** **=** "Incident"

280 angle2**.**show\_theta **=** theta2

281 angle2**.type** **=** "Response"

282 angle**.**orientation **=** orientation

283 angle2**.**orientation **=** orientation

284 angle**.**theta1 **=** **eval(**sign **+** **str((**theta1 **\*** 180**)** **/** math**.**pi**))** **+** nor

285 angle**.**theta2 **=** nor

286 angle**.**x **=** coord**[**1**]**

287 angle**.**y **=** coord**[**2**]**

288 angle2**.**theta2 **=** nor2

289 angle2**.**theta1 **=** **eval(**sign **+** **str((**theta2 **\*** 180**)** **/** math**.**pi**))** **+** nor2

290 angle2**.**x **=** coord**[**1**]**

291 angle2**.**y **=** coord**[**2**]**

292

293 ##### normalfunc #######

294 # Parameters :- orientation:string, source:Memory Address of Object, coord:list

295 # Return Type :- None

296 # Purpose :- creates a normal line dependant on the propagation position

297 ###########################

298 **def** normalfunc**(**self**,** orientation**,** source**,** coord**):**

299 Tag **=** tag\_generator**(**tag**,** "N"**)**

300 normal **=** Source**(**Tag**)**

301 IncidentRays**[**source**.**defined\_name**].**append**(**normal**)**

302 source**.**normals**.**append**(**normal**)**

303 normal**.**blackout**()**

304 normal**.**thickness **=** 2

305 **if** orientation **==** "top"**:**

306 normal**.**xp **=** coord**[**1**]**

307 normal**.**yp **=** coord**[**2**]** **-** 50

308 normal**.**endx **=** coord**[**1**]**

309 normal**.**endy **=** coord**[**2**]** **+** 50

310 **else:**

311 normal**.**xp **=** coord**[**1**]** **-** 50

312 normal**.**yp **=** coord**[**2**]**

313 normal**.**endx **=** coord**[**1**]** **+** 50

314 normal**.**endy **=** coord**[**2**]**

315 normal**.**colour2 **=** **(**255**,** 0**,** 0**)**

316

317 ##### redraw1 #######

318 # Parameters :- None

319 # Return Type :- None

320 # Purpose :- updates instance of object on screen

321 ###########################

322 **def** redraw1**(**self**):**

323 self**.**graphical **=** pygame**.**draw**.**line**(**screen**,** self**.**colour2**,** **(**self**.**xp**,** self**.**yp**),** **(**self**.**endx**,** self**.**endy**),**

324 self**.**thickness**)**

325 self**.**graphical2 **=** pygame**.**draw**.**line**(**screen**,** self**.**colour**,** **(**self**.**redx**,** self**.**redy**),** **(**self**.**rendx**,** self**.**rendy**),**

326 self**.**thickness**)**

327

328

329##### Block #######

330# Inheirts :- Main\_Objects

331# Parameters :- name:String

332# Purpose :- Holds every attribute and operation that can be done on a block object

333###########################

334class Block**(**Main\_Objects**):**

335 **def** \_\_init\_\_**(**self**,** name**):**

336 **super().**\_\_init\_\_**(**name**)**

337 self**.**colour **=** **(**255**,** 105**,** 105**)**

338 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

339 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

340 self**.**full\_name **=** "Block" **+** **str(**self**.**defined\_name**[-**1**])**

341 self**.**Refractive\_Index **=** 1.5

342

343 ##### redraw1 #######

344 # Parameters :- None

345 # Return Type :- None

346 # Purpose :- updates instance of object on screen

347 ###########################

348 **def** redraw1**(**self**):**

349 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

350 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

351 self**.**update**()**

352

353

354##### GlassBlock #######

355# Inheirts :- Main\_Objects

356# Parameters :- name:String

357# Purpose :- Holds every attribute and operation that can be done on a glassblock object

358###########################

359class GlassBlock**(**Main\_Objects**):**

360 **def** \_\_init\_\_**(**self**,** name**):**

361 **super().**\_\_init\_\_**(**name**)**

362 self**.**Refractive\_Index **=** 1.52

363 self**.**colour **=** **(**168**,** 204**,** 215**)**

364 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

365 self**.**full\_name **=** "GlassBlock" **+** **str(**self**.**defined\_name**[-**1**])**

366

367 ##### redraw1 #######

368 # Parameters :- None

369 # Return Type :- None

370 # Purpose :- updates instance of object on screen

371 ###########################

372 **def** redraw1**(**self**):**

373 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

374 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

375 self**.**update**()**

376

377

378##### Mirror #######

379# Inheirts :- Main\_Objects

380# Parameters :- name:String

381# Purpose :- Holds every attribute and operation that can be done on a mirror object

382###########################

383class Mirror**(**Main\_Objects**):**

384 **def** \_\_init\_\_**(**self**,** name**):**

385 **super().**\_\_init\_\_**(**name**)**

386 self**.**Property **=** "Reflective"

387 self**.**width **=** 10

388 self**.**colour **=** **(**192**,** 192**,** 192**)**

389 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

390 pygame**.**Rect**(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

391 self**.**full\_name **=** "Mirror" **+** **str(**self**.**defined\_name**[-**1**])**

392

393 ##### redraw1 #######

394 # Parameters :- None

395 # Return Type :- None

396 # Purpose :- updates instance of object on screen

397 ###########################

398 **def** redraw1**(**self**):**

399 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

400 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

401 self**.**update**()**

402

403

404##### Screen #######

405# Inheirts :- Main\_Objects

406# Parameters :- name:String

407# Purpose :- Holds every attribute and operation that can be done on a screen object

408###########################

409class Screen**(**Main\_Objects**):**

410 **def** \_\_init\_\_**(**self**,** name**):**

411 **super().**\_\_init\_\_**(**name**)**

412 self**.**width **=** 10

413 self**.**colour **=** **(**169**,** 169**,** 169**)**

414 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

415 self**.**Property **=** "Stop"

416

417 self**.**full\_name **=** "Screen" **+** **str(**self**.**defined\_name**[-**1**])**

418

419 ##### redraw1 #######

420 # Parameters :- None

421 # Return Type :- None

422 # Purpose :- updates instance of object on screen

423 ###########################

424 **def** redraw1**(**self**):**

425 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** self**.**colour**,**

426 **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

427 self**.**update**()**

428

429

430##### Diffraction #######

431# Inheirts :- Main\_Objects

432# Parameters :- name:String

433# Purpose :- Holds every attribute and operation that can be done on diffraction objects

434###########################

435class Diffraction**(**Main\_Objects**):**

436 **def** \_\_init\_\_**(**self**,** name**):**

437 **super().**\_\_init\_\_**(**name**)**

438 self**.**width **=** 5

439 self**.**length **=** 50

440 self**.**slitdis **=** 0.0002

441 self**.**scrlength **=** 1500

442 self**.**screendis **=** 500

443 self**.**sourcexp **=** self**.**xp **-** 100

444 self**.**sourceyp **=** self**.**yp **-** self**.**length **-** self**.**slitdis **+** self**.**length **/** 2

445 self**.**source\_draging **=** **False**

446 self**.**wavelength **=** 600 **\*** 10 **\*\*** **-**9

447 self**.**yp **=** **(**length **/** 2**)** **-** **(**self**.**slitdis **+** self**.**length**)**

448 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

449 **(**self**.**xp**,** self**.**yp **-** self**.**length **-** self**.**slitdis**,** self**.**width**,**

450 self**.**length**))**

451 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

452 self**.**graphical3 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

453 **(**self**.**xp**,** self**.**yp **+** self**.**length **+** self**.**slitdis**,** self**.**width**,**

454 self**.**length**))**

455 self**.**graphical4 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

456 **(**self**.**xp **+** self**.**screendis**,** 0**,** self**.**width**,** self**.**scrlength**))**

457 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**self**.**sourcexp**,** self**.**sourceyp**),** 7**)**

458 self**.**Property **=** "D"

459 self**.**a **=** 2 **\*** self**.**slitdis **\*** 3 **\*** 10 **\*\*** **-**3 **+** self**.**length **\*** 3 **\*** 10 **\*\*** **-**5

460 self**.**full\_name **=** "Diffraction" **+** **str(**self**.**defined\_name**[-**1**])**

461 self**.**fringe **=** **(**self**.**wavelength **\*** self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **/** self**.**a

462 self**.**update**()**

463

464 ##### update #######

465 # Parameters :- None

466 # Return Type :- None

467 # Purpose :- checks if properties of diffraction experiment is allowed to be

468 # displayed and displays them if so

469 ###########################

470 **def** update**(**self**):**

471 self**.**a **=** 2 **\*** self**.**slitdis **\*** 3 **\*** 10 **\*\*** **-**3 **+** self**.**length **\*** 3 **\*** 10 **\*\*** **-**5

472 self**.**fringe **=** **(**self**.**wavelength **\*** self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **/** self**.**a

473 **if** m**.**SlitS**:**

474 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

475 text **=** font**.**render**(**"Slit Seperation:" **+** **str(**self**.**a**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

476 textrect **=** text**.**get\_rect**(**center**=(**150**,** 80**))**

477 screen**.**blit**(**text**,** textrect**)**

478 **if** m**.**FS**:**

479 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

480 text **=** font**.**render**(**"Fringe seperation:" **+** **str(**self**.**fringe**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

481 textrect **=** text**.**get\_rect**(**center**=(**150**,** 95**))**

482 screen**.**blit**(**text**,** textrect**)**

483 **if** m**.**W**:**

484 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

485 text **=** font**.**render**(**"Wavelength:" **+** **str(**self**.**wavelength**)** **+** "nm"**,** **True,** **(**105**,** 105**,** 105**))**

486 textrect **=** text**.**get\_rect**(**center**=(**150**,** 110**))**

487 screen**.**blit**(**text**,** textrect**)**

488 **if** m**.**SourceS**:**

489 font **=** pygame**.**font**.**SysFont**(**'cambriacambriamath'**,** 20**)**

490 text **=** font**.**render**(**"Screen distance:" **+** **str(**self**.**screendis **\*** 3 **\*** 10 **\*\*** **-**3**)** **+** "m"**,** **True,** **(**105**,** 105**,** 105**))**

491 textrect **=** text**.**get\_rect**(**center**=(**150**,** 125**))**

492 screen**.**blit**(**text**,** textrect**)**

493

494 ##### redraw1 #######

495 # Parameters :- None

496 # Return Type :- None

497 # Purpose :- updates instance of object on screen

498 ###########################

499 **def** redraw1**(**self**):**

500 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

501 self**.**graphical **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

502 **(**self**.**xp**,** self**.**yp **-** self**.**length **-** self**.**slitdis **-** 200**,** self**.**width**,**

503 self**.**length **+** 200**))**

504 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),** **(**self**.**xp**,** self**.**yp**,** self**.**width**,** self**.**length**))**

505 self**.**graphical3 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

506 **(**self**.**xp**,** self**.**yp **+** self**.**length **+** self**.**slitdis**,** self**.**width**,**

507 self**.**length **+** 200**))**

508 self**.**graphical4 **=** pygame**.**draw**.**rect**(**screen**,** **(**169**,** 169**,** 169**),**

509 **(**self**.**xp **+** self**.**screendis**,** 75**,** self**.**width**,** self**.**scrlength**))**

510 self**.**source **=** pygame**.**draw**.**circle**(**screen**,** **(**255**,** 193**,** 110**),** **(**self**.**sourcexp**,** self**.**sourceyp**),** 7**)**

511 self**.**update**()**

512

513

514##### Semi\_Circle #######

515# Inherits :- Main\_Objects

516# Parameters :- name:String

517# Purpose :- Holds every attribute and operation that can be done on a semi circle object

518###########################

519class Semi\_Circle**(**Main\_Objects**):**

520 **def** \_\_init\_\_**(**self**,** name**):**

521 **super().**\_\_init\_\_**(**name**)**

522 x**,** y **=** sympy**.**symbols**(**"x,y"**)**

523 self**.**Radius **=** 100

524 self**.**graphical **=** pygame**.**draw**.**circle**(**screen**,** **(**168**,** 204**,** 215**),** **(**self**.**xp**,** self**.**yp**),** self**.**Radius**)**

525 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

526 **(**self**.**xp **-** self**.**Radius**,** self**.**yp**,** self**.**Radius **\*** 2**,**

527 self**.**Radius**))**

528 self**.**Refractive\_Index **=** 1.52

529 self**.**Equation **=** **(**"((x-{0})\*\*2 + (y-{1})\*\*2)-{2}"**.format(int(**self**.**xp**),** **int(**self**.**yp**),** **int(**self**.**Radius **\*\*** 2**)))**

530 self**.**Gradient **=** sympy**.**idiff**(eval(**self**.**Equation**),** y**,** x**)**

531 self**.**full\_name **=** "SemiCircle" **+** **str(**self**.**defined\_name**[-**1**])**

532

533 ##### redraw1 #######

534 # Parameters :- None

535 # Return Type :- None

536 # Purpose :- updates instance of object on screen

537 ###########################

538 **def** redraw1**(**self**):**

539 self**.**graphical **=** pygame**.**draw**.**circle**(**screen**,** **(**168**,** 204**,** 215**),** **(**self**.**xp**,** self**.**yp**),** self**.**Radius**)**

540 self**.**graphical2 **=** pygame**.**draw**.**rect**(**screen**,** **(**0**,** 0**,** 0**),**

541 **(**self**.**xp **-** self**.**Radius**,** self**.**yp**,** self**.**Radius **\*** 2**,**

542 self**.**Radius**))**

543 self**.**update**()**

544

545

546##### Button #######

547# Parameters :- id:int, def\_name:string

548# Purpose :- creates button and displays it on screen

549###########################

550class Button**:**

551 **def** \_\_init\_\_**(**self**,** **id,** def\_name**):**

552 self**.**defined\_name **=** def\_name

553 self**.**x**,** self**.**y **=** **(**80**,** 45**)**

554 self**.id** **=** **id**

555 toolbar\_button**.**append**(**def\_name**)**

556

557 **def** draw**(**self**,** text**):**

558 self**.**rect **=** pygame**.**draw**.**rect**(**screen**,** **(**160**,** 160**,** 160**),** **(**10 **+** **(**12 **\*** self**.id),** 15**,** self**.**x**,** self**.**y**))**

559 **if** text **!=** ''**:**

560 font **=** pygame**.**font**.**SysFont**(**'comicsans'**,** 20**)**

561 text **=** font**.**render**(**text**,** **True,** **(**0**,** 0**,** 0**))**

562 textrect **=** text**.**get\_rect**(**center**=((**self**.**x **/** 2**)** **+** 10 **+** **(**12 **\*** self**.id),** **(**self**.**y **/** 2**)** **+** 15**))**

563 screen**.**blit**(**text**,** textrect**)**

564

565 **def** click**(**self**,** event**):**

566 x**,** y **=** pygame**.**mouse**.**get\_pos**()**

567 **if** event**.type** **==** pygame**.**MOUSEBUTTONDOWN**:**

568 **if** self**.**rect**.**collidepoint**(**x**,** y**):**

569 **return** self**.**defined\_name

570 **else:**

571 **return** **False**

572

573

574##### ShowGUI #######

575# Parameters :- None

576# Return Type :- None

577# Purpose :- dictates which properties of an experiment can be displayed

578###########################

579class ShowGUI**:**

580 **def** \_\_init\_\_**(**self**):**

581 self**.**IA **=** **False**

582 self**.**RA **=** **False**

583 self**.**RI **=** **False**

584 self**.**SlitS **=** **False**

585 self**.**FS **=** **False**

586 self**.**SourceS **=** **False**

587 self**.**W **=** **False**

588

589

590m **=** ShowGUI**()**

591run **=** **True**

592conn **=** Network**()**

593connect **=** conn**.**getP**()**

594previous\_ol **=** **[]**

595previous\_sl **=** **[]**

596previous\_al **=** **[]**

597chatter**=None**

598chat**=False**

599while run**:**

600 pygame**.**draw**.**rect**(**screen**,** **(**105**,** 105**,** 105**),** pygame**.**Rect**(**0**,** 0**,** width**,** 75**))**

601 Quit **=** Button**(**8**,** "Quit"**)**

602 Quit**.**draw**(**"Quit"**)**

603 Chat **=** Button**(**0**,** "Chat"**)**

604 Chat**.**draw**(**"Chat"**)**

605 toolbar\_button **=** **list(dict.**fromkeys**(**toolbar\_button**))**

606 true\_load **=** conn**.**rec**()**

607 **while** true\_load **is** **None:**

608 **for** event **in** pygame**.**event**.**get**():**

609 pos **=** pygame**.**mouse**.**get\_pos**()**

610 y\_axis **=** pos**[**1**]**

611 **if** event**.type** **==** pygame**.**QUIT**:**

612 conn**.**disconnect**()**

613 pygame**.quit()**

614 sys**.exit()**

615

616 **elif** event**.type** **==** pygame**.**MOUSEBUTTONDOWN **and** y\_axis **<=** 75**:**

617 **for** i **in** toolbar\_button**:**

618 clicktest **=** **eval(**i**).**click**(**event**)**

619 **if** clicktest **==** "Chat"**:**

620 chatter**=**ChatboxClient

621 **try:**

622 chat**=**chatter**.**Client**()**

623 **except** **ConnectionRefusedError:**

624 **continue**

625 **elif** clicktest **==** "Quit"**:**

626 conn**.**disconnect**()**

627 pygame**.quit()**

628 **if** chat**:**

629 chat**.**stop**()**

630 sys**.exit()**

631 true\_load**=**conn**.**rec**()**

632 loaded **=** **[]**

633 **while** "PAUSE" **not** **in** **str(**true\_load**):**

634 **if** "STOP" **in** **str(**true\_load**):**

635 conn**.**disconnect**()**

636 pygame**.quit()**

637 **if** chat**:**

638 chat**.**stop**()**

639 sys**.exit()**

640 **if** true\_load **is** **not** **None:**

641 loaded**.**append**(**true\_load**)**

642 true\_load **=** conn**.**rec**()**

643 **if** loaded **is** **not** **None** **and** loaded **!=** "{1: 'dont redraw'}"**:**

644 screen**.**fill**((**0**,** 0**,** 0**),** **(**0**,** 75**,** 1920**,** 1120**))**

645 **if** "Source" **in** **str(**loaded**):**

646 sources\_loaded **=** **[]**

647 **if** "Block" **in** **str(**loaded**)** **or** "Screen" **in** **str(**loaded**)** **or** "Circle" **in** **str(**loaded**)** **or** "Diffraction" **in** **str(**

648 loaded**)** **or** "Mirror" **in** **str(**loaded**):**

649 objects\_loaded **=** **[]**

650 **if** "theta" **in** **str(**loaded**):**

651 ang **=** **[]**

652 loaded**=eval(str(**loaded**).**replace**(**"]["**,**"],["**))**

653 **if** "Diffraction" **in** **str(**loaded**):**

654 name**=**""

655 **for** pickled **in** loaded**:**

656 pickled **=** **eval(**pickled**)**

657 pickled **=** pickled**[**0**]**

658 **if** "Diffraction" **in** **str(**pickled**):**

659 Tag **=** tag\_generator**(**tag**,** "D"**)**

660 name**=**Tag

661 pick **=** Diffraction**(**Tag**)**

662 objects\_loaded**.**append**(**pick**)**

663 **for** key **in** pickled**:**

664 **if** "graphical" **in** **str(**key**):**

665 **pass**

666 **else:**

667 **setattr(**pick**,** key**,** pickled**[**key**])**

668 **elif** "ShowGUI" **in** **str(**pickled**):**

669 **if** **len(**pickled**)** **==** 1**:**

670 **for** key **in** pickled**[**0**]:**

671 **setattr(**m**,** key**,** pickled**[**0**][**key**])**

672 **else:**

673 **for** key **in** pickled**:**

674 **setattr(**m**,** key**,** pickled**[**key**])**

675 **elif** "Source" **in** **str(**pickled**):**

676 Tag **=** tag\_generator**(**tag**,** "R"**)**

677 pick **=** Source**(**Tag**)**

678 sources\_loaded**.**append**(**pick**)**

679 **if** **len(**pickled**)** **==** 1**:**

680 **for** key **in** pickled**[**0**]:**

681 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

682 **else:**

683 **for** key **in** pickled**:**

684 **setattr(**pick**,** key**,** pickled**[**key**])**

685

686 **else:**

687 **for** pickled **in** loaded**:**

688 **if** pickled **is** **not** **None:**

689 pickled **=** **eval(**pickled**)**

690 pickled **=** pickled**[**0**]**

691 **if** "GlassBlock" **in** **str(**pickled**):**

692 Tag **=** tag\_generator**(**tag**,** "GB"**)**

693 pick **=** GlassBlock**(**Tag**)**

694 objects\_loaded**.**append**(**pick**)**

695 **for** key **in** pickled**:**

696 **if** **len(**pickled**)** **==** 1**:**

697 **for** key **in** pickled**[**0**]:**

698 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

699 #try execpt type error

700 **else:**

701 **for** key **in** pickled**:**

702 **setattr(**pick**,** key**,** pickled**[**key**])**

703 **elif** "ShowGUI" **in** **str(**pickled**):**

704 **if** **len(**pickled**)==**1**:**

705 **for** key **in** pickled**[**0**]:**

706 **setattr(**m**,** key**,** pickled**[**0**][**key**])**

707 **else:**

708 **for** key **in** pickled**:**

709 **setattr(**m**,** key**,** pickled**[**key**])**

710 **elif** "Block" **in** **str(**pickled**):**

711 Tag **=** tag\_generator**(**tag**,** "B"**)**

712 pick **=** Block**(**Tag**)**

713 objects\_loaded**.**append**(**pick**)**

714 **for** key **in** pickled**:**

715 **if** **len(**pickled**)** **==** 1**:**

716 **for** key **in** pickled**[**0**]:**

717 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

718 **else:**

719 **for** key **in** pickled**:**

720 **setattr(**pick**,** key**,** pickled**[**key**])**

721 **elif** "Screen" **in** **str(**pickled**):**

722 Tag **=** tag\_generator**(**tag**,** "S"**)**

723 pick **=** Screen**(**Tag**)**

724 objects\_loaded**.**append**(**pick**)**

725 **for** key **in** pickled**:**

726 **if** **len(**pickled**)** **==** 1**:**

727 **for** key **in** pickled**[**0**]:**

728 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

729 **else:**

730 **for** key **in** pickled**:**

731 **setattr(**pick**,** key**,** pickled**[**key**])**

732 **elif** "Semi\_Circle" **in** **str(**pickled**):**

733 Tag **=** tag\_generator**(**tag**,** "SC"**)**

734 pick **=** Semi\_Circle**(**Tag**)**

735 objects\_loaded**.**append**(**pick**)**

736 **for** key **in** pickled**:**

737 **if** **len(**pickled**)** **==** 1**:**

738 **for** key **in** pickled**[**0**]:**

739 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

740 **else:**

741 **for** key **in** pickled**:**

742 **setattr(**pick**,** key**,** pickled**[**key**])**

743 **elif** "Mirror" **in** **str(**pickled**):**

744 Tag **=** tag\_generator**(**tag**,** "M"**)**

745 pick **=** Mirror**(**Tag**)**

746 objects\_loaded**.**append**(**pick**)**

747 **for** key **in** pickled**:**

748 **if** **len(**pickled**)** **==** 1**:**

749 **for** key **in** pickled**[**0**]:**

750 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

751 **else:**

752 **for** key **in** pickled**:**

753 **setattr(**pick**,** key**,** pickled**[**key**])**

754 **elif** "Source" **in** **str(**pickled**):**

755 Tag **=** tag\_generator**(**tag**,** "R"**)**

756 pick **=** Source**(**Tag**)**

757 sources\_loaded**.**append**(**pick**)**

758 **for** key **in** pickled**:**

759 **if** **len(**pickled**)** **==** 1**:**

760 **for** key **in** pickled**[**0**]:**

761 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

762 **else:**

763 **for** key **in** pickled**:**

764 **setattr(**pick**,** key**,** pickled**[**key**])**

765 **elif** "theta" **in** **str(**pickled**):**

766 Tag **=** tag\_generator**(**tag**,** "A"**)**

767 pick **=** Angle**()**

768 ang**.**append**(**pick**)**

769 **if** **len(**pickled**)==**1**:**

770 **for** key **in** pickled**[**0**]:**

771 **setattr(**pick**,** key**,** pickled**[**0**][**key**])**

772 **else:**

773 **for** key **in** pickled**:**

774 **setattr(**pick**,** key**,** pickled**[**key**])**

775 **if** **len(**objects\_loaded**)** **>=** 1**:**

776 **for** n **in** objects\_loaded**:**

777 **if** n **==** 1**:**

778 **pass**

779 **else:**

780 n**.**redraw1**()**

781 **else:**

782 **pass**

783 **if** **len(**sources\_loaded**)** **>=** 1**:**

784 **for** n **in** sources\_loaded**:**

785 **if** n **==** 1**:**

786 **pass**

787 **else:**

788 n**.**redraw1**()**

789 **else:**

790 **pass**

791

792 **if** **len(**ang**)** **>=** 1**:**

793 **for** n **in** ang**:**

794 **if** n **==** 1**:**

795 **pass**

796 **else:**

797 n**.**redraw1**()**

798 **else:**

799 **pass**

800

801 **if** **len(**objects\_loaded**)** **>=** 1**:**

802 **for** n **in** objects\_loaded**:**

803 **if** n **==** 1**:**

804 **pass**

805 **else:**

806 n**.**redraw1**()**

807 **else:**

808 **pass**

809 **if** **len(**sources\_loaded**)** **>=** 1**:**

810 **for** n **in** sources\_loaded**:**

811 **if** n **==** 1**:**

812 **pass**

813 **else:**

814 n**.**redraw1**()**

815 **else:**

816 **pass**

817 **if** **len(**ang**)** **>=** 1**:**

818 **for** n **in** ang**:**

819 **if** n **==** 1**:**

820 **pass**

821 **else:**

822 n**.**redraw1**()**

823 **else:**

824 **pass**

825 **elif** loaded **is** **None:**

826 **continue**

827

828 pygame**.**display**.**flip**()**

## ChatboxServer.py

1 **import** math

2 **import** socket

3 **import** threading

4 **import** tkinter

5 **import** tkinter**.**scrolledtext

6 **from** math **import** **\***

7 **import** time

8

9

10 ##### Server #######

11 # Parameters :- None

12 # Purpose :- handles incoming chat messages from clients and allows teacher to broadcast messages to students

13 # it also calculates the answers to questions set and sets a timer for time to answer question

14 ###########################

15 **class** **Server:**

16 **def** \_\_init\_\_**(**self**):**

17 self**.**host **=** socket**.**gethostname**()**

18 self**.**port **=** 55559

19 self**.**clients **=** **[]**

20 self**.**nicknames **=** **[]**

21 self**.**answered **=** **{}**

22 self**.**answers **=** 0

23 self**.**question **=** **[]**

24 self**.**sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**

25 self**.**sock**.**bind**((**self**.**host**,** self**.**port**))**

26 self**.**sock**.**listen**()**

27 self**.**nickname **=** "Server"

28 self**.**gui\_done **=** **False**

29 self**.**running **=** **True**

30 self**.**timer **=** 0

31 self**.**asked **=** **False**

32 self**.**t0 **=** 0

33 gui\_thread **=** threading**.**Thread**(**target**=**self**.**gui\_loop**)**

34 receive\_thread **=** threading**.**Thread**(**target**=**self**.**receive**)**

35 gui\_thread**.**start**()**

36 receive\_thread**.**start**()**

37

38 ##### guestions ######

39 # Parameters :- timeset:int, question:string, inputs:list

40 # Return Type :- None

41 # Purpose :- sets up the questions on the server and calculate the answer for each question

42 # sets the time everyone has to answer the question

43 # displays the question to everyone

44 ###############################

45 **def** questions**(**self**,** timeset**,** question**,** inputs**):**

46 **if** inputs**[-**1**]** **==** "created"**:**

47 self**.**broadcast**((**"QUESTION:" **+** question**).**encode**())**

48 self**.**asked **=** **True**

49 self**.**question**.**append**(**question**)**

50 self**.**answers **=** inputs**[**0**]**

51 **print(**timeset**)**

52 self**.**timer **=** **int(**timeset**)**

53 self**.**t0 **=** time**.**time**()**

54 **elif** "index" **in** question**:**

55 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

56 self**.**asked **=** **True**

57 self**.**question**.**append**(**question**)**

58 self**.**answers **=** **round(**sin**(**math**.**radians**(**inputs**[**1**]))** **/** sin**(**math**.**radians**(**inputs**[**0**])),**1**)**

59 self**.**timer **=** **int(**timeset**)**

60 self**.**t0 **=** time**.**time**()**

61 **elif** "critical" **in** question **and** **len(**inputs**)** **==** 1**:**

62 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

63 self**.**asked **=** **True**

64 self**.**question**.**append**(**question**)**

65 self**.**answers **=** **round(**asin**(**1 **/** math**.**radians**(**inputs**[**0**]))** **\*** 180 **/** pi**,**1**)**

66 self**.**timer **=** **int(**timeset**)**

67 self**.**t0 **=** time**.**time**()**

68 **elif** "fringes" **in** question**:**

69 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

70 self**.**asked **=** **True**

71 self**.**question**.**append**(**question**)**

72 self**.**answers **=** **(round((**inputs**[**0**]** **\*** inputs**[**2**]** **/** inputs**[**1**])),**1**)**

73 self**.**timer **=** **int(**timeset**)**

74 self**.**t0 **=** time**.**time**()**

75 **elif** "slits" **and** "screen" **in** question**:**

76 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

77 self**.**asked **=** **True**

78 self**.**question**.**append**(**question**)**

79 self**.**answers **=** **(round((**inputs**[**0**]** **\*** inputs**[**2**]** **/** inputs**[**1**])),**1**)**

80 self**.**timer **=** **int(**timeset**)**

81 self**.**t0 **=** time**.**time**()**

82 **elif** "slits" **in** question**:**

83 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

84 self**.**asked **=** **True**

85 self**.**question**.**append**(**question**)**

86 self**.**answers **=** **(round((**inputs**[**0**]** **\*** inputs**[**2**]** **/** inputs**[**1**])),**1**)**

87 self**.**timer **=** **int(**timeset**)**

88 self**.**t0 **=** time**.**time**()**

89 **elif** "wavelength" **in** question**:**

90 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

91 self**.**asked **=** **True**

92 self**.**question**.**append**(**question**)**

93 self**.**answers **=** **(round((**inputs**[**0**]** **\*** inputs**[**1**]** **/** inputs**[**2**])),**1**)**

94 self**.**timer **=** **int(**timeset**)**

95 self**.**t0 **=** time**.**time**()**

96 **elif** "order" **in** question **and** **len(**inputs**)** **==** 3**:**

97 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

98 self**.**asked **=** **True**

99 self**.**question**.**append**(**question**)**

100 wavelength **=** **((**inputs**[**0**]** **\*** inputs**[**1**])** **/** inputs**[**2**])**

101 self**.**answers **=** **round(**asin**(**math**.**radians**((**inputs**[**3**]** **\*** wavelength**)** **/** inputs**[**2**]))** **\*** 180 **/** pi**,**1**)**

102 self**.**timer **=** **int(**timeset**)**

103 self**.**t0 **=** time**.**time**()**

104 **elif** "refraction" **in** question**:**

105 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

106 self**.**asked **=** **True**

107 self**.**question**.**append**(**question**)**

108 self**.**answers **=** **round(**asin**(**math**.**radians**(**inputs**[**1**])** **\*** sin**(**math**.**radians**(**inputs**[**0**])))** **\*** 180 **/** pi**,**1**)**

109 self**.**timer **=** **int(**timeset**)**

110 self**.**t0 **=** time**.**time**()**

111 **elif** "incidence" **in** question**:**

112 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

113 self**.**asked **=** **True**

114 self**.**question**.**append**(**question**)**

115 self**.**answers **=** **round((**asin**(**sin**(**math**.**radians**(**inputs**[**0**])** **/** math**.**radians**(**inputs**[**1**])))),**1**)**

116 self**.**timer **=** **int(**timeset**)**

117 self**.**t0 **=** time**.**time**()**

118 **elif** "order" **in** question**:**

119 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

120 self**.**asked **=** **True**

121 self**.**question**.**append**(**question**)**

122 self**.**answers **=** **round(**asin**(**math**.**radians**((**inputs**[**0**]** **\*** inputs**[**2**])** **/** inputs**[**1**])),**1**)**

123 self**.**timer **=** **int(**timeset**)**

124 self**.**t0 **=** time**.**time**()**

125 **elif** "critcal"**:**

126 self**.**broadcast**((**"QUESTION:" **+** question **+** "\n"**).**encode**())**

127 self**.**asked **=** **True**

128 self**.**question**.**append**(**question**)**

129 RI **=** **(**sin**(**math**.**radians**(**inputs**[**0**]))** **/** sin**(**math**.**radians**(**inputs**[**1**])))**

130 self**.**answers **=** **round(**asin**(**1 **/** RI**)** **\*** 180 **/** pi**,**1**)**

131 self**.**timer **=** **int(**timeset**)**

132 self**.**t0 **=** time**.**time**()**

133 # return self.asked, self.timer,self.t0,self.answers,self.question

134

135 ##### gui\_loop ######

136 # Parameters :- None

137 # Return Type :- None

138 # Purpose :- creates the gui for the chatbox

139 ###########################

140 **def** gui\_loop**(**self**):**

141 self**.**win **=** tkinter**.**Tk**()**

142 self**.**win**.**title**(**"Chatbox"**)**

143 self**.**win**.**configure**(**bg**=**"lightgray"**)**

144 self**.**chat\_label **=** tkinter**.**Label**(**self**.**win**,** text**=**"Chat:"**,** bg**=**"lightgray"**)**

145 self**.**chat\_label**.**configure**(**font**=(**"Arial"**,** 12**))**

146 self**.**chat\_label**.**pack**(**padx**=**20**,** pady**=**5**)**

147

148 self**.**text\_area **=** tkinter**.**scrolledtext**.**ScrolledText**(**self**.**win**)**

149 self**.**text\_area**.**pack**(**padx**=**20**,** pady**=**5**)**

150 self**.**text\_area**.**config**(**state**=**"disable"**)**

151

152 self**.**msg\_label **=** tkinter**.**Label**(**self**.**win**,** text**=**"Message:"**,** bg**=**"lightgray"**)**

153 self**.**msg\_label**.**configure**(**font**=(**"Arial"**,** 12**))**

154 self**.**msg\_label**.**pack**(**padx**=**20**,** pady**=**5**)**

155

156 self**.**input\_area **=** tkinter**.**Text**(**self**.**win**,** height**=**3**)**

157 self**.**input\_area**.**pack**(**padx**=**20**,** pady**=**5**)**

158

159 self**.**send\_button **=** tkinter**.**Button**(**self**.**win**,** text**=**"Send"**,** command**=**self**.**write**)**

160 self**.**send\_button**.**config**(**font**=(**"Arial"**,** 12**))**

161 self**.**send\_button**.**pack**(**padx**=**20**,** pady**=**5**)**

162

163 self**.**gui\_done **=** **True**

164 self**.**win**.**protocol**(**"WM\_DELETE\_WINDOW"**,** self**.**stop**)**

165 self**.**win**.**mainloop**()**

166

167 ##### broadcast ######

168 # Parameters :- message:string

169 # Return Type :- None

170 # Purpose :- sends parameter message to everyone connected to the server

171 ###########################

172 **def** broadcast**(**self**,** message**):**

173 **for** client **in** self**.**clients**:**

174 client**.**send**(**message**)**

175

176 ##### receive ######

177 # Parameters :- None

178 # Return Type :- None

179 # Purpose :- establishes connection between client and receives nickname and student id of client and stores it

180 ###########################

181 **def** receive**(**self**):**

182 **while** self**.**running**:**

183 client**,** address **=** self**.**sock**.**accept**()**

184 **if** self**.**gui\_done**:**

185 self**.**text\_area**.**config**(**state**=**"normal"**)**

186 self**.**text\_area**.**insert**(**"end"**,** f"Connected with{**str(**address**)**}!\n"**)**

187 self**.**text\_area**.**yview**(**"end"**)**

188 self**.**text\_area**.**config**(**state**=**"disable"**)**

189 client**.**send**(**"NICK"**.**encode**(**"utf-8"**))**

190 nickname **=** client**.**recv**(**1024**).**decode**()**

191 client**.**send**(**"ID"**.**encode**(**"utf-8"**))**

192 stuID **=** client**.**recv**(**1024**).**decode**()**

193 self**.**answered**[**stuID**]** **=** **[]**

194 self**.**nicknames**.**append**([**nickname**,** stuID**])**

195 self**.**clients**.**append**(**client**)**

196 **if** self**.**gui\_done**:**

197 self**.**text\_area**.**config**(**state**=**"normal"**)**

198 self**.**text\_area**.**insert**(**"end"**,** f"Nickname of the client is {nickname}\n"**)**

199 self**.**text\_area**.**yview**(**"end"**)**

200 self**.**text\_area**.**config**(**state**=**"disable"**)**

201 self**.**broadcast**(**f"{nickname} joined the chat!\n"**.**encode**(**"utf-8"**))**

202 **if** self**.**gui\_done**:**

203 self**.**text\_area**.**config**(**state**=**"normal"**)**

204 self**.**text\_area**.**insert**(**"end"**,** f"{nickname} joined the chat!\n"**.**encode**(**"utf-8"**))**

205 self**.**text\_area**.**yview**(**"end"**)**

206 self**.**text\_area**.**config**(**state**=**"disable"**)**

207 client**.**send**(**"Connected to the server\n"**.**encode**(**"utf-8"**))**

208 thread **=** threading**.**Thread**(**target**=**self**.**handle**,** args**=(**client**,))**

209 thread**.**start**()**

210

211 ##### stop ######

212 # Parameters :- None

213 # Return Type :- None

214 # Purpose :- stops server and returns values needed to create report

215 ###########################

216 **def** stop**(**self**):**

217 self**.**running **=** **False**

218 self**.**broadcast**(**"^^++"**.**encode**(**"utf-8"**))**

219 self**.**sock**.**close**()**

220 **with** **open(**"return.txt"**,** "r"**)** **as** f**:**

221 t **=** f**.**read**()**

222 t **=** **eval(**t**)** **+** **[**self**.**question**,** self**.**answered**]**

223 f**.**close**()**

224 **with** **open(**"return.txt"**,** "w"**)** **as** f**:**

225 f**.**write**(str(**t**))**

226 f**.**close**()**

227

228 ##### handle ######

229 # Parameters :- client:object

230 # Return Type :- None

231 # Purpose :- handles the incoming message

232 ###########################

233 **def** handle**(**self**,** client**):**

234 **while** self**.**running**:**

235 **if** self**.**asked**:**

236 current\_time **=** time**.**time**()**

237 dur **=** **(**current\_time **-** self**.**t0**)**

238 **if** dur **>=** self**.**timer**:**

239 self**.**answers **=** 0

240 self**.**timer **=** 0

241 self**.**asked **=** **False**

242 **for** key **in** self**.**answered**:**

243 **while** **len(**self**.**answered**[**key**])** **!=** **len(**self**.**question**):**

244 self**.**answered**[**key**].**append**(**0**)**

245

246 **try:**

247 message **=** client**.**recv**(**1024**)**

248 **if** self**.**asked**:**

249 **if** "/answer" **in** message**.**decode**()** **or** "/Answer" **in** message**.**decode**():**

250 student **=** self**.**nicknames**[**self**.**clients**.**index**(**client**)][**1**]**

251 **if** "/answer" **in** message**.**decode**():**

252 answer **=** message**.**decode**().**partition**(**"/answer "**)[**2**]**

253 **elif** "/Answer" **in** message**.**decode**():**

254 answer **=** message**.**decode**().**partition**(**"/Answer "**)[**2**]**

255 **if** **float(**answer**)** **==** **float(**self**.**answers**)** **and** **len(**self**.**answered**[**student**])** **!=** **len(**self**.**question**):**

256 self**.**answered**[**student**].**append**(**1**)**

257 **elif** **float(**answer**)** **!=** **float(**self**.**answers**)** **and** **len(**self**.**answered**[**student**])** **!=** **len(**self**.**question**):**

258 self**.**answered**[**student**].**append**(**0**)**

259 **else:**

260 **print(**f"\n{self**.**nicknames**[**self**.**clients**.**index**(**client**)][**0**]**} says {message}"**)**

261 self**.**broadcast**(**message**)**

262 **if** self**.**gui\_done**:**

263 self**.**text\_area**.**config**(**state**=**"normal"**)**

264 self**.**text\_area**.**insert**(**"end"**,** message**)**

265 self**.**text\_area**.**yview**(**"end"**)**

266 self**.**text\_area**.**config**(**state**=**"disable"**)**

267

268 **else:**

269 **if** "/answer" **in** message**.**decode**()** **or** "/Answer" **in** message**.**decode**():**

270 **print(**"Blocked"**)**

271 **else:**

272 **print(**f"{self**.**nicknames**[**self**.**clients**.**index**(**client**)][**0**]**} says {message}"**)**

273 self**.**broadcast**(**message**)**

274 **if** self**.**gui\_done**:**

275 self**.**text\_area**.**config**(**state**=**"normal"**)**

276 self**.**text\_area**.**insert**(**"end"**,** message**)**

277 self**.**text\_area**.**yview**(**"end"**)**

278 self**.**text\_area**.**config**(**state**=**"disable"**)**

279 **except:**

280 index **=** self**.**clients**.**index**(**client**)**

281 self**.**clients**.**remove**(**client**)**

282 client**.**close**()**

283 nickname **=** self**.**nicknames**[**index**]**

284 self**.**nicknames**.**remove**(**nickname**)**

285 **break**

286

287 ##### write #######

288 # Parameters :- None

289 # Return Type :- None

290 # Purpose :- writes what the server/teacher/host has typed in the chat box

291 ###########################

292 **def** write**(**self**):**

293 message **=** f"{self**.**nickname}:{self**.**input\_area**.**get**(**'1.0'**,** 'end'**)**}"

294

295 **if** self**.**gui\_done**:**

296 self**.**text\_area**.**config**(**state**=**"normal"**)**

297 self**.**text\_area**.**insert**(**"end"**,** message**)**

298 self**.**text\_area**.**yview**(**"end"**)**

299 self**.**text\_area**.**config**(**state**=**"disable"**)**

300 self**.**input\_area**.**delete**(**'1.0'**,** 'end'**)**

301 self**.**broadcast**(**message**.**encode**(**"utf-8"**))**

## ChatboxClient.py

1 **import** socket

2 **import** threading

3 **import** tkinter

4 **import** tkinter**.**scrolledtext

5 **from** tkinter **import** simpledialog

6

7

8 **class** **Client:**

9 **def** \_\_init\_\_**(**self**):**

10 self**.**host **=** socket**.**gethostname**()**

11 self**.**port **=** 55559

12 self**.**sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**

13 self**.**sock**.**connect**((**self**.**host**,** self**.**port**))**

14

15 msg **=** tkinter**.**Tk**()**

16 msg**.**withdraw**()**

17

18 self**.**nickname **=** simpledialog**.**askstring**(**"Nickname"**,** "Please choose a nickname"**,** parent**=**msg**)**

19 self**.**stuId **=** simpledialog**.**askstring**(**"StudentID"**,** "Enter your student id"**,** parent**=**msg**)**

20 self**.**gui\_done **=** **False**

21 self**.**running **=** **True**

22 gui\_thread **=** threading**.**Thread**(**target**=**self**.**gui\_loop**)**

23 receive\_thread **=** threading**.**Thread**(**target**=**self**.**receive**)**

24 gui\_thread**.**start**()**

25 receive\_thread**.**start**()**

26

27 # creates the gui for the chatbox

28 **def** gui\_loop**(**self**):**

29 self**.**win **=** tkinter**.**Tk**()**

30 self**.**win**.**title**(**"Chatbox"**)**

31 self**.**win**.**configure**(**bg**=**"lightgray"**)**

32 self**.**chat\_label **=** tkinter**.**Label**(**self**.**win**,** text**=**"Chat:"**,** bg**=**"lightgray"**)**

33 self**.**chat\_label**.**configure**(**font**=(**"Arial"**,** 12**))**

34 self**.**chat\_label**.**pack**(**padx**=**20**,** pady**=**5**)**

35

36 self**.**text\_area **=** tkinter**.**scrolledtext**.**ScrolledText**(**self**.**win**)**

37 self**.**text\_area**.**pack**(**padx**=**20**,** pady**=**5**)**

38 self**.**text\_area**.**config**(**state**=**"disable"**)**

39

40 self**.**msg\_label **=** tkinter**.**Label**(**self**.**win**,** text**=**"Message:"**,** bg**=**"lightgray"**)**

41 self**.**msg\_label**.**configure**(**font**=(**"Arial"**,** 12**))**

42 self**.**msg\_label**.**pack**(**padx**=**20**,** pady**=**5**)**

43

44 self**.**input\_area **=** tkinter**.**Text**(**self**.**win**,** height**=**3**)**

45 self**.**input\_area**.**pack**(**padx**=**20**,** pady**=**5**)**

46

47 self**.**send\_button **=** tkinter**.**Button**(**self**.**win**,** text**=**"Send"**,** command**=**self**.**write**)**

48 self**.**send\_button**.**config**(**font**=(**"Arial"**,** 12**))**

49 self**.**send\_button**.**pack**(**padx**=**20**,** pady**=**5**)**

50

51 self**.**gui\_done **=** **True**

52 self**.**win**.**protocol**(**"WM\_DELETE\_WINDOW"**,** self**.**stop**)**

53 self**.**win**.**mainloop**()**

54

55 **def** write**(**self**):**

56 message **=** f"{self**.**nickname **+** self**.**stuId}:{self**.**input\_area**.**get**(**'1.0'**,** 'end'**)**}"

57 self**.**sock**.**send**(**message**.**encode**())**

58 self**.**input\_area**.**delete**(**'1.0'**,** 'end'**)**

59 #disconnects client from server and returns values needed to create report

60 **def** stop**(**self**):**

61 self**.**running **=** **False**

62 self**.**win**.**destroy**()**

63 self**.**sock**.**close**()**

64 #sends nickname and student id of client to server

65 **def** receive**(**self**):**

66 **while** self**.**running**:**

67 **try:**

68 message **=** self**.**sock**.**recv**(**1024**).**decode**()**

69 **if** message **==** "NICK"**:**

70 self**.**sock**.**send**(**self**.**nickname**.**encode**(**"utf-8"**))**

71 **elif** message **==** "ID"**:**

72 self**.**sock**.**send**(**self**.**stuId**.**encode**(**"utf-8"**))**

73 **elif** message**==**"^^++"**:**

74 self**.**stop**()**

75 **else:**

76 **if** self**.**gui\_done**:**

77 self**.**text\_area**.**config**(**state**=**"normal"**)**

78 self**.**text\_area**.**insert**(**"end"**,** message**)**

79 self**.**text\_area**.**yview**(**"end"**)**

80 self**.**text\_area**.**config**(**state**=**"disable"**)**

81

82

83 **except:**

84 self**.**sock**.**close**()**

85 **break**

## Network.py

1 **import** socket

2 **import** select

3

4 ##### Network #######

5 # Parameters :- None

6 # Purpose :- handles data coming and going to server

7 ###########################

8 **class** **Network:**

9 **def** \_\_init\_\_**(**self**):**

10 self**.**client **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**

11 self**.**server **=** socket**.**gethostbyname**(**socket**.**gethostname**())**

12 self**.**port **=** 55633

13 self**.**addr **=** **(**self**.**server**,** self**.**port**)**

14 self**.**p **=** self**.**connect**()**

15

16 **def** getP**(**self**):**

17 **return** self**.**p

18

19 ##### connect ######

20 # Parameters :- None

21 # Return Type :- None

22 # Purpose :- connects client to server

23 ####################

24 **def** connect**(**self**):**

25 **try:**

26 self**.**client**.**connect**(**self**.**addr**)**

27 **return** self**.**client**.**recv**(**16384**)**

28 **except** socket**.**error **as** e**:**

29 **print(**e**)**

30

31 ##### rec ######

32 # Parameters :- None

33 # Return Type :- None

34 # Purpose :- used to handle incoming data

35 ####################

36 **def** rec**(**self**):**

37 timeout **=** 0.1

38 ready\_sockets**,** \_**,** \_ **=** select**.**select**(**

39 **[**self**.**client**],** **[],** **[],** timeout**)**

40 **if** ready\_sockets**:**

41 **return** self**.**client**.**recv**(**16384**).**decode**(**"utf-8"**)**

42 **else:**

43 **return** **None**

44

45

46

47 ##### send ######

48 # Parameters :- data:byte

49 # Return Type :- None

50 # Purpose :- sends data to server

51 ####################

52 **def** send**(**self**,** data**):**

53 **try:**

54 self**.**client**.**send**(**data**)**

55 **return** self**.**client**.**recv**(**16384**)**

56 **except** socket**.**error **as** e**:**

57 **print(**e**)**

58

59 ##### disconnect ######

60 # Parameters :- None

61 # Return Type :- None

62 # Purpose :- disconnects client from server

63 ####################

64 **def** disconnect**(**self**):**

65 f **=** **open(**"disconnect.txt"**,** "a+"**)**

66 f**.**seek**(**0**)**

67 data **=** f**.**read**(**100**)**

68 **if** **len(**data**)** **>** 0**:**

69 f**.**write**(**"\n"**)**

70 f**.**write**(str(**self**.**client**))**

71 self**.**client**.**close**()**

72 **def** disconnect2**(**self**):**

73 self**.**send**(**"OUT"**.**encode**())**

## Server-Teacher.py

1 **import** socket

2 **import** threading

3 **from** \_thread **import** **\***

4 **import** pickle

5 **import** sys

6

7 server **=** socket**.**gethostbyname**(**socket**.**gethostname**())**

8 port **=** 55633

9 clients **=** **set()**

10clients\_lock **=** threading**.**Lock**()**

11s **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**

12s.bind**((**server**,** port**))**

13s.listen**(**1**)**

14print**(**"Waiting for a connection, Server Started"**)**

15d **=** **{**1**:** 'dont redraw'**}**

16

17

18##### threaded\_clients ######

19# Parameters :- conn:object, client:set, x:int

20# Return Type :- None

21# Purpose :- handles incoming data for each client

22###########################

23def threaded\_client**(**conn**,** clients**,** x**,**server**):**

24 conn**.**send**(**pickle**.**dumps**(**d**))**

25

26 **while** **True:**

27 **try:**

28 data **=** conn**.**recv**(**16384**)**

29 **except** **ConnectionAbortedError:**

30 **break**

31 **if** **str(**data**.**decode**)==**"OUT"**:**

32 server**.**close**()**

33 sys**.exit()**

34 **else:**

35 **with** clients\_lock**:**

36 r **=** **open(**"disconnect.txt"**,** "r"**)**

37 dis **=** r**.**readlines**()**

38 **for** line **in** dis**:**

39 **if** line **in** **str(**clients**):**

40 **print(**"Disconnected to:"**,** **eval(**line**))**

41 clients**.**discard**(eval(**line**))**

42 **open(**'disconnect.txt'**,** 'w'**).**close**()**

43 # checks if the client is the top client

44 # then sends its incoming data to every other client

45 **if** x **==** 1**:**

46 **if** data **is** **None:**

47 clients**.**clear**()**

48 conn**.**close**()**

49 **break**

50 **else:**

51 **if** **len(**clients**)** **>** 1**:**

52 **for** c **in** clients**:**

53 **try:**

54 c**.**sendall**(**data**)**

55 **except** **OSError:**

56 **break**

57

58

59 **else:**

60 **continue**

61 **print(**"Disconnected"**)**

62 conn**.**close**()**

63

64

65x **=** 0

66while **True:**

67 conn**,** addr **=** s**.**accept**()**

68 f **=** **open(**"disconnect.txt"**,** "w"**)**

69 **with** clients\_lock**:**

70 clients**.**add**(**conn**)**

71 **print(**"Connected to:"**,** addr**)**

72 x **=** x **+** 1

73 # starts new thread for new client that join server

74 start\_new\_thread**(**threaded\_client**,** **(**conn**,** clients**,** x**,**server**))**

75

## Report.py

0

1 **from** email**.**mime**.**base **import** MIMEBase

2 **from** email **import** encoders

3 **import** xlsxwriter

4 **import** sqlite3

5 **import** smtplib**,** ssl

6 **from** email**.**mime**.**text **import** **\***

7 **from** email**.**mime**.**multipart **import** MIMEMultipart

8 **import** os

9 **import** csv

10

11

12 **def** Reported**(**time**):**

13 db**=**sqlite3**.**connect**(**"Account.db"**)**

14 cursor**=**db**.**cursor**()**

15 MasterCode**=**time**[**0**]**

16 accId**=**time**[**1**]**

17 sql **=** "SELECT ClientIP FROM Code WHERE MasterCode=?"

18 cursor**.**execute**(**sql**,** **(**MasterCode**,))**

19 stu **=** cursor**.**fetchall**()**

20 stu **=** **list(dict.**fromkeys**(**stu**))**

21 # creates file reports for end of session

22 CSV\_Report**(**MasterCode**,** time**[**3**],** time**[**2**])**

23 # inserts files into database

24 sql **=** " INSERT INTO Files (File,File2, AccountID) VALUES(?,?,?)"

25 path **=** os**.**path**.**abspath**(**"{0}.csv"**.format(**MasterCode**))**

26 path2 **=** os**.**path**.**abspath**(**"{0}.xlsx"**.format(**MasterCode**))**

27 insert **=** **(**path**,**path2**,** accId**)**

28 cursor**.**execute**(**sql**,** insert**)**

29 db**.**commit**()**

30 sql **=** "SELECT FilesID FROM Files"

31 cursor**.**execute**(**sql**)**

32 ids **=** cursor**.**fetchall**()**

33 sorting **=** **[]**

34 **for** i **in** ids**:**

35 **if** **list(**i**)[**0**]** **is** **None:**

36 sorting**.**append**(**0**)**

37 **else:**

38 sorting**.**append**(list(**i**)[**0**])**

39 sorting **=** **sorted(**sorting**)**

40 last **=** sorting**[-**1**]**

41 # sends files to email

42 sql **=** "SELECT Email FROM Account WHERE AccountID=?"

43 cursor**.**execute**(**sql**,** **(**accId**,))**

44 msg **=** MIMEMultipart**()**

45 MESSAGE\_BODY **=** "Hello \n Find the report for your session on Optics Experimentation Environment attached"

46 body\_part **=** MIMEText**(**MESSAGE\_BODY**,** 'plain'**)**

47 msg**[**'Subject'**]** **=** "Session File"

48 msg**[**'From'**]** **=** "christmasshopmail@gmail.com"

49 msg**[**'To'**]** **=** cursor**.**fetchall**()[**0**][**0**]**

50 # Add body to email

51 msg**.**attach**(**body\_part**)**

52 # Attach the file with filename to the email

53 part **=** MIMEBase**(**'application'**,** "octet-stream"**)**

54 part**.**set\_payload**(open(**"{0}.csv"**.format(**MasterCode**),** "rb"**).**read**())**

55 encoders**.**encode\_base64**(**part**)**

56 part**.**add\_header**(**'Content-Disposition'**,** 'attachment; filename="{0}.csv"'**.format(**MasterCode**))**

57 msg**.**attach**(**part**)**

58

59 part2 **=** MIMEBase**(**'application'**,** "octet-stream"**)**

60 part2**.**set\_payload**(open(**"{0}.csv"**.format(**MasterCode**),** "rb"**).**read**())**

61 encoders**.**encode\_base64**(**part2**)**

62 part2**.**add\_header**(**'Content-Disposition'**,** 'attachment; filename="{0}.xlsx"'**.format(**MasterCode**))**

63 msg**.**attach**(**part2**)**

64

65 smtp\_obj **=** smtplib**.**SMTP\_SSL**(**"smtp.gmail.com"**,** 465**)**

66 # Login to the server

67 smtp\_obj**.**login**(**"christmasshopmail@gmail.com"**,** "testpassword123+"**)**

68 # Convert the message to a string and send it

69 smtp\_obj**.**sendmail**(**msg**[**'From'**],** msg**[**'To'**],** msg**.**as\_string**())**

70 smtp\_obj**.quit()**

71 # adds session info into databse

72 x **=** **len(**stu**)** **-** 1

73 sql **=** "INSERT INTO Session(ClassID,Files,Duration,StudentsPresent) VALUES(?,?,?,?)"

74 cursor**.**execute**(**sql**,** **(**MasterCode**.**partition**(**"-"**)[**0**],** last**,** time**[**0**],** x**))**

75 db**.**commit**()**

76 sql **=** "DELETE FROM Code WHERE MasterCode=?"

77 cursor**.**execute**(**sql**,** **(**MasterCode**,))**

78 db**.**commit**()**

79 ##### CSV\_Report #######

80 # Parameters :- code:string, answered:list, question:list

81 # Purpose :- creates files for session one file that displays a bar chart and another that displays

82 # a more in depth analysis of each student and identify who may need help or more of a challenge

83 ###########################

84 **def** CSV\_Report**(**code**,** answered**,** question**):**

85 header **=** **[**"StudentID"**,** "Name"**]**

86 f **=** **open(**"{0}.csv"**.format(**code**),** 'w'**)**

87 data **=** **[]**

88 **for** qs **in** question**:**

89 data**.**append**(**0**)**

90 **for** i **in** answered**.**values**():**

91 x **=** 0

92 **for** p **in** i**:**

93 data**[**x**]** **=** data**[**x**]** **+** **int(**p**)**

94 x **+=** 1

95 **for** d **in** **range(**0**,** **len(**data**)):**

96 data**[**d**]** **=** data**[**d**]** **/** **len(**answered**.**keys**())** **\*** 100

97 f**.**close**()**

98 # creates bar chart for average correct against questions set

99 workbook **=** xlsxwriter**.**Workbook**(**"{0}.xlsx"**.format(**code**))**

100 worksheet **=** workbook**.**add\_worksheet**()**

101 chart **=** workbook**.**add\_chart**({**'type'**:** 'bar'**})**

102 worksheet**.**write**(**"A1"**,** "Question Average"**)**

103 worksheet**.**write\_column**(**"A2"**,** data**)**

104 chart**.**add\_series**({**'name'**:** '=Sheet1!$A$1'**,** 'values'**:** '=Sheet1!$A$2:$A${}'**.format(len(**data**)** **+** 1**)})**

105 chart**.**set\_title**({**'name'**:** 'Class Average Per Question'**})**

106 chart**.**set\_x\_axis**({**'name'**:** 'Percentage(%)'**})**

107 chart**.**set\_y\_axis**({**'name'**:** 'Question No.'**})**

108 chart**.**set\_style**(**13**)**

109 worksheet**.**insert\_chart**(**'B1'**,** chart**,** **{**'x\_offset'**:** 25**,** 'y\_offset'**:** 10**})**

110 workbook**.**close**()**

111 f **=** **open(**"{0}.csv"**.format(**code**),** 'a'**)**

112 writer **=** csv**.**writer**(**f**,** delimiter**=**','**)**

113 header **=** header **+** question

114 writer**.**writerow**(**i **for** i **in** header**)**

115 ID **=** answered**.**keys**()**

116 rankings **=** **{}**

117 # gets percentage of correct answers for each student

118 **for** student **in** ID**:**

119 total **=** 0

120 **for** i **in** answered**[**student**]:**

121 total **=** total **+** **int(**i**)**

122 total **=** total **/** **len(**question**)** **\*** 100

123 rankings**[**student**]** **=** total

124 # updates students progress for that class in database

125 **with** sqlite3**.**connect**(**"Account.db"**)** **as** db**:**

126 cursor **=** db**.**cursor**()**

127 classId **=** code**.**split**(**"-"**)[**0**]**

128 **for** student **in** ID**:**

129 details **=** **[]**

130 sql **=** "SELECT Firstname, Surname FROM Students WHERE StudentsID=?"

131 cursor**.**execute**(**sql**,** **(**student**,))**

132 name**=**cursor**.**fetchone**()**

133 details**.**append**(**student**)**

134 details**.**append**(**"{} {}"**.format(**name**[**0**],** name**[**1**]))**

135 **for** i **in** **(**""**.**join**(repr(**e**)** **for** e **in** answered**[**student**])):**

136 details**.**append**(**i**)**

137 writer**.**writerow**(**i **for** i **in** details**)**

138 sql **=** "UPDATE Class SET Progress=? WHERE ClassID=? AND StudentsID=?"

139 cursor**.**execute**(**sql**,** **(**rankings**[**student**],** classId**,** student**))**

140

141 top **=** **[]**

142 **help** **=** **[]**

143 **for** i **in** ID**:**

144 # identifies those who need more a challenge

145 **if** rankings**[**i**]** **>=** 95**:**

146 top**.**append**(**i**)**

147 # identifies those who need help

148 **elif** rankings**[**i**]** **<=** 60**:**

149 **help.**append**(**i**)**

150 writer**.**writerow**(**i **for** i **in** **[**"StudentID of Those That Need a Challenge"**])**

151 writer**.**writerow**(**i **for** i **in** top**)**

152 writer**.**writerow**(**i **for** i **in** **[**"StudentID of Those That Need Help"**])**

153 writer**.**writerow**(**i **for** i **in** **help)**

154 f**.**close**()**

## Unique\_Code.py

1 **from** datetime **import** datetime

2

3 # creates new code using current time and class id

4 **def** uni\_code**(**cID**):**

5 code **=** ""

6 now **=** datetime**.**now**().**strftime**(**'%Y%m%d%H%M%S'**)**

7 **for** i **in** **range(**7**):**

8 no **=** **int(**now**[**0**])** **+** **int(**now**[**1**])**

9 code **=** code **+** **str(**no**)**

10 now **=** now**[**2**:]**

11 code **=** **str(**cID**)** **+** "-" **+** code

12 **return** code

## Algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| Group | Skill | Description | Line/Page |
| A | Recursive Algorithm | Used to recursively go through propargation of one source while interacting with object | Line 380  Pg 75  Line 683  Pg81 |
| A | Complex scientific model | Used to calculate the result of an interaction between a light ray and an object using scientific models (e.g angle of refraction, critical angle and total internal refraction) | Line 380  Pg 75  Line 683  Pg81 |
| A | Complex mathemathical model | Used to translate from or translate into the scientific results and interpretating it into or from a geometric plane using several different mathemathical eqaution | Line 380  Pg 75  Line 683  Pg81 |
| A | Object generation.  Inheritance, polymorphism and composition | Creates objects.  In OSD.py inheritance and polymorphism takes place Main\_Object is the parent class and all other manipulatable object classes inherit from it, however objects may have different dimensions or colours or may need additional attributes or override existing methods to match object, where polymorphism is used.  In Accounts.py composition is used in classes to gain access to another class and its methods. | Pg68-114 |
| A | Several interlinked tables | foreign keys (and sometimes composite keys composed of foriegn) used to connect tables together | Pg122-139 |
| A | Complex client-server model | Once connected to server only the first client (Top Client/Teacher) to connect can send data over the server and all other clients (Low Clients/Students) can only receive and not send. Teachers are not the server here as the sending and receiving data will interfere with the loop. To allow for multiple clients to connect to server threading is used, every time a client connects an instance of the threading function is initiated.  Also used when handling data from answering questions and sets timer for each question | Pg154-163 |
| B | Dictionaries | Stores the resulting rays from propargation of a generated ray.  Used to transfer class objects over socket | Line 29-30 Pg 68  Line 20 Pg 139  Line 21 Pg 154  Line 116 Pg 165 |
| B | Mathemathical algorithm | Vernier cipher used to encrypt passwords going into the database and encrypt passwords when they are needed to verify. | Line 721 Pg 135 |
| B | Writing and reading from files | stores objects loaded in files for reloading | Line 721 Pg 135  Line 2712 Pg 120  Line 2734 Pg 120  Line 243 Pg 126  Line 220 Pg 158  Line 224 Pg 158  Line 65 Pg 162  Line 36 Pg 163  Line 42 Pg 163  Line 68 Pg 163 |
| B | Cross-table paramaterised SQL | Used to gather information from interlinked tables, using queries containing join. | Pg122-139 |

# Testing

### Test Strategy

I need to test my program so that I can be sure that all of the systems in my project work as expected. Any that don’t need to be fixed so that they produce the expected outcome. To test my project, I will create a test table to organise all of my tests and make sure that all are completed. To prove that I have tested my project I will use video evidencing, with timestamps marking where each test is done in the video noted in a column of my table.

To test my account algorithm, I will test each field with normal, erroneous and boundary data to make sure that only vaild fields can allow you progress through menu or add to database. Then I will test the student access of students by entering a valid and invalid code to test that a valid code is needed and an invalid student id.

I can then test my database by making sure the user has been created as well as the linked profile. This will prove that my database exists and that data is being saved to it correctly. To test my password reset system I can test a valid and invalid email address, and then test inputting a valid and invalid password to prove that accounts are secure.

To test that my when a new student or account is created the algorithm only accepts valid email addresses, I will be using normal, boundary and erroneous data. To test that my emails are sent when an envoirnment is created I will set up fake email addresses and add them as students to a class and run a session with that class and investigate whether an email of the code has been sent or not.

To test my simulation algorithm, I will be investigating refraction, reflection, total internal refraction on the following objects block, glassblock and semi-circular glass block at various different angles, and postion of entry, on other blocks like mirror and screen I will be testing the reflection and the stopping of the source respectively at different angles. I will also test the show gui and change gui algorithm simultaneously, in the inputs I will input valid and erroneous data to see how the algorithm processes the data.

To test my diffraction mode, I will try to load prohibited objects to see if they will be displayed or not, I will also test that the change gui algorithm for the diffraction mode and see if they change in real time.

I will test the question gui algoritm by connecting students to the envoirnment then use the algortihnm to ask questions I will input various different valid and erroneous answer to see how the algorithm will react, as part of this testing I will also test the report algorthim after the session ends a report should be sent the host email, I will check the email and investigate the report and check whether it has inputted and plotted the right data.

I will perform a test that run through the entire system to prove that it works as a whole. The two path I will follow is as follows:

I will first sign-up as a new user than login into the account I will then add new class and add a mixture of existing and new students (filling out each of their information), once all students added I will run a session. Once the envoirnment is loaded I will open the chatbox and add objects to screen, then using change gui change the property of some objects and investigate refraction to show the angles or refractive index using the show gui. Then using the question gui I will use one of the automatically generated questions to set a question and set a time, then I will wait until the time is done before moving on, after I will repeat same process for total internal refraction and diffraction but showing the appropriate properties of each experiment, I will then create my own custom question. Once the session is done I will check the email to see if the report has been sent and examine it then, in my account main menu check class progress, then edit class by deleting a student.

Parallel to what is happening, in the perspective of a student I will be waiting for the email that holds the unique code that I can use to enter the envoirnment, once I have gotten the code I will enter the envoirnment through student access and enter my student id and code received, once entered I will open the chatbox, what every objects the teacher loads within the envoirnment should display on the students envoirnment but has no ability to move the object. When a question is asked, I will answer it. Then when the session is over I will re enter the student access menu and enter the offline access.

### Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Purpose | Test data | Expected Outcome | Outcome | Changes Required | Timestamp |
| To test that invalid usernames (ones already taken), prevent account creation | Username: JamillaK  Firstname: Jamilla  Lastname: Kone  Email: jamillakone14@gmail.com  Password: topgun  Re-enter Password: topgun | Pop up window will indicate that an error has occurred that username already taken | A pop-up window was displayed indicating that an error has occurred that username already taken | None required | 1:14 |
| To test that invalid password, prevent account creation | Username: JamillaKone  Firstname: Sarah  Lastname: Jones  Email: jamillakone14@gmail.com  Password: topgun  Re-enter Password: top | Pop up window will indicate that an error has occurred that passwords don’t match | A pop-up window was displayed indicating that an error has occurred that passwords don’t match | None required | 2:16 |
| To test that invalid password prevents account access | Username: JamillaK  Password: topg | Pop up window will indicate that an error has occurred that password or username incorrect | A pop-up window displayed indicating that an error has occurred that password or username incorrect | None required | 3:48 |
| To test that invalid username prevents account access | Username: Jamill  Password: topgun | Pop up window will indicate that an error has occurred that password or username incorrect | A pop-up window displayed indicating that an error has occurred that password or username incorrect | None  Required | 4:11 |
| To test that when account created the account is added to the database | Check database after inputs entered correctly:  Username: JamillaKone  Firstname: Sarah  Lastname: Jones  Email: jamillakone14@gmail.com  Password: topgun  Re-enter Password: topgun | Sent back to first window | User was sent back to first window | None required | 3:34 |
| To test that when choosing class to run session with invalid class id prevents continuation | ClassID:100000 | Pop up window will indicate that an error has occurred that class id is invalid | A pop-up window displayed a message that an error has occurred that email format is invalid | None required | 7:00 |
| To test that when choosing class to run session with email of unique code emailed to everyone in that class | Check email of one of the students is class, and enter it into the student access window with student id | Email is sent with code | Email was sent with code | None required | 20:00 |
| To test that when choosing to edit class, invalid class id prevents continuation | ClassID=10000 | Pop up window will indicate that an error has occurred that invalid class id was entered | Pop up window is displayed indicating that an error has occurred that invalid class id was entered | None required | 19:00 |
| To test that when choosing to delete student from class invalid student id prevents continuation | StudentID:1000 | Pop up window will indicate that an error has occurred that invalid student id was entered | Pop up window is displayed indicating that an error has occurred that invalid student id was entered | None required | 19:10 |
| To test that when adding class, if adding existing student that an invalid student id prevents continuation | StudentID:10000 | Pop up window will indicate that an error has occurred that invalid student id was entered | Pop up window is displayed indicating that an error has occurred that invalid student id was entered | None required | 19:23 |
| To test that no account is required to enter offline mode | Go through student access and click the offline access button | through student access clicking the offline access will take you to a offline envoirnment | through student access clicking the offline access takes you to a offline envoirnment | None required | 8:25 |
| To test that the old password can no longer be used on login. | Within an account main menu select to edit password (same algorithm used to check whether password matches tested previously) and logout and try to relogin using the old password | Pop up window will indicate that an error has occurred that password or username incorrect | A pop-up window displayed indicating that an error has occurred that password or username incorrect | None required | 18:41 |
| Test that environment can load and move every object available | Selecting the button that say any of the moveable objects click them and | When clicked object will be loaded and can be moved | When clicked object isloaded and can be moved | None required | 8.25 |
| Test that refraction occurs in blocks, glass blocks and semi circles |  | when source at an angle and interacts with an object (block, glassblock,semi-circle) refraction will occur | when source at an angle and interacts with an object (block, glassblock,semi-circle) refraction occurs | None required | 9:52 |
| Test that at the critical angle or greater total internal refraction occurs, in the following objects, semi-circle, block and glass block |  | when source at an angle and interacts with an object (block, glassblock,semi-circle) at a certain angle total internal refraction will occur | when source at an angle and interacts with an object (block, glassblock,semi-circle) at a certain angle total internal refraction occurs | None required | 9:52 |
| Test that the source is stopped when it comes across a screen object |  | when source at an angle and interacts with screen it will stop at the point of incidence | when source at an angle and interacts with screen it stops at the point of incidence | None required | 15:40 |
| Test that sources can interact with multiple objects at a time |  | when source at an angle and interacts with an object (block, glassblock,semi-circle) and comes out of that object the source can interact with another object | when source at an angle and interacts with an object (block, glassblock,semi-circle) and comes out of that object the source interacts with another object | None required | 14:48 |
| Test that diffraction mode disable the ability to see and move previously loaded objects |  | when diffraction button pressed all other objects will disappear and young double slit experiment is displayed | when diffraction button pressed all other objects disappear and young double slit experiment is displayed | None required | 15:15 |
| Test that young’s double slit experiment is carried out on this mode and can be manipulated in real time | Slit Separation:10 | while in diffraction mode change button can be pressed to change properties that will change the experiments outcomes | while in diffraction mode change button can be pressed to change properties that will change the experiments outcomes | None required | 15:15  14:44 |
| Source rotates when red line clicked and dragged, creating various angles about the horizontal plane |  | When dragged up source will be angled upwards and when dragged down source will be angled downwards | When dragged up source angled upwards and when dragged down source angled downwards | None required | 12:00 |
| Test that when change button clicked, user able to manipulate attributes of each object | Width:800  Length:300  Radius:150 | after okay is pressed properties of the object will have changed | after okay is pressed properties of the object has changed | None required | 10:24 |
| Test that when show button clicked user able to select what they want to be able to view and it shows it in real time | Click all tickboxes  and  Click all tickboxes individually | after okay is pressed properties of the object will be displayed | after okay is pressed properties of the object is displayed | None required | 14:30 |
| Test that if host chatbox server is not running what happens when client tries to open chatbox |  | Nothing will happen | Nothing happened | None required | 12:20 |
| Test that shows that every time a code is generated it’s a unique code | Run unique\_code python file with multiple parameters | a unique code will be generated each time | a unique code is generated each time | None required | 13:00 |
| Teacher can set custom questions | Question:”How are you (1 for good or 0 for bad)”  Timeset:20  Answer:1 | question will be broadcast to everyone connected to chatbox | question was broadcast to everyone connected to chatbox | None required | 13:30 |
| Teacher can set automatically generated questions | Question:”Find the refractive Index”  Timeset:30  Enter Angle of Refraction:23  Enter Angle of Incidence:43 | question will be broadcast to everyone connected to chatbox | question was broadcast to everyone connected to chatbox | None required | 21:30 |
| Student answers the question correctly |  | On file student will have answered all correctly and be identified as needing a challenge | On file student has answered all correctly and is identified as needing a challenge | None required | 24:00 |
| Student answers the question incorrectly |  | On file student will have answered all incorrectly and be identified as needing help | On file student has answered all incorrectly and is identified as needing help | None required | 24:00 |

### Video

<https://youtu.be/DwGC4RqVRf8>

# Evaluation

## Objective Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Objective | Met? | Comment | Test Reference |
| Able to investigate refraction experiments | yes | Block, glassblock and semi-circular glass block are the objects where refraction can occur when source angle about the horizontal axis is not zero, angles of refraction and incidence can be viewed using the show gui. | Video:8:25 |
| Able to investigate reflection experiments | yes | Mirror is the only object where reflection can occur when source angle about the horizontal axis is not zero, angles of reflection can be viewed using the show gui. | Video:8:25 |
| Able to investigate Young’s Double Slit Experiment | yes | When the diffraction button is clicked it enters a mode where young’s double slit experiment can be carried out, in this mode you can’t see any previously loaded blocks (they reappear when you exit the mode), and the objects needed to carry out young’s double slit experiment are automatically loaded and can be moved along the horizonal plane, to change the elememts of the experiment change gui button can be pressed to change the elements. To show the fringe separation, slit separation, wavelength and screen distance the show gui can be clicked and corresponding boxes can be clicked | Video: 15:15  14:44 |
| Able to investigate total internal refraction | yes | Block, glassblock and semi-circular glass block are the objects where total internal refraction can occur when source angle about the horizontal axis is not zero, angles of refraction and incidence can be viewed using the show gui | Video:8:25 |
| Able to investigate theses experiments using the following object: a light source mirror, glass block, slit, semicircular glass block, screen, and stock block (a block that’s properties can be changed to investigate the behaviour of light with material of different properties) | yes | The objects can be loaded using the tool bar at the top of the screen, and properties can be changed when the change gui button is clicked | Video:8:25 |
| One light source can interact with more than one object at a time | yes | By using recursion of the simulation and inside simulation method sources can interact with multiple objects | Video:14:48 |
| Host must have the ability to change the properties of objects | yes | Using the change gui host has the ability to change properties of the objects on screen | Video:20:00 |
| The program must be networked to allow for students to access envoirnments from different computers on local network | yes | students can access from local network but not over the internet/ from different network | Video:20:00 |
| When a pupil exits the environment, the program must be able to disconnect the pupil from the server. | yes | As threading is being used clients can disconnect from server individually and are remove from the client list | Pg 150  Line: 612, 626 |
| Pupils must be able to connect to an environment with a name using a unique code generated based on the time. | yes | Before entering the session, students have to enter a unique code, student id, and nickname. | Video: 20:00 |
| Pupils must also can communicate with each other and the host using a chat box where the name of the chatter is also displayed beside text. When the chat box is opened a new window opens and connected to another server hosted by the host specifically for chat and answering questions. | yes | Using a button on their gui students can open a chatbox which automatically connects to the chatbox server (handles incoming messages, hosted by host), and displays all inputted messages, unless answering a question | Video: 21:36 |
| When pupils connect to the environment, they must not have the ability to move or change anything within the environment | yes | As the client-side uses a while loop to handle events, when the client clicks anything under the toolbar there is no statement to handle that event so it passes (does nothing), therefore objects can’t be moved. | Video: 12:10 |
| The program must allow the Host to save and reload environments they have saved | yes | Envoirnments can be saved but only within the session its running, can be improved by envoirnments being saved externally so they can be opened during different sessions | video:8:25 |
| A login system to access their account all information must be stored on a database and the passwords for each account must be encrypted and have an interface where host can add classes, remove classes, view classes, edit their own account | yes | Login system has 3 different options; login, sign up and student access.  Signup allows users to create and account they can log into  Student access allows students to connect to a specific envoirnment  Login allows user to add classes, remove classes, view classes, edit their own account. | Video:0:00 |
| The program must be set up as a Client-Server model when setting up the network, where there is a top client(teachers) and lower clients(students). | yes | As ‘top client’ always enters the server first, and all the other clients are added after, its easy to send data to the ‘lower clients’. | Pg 154-163 |
| Unique code is sent via email to every student | yes | When session starts up for a class automatically an email is sent to each member of the class | Video:20:00 |
| Objects must be rendered in 2D environment by using a 2D environment the host will be able to move the objects around on the X and Y axis using a mouse, using pygame. | yes | Using the pygame draw function, objects are drawn in a 2D envoirnment | Video:8:25 |
| Program able to add and delete objects on screen when specific button clicked | yes | When delete all button clicked removes objects from sources\_loaded and objects\_loaded, then clears the screen keeping the toolbar | Video:8:25 |
| Program can be used offline; this function serves to benefit students who want to investigate by themselves. This mode can be used without an account, or with an account | yes | Fairly simple to implement runs the program without asking for class id or connecting to the server or creating a report | Video:8:25 |
| Program must allow teachers to set question depending on the experiment that is being carried out in the environment, the program must automatically suggest question that the teacher can set depending on the elements that are being displayed, elements referring to incident angle, refracted angle, refractive index, critical angle, and elements of the young’s double slit experiment. | yes | As program checks if there are objects being displayed then determines if they have interacted with a source, then generates question dependant on what is being shown via the show gui algorithm. | Video:21:30 |
| Student must only have a set amount of time to answer the question, the amount of time is set by the host | yes | This is done by measuring the time taken from the time the question was set to the time an incoming message has come in it compares that time with the time that was set if it still within time the answer or message will be handled and if not it will not allow for the answer to be registered. | Pg 158  Line 236,244 |
| Questions must be answered in chat box but not displayed to other clients. | yes | When a question is asked and a student answers within the time set, they must use /answer or /Answer to register the answer, and to also let the algorithm know not to broadcast your message | Video: 21:40 |
| After each session the program must produce a report of the class’ progress. The program must be able to report if the answer is correct or incorrect for every student and formatted into an excel file using csv and should be downloadable for teachers. | yes | Two files will be created one of a more indepth analysis of the session containing every student result for each question and who needs help/ is struggling.  These files of the session are sent to the teachers email which then can be downloaded. | Video: 23:47 |
| File must include a graph showing the classes overall performance per question | yes | On a second file contains a bar chart of every questions class average with the questions displayed in the adjacent cells. | Video:26:00 |
| File will be sent to the host email after each session | yes | These files of the session are sent to the teachers email which then can be downloaded, an unedited version is saved on the database for access to later. | Video:23:37 |
| Details on Account, Students, Session, Files, Class and Code should be stored on a relational database | yes | Details are stored on a database with different relationships, tables can be edited by account holders only on entities that are related to them. Queries include; select, update, delete and join |  |
| Pupils can connect to the environment over the internet. As this will require more time to do if program can successfully meet all objectives, then extension can be tried to implement. | no | Students cannot connect to the environment via the internet as there was not a lot of time for me to implement this as I spent a lot of time trying to find a good way of sending objects over a socket server |  |
| The objects are rendered in a 3D environment, and can be viewed from different angle and along the X, Y and Z axis | no | Due to time restriction I want able to create a 3d environment for the program as it would require ray tracing. |  |

## Teacher User Feedback

How easy is the system to use?

“I had little problem with using it, everything I needed was laid out in a straight forward manner. to note when going through some areas of the program there was a lot of windows being made which was a bit overwhelming”

Does it meet the objectives?

“from I what I can see yes”

Any criticism?

“I think as a teacher being able to use it over the internet would be more helpful, I would also like to see more experiments on the app not just wave experiments but from other topics you are learning in physics. And make it so that users only dealing with maximum of 2 windows at a time so that we don’t lose where we are ”

## Student User Feedback

How easy is the system to use?

“It was pretty easy to use I like how everything was obvious, but I feel like it would have been cooler if there was some more colour involved I think it will help more people engage with it, it is nice that you implemented a way for students to carry out there own experiments.”

Does it meet the objectives?

“yeah it looks like they have meet most of the objectives ”

Any criticism?

“like mentioned before I think more colour and more of a more modern GUI would be better”

## Analysis of Feedback and improvements

Feedback was positive all around and I fully agree with the criticisims made, looking forward I think an improvement that can be made is defintely adding more colour to the GUI and making it look more modern. I am happy that almost everything was straightforward and easy to navigate as it was one of my fears as the project was made up of two different GUI’s (pygame and tkinter) but a problem which was addressed by my teacher as they found it overwhelming to have more than 2 windows open at a time which is an area which will be fairly easy to improve on with the use of a stack data structure to hold previous windows.

To improve the program I think adding an algorithm that saves the envoirnment and stores it in a file or database which can be accessed later also I will also like to add the ability to play around with different shapes such as a circle or a free shape (shape created by the user), it would be interesting as an extention to investigate concave and convex lens. Also to improve it would be nice to have the ability to be able to carry experiments from other areas of physics such as investigating energy, projectile motion and others would really make the project more diverse .

Another improvement coulde be to be able to displays all the files that have been created for that account and allows users to select which one they want to get a copy of and it gets sent to there email.