First to understand the topic better I started watching videos about blockchain and the relationship between blocks and how they are connected, what they include in them, how a blockchain works etc. I wasn't very familiar with the syntax of Python so I also took my time to get used to the syntax of it. After I grasped the topic and the assignment I started implementing the first module, block class.

First I started off with creating a constructor for the block class because I would need to create multiple block objects for the blockchain. To make the data private I copied it and didn't access it directly. After that in order to calculate the hash of the block, I needed to receive the passwords in data with underscores between them in an efficient way, so I created a method "concatenate_data" for that usage. Then I created the "calculate_md5_hash" method and first I concatenated the necessary attribute. With the help of the link about the hashes in the assignment file, I imported the hashlib module and converted the concatenated string to a MD5 hash. After that, I created the “check_hash” method with the help of the "calculate_md5_hash" method.

Then I continued with the second module, blockchain class. First I imported Block from block.py file to actually create and use blocks. Then I started creating the constructor of the blockchain class with the empty chain array. I also wanted to implement a method that would help me to create an initial block, and I wanted to use that method in my constructor because it would be the way the blockchain is by default in this assignment. However, to initialize a block, I needed to calculate the timestamp of it in real time. Because I would need a timestamp for my “create_initial_block” method, I started implementing the “timestamp_calculate” method. I searched for how I could use the real time data and finished the method with the right syntax for the date and time. Also because it doesn't rely on any attributes or state of a Blockchain instance, it only calculates and returns the current timestamp, which is independent of any specific blockchain instance. That’s why I made the method static. After that, also using the “timestamp_calculate” method, I finished the “create_initial_block” method and used it in the constructor of the blockchain class. After that I started implementing the “add_block” method and to figure out the index, I checked the length of the chain. Then I accessed the previous block by self.chain[-1] and then to its hash. After that, using the “timestamp_calculate” method and given data by parameter, I created a new block object and appended it to the chain. Then I implemented the “print_chain” method by iterating the blockchain and accessing its block’s attributes. I also added a line in the end where it prints “-” 50 times for easier reading. Lastly, iterating over the blockchain except the first block and using the “check_hash” method of the block class, I implemented the “check_chain” method.

Lastly, I started with the final module, test script. The coding part wasn’t challenging but the term “test script” was new to me, so I had to search about it. I implemented 4 methods, the first one is the setup method. With the help of the tip in the assignment

(sleep function), I iterated an array that includes 3 example data and added a block with the specific data to the blockchain in each iteration. For the method of the first test, I printed the blockchain and added a condition to see if the result of “check_chain” would be true, if yes, SUCCESSFUL message is printed, FAILED otherwise. For the second test I had to change the timestamp of the third block to the current date, but I also had to use the original timestamp for the third test. So after the main condition before executing the second test, I saved the original timestamp to a “original_timestamp” attribute, so I could change the modified timestamp before executing my third test. With this addition I continued writing the second test method and modified the timestamp with the current one. The rest of the steps were the same with the first test method. For the third test method, I have an extra parameter “initial_timestamp” to use the “original_timestamp” attribute I created earlier to use the original chain, with this addition I set the chain back the way it was created initially. Then I had to set the hash of the latest block to a random hash. For it to have a realistic MD5 hash, I set it to the first blocks hash. Then from the last block to the first block (initial block excluded), I called the “check_hash” method for each block and appended the result to an empty “output_array” string. After that I created the “ref_array” that was given in the assignment file and checked if the “output_array” and the “ref_array” would be the same, if yes it’d return SUCCESSFUL, FAILED otherwise. When I was done with my method implementations, I tested these methods under the main condition one by one (also added three empty lines for it to be easier to read), and with that “test_blockchain” script was done too.

This assignment was really helpful for me to get used to python syntax and grasp the blockchains basic logic, and to improve my skills to search and find helpful resources online (real time and date usage, sleep function, test script logic etc.).