I didn't face very many issues in my coding up until XOR Caesar and Vigenere. Salting was easy, just add the salt, and return the string minus the length of the cipher to decrypt. Reversing used some simple for loops. Next was the XOR cipher. This one was partially difficult to figure out how to get started. I learned some new python methods like ord() and chr() to turn the string into numbers and back into characters. I used the built-in XOR function ^ to complete the operation and turned them back into characters. I figured out that this works in reverse on accident because of the way I had originally set things up, not realizing we were supposed to decrypt as well. Turns out to decrypt you do the same thing with the same key.

Caesar cipher mostly was difficult due to my own organization. I had some pretty funky things happen while trying to loop back around from z back to a because of uppercase and lowercase characters. I learned some new character methods like upper(), lower(), and alpha() - (From talking with you!!). These helped categorize things easier rather than using the ord() numbers. After categorizing it makes it much easier to handle what to do in each case. Using while loops will work for any large number, although mod might've worked I didn't really want to mess with that.

Vigenere cipher was difficult for very similar reasons, and after also completely reorganizing and writing the code for it everything started to work properly. Instead of using the ord() numbers to categorize things and try and do everything in the same spot it works better to treat the lowercase and uppercase letters to their own separate treatments in groups. The biggest difference comparatively is subtracting 65 from the number you get when converting the key to "standardize" it so that A=0, B=1, C=2… etc.

Surprisingly, I got custom mapping to work on my first try. Figuring out how to decrypt it was the slightly trickier part. Unfortunately there's no good way to directly search a dictionary by its values and get the keys as an output, but instead you can convert the keys and values to lists, and find the index of the value and plug that index into the key list. I found this strategy off the internet while trying to figure out if you could simply plug in a value into the dictionary and get a key.

When it came to making the tests, I went what I would consider to be above and beyond. I thought of every possible way code could be messing up and I have a ton of alternative tests for blank inputs, wrong type inputs, and lots of tests for Caesar and Vigenere. Big positive and negative values for Caesar, as well as special characters for Vigenere. I tested the encrypted and decrypted values for all of these different key inputs that might mess up someone's code. I also made sure to check in the ciphers with keys that when creating the object the key value is correct, and that after decrypting it, the key value had not been changed.

All in all, I didn't focus very much on generalizing code and creating methods or anything like that. One thing I did do is as a part of reorganizing Caesar and Vigenere, I made sure to only put things that were specific to lowercase and uppercase inside their respective if and else statements, doing everything else before entering these separate cases. It really helps to do as much as you can before the if else statement that way it all only shows up once in the code which saves memory as well as makes the code more readable. Overall this was a fun project, and I really enjoyed coding these different forms of ciphers.