**Key =**

**Green = need to do**

**Yellow = unsure to add or not yet, do later**

**Red (dark) = Greens that have been done and can now ignore**

**Black = done in the update**

**Pink = Temporary**

**Purple = Errors encountered and fixes to them found**

**Orange = Purples that were edited**

**Teal = Noticed / Interesting point encountered during implementation**

**Update 1 - 24th March 2025**

* Started off with pygame template
* Created basic class for the player’s character with the character’s dimensions
  + Needed a class because player later on will have methods like certain actions such as picking up collectibles, interacting with buttons and that so it would keep things organised by having a class and everything the player can do with their character in one place.
    - This also is needed because later on, I will be adding properties of the player’s character, such as their health and any other reasonable properties
      * Maybe health affects speed? (lower hp -> slower running speed)
* Added very basic movement system, very temporary
  + Did this because first I need to get a grasp of how I can get the player’s character to start moving. I initially tried to use a while loops, but I noticed that it just froze the game. This is because when you use a while loop to detect when the key is being held down, it waits for you to let go before continuing, freezing the game. I wanted to do this because I thought what it would do was temporarily cause the character to move for the duration of it being held down, but I failed to notice that, since the while loop only runs what is inside of it, it wont turn it back off the moving condition when I stop holding the key, you cant really do that. This lead me to find out about dictionaries for this movement system, since I can make a dictionary that says like, each key references a different movement condition, like ‘A’ references the moving left condition (a variable), and when that variable is set to True, the character’s X position is changed
    - Discovered that dictionaries could work for this, so I don’t have to use a thousand if statements for basic movements. Kind of have an idea for this, if I make a dictionary with the keys being the pygame key events, like ‘pygame.K\_w’, and then have the value pair be the movement conditions (left right up down),
* Added a variable which is a list of all the sprites in the game. This may be useful later on because I will need to hold the sprites somewhere so they can be referenced. \*\*Might need more lists later for different sprites, like one specifically for all the platforms, etc.
  + Needed for detecting collisions between player and blocks, so that the player doesn’t just fall through the ground for example.
* Temporarily added a line of code that draws a circle, just using this for now as the character to test the movement to make sure it works.

**Update 2-5 – 25th March 2025 – 26th April 2025**

* Created new files for levels. My game will have levels so I will be doing this by having other files that are the levels, then importing them like modules to be drawn into the main game every time a player beats a level.
  + Thinking of making a bunch of functions in each level and what the functions do is just draw the level, then when I import it and call that function, it does all the drawing the function does in each map.
    - Think of some simple level designs for now just to test if this would work
* Testing jumping mechanic, using space bar as the input, I will make it so that for now, jumping will let you reach a certain height, then decrease your Y axis value back to original – later on, change this so that it just continuously drops the Y until you hit something or fall out of bounds. Could probably use a loop to increment your height a few times at a decreasing rate, then from then on until a boundary is reached, continuously increment your altitude in the positive y direction (down) at an increasing rate.
  + I think this would be made easy with sprites because when I was checking all the methods for sprites in “sprites.py” of the official pygame dictionary, there was a method called “spritecollide”, I realised I can use this to detect if the player (already a sprite) collides with any platforms (for now I will do it as if they only collide with one platform. By detecting which platforms they collide with, I can compare the position of the character with the detected platform that they have collided with, so like for example if the character’s X is within the range of X values the platform takes up, and the Y of the character is smaller (meaning they are above the platform) than the smallest Y value the platform can take, then I know the player is above the platform and standing on it, therefore I can change the movement conditions so that while this is active, the player cannot just fall through the platform. The player passively falls due to gravity, but they won’t while on the block. Also applies to next point.
* Added boundaries so that the character cant just escape the screen and disappear. I want to later on make certain parts of the boundaries kill blocks so if the player falls into it, i.e if it’s the void in a map, then they get respawned to previous checkpoint. – I can do this via sprites as mentioned above, then simply reset the player’s class’s x and y position back to the starting x y position of the level, then decrement lives by 1.
  + orange
* Was testing modules to try to use level 1s module in the main game, but resulted in “circular import”, meaning level 1 executed main game in the process, the solution to this is by making a third file with all the base settings I might need to use, so I can import from that file, without accidentally running the main game since python when you import a file, it runs it as well from what I’ve seen so far.
  + Or could also solve this issue by passing in anything I need directly by producing a function then putting the variable I need from main game directly in, though probably more inefficient, at the moment I’m going to use the first method. This method worked out better, for now I’ll stick with this new method, the first method didn’t really implement that well and required me to make a bunch more files which may get messy later on. Keeping it simple for now
* Level 1 successfully made, will now use this method to utilise modules to draw my maps and levels. Also made a few if statements for later on because I will need to update the player’s level once they beat a level, so that after they beat it, the game draws up the next level. While doing this I noticed that when you draw things in order in pygame, pygame draws them on top of each other, like I had my test character drawing code below the if statements, so it drew the map first, then it drew the character so when I move it around (I haven’t added collisions yet up to this point), the character when moved into the map’s platforms, instead of being beneath the platform and hidden, it is on top of the platform, but if I put the drawing code for the character first, it will be underneath the platform and hidden when it enters the dimensions where the platforms are drawn.
* Replaced the test character with the actual character so what is being drawn on the screen is the actual player
  + Did this by making inside of the character class, a method called “drawPlayer” which just draws the player’s body which is made up of multiple pygame draw functions to make up the character’s body, then to make the movement from before work with the new character body, I already had the variables on the information about the character’s body, so that includes the central X and Y position the character will be in, and then just replaced my original conditionals with them and the X and Y place holders as well. It worked as I expected because I made sure to keep the class simple for now and not do too many things with it yet, because if I did that that may result in making it harder to link the class to my movement system. I’m going to get the foundation of the game (maps, base systems like linking different maps together and making the game ‘s graphics work etc) implemented first before I start working on all the controls and dynamic things because the dynamic mechanics of the game may end up being changed by the player, like controls.
* Also started planning game design i.e graphics for all the obstacles, their visual appearances, placements on each level. -------
* Created a new module specifically for storing a bunch of colours, this way I can keep my code in main game as tidy as possible. I want to make sure that my main game is literally just the game, all the mechanics and that etc (for now, later on modules may be needed for more advanced mechanics to keep the game tidy). This way I can develop my levels easier and when I need to use my levels (in the form of modules) I can easily reference a colour from the colour module in my main game and draw the level without wasting lines of code on my “maingame.py” file. My game idea is pretty large for one of my first ever game projects that I will follow through on so it is important to make sure it is organised so that I can implement everything I need to implement without looking everywhere for one small part of the game.