Euclidean Distance Thought Process

Omar Bustos

So for this assignment I had to calculate the Euclidean distance between two points. I already knew the formula from math, but turning it into Python was not as simple as I expected. At first it felt like everything I wrote broke for reasons I didn't totally understand, and I had to keep fixing little things until the whole program made sense.

I started with the two points: (6, 6) and (-6, -6). At first I was using the variables x, y, h, and u by themselves, but it got confusing really fast. That's why I made two functions, point1 and point2. Both of them just return a tuple, but having two separate functions helped me keep track of which point was which. Before I fixed them, I accidentally wrote the functions in a way where they called themselves over and over. I honestly didn't even realize that would happen until the code obviously didn't work. After fixing that part, both functions finally gave me points I could actually pass around.

Next I made the absolute(p1, p2) function. This is the one that calculates a and b, which are the differences in the x-direction and y-direction. Inside the function I unpacked each point into x, y and h, u because it made it easier to read. Then I used abs(x - h) and abs(y - u) so I never have to think about negative distances. This part actually worked the first time, and I felt like "okay, at least something is working now."

The next part that gave me issues was trying to write the Pythagorean theorem in Python. I kept wanting to write it exactly like the equation "$a^2 + b^2 = c^2$," and Python just did not accept that. I didn't know why at first, but eventually I realized that Python doesn't let you write equations like that. You can only assign something to a variable or return it. So inside my theorem(a, b) function, I changed it to c_squared = a*a + b*b and that finally worked. That was one of the moments where it clicked for me that I needed to stop thinking so much like math class and more like how Python actually runs things line by line.

Originally I tried making another function that would take c^2, "fix it" somehow, and then give me c. But that whole idea fell apart because I didn't understand how to pass c^2 into the function, and I even tried writing c**2 in the function header at one point, which obviously didn't work. Later I realized that the whole step wasn't even needed because c^2 can never be negative anyway. So I just wrote a simple function that returns n ** 0.5 to take the square root without using any libraries.

One thing I accidentally did before fixing the code was calculating a and b twice: once manually using the raw variables and once using my function. It didn't break anything in a huge way, but it felt messy and I knew that wasn't how the program was supposed to be structured. So I deleted the extra part and only kept the version that comes from the absolute() function.

After putting everything together, the program actually worked exactly how it was supposed to. It printed the right distance and the whole thing finally felt clean. The biggest thing I learned from this assignment is that math and Python look similar but they're not written the same way. I had to stop using "math brain" and start using "programming brain," especially with equals signs and how functions return things. Once I understood that, everything else started falling into place.