Praful Patel - Develop an Outcome L2 Evidence of Learning

NSN NUMBER - 0137483603

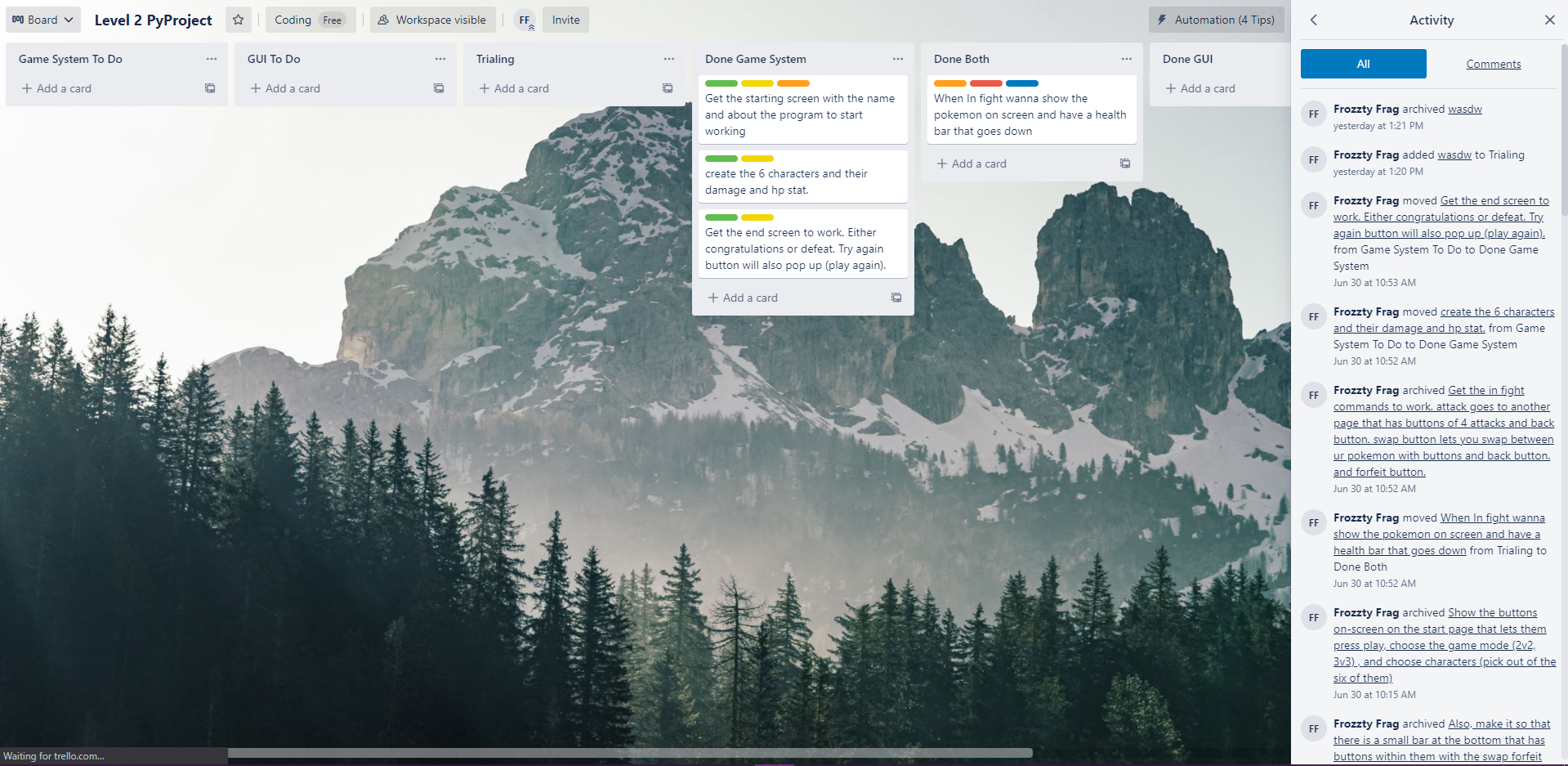
Project Title - Pokimals

[Google Drive Link- Pokimals](https://drive.google.com/file/d/1ppEQJVwlYwRQcHxTPIOcCMtIn5mYbSk2/view?usp=sharing)

[Trello Board Link - Project Management](https://trello.com/invite/b/BpMae02D/856aeefae5a2df249863bc2c76346616/level-2-pyproject)

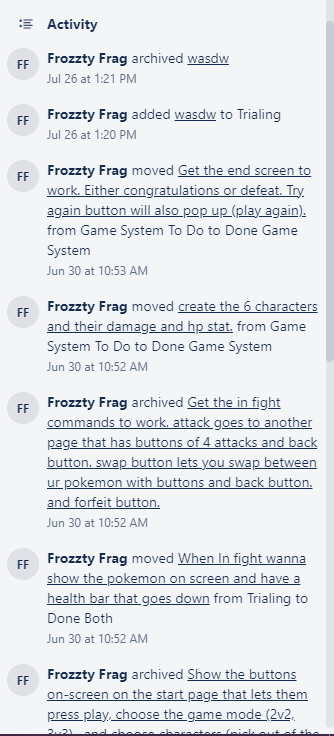
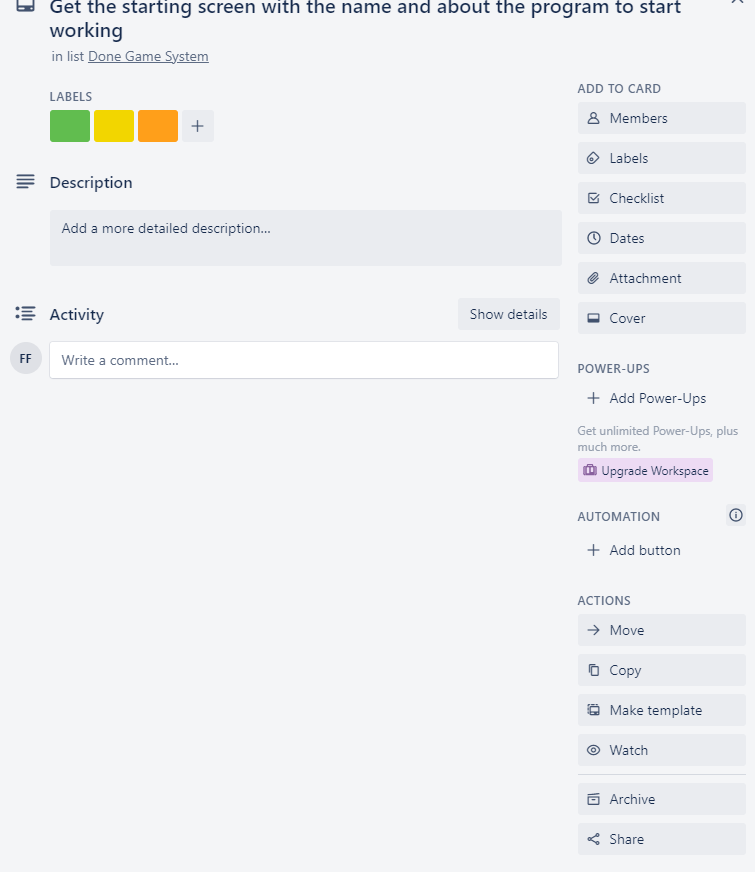
**Project Management - Setup**

Record evidence that you have set up project management tools to manage the development of your outcome.

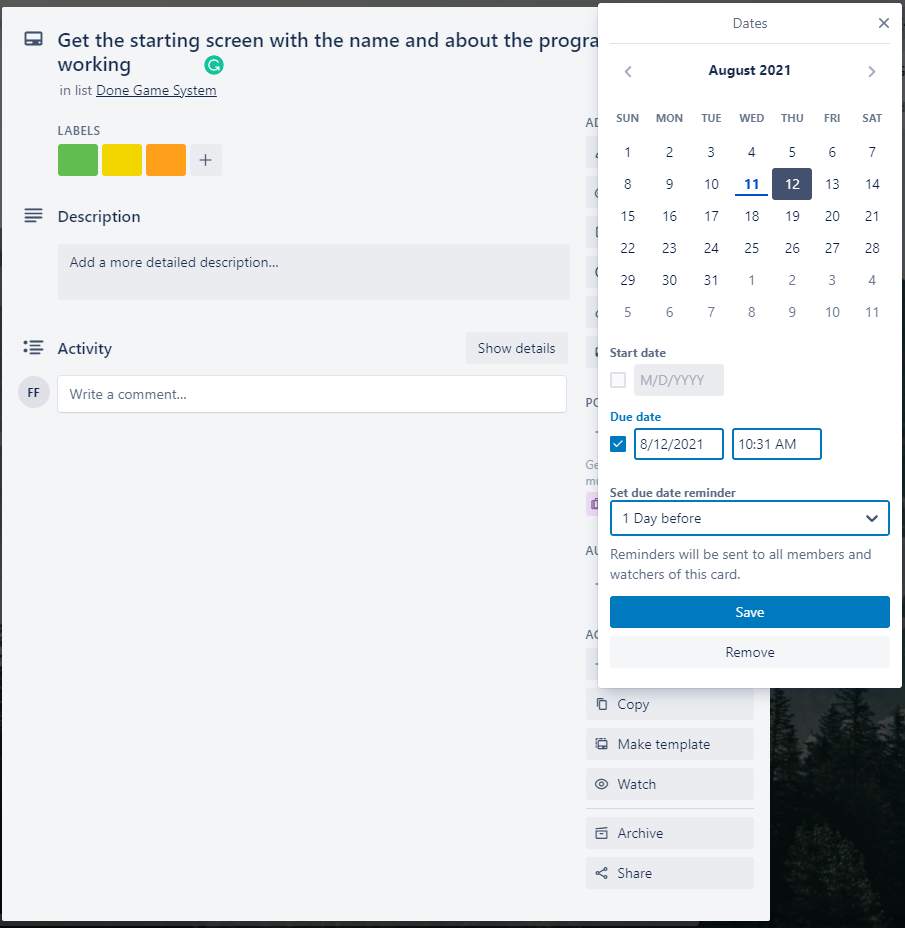
I used Trello.com as I was very familiar with the website to track my progress for my project. I added tags to each of the lists so that once it had reached that list, that task would get a coloured indicating it has gone into that list. For example, when I move any of the cards from the to-do list’s into the trialling list, it would give it a yellow mark. This is useful for me to track down what was done and if I still needed to trial it or if it was all finished and polished. It is also effortless to share and let other users collaborate. If I were to have done it with someone else, it would make it very easy for each of us to know precisely what had been done, if it needed to be trailed still or if it was fully completed to be set aside until it all needed to be put together. Trello was an excellent pick compared to other project management tools for many reasons. It was online, meaning that it was easily accessible and available on all platforms. As long as I had internet access, I could access my board from anywhere. This allowed me (and if I were to have collaborators) to add, change or remove objectives from the board and help track the project’s progress on the go without having to go to a physical calendar or board to tick off the sections. 

Record evidence that you have chosen project management techniques to manage the development of your outcome.

Trello saves the history of the Trello board, so anything changes that occur to the board are recorded, making it easier for me to track down what I have done and when I exactly made that change. I used the archive feature to get rid of the cards that were not relevant to the project. I had created the board during my initial concepts of the project in which I wanted to do a full GUI game. However, after a lot of coding, I figured out that it was very hard to do, and with a lack of extensive knowledge in python, it was not going to be finished before the due date. So I decided to archive all the cards related to the fight part of the project as I was not doing it anymore. Another fantastic feature that Trello has is the watch feature. When I struggled with a particular aspect of the code, I clicked on the card I was struggling on and selected watch. This would put an eye icon on the card, so I know that this is a part I need to work on until I have completed it and unmark the card. Another two features that also helped me a lot was the due date/reminder one. For each card, I was able to add a reminder. This would send an email to me before the due date to help me stay on track with all the smaller code sections. The reminder feature was great as I could make sure I never started slacking. The activity feature was also pretty nice to have around as it was never extra effort always to make sure each tiny thing is remembered. Instead, if I was to have forgotten something important, I could always go to the activity tab and go over what had been done to ensure I had done what I was supposed to.

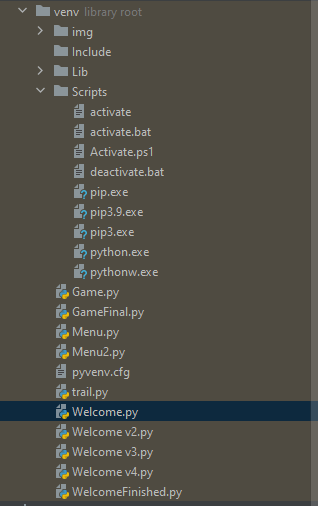






\* Showing how each feature I used can be found within Trello. The reminder comes under dates.

Record evidence that you have set up / chosen version control tools/techniques to manage the development of your outcome.

I decided to make versions of my code after every significant change to that part of my game. This allowed me to go back to a later version if I needed to. It also gave me a visible contrast between the code to look for any new errors. I did this by copying the current version and changing anything significant that I wanted, adding the version number. This helped me keep track of each iteration of code for that particular part of my project. This was also good for when I ran into bugs. Rather than going through a whole final chunk of code and trying to debug it, I knew precisely what iteration was causing issues, making it easy to narrow down the cause faster and easier for a quicker fix. 

Versions were instrumental when it came to the welcome page, where I had made many small definitions. Multiple definitions caused quite a complex array of code; thus, making versions helped fix errors between versions for a more straightforward fix. For example, the welcome page had four versions.

The first version was a basic layout of the code and achieved what I had initially wanted. However, it turned out to be very chunky, and there was too much unnecessary code.

I added a new window for an information page in the second version and cleaned up most unnecessary code.

In the third version, I got rid of some of the code as I no longer needed that part of the code due to changing the project slightly.

The fourth version of the welcome page was just a cleanup version. This is where I made sure everything was laid out well and annotated correctly to ensure that everything that worked together was together.

The last version was where I added a significant change. I added a new greeting page asking for the user’s name and greeted the user using a randomly generated greeting. I added robusticity to my code by making sure they could only enter their name and not integers. The new greeting page was added to all my other code to create a final part of my whole code. Overall, versions helped me look between iterations of code to make sure everything was working, and if anything did go wrong, it helped me narrow it down to the root cause quicker.

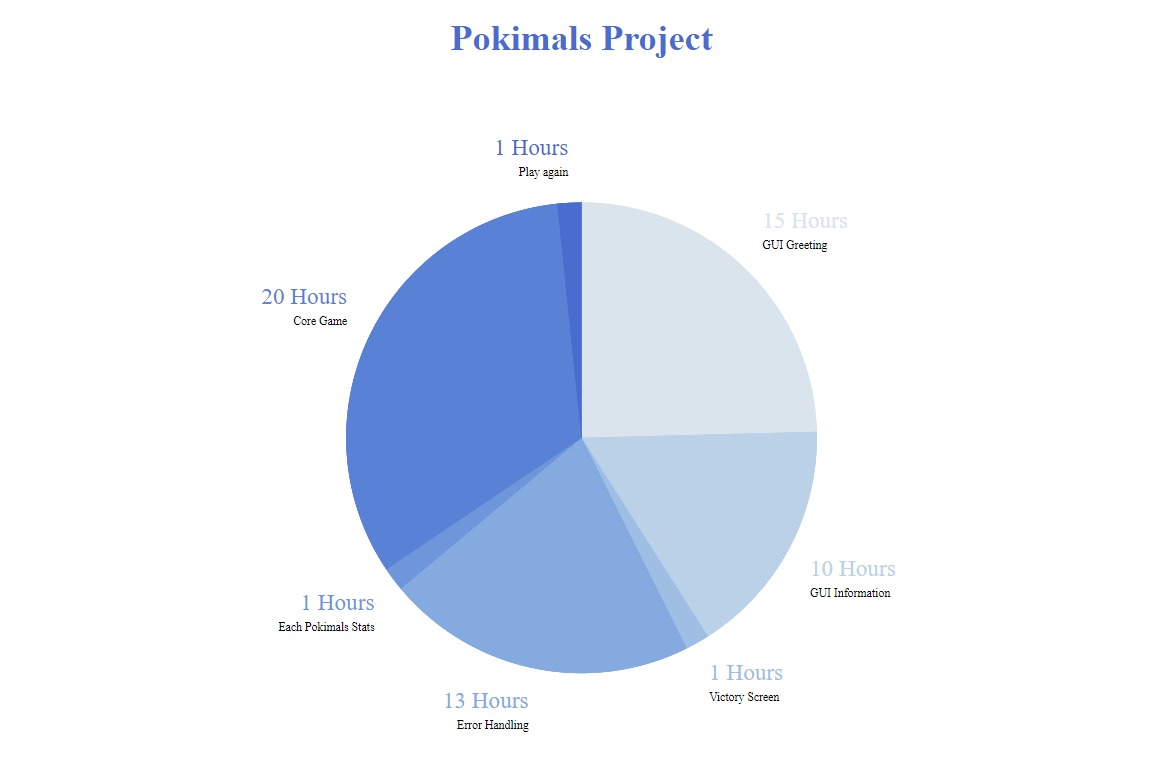
There were multiple other versions created which were for adding robustness to the code. This was done by adding restrictions that would

Explain why you believe that these tools and techniques are appropriate for the development of your outcome.

Both Trello and version control was beneficial within my code as they allowed me to keep track of my progress and made it easier for me to fix errors within the code. They allowed me to show and see my progress for each decomposed section of my project. Trello helped me stay organised and on-task, where I start by making a group for the game system, GUI sections to do and complete, and a trialling section. I made a separate to-do and finished pile for them for components of the code that landed in between GUI and the Game system. I had also added colour tags for each group to know what groups the tasks had travelled through. This allowed me to see if some had not been trailed yet or were fully completed. The version control was helpful for the trialling process and the creation of each decomposition. It allowed me to work on each decomposition more efficiently and effectively. This made it easier to track in which iteration the problem occurred for a faster fix, allowing me to build up my work in layers. Starting with a solid compound, slowly polishing and adding any necessary improvements.

**Decomposing Outcome**

Record evidence that you have broken the outcome into smaller components.

For my project, I decided to break it up into more valuable chunks for a smoother coding experience and work on each important feature one at a time. I did this by creating a pie chart consisting of each of my decomposed parts. These included: GUI Greeting, GUI Information, Core Game, Each Pokimlas stats, Error Handling, Play again, and Victory screen. I put down a rough estimate of the time I would like to spend on each component for each component.

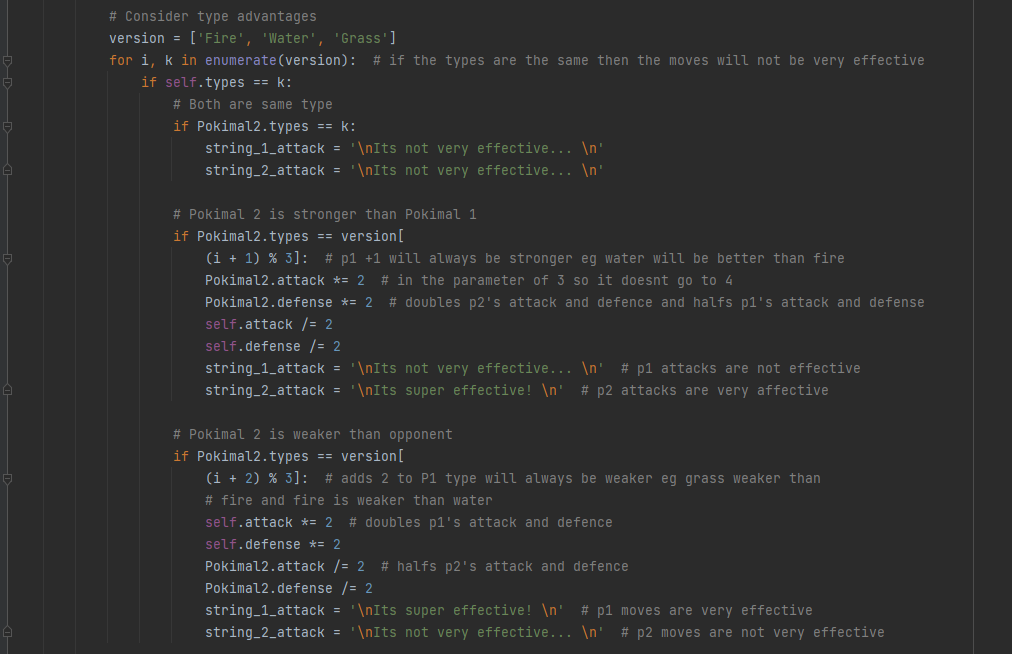
In this case, all the minor features that required tiny amounts of code were given 1 hour, whereas the core game was given the most hours as it was the most intensive and vast part of the project. Since it was my first time working with a GUI, I made sure that I had enough time to learn as I coded the components, So I gave them at least 10hrs each to learn and apply it to my project.

Decomposition was helpful when it came to making sure I am on track to completing each component and was not wasting way too much time on parts I shouldn’t be.

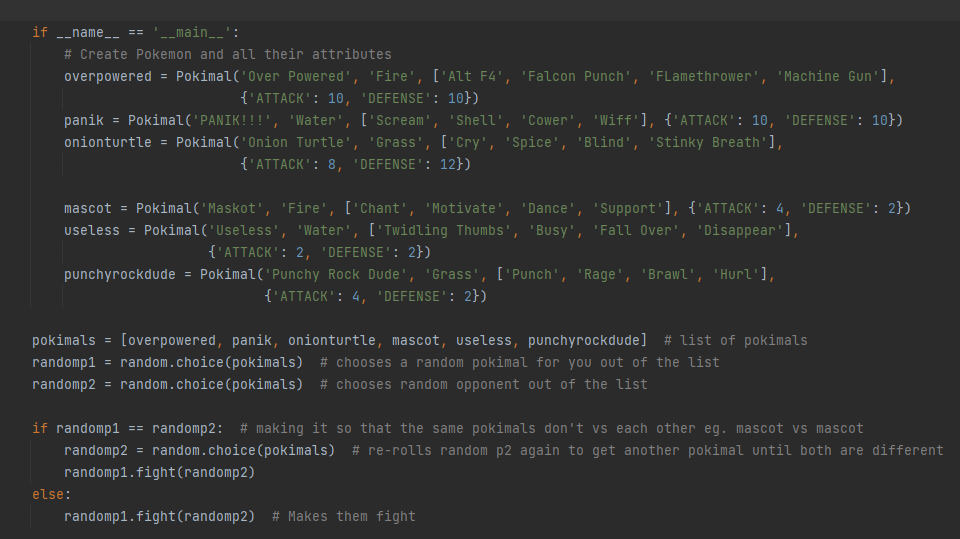
GUI Greeting was where I implemented a GUI for more user interaction. The GUI used an input box to ask for the user’s name. When entered, it then randomly greeted the user with a greeting from a selected amount of languages.

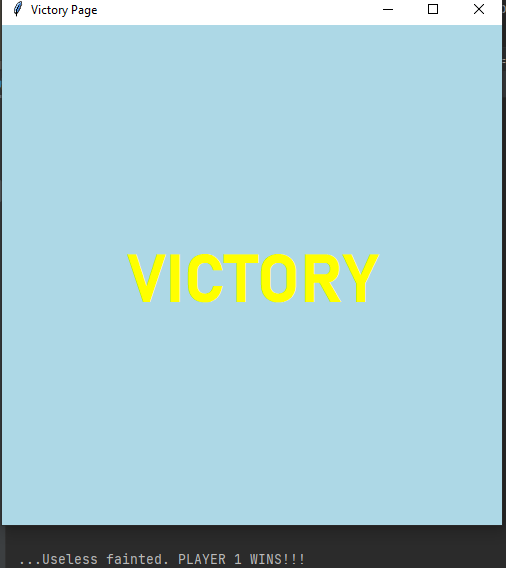
GUI Information was just where the information of what my program did was placed for the user to understand. It required minimal interaction and was very easy to understand and implement. However, since it was my first time making a GUI, it required more intensive thinking. After completion, the information GUI turned out to be very simple and effective.

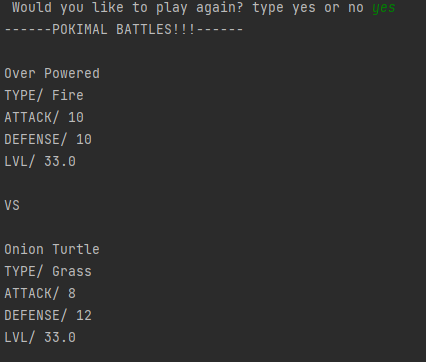
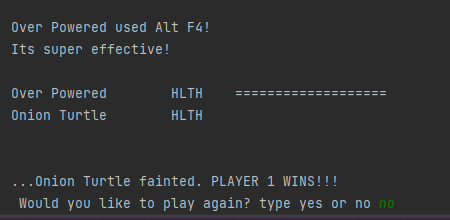
The core game consisted of the mainframe and was introduced to many mind-boggling pieces of code. Classes were the critical component of the core games’ structure. I ended up using one class that had all the features of the Pokimal, such as its attack, defence, Health, both displayed and calculated, its name and moveset. This was then used for Pokimal 1 (self) and Pokimal 2 (opponent) to make the coding very easy. If I were to add more opponents, I could add another variable called Pokimal 3. I decided to use classes over individual lists. It helps my code stay very concise and easy to alter if I add more Pokimals, different attack effects (e.g. Burning, Poison, Continuous earthquakes, Sleep, Confusion where it attacks itself.). However, if I were to use list’s, it would be tough to implement them. It would require me to change each Pokimal and add it individually rather than adding the component to the class and creating the rest of the element itself. Classes also helped me add to all other data storage methods, such as dictionaries, lists and strings.

I also implemented how effective and ineffective the moves were to the Pokimal’s based on its type. Such as fire being very effective against water and water being very weak against fire. If the Pokimal were ineffective against it, I would half that Pokimal’s attack and defence stats. However, if it were very effective against that Pokimal, Pokimal’s attack and defence would double. I did this by making a list of the versions. If the opponent was weak against the self, then it had to be the element two more in the list of 3, e.g. Grass is two higher than Fire in the list with grass being opponent (weaker) and self being fire (stronger).

For each Pokimal’s attributes, I used a list for their attack and defence stat and all Pokimals different attacks. This was better than creating a variable for each as it saved space and how much processing was being done and made it easy to alter each Pokimal’s stats efficiently. I could also add more Pokimal’s if I wanted to and not make any significant changes. This was also where I randomly selected each pokimal to battle out of the list. I did this by creating a list called Pokimals with all the Pokimals inside. Then, two more variables called randomop1 and 2 randomly selected a Pokimal out of the list to use. If the Pokimals’ were the same, then randomop2 would rechoose until each Pokimal was different. Then the fighting occurred when called using x.fight(y), in this case, randomop1fight(randomop2).

The victory screen was shortcode which just popped up a GUI screen that said Victory to whoever won. This was just a few lines of code.



The Play again features was also very short, where I asked the user if they would like to play again in which if they responded yes, it would keep play (a variable) true, which is pronounced true at the very start of the code, otherwise it would turn play into false. This stops the code from running anymore. 

Decomposition was very useful in separating all the parts of the code whilst pairing them with Trello allowed me to keep up with all the decomposed features and their given times of completion.

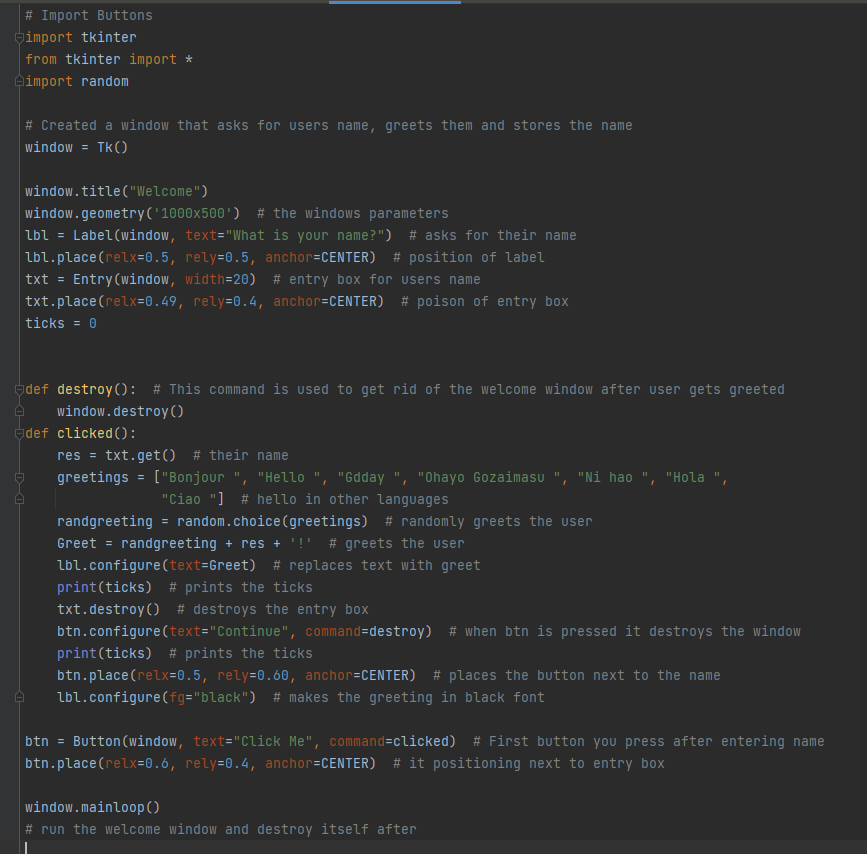
**Trialling Components**

Record evidence that you have trialled components. Show that you gathered information to inform your decision making.

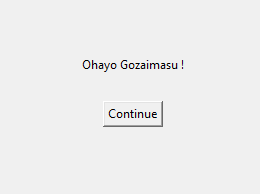
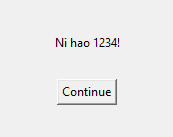
Record evidence that you have trialled multiple components and have selected those most suitable.

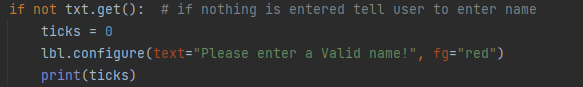
Record evidence that you have used information from trialling appropriately to improve the functionality of the outcome.

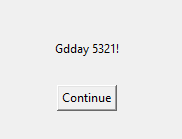
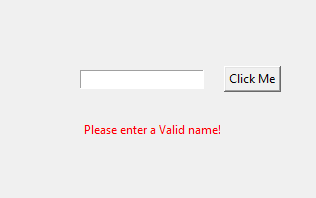
GUI Welcome - Trail 1

The debugging part was a crucial part of my project as it allowed me to also and minor bugs and change any major points where my code was not strong enough. There were countless ways I tried and tested fixing this problem; however, after researching about the text entry system of Tkinter (the GUI system that I was using), I found out that the text entered by the user was not treated the same way as strings are treated, so I ended up using the way I did below.

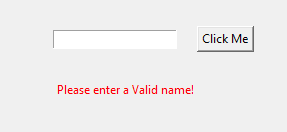
Initially, my code just asked for the user’s input and then greeting the user by taking their input and adding a greeting before it regardless if they input nothing or integers. I created a GUI window with a text entry box, some text asking for their name and a button. After the user enters their name, the text box disappears, and instead, text appears, greeting the user with a button that says continue. This button destroys the window and is also programmed to open the second window with the information. This was not robust at all and caused the code the break every time. It did what I wanted; however, it would not regard the actual input and assume the user entered a real name without checking.

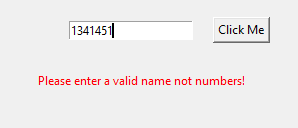
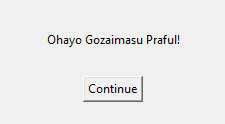
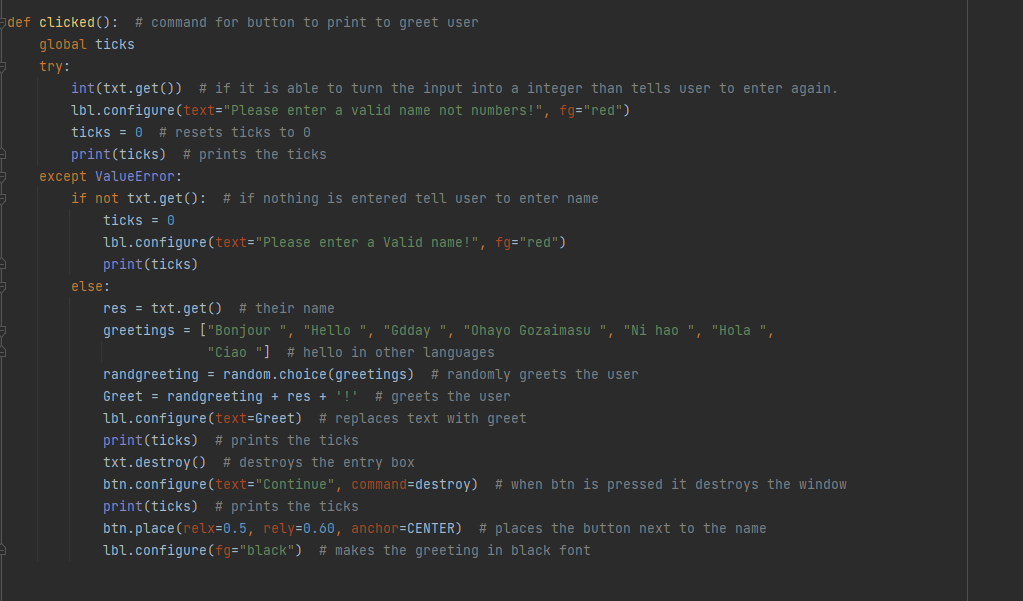


However, as part of the debugging process, I wanted to improve the rigidity of the code by making sure that the code would not break in any shape or form, which sometimes caused it to fail, but also make sure that all inputs were what I wanted. Making sure the code was checking for these user errors. So I started by making sure that the user had to enter something so. I began by asking if the enter box had been used and something had been entered. If nothing is entered, the code should stop there, and a message should pop up asking the user to input something. This occurs until the user inputs something. This worked perfectly every time, and I made sure that the text was red, indicating to the user that something was not right to make sure the user reads that they have to enter something. 



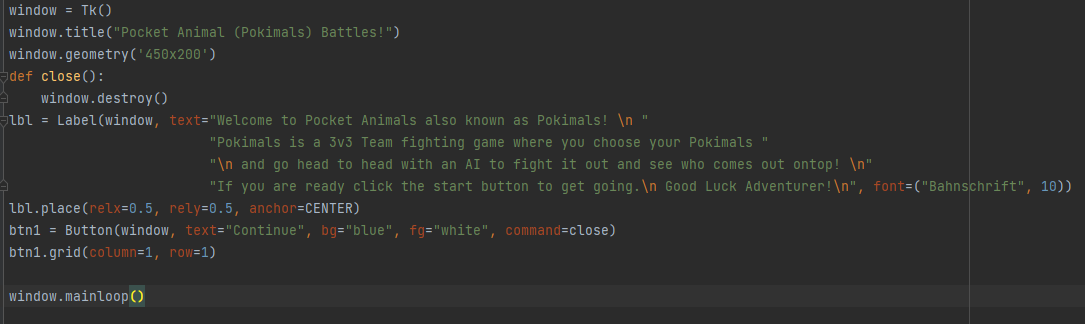
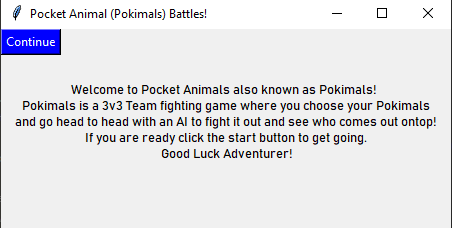
I went on researching ways to check if there are integers within the entry box. Since it was my first time working with a GUI, I used many external resources to understand and fix this error. I finally figured out how to make the code to try and see if the input was integers. If it were integers, it would let the user know that they can’t input integers and have to input a real name in red text. This occurred until the user entered a real name. And if no integers were inputted, it would first make sure that something was entered; after that, it would greet the user. I decided to do it this way because it ensured I had control over what I wanted the user to enter, limiting the amount of user error in my project allowing a smoother user experience. This was better compared to some of the other ways as it was the most understandable to me since I was new to GUI and was also one of the better ways from my research. This improved the user experience significantly as it minimised the amount of user error and gave the user more chances to interact with the project rather than watching everything occur.



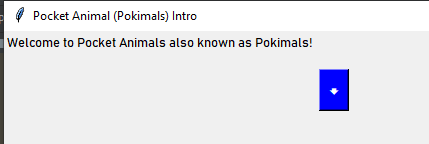
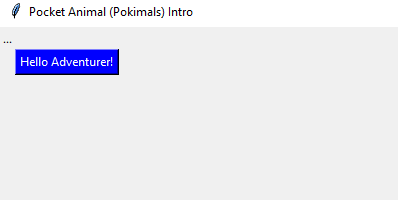


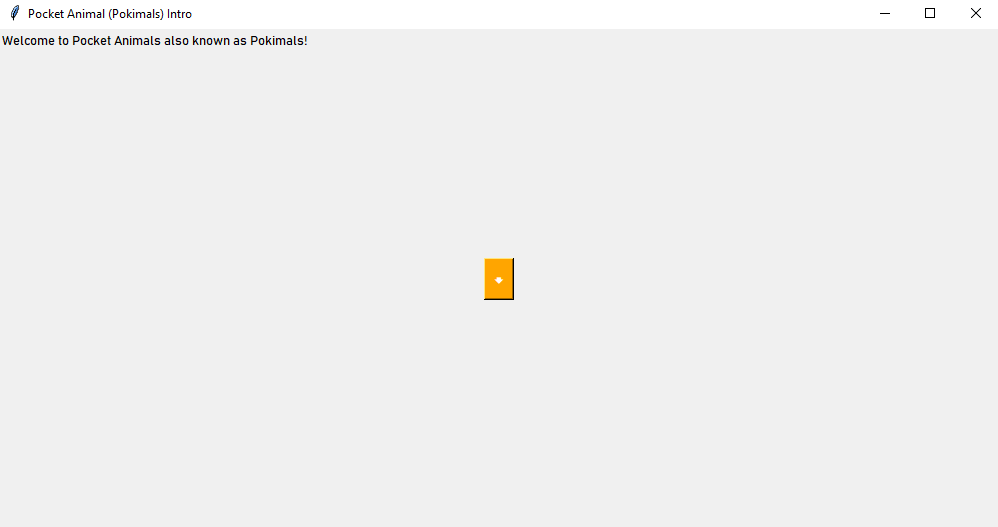
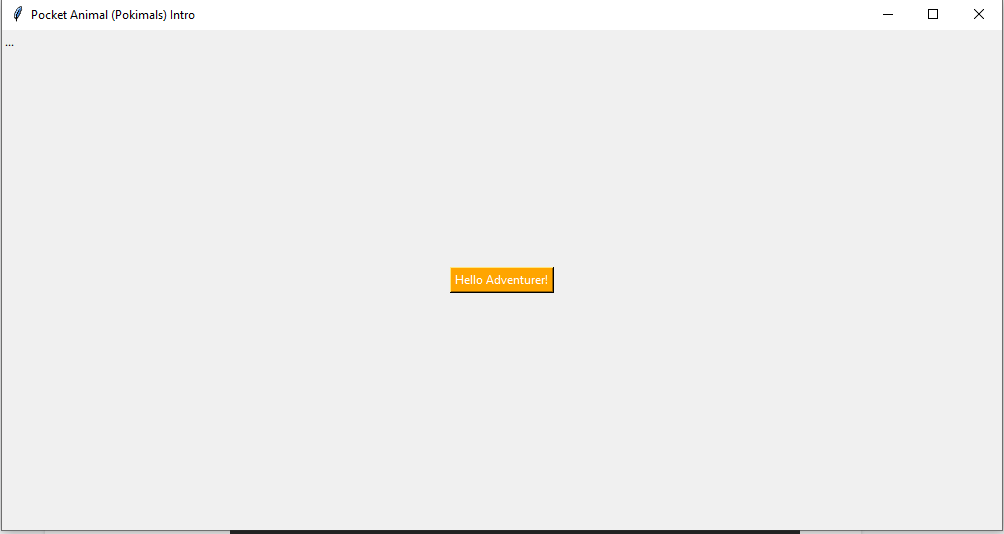
I made sure to run the finished code through the pep8 checker to make sure everything was validating and made sure once more that the code was not taking any names or integers as inputs. I did this by continuously checking and addressing the functionality and ranI had set in place, ensuring it was up to my desired standards.

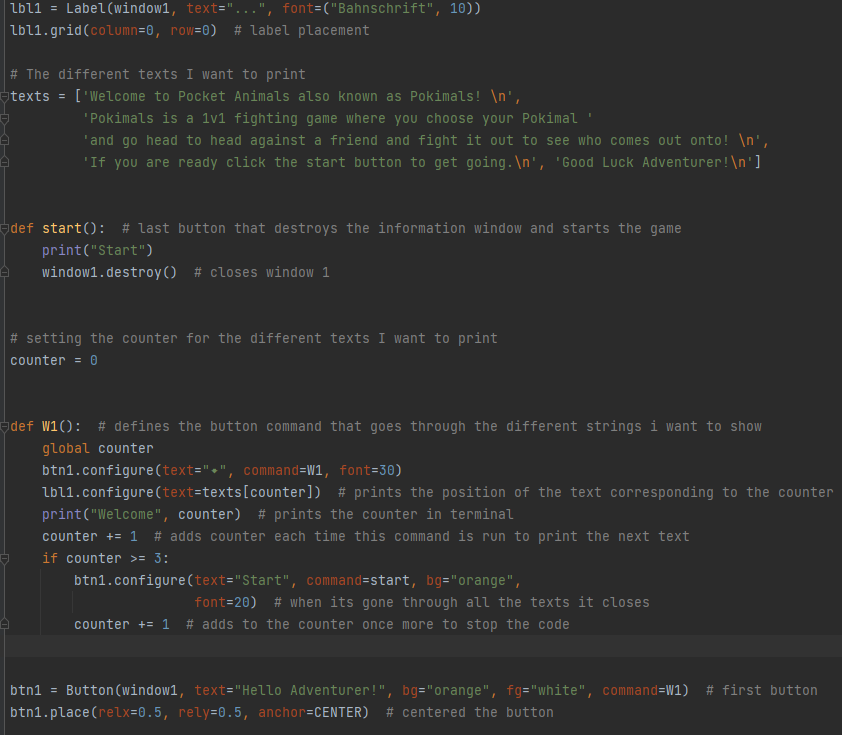
GUI Information - Trail 2

The GUI information was a great way to let the user know what the code was about and exactly what they can expect from the code. I initially started by having it so that all of the information came up at once with a continue button in the top left corner. I initially kept the button in the top left because I lacked knowledge of how to centre it.

However, initially, I wanted more user interactions and felt as if this were to be minimising it. So I decided to make a list of the text to appear bit by bit. This would be done by adding the initial label(text shown) and button. Whenever the button clicked, it added 1 to the counter, which was linked to the list of texts printing the next one out every time it was pressed. This new text replaces the previous one rather than printing a paragraph. Once all the texts were printed, the button was programmed to destroy the window like the last versions continue button worked. 

This gave more user interactions; however, it was a bad user experience due to the button position constantly changing due to the change in sentence lengths of each bit of text. I decided to research different ways I could fix the position of the buttons within the GUI. 

I also did not like the colour of the button as it was very contrasting, causing eye strain, and it would be challenging for users with colour blindness to understand what the buttons had written on them. I found a way to fix the position of the buttons to the centre by using the .place function. This function allowed me to position the buttons to a given anchor (I used centre in this case) relative to that position’s y (rely) and the x-axis (relx). This kept the control (s) always centred to the window regardless of the size.

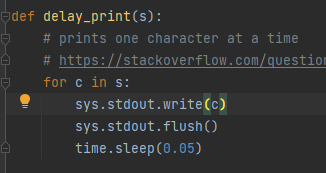
I also changed the colour of the button to a less straining colour on the eyes. This aided colour blinded users with determining what was written on the button. I decided to make these changes as it allowed for more user interactions with the code while keeping the interactions very straightforward and user friendly. This was achieved by keeping the button centred as the texts were displayed. This also allowed the user to choose when the following text is displaced. Another way to implement the text would be to replace the text after a certain amount of time. However, this would not be user friendly as each user takes a different amount of time to read the exact text. Some may finish earlier than the timer meaning that they have to wait till the text changes, and some may not finish reading the text making the timer too short for them. After looking at the pros and cons of each type of implementation, I decided to allow the user to control the text speed. This also adds more user engagement, which was essential for me as I wanted the most user engagement to happen. This is the completed code after all the altercations. 

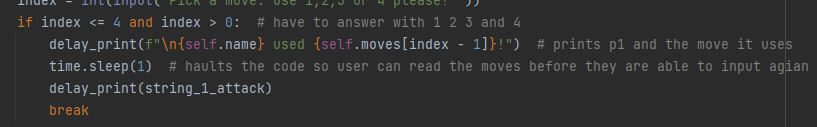
I like the last one made sure the debug and errors that would appear and ran it through the pep8 checker to make sure everything was validating and done correctly. I constantly ensured that it stayed up to my functionality implications’ standards and ran with no hiccups.

Slow Print - Trail 3

Printing the output of the code was one of the most significant sections of my code. This was because my game section of the code ran purely within the Terminal. Since the Terminal printed all the outputs of my code, I intended to make it a better experience. At first, I was using the primary print function for all of my outputs; however, I did not like the feel of my outputs. They felt too dull, and I wanted to give it a more lively experience. I decided to make some sort of text animation where the text would be printed a character at a time, giving it a typewriter feels in games such as Pokemon. It looked pretty interesting, and after seeing some examples of it, I decided to implement it.

I first imported a new python extension called time and sys (system), which allowed me to create the slow printing function. Then I made a def for this text animation which I called delay\_print(s). [I found this version on a stack overflow post.](https://stackoverflow.com/questions/9246076/how-to-print-one-character-at-a-time-on-one-line) I treated the characters (c) within this new function (s) individually and ensured that the next character isn’t accidentally printed. The code is then put into “sleep” for .50 of a second. This occurs after each character is printed off the output until everything is printed.



If something needs to be printed, you use delay\_print instead of using the ordinary print function to use the function. Here is an example of me using delay\_print() in my code.

Whenever someone uses an attack of their Pokimal, it delays prints the Pokimal name and the move they wanted to use.

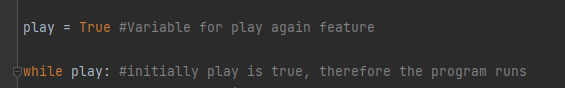
**\*Sorry, I am unable to screen the record to show it working correctly.**

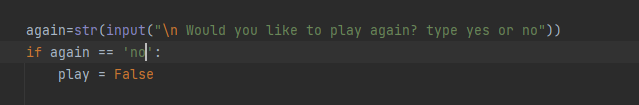
This allowed for a more exciting feel to the code. Rather than the text appearing out of nowhere, it feels like it is being typed out in front of the user. The slow printing gave a better user experience and, after peer feedback, was discovered to be a liked feature.

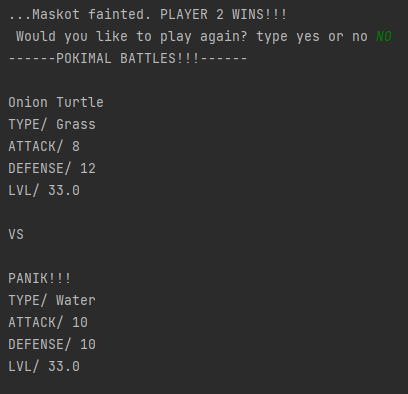
Play Again Feature - Trial 4

The play again feature was the fourth most important feature of my code as it allowed the users to carry on playing without having to restart the code every time. This feature comes up at the end of the code whenever player one or two wins without re-entering their name and going through the information.

For this to work, I had to create a variable called play. At the start of the code, it is made true since the user has started up the program to use it. The rest of the code is put into a while loop that only lets the code run while play stays true.

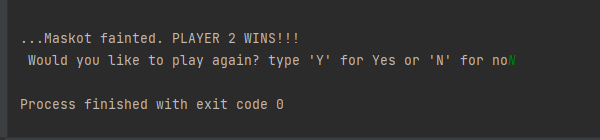
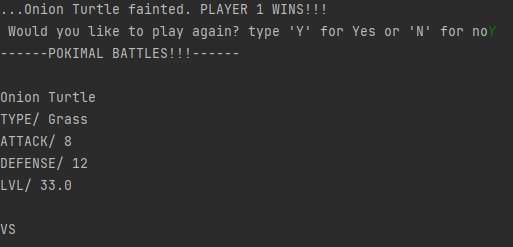
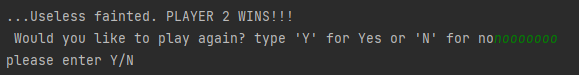
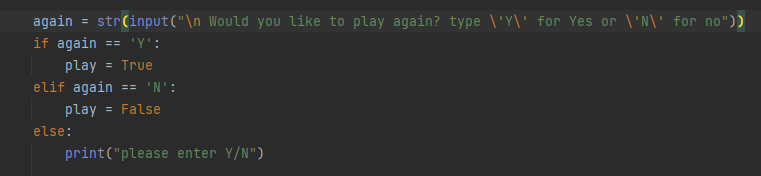




This ended up working, allowing the user to pick whether they would like to replay or exit the program. However, I found a problem during its testing and debugging where you could enter anything that was not directly “no”, and it would continue. Even if the user wrote “no” as “NO” or “No”, it would only pick up “no”; thus, the program would continue until the next time they get asked to play again. This was not an excellent user experience and conflicted with my functionality implication for the project.

This is where I had entered NO; however, since it was not spelt the way I had in my code, it did not pick up, making the fight part run again.

I decided to set the answer as Y and N. when they are asked to play again to address this issue. If they answer Y, then play remains true. If they answer N, then play turns false, stopping the program. If none are inputted, it asks the user to reenter their answer with either Y or N.

This worked well at making sure that what the program did was precisely what the user asked for. This also helped with user interactions as it helped let the user know exactly what I wanted them to answer rather than just insinuating that the user will spell out no to stop the code the way I wanted them to. This resolved all the functionality problems I was having, ensuring that I met that implication’s desired expectations. 

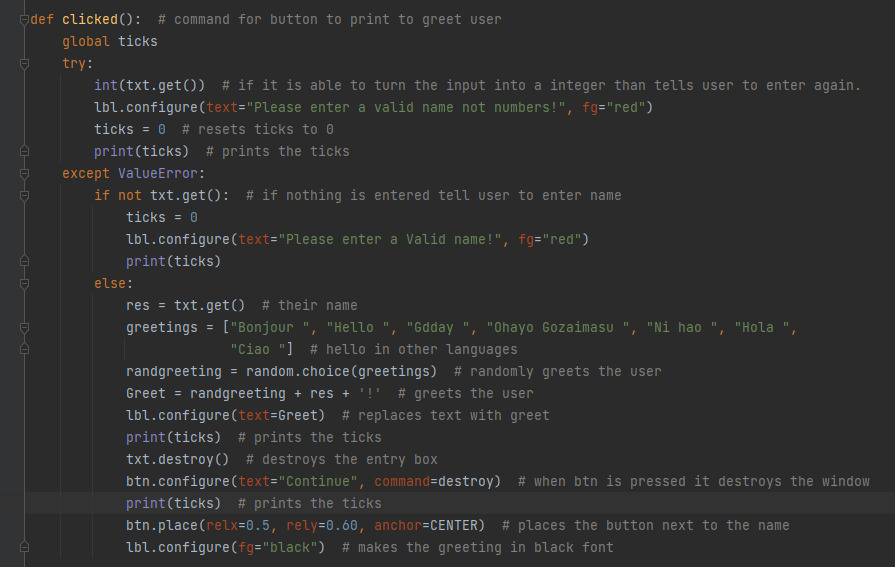
**Testing Outcome**

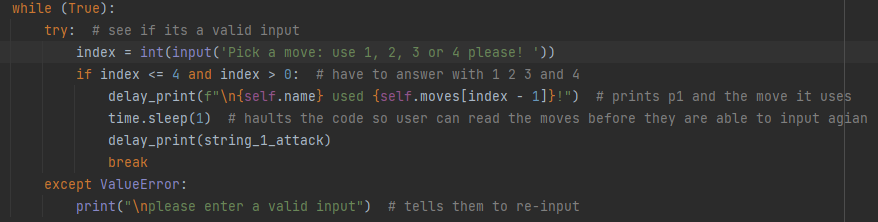
Record evidence that you have tested your outcome.

Explain how the testing made sure your outcome functions as intended to meet its purpose and requirements.

Record evidence that you have used information from testing appropriately to improve the functionality of the outcome.

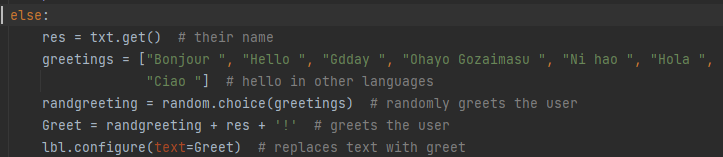
Debugging - Testing

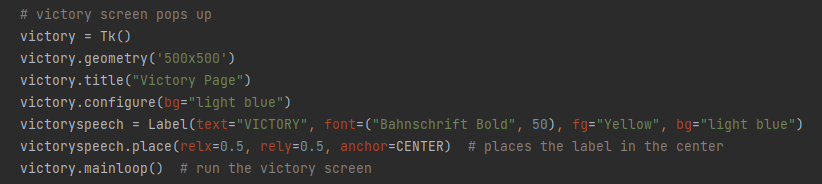
For this, I made sure that any user inputs were directed by me, making sure that the user inputted what I intended for them to enter, maximising the program’s functionality whilst ensuring that the user cannot break my code. Multiple issues were pointed out by peer testing (from my friends and family) and myself, becoming very important for me to fix these issues. I had restrictions to stop the user from entering nothing or integers on the welcome page whilst asking for their name to greet them. This was done as shown below by restricting the user from entering integers or leaving the entry box blank. Since the entry box was treated as just a string, I had first to change the input to an integer, and if this does happen successfully, the user has entered integers. So I remind them to enter a name, not integers, and if they do, they get greeted.

I did these tests throughout my code and found another spot where the user could break it through an invalid input. During the battle part of the program, where the user can choose what move they want to use, they can break it by not inputting an integer between one and four. This fix was crucial as it was the essential part of the program, and without these fixes would allow for the user to break the code consistently due to purposeful or accidental miss inputs. These fixes were taken place to stop this constant breaking of the code.

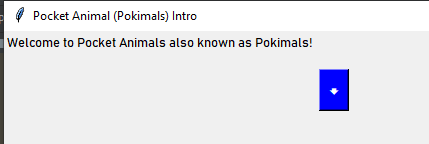
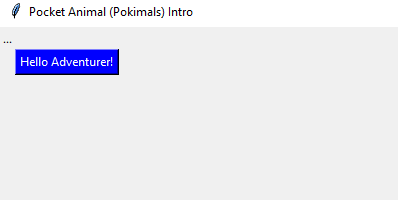
Another I also added was mentioned within trialling and my initial one where I decided to make the play again feature more robust. As stated within the trailing section, this was to ensure that I met my functionality implication and let the user know exactly what I wanted them to input to stop the code from breaking. This was done by adding the following changes to the code where I ask them to answer with Y or N, and if they don’t, it reruns, reminding them to respond with Y or N until they do.

Outcome - Testing

Testing each part of the code, I got peers to try it to see if they could find any improvements I could make to the program and make sure that my code was unable to break. While one of my friends was going through the code’s greeting part, they stated that I should add multiple greetings for the user rather than just Hello. I took this opportunity to greet people in languages other than English as I felt this was an excellent idea. This also helped me address the cultural implication, ensuring that my code was not rude towards certain cultures and was inclusive to all. I decided to add a fair amount of greetings, one from each culture that I could remember at the time.

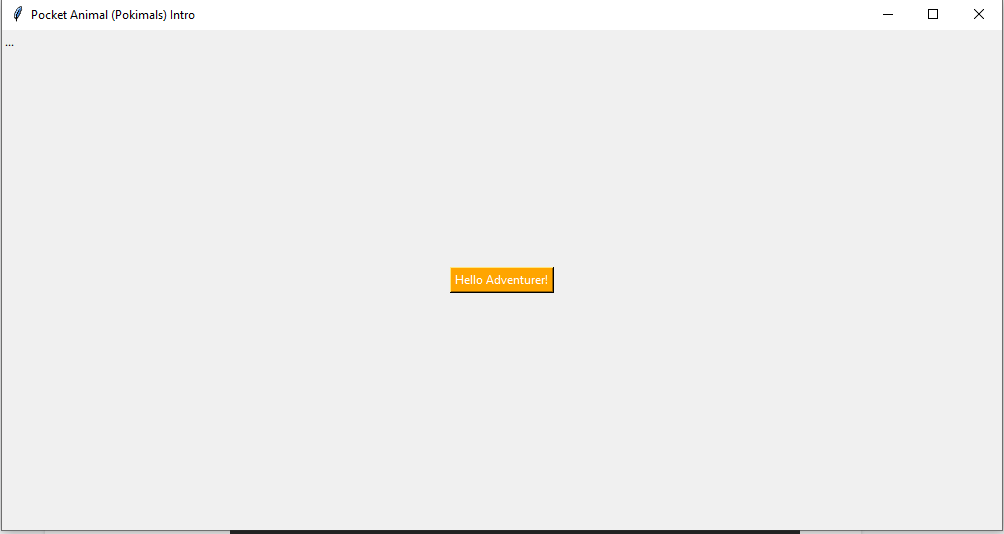
Another thing that was recommended to me by peers was to add a victory screen at the end of the battles to make it less robotic and celebrate the user’s success. Before, I did not have this as I had not thought about it. I decided to add it as I liked the idea. It made it more exciting to win and felt as if they gained recognition for it rather than it being ignored in a way. This was a very easy add and turned out to be liked by peers once added. This is what it looks like.

Something that was also pointed out to me was when I initially had only one page for the GUI information section of the project. My peers wanted more interactions with the GUI, so most of them recommended adding a simple button to the GUI or making it more interesting for the user while the text is being displayed. This was mentioned in my trialling section. I first implemented the button that allowed the user to control the pace of the text displayed and, when pressed, moved onto the following sentence of information. I got this to be tested by many peers who said that they loved the concept of having more interactions but found the exchange to be very awkward due to the button constantly changing positions. A peer of mine is colour blind and explained that the blue button was not an excellent colour choice as he struggled to determine what was written inside the button. He would prefer another colour such as orange. I considered all of this as I wanted to ensure that the usability and functionality implication was met.



I fixed one issue by changing the button colour to orange. He then tested it and found it easier for him to read the text inside the buttons. I addressed the shifting button problem by finding a way to centre the button inside the window. This was achieved by using the following code.





I got my peers to test this section once more to see if it was what they had wanted. It was agreed upon that this was a very good improvement, and they all loved it. They liked how interactive the GUI was now without it being too much. They also wanted the button to be centred as it made the interactions not awkward anymore, and they all preferred the orange button to the blue button.

**Project Management - Applied**

Record evidence that you have used your project management tools to manage the development of your outcome.

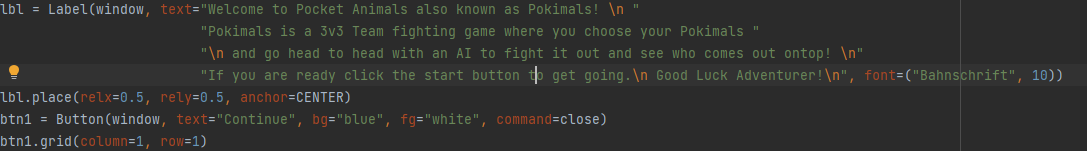
Record evidence that you have used your project management techniques to manage the development of your outcome.

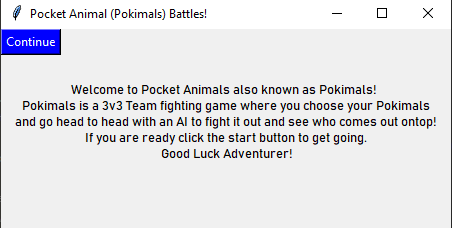
Record evidence that you have used version control tools to manage the development of your outcome.

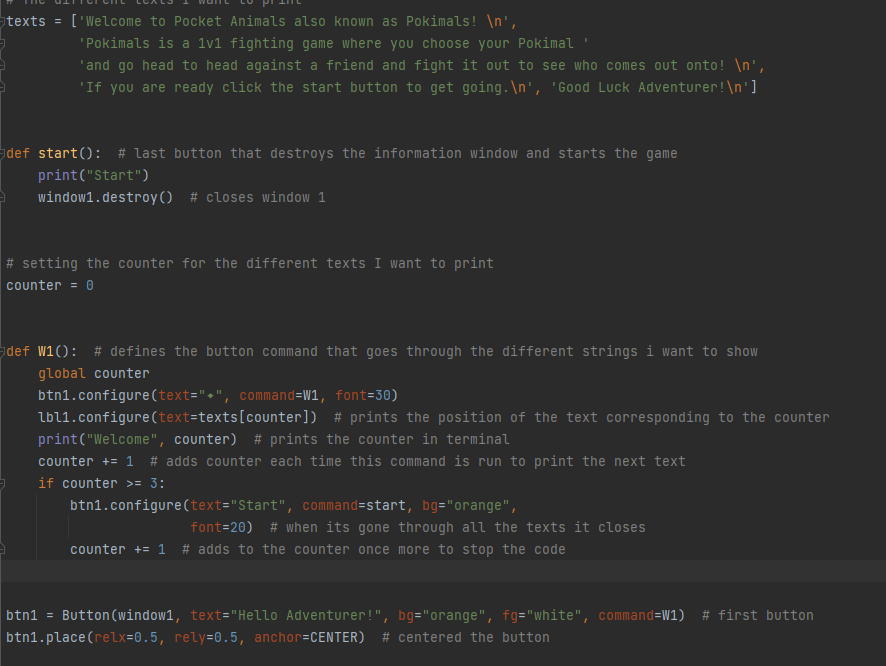
Project Management - Tools

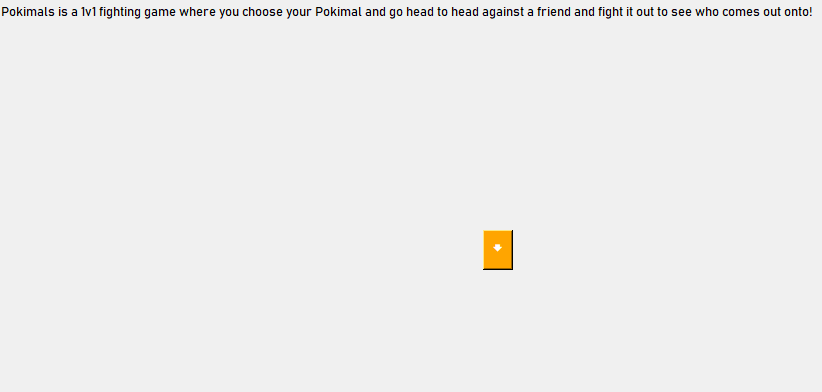
As stated in the setup section of project management, I decided to use a Trello board as my primary way of managing my project. I used it to organise and plan out my project by allowing me to track the progress of my project. This was good as I could see what still needed to be created and tested while working parallel with my decomposition graph to see if I was working inside my assigned time for each section of my project. Features that were useful of Trello were: Archive, watch, History, and tag function, which allowed me to add a coloured tag when a card is moved into a section. This was good to see if a card had gone through a group or not.

Project Management - Techniques

The project management method that I had used was the agile method compared to the waterfall method (traditional method) to create the project. This was due to the many advantages that the Agile methodology has over the waterfall method. This is due to it being a repeated and iterative process. Rather than completing the whole project and then making improvements and testing, the waterfall method is. The agile methodology allows the user to make design decisions as coding. If they find that something could be better rather than waiting till the end, they iterate their design until they reach what they think is the best they can get. Agile methodology is a cycle rather than a linear line. It provides more flexibility to make changes that can impact without hurting the rest of the project. An example where I used the agile methodology was during the creation of my GUI information screen. I initially started with a more straightforward GUI with all the text available and one button to continue. This part was to give information about my project to the user. My first creation, which was very basic, ticked all the boxes with my end goal in mind. After testing it with my peers, it was said to be too tedious and very basic. So after some research, I found that there was a way to get a button that would go to the next bit of information when clicked.

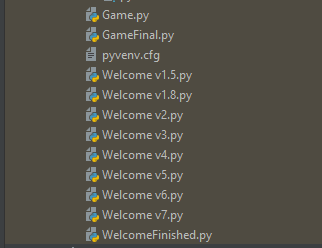
**\*This is the first iteration**

**Second Iteration -**

****

**\*note the button is centred within the window; however, the screenshot makes it seem as it is not centred**

Project Management - Version Control

Version control was a critical tool that I used to manage my project more straightforward, proving to be very useful. It allowed me to see any errors that were caused due to the addition of new code, as this occurred many times. I was able to refer between the version causing the error and the version before to help isolate the possible suspects of the issue for a faster fix without affecting the previous versions. If I were not to find the problem and fix it, I could always revert to the older version that works and rebuild the new code from there. I was using Pycharm for my coding and was very relieved to know that it has an autosave feature, allowing me not to have the stress of constantly saving during any new additions of codes. Version control helps build the sections layer by layer, ensuring each layer was error-free before moving to the next one. If an error were to occur, I could always rebuild it since the previous layer was strong and allowed an excellent foundation to build on each time. 

**Version control**

*Your version control evidence should go here.  This could be in the form of annotated screenshots which show you managed this process, or you could make a brief screencast explaining how you implemented version control.*

I demonstrated a great use of version control through 11 full versions of my Pokimals project. These versions were split between Welcome being the GUI greeting and Information part of the project and Game is the core game, Pokimal stats, Victory Screen and play the part of the project. Each version implemented more difficulty to the code from the next. The final version (GameFinal) was the most polished game and Welcome versions, integrating one. Welcome 1.5 being the first version where it was the basic information GUI. Welcome v1.8 was another version that implemented a list of information and a button used to display each part of the information from the list. Between these versions, I found a lot of bugs and issues within my code due to my first interaction with a GUI; thus, I decided to make another version before making any of these significant changes to help debug and having the opportunity to revert to an earlier version if need be. Welcome v2 was where some considerable debugging took place. It allowed me to polish any errors between v2 and V1.8. Welcome v3 was just an experimental version where I trialled many ways of centring the button within the window. It created many bugs until I found a way I could which was implemented into v4. This version was to be the last of the GUI Information screen and was where I polished out the remaining bugs and made sure it was perfect before moving on to the GUI greeting. Welcome v5 was where I was first created my greeting GUI, which contained many bugs and was easy to break. Welcome v6 was where I added the restriction to ensure that the user entered something to get greeted. Welcome v6 was also where I fixed all the bugs that I could. In Welcome v7, I implemented the restriction not to let the user enter integers for their name. And Welcome Finished being the Final version of the Welcome part of the project. This version was combined with the integer, and no entry restrictions and bugs are fixed to create a polished and final product.

The First game version consists of just the game and the Pokimals stats. This was where most bugs were fixed and was a pretty solid version. GameFinal was the last version for the Project where I import Welcome Finished and add a victory page and play again feature. I also fixed some bugs where the user could enter a random number instead of 1-4 of the moves of their Pokimal and cause an error code to appear.

Version Control is crucial when coding any project due to how much it can help while debugging. It enables you to understand your code in stages and identify bugs by referencing previous versions for a quicker and steadier fix, allowing for more efficient coding.

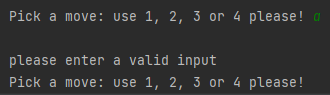
**Relevant Implications**

*Explain the relevant implications here. Please* [*watch this video*](https://masseyhighschoolnz-my.sharepoint.com/personal/jgottschalk_masseyhigh_school_nz/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fjgottschalk%5Fmasseyhigh%5Fschool%5Fnz%2FDocuments%2FProjects%2F01%5FTutorials%2F12I%5F91896%5F7%5Fv2%5FAdv%5FProgramming%5FRevisited%2F91896%5F7%5FVideos%2F02%5FRelevant%5FImplications%2Emp4&parent=%2Fpersonal%2Fjgottschalk%5Fmasseyhigh%5Fschool%5Fnz%2FDocuments%2FProjects%2F01%5FTutorials%2F12I%5F91896%5F7%5Fv2%5FAdv%5FProgramming%5FRevisited%2F91896%5F7%5FVideos&originalPath=aHR0cHM6Ly9tYXNzZXloaWdoc2Nob29sbnotbXkuc2hhcmVwb2ludC5jb20vOnY6L2cvcGVyc29uYWwvamdvdHRzY2hhbGtfbWFzc2V5aGlnaF9zY2hvb2xfbnovRVRHVzFaNmZ6anhHbkNJYlFvblVpcDhCbVNaV2g3djYtMzF3N0w5OHdiRzktQT9ydGltZT05SGxJaFMwODJVZw) *to learn how to do this.4*

Explain the Implications that are relevant to your outcome.

Record evidence that you have addressed the Implications that are relevant to your outcome.

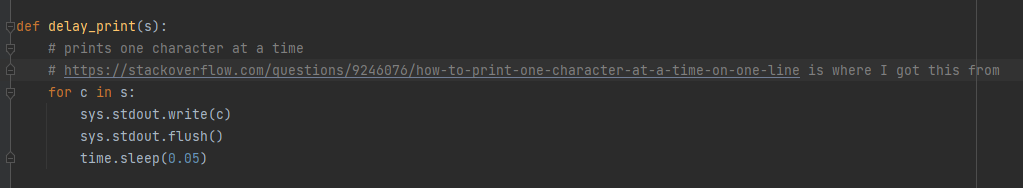
Functionality

A functionality implication is there to see if a given item achieves the tasks it has projected to accomplish and what can be done to achieve these tasks and function how intended. The functionality implication intertwines with the usability implication; however, it’s based more around if the code achieves the purpose intended rather than if the way it achieves this purpose is user friendly or not. I addressed this implication during Welcome versions 6 & 7 and version 2 of the Game. Many minor bugs caused the game to break throughout my project, which is addressed and resolved. These were achieved by letting the user know that they could only enter a set amount of things, and if the user still did not listen, it would not take the input and tell the user to make sure that they are entering a valid input. This was to allow the user to use the program as intended. I made sure that I guided them into entering an incorrect input when they entered a wrong input. I did this by adding hints such as telling the user to enter Y or N at the end to play again when they enter incorrectly.

Usability

It is needed to let the end-users understand the actual objective of the project with minimal navigation required for them to be able to achieve their goals within the code or game. This needs to be shown visually to the end-user and is up to the developers to make it easy for the end-user to understand. If an app or game were to be created with a purpose but with no way of achieving the goal due to minimal help from the developer, unless intended will cause a lack of potential users of the app. This is because we are very reliant on hints from others in a new environment. I have addressed the Usability implication by guiding the user to help them play the game as intended with minimal frustration. This I accomplished by constantly letting the user know exactly what I wanted them to input and, when incorrect, was willing to help them correct it. These pop up in error messages assisting the user in playing my game, addressing my Usability Implication.

Copyright Implication

The misuse of others work can breach the user’s rights over them as longs as they are copyrighted. With Google being a very open-source browser, there is still a lot of work with restricted privileges for others, and many people gain help from this source to achieve their objectives. Without proper permission of the creator or agreed compensation, it would indeed be a breach of copyright. This is a legal matter that could cause lawsuits. Google CEO is currently debating whether all their privacy breaches were worth the lawsuits. Sticking to the implication is very important. I made sure to address this Implication by using my characters rather than premade ones from the Pokemon franchise. All my work was not ripped code or work illegally or wrongly. I credited where due to helpful articles on the web that helped me achieve given parts of my code, such as the slow print function shown below.

Maintenance

Once a program is created, it is essential that changes are easy to make and it is easy to maintain for it to keep on running. Many bugs can appear after extended use of time, and it is vital that these tiny bugs can be fixed quickly to minimise the program’s downtime to show its reliability. This can be achieved in many ways, such as annotating each part of the code so that it is easy to identify what code controls what part of the program. It is also helpful if the code is arranged neatly to help navigate it and understand each part of the code. I addressed the Maintenance Implications by neatly placing my code in order while annotating each piece. This made sure that the code was easily accessible and the purpose of each part of the code was understandable. This minimises confusion when maintenance is needed in some regions to lower the software’s downtime.

**Evaluate**

Discuss how the information from planning, testing and trialling of components assisted in developing a high-quality outcome.

To conclude, I used the project management tool Trello to create a board to organise my project. Some excellent features used as I went on with my project were the archive, history, watch and tag features. I made sure to decompose my overall project into smaller sections, assigning each with a given time frame to work parallel with my Trello board, keeping me on track to completing my project on time. The Trello board helped me project stress-free towards my end goal and tracked my progress for each part of the project. This gave my project a smooth road to travel on, paired with the agile project management method, allowing me to create a bug-free and objective completing project.

I trailed my code a lot to ensure that any improvements that could be made were taken care of. I trialled and got my peers to test my project to help me create a more user interactive experience while keeping it user-friendly. This created a more publicly appreciated code for my game to grow into a broader audience and appeal to them.

Testing was critical to fix the major issues with user inputs that my code had. These were code breaking bugs and problems when the user was to enter an invalid input. An example of this was when I ask the user to select a move they would like their Pokimal from 1-4. However, the user could enter any other integer or a string that would cause the code to throw an error message and break. I had not taken this in as I had thought the user will always input the correct answer as I had stated for them to pick 1,2,3, or 4. Knowing that this was a significant flaw within my code, I created a restriction where the user had to select a number between 1-4. Otherwise, the user would be told to enter a valid input. This would keep on occurring until the user enters the correct input. Other minor game bugs where the code would stay in a continuous loop were fixed. Doing this, my project came out to be very robust, withstanding the purposeful and accidental invalid inputs.

How I could improve

Although I was thrilled with the GUI and project as a whole, I was sure that if I were to have a sufficient amount of time, I would be able to create a project that would be way more impressive and robust than this one. I would have liked to develop the core game within the GUI rather than switching to the terminal as it was my first time interacting with the GUI. I was not ready to throw myself into the deep end and not being able to achieve it. This would cause me to scratch everything and having to restart everything from the start. I am confident that now with my knowledge and experience with the GUI, I would be able to create a similar project where everything worked within the GUI well and efficiently. This would allow me to add drawings of each sprite and a visual attack animation, and a lively health bar. This game would impress my peers and me; however, with being fresh to GUI’s and my first proper time using classes within a project, I was happy with the core game and what I had achieved as a whole.