When encountering an issue, defect, or feedback, what did you do to resolve the issue in general?... (r0_DebugIssue)

We used lots of debuggers and wrote small unit tests. (r1_DebugIssue)

I used the debugger to see where the error was thrown and traced backward to the actual point of failure. (r2_DebugIssue)

first find the line where the error occurred, looked into the message and checked if it was something simple. if not simple to debug, would use console logs to try and find flow/logic errors, usually would be async problems (r3_DebugIssue)

order - read the spec again, check that the implementation matches it and then see if any testes need to be added (r4_DebugIssue)

Usually I would look at my code first, and try to figure with some test. Then I would go to office hours and if they weren't able to help I would then try further debugging on my own. (r5_DebugIssue)

If I found a test defect locally (through a test failure) I would step through the failure and see which area was going wrong, then make an adjustment and run the test again. I would run the test within a group of tests too to make sure that it passes (eg. running yarn test). Then i'd usually submit to the auto-grader (if all tests were passing) so that I could see how we were doing. (r6_DebugIssue)

Figure out the functionality that's causing the issue, then narrow it down to the details that I could change to resolve the issue. First I read the feedback/issue, then run the code of the function that's causing problem, afterwards I reason about the issue and make changes to resolve it. (r7_DebugIssue)

When I encounter an issue, it is usually an error from my self made tests, since those are the first set of tests I run. When those fail I can pinpoint which function it is coming from. I will then read through the code o try and debug it. If that doesn't work then I run it through the debugger. If I cannot find the bug then I usually go to office hours or piazza for help. (r8_DebugIssue)

understand where my test case went wrong, go back to the spec, check if my understanding is reflected in the code (r9_DebugIssue)

Whenever I hit a snag, I'd first hit up the docs, then try out some code, maybe bug a mentor or friend, and keep tweaking and testing until things clicked. (r10_DebugIssue)

When encountering an issue, I would look at either piazza, or google similar errors and how people dealt with similar situations. Along with this, adding more intensive testing so that I knew if my approach was correct or not was also a strategy I employed. However, a lot of my coding resolutions come from coming up with solutions through trying different snippets of coding and using the debugger to find out where problems exist. I normal would look at my code to see if there are any major issues, then run with a debugger, then read online materials about similar issues if I haven't found a solution myself, the code and the run again. (r11_DebugIssue)

try console logging and running debugger to see the objects and values affected, then identify which step did things go wrong and go fix code. ChatGPT if not sure how to fix the code (r12_DebugIssue)

Normally I would first do a careful rereading of the spec, then give one to two attempts just coding a solution myself, then would turn to my groupmates (we were an odd group of 3) to see if they have suggestions or ideas. (r13_DebugIssue)

if we ran into something not passing on the smoke tests or hints to improve we often went back and tried to think of any other tests we may be missing, if we felt our test suite was comprehensive but we still had issues we then checked Piazza to see if others were experiencing similar issues - and lastly we attended OH if needed (r14_DebugIssue)

in case of a defect, usually talked to TA's in OH, when I was experiencing a really bad defect it was usually bc everything seemed right to me and the local tests were passing, but autobot wasn't passing. (r15_DebugIssue)

Reading the error by the complier if any, re-reading my code and debugging, re-reading project specifications, adding additional tests, asking TA/Piazza, and referencing documentation online. (r16_DebugIssue)

Seeing inputs and outputs of tests, using console.log statements (r17_DebugIssue)

Run code, get unexpected behavior, add print statements to areas of suspicion, run code again to see what was printed, debug where necessary, highlight where the bug was, fix the bug. Repeat until the bug is fixed and all of my tests pass. (r18_DebugIssue)

print statement debugging (r19_DebugIssue)

Ask TA, read tests, check coverage, pair program (r20_DebugIssue)

general debugging strategies (console logging) (r21_DebugIssue)

I try to replicate the bug, then go into the code section to try to find the issue. (r22_DebugIssue)

Read the feedback from the autograder to determine what was going wrong, although the feedback was very vague often. (r23_DebugIssue)

I commented out the other tests and then stepped through what was going on. I had introduced very thorough error messaging so that I would always have at least a rough idea of the location of a bug (r24_DebugIssue)

Office hours, feedback from TA's about the test logs, local tests, pair programming. (r25_DebugIssue)

console logs, debugging tool, Piazza, spec, checkpoint page (r26_DebugIssue)

Used the debugger to step through code that was not executing as expected. Ran each tests individually instead of running all of my tests at once. (r27_DebugIssue)

Autograder would indicate the right portion of code to focus on. Then would write unit tests, looking for failure. (r28_DebugIssue)

I would go back to read the issue firstly. Try, and code again. Run again and if it's still not passed then ask. (r29_DebugIssue)

I would usually read the spec more carefully, to figure out anything I may have missed. Usually, my errors were edge cases I missed in the spec. However, when I didn't have many local tests, I often had to step through my code and actually find the issues and had to fix them. My most common issues, or atleast the failing tests on auto-grader were from missing particular details in the spec. (r30_DebugIssue)

I just went through the code that AutoTest said was not working and usually I was able to spot the bug. (r31_DebugIssue)

I would read the problem, go through the debugger to see where the issue occurs in the code, try to fix it and then repeat until the issue was resolved. (r32_DebugIssue)

Autotest gives a notion of the type of test which fails, however, it is often misleading as most of the times we found bugs which were from different places that what the autotest suggested but this might be because of the impact of a bug in one place can cause tests elsewhere to fail.\\We mostly just went to TAs when encountering bugs and based on their feedback spent more time writing tests ourselves. Once we could replicate the test failure, we debug line by line to see where things break. (r33_DebugIssue)

I read the error message first, then tried to figure out what general area it came from. If it is an error message from a test, I'll step through that test to see what went wrong. If its an error message from the autobot, I'll first reread the specs to make sure i'm not interpreting it wrong. If i'm not able to find a general direction for the cause, I'll first ask my partner if she has any ideas, and if not, i'll go to office hours and ask. (r34_DebugIssue)

What went wrong? Oh, maybe i should try some log statements first. Run. Oh this variable isn't what it's supposed to be. I then code to try and fix the logic for that variable (r35_DebugIssue)

I tried to assess over the function that i was not fully get the mark. And used debugger to make sure it runs as desired results. I went to piazza and read the spec again, and if I still could not figure it out, I had to go to office hours. (r36_DebugIssue)

I would read the issue and rerun the code (r37_DebugIssue)

Ask TA. The feedbacks are often too vague, and writing tests would take too long. (r38_DebugIssue)

When facing an issue, defect, or feedback, my typical troubleshooting process follows a structured approach:\(1) Read the checkpoint requirements: I start by revisiting the specific requirements outlined for the task to ensure I understand the expectations clearly.\(2) Ask my teammate/TAs: I seek assistance from teammates or Teaching Assistants (TAs).\(3) Revise the code/ try different approaches: Based on the understanding gained from the requirements and discussions, I revise my code accordingly. This may involve debugging, refactoring, or exploring alternative solutions to address the issue at hand. \(4) Run the code locally: After making changes, I test the code locally to observe its behaviour and verify if the issue has been resolved.\(5) Submit the code for more feedback. (r39_DebugIssue)

I discuss it with my friend, sometimes piazza posts are useful. (r40_DebugIssue)

Usually, an issue would get caught by the unit tests, so I would go through the read through the method it is associated with. I know I should probably use more sophisticated methods for debugging, but console logs are my favorite tool to use. Rough order would be to run the specific part, then read through, sprinkle some console logs, then run it again. Sometimes chat also helps. (r41_DebugIssue)

Normally I would try to check in piazza, as most of the times another student has encountered that issue and could potentially help me to guide me throw a solution. Now if there were no post on piazza, I would try to read in the spec if there something extra mentioned. After getting a general idea I would come up with a test case that it would make that specific thing fail. Write the test case and then see if it fails. If it fails then check which part of the code maybe failed and after that create code and make it pass the test case (r42_DebugIssue)

First find unexpected behaviour. Then follow where I think the code should have mistakes and read through. If I can't find a problem I may add console.log to diagnose. If I can't find problem after repeating above steps a couple of times then I'll ask partner and then TA. (r43_DebugIssue)

Read Piazza to see if there was anyone with the same issue\then ask a TA (r44_DebugIssue)

I read the autobot feedback, and then asked the TA if they could help me understand it better. Then I also re-read my implementation, debug and re-run my tests to figure out what was going wrong. (r45_DebugIssue)

LOTS of debugging using the debugger and stepping through each line (r46_DebugIssue)

using the debugger was often helpful as was console logging. also used a bunch of tests (r47_DebugIssue)

I tried to read the code from the first method call to the last, as well as debug console.log or debugger statements. (r50_DebugIssue)

Read the feedback given by autobot, which informs us about which functions are not giving the correct results. I then run code, try to pinpoint spots that caused these errors, rewrite and run code, and test again to see if the errors have been fixed. (r48_DebugIssue)

We didn't really encounter any issues (r49_DebugIssue)

I will first ensure my local test passes first. If there's issue, I usually re-read my code, check output difference, if I have not idea about how to solve I will use Intellj debugger that's usually a efficient tool to debug\if I don't pass the local test I will just look at Piazza if see if someone else encounter the same problem -- usually due to misinterpretation (r51_DebugIssue)

To fix a bug, I usually used the debugger to step through the failed test. If the fix was not immediately apparent, I would create more tests that divided the failed test into subcategories, and try to reason about/debug those tests. (r52_DebugIssue)

'- I tried to use 'binary-search debugging' to pinpoint where my expectations and the code were diverging. I also used print-line debugging.\- I read the spec further.\- I searched on Piazza.\- I asked TAs for help. (r53_DebugIssue)

I would attempt to write tests (validated against #check) that failed on the given issue. Then I would run these tests and compare expected vs actual, possibly with the IDE debugger active to step through. I would ask ChatGPT for ideas after sharing snippets of code and test details. I might also ask my partner, or a TA in office hours. (r54_DebugIssue)

First, we would get our tests to print the error and that usually gives us an idea of where the code was failing. Then, we would then reread that code carefully to find any missed edge cases. Finally, if that did not work, we will check for any existing related-Piazza posts. If there are none, we posted the question on Piazza ourselves. (r55_DebugIssue)

First, I'd try to debug myself by logging to find where the bug is. Sometimes I'd try local tests if the behaviour is off, but if it's a syntactical or language thing, I'd try googling or asking ChatGPT. My last resort is always office hours to ask the TA (r56_DebugIssue)

I would first locate the error, read the code producing the error, and then run the code in debugging mode to try to identify the root cause. If it was a function returning a wrong value and I had not written the function (library), I would read the documentation to try to debug. (r57_DebugIssue)

I write code according to the spec, then I run tests (or run #check for C0). If any tests fail, I debug mostly using console statements until everything is passing locally. If I can't get my tests to pass after debugging for a long time, I will ask my partner for help. If my partner also can't figure it out, I'll ask a TA. Once everything is passing locally, I'll submit to the autograder, and repeat the process for any failing tests. (r58_DebugIssue)

Typical debugging strategies like breakpoints. (r59_DebugIssue)

In general, most of my issues were able to be solved by further readings/forums on sites like stack overflow or documentation for whatever it was I was doing at the time. \\The first thing I would do I google the issue I was having. If nothing came up then I would try to ask a more general version of my question to a chatbot like chat-GPT (since in my experience it is pretty decent at giving documentation based questions). If this also led to a dead end I would have to ask a TA in office hours. (r60_DebugIssue)

Piazza, office hours, stack overflow (r61_DebugIssue)

When facing issues, my process was: 1) Reproduce the problem consistently; 2) Consult project documentation and online resources; 3) Debug, using print statements or a debugger to trace code execution; 4) Experiment with small code changes; 5) Reach out to peers or mentors for assistance if needed. (r62_DebugIssue)

Due to what felt like luck to a certain extent, issues were rarely encountered. The few times issues were encountered during development, I generally read the code first to look for glaring issues. If none were round, I generally fire up the debugger to get better insight into the state of the program and how that state evolves during execution.\\My partner and I were fortunate in that when we would submit to the bot we would generally get proficient on the first attempt. However, one exception to this was C2. My partner is a PL person and was therefore an excellent fit for writing the query language/ processor. However, they weren't as thorough about writing tests as I was. So on the first submission to the atuobot for C2 there were issues. What I did to address this was write a series of tests that failed but #check reported as correct. My partner then merged the PR with these tests and used them to fix their implementation. (r63_DebugIssue)

Use debugger to find error, write tests if needed, step through to see problem. (r64_DebugIssue)

I debug using print statements then the debugger to find and fix the issue. If I didn't find it, google. If not, ChatGPT. Then the office hours. (r65_DebugIssue)

first, I go through the spec for what went wrong and compare it to the code, making sure what I expect from the code follows the specification. If that doesn't work, then I try making new tests to pinpoint the issue. (r66_DebugIssue)

try running tests, read where it might fail and try to diagnose problem, if don't know from looking try looking up docs for proper usage of functions, chatGPT, or other tools like copilot (r67_DebugIssue)

We wrote more tests to pinpoint the issue, sometimes asking for help once we find it and get stuck. (r68_DebugIssue)

I wrote more tests to figure out the bug, and also reviewed the functionality of my code, specifically in areas where the smoke tests were failing (r70_DebugIssue)

Normally, we just tried to read over the code and the specs we added, then added a bunch of console.log(JSON.stringify(...)) statements to compare the results at each line of execution to the expected results. We also made heavy use of string comparison and JSON comparison websites to do this. (r69_DebugIssue)

Read the feedback, make relevant test case, run code (r71_DebugIssue)

In general, we would first add more tests to verify the exact behaviour that was causing the issue. Then once we can consistently reproduce the issue, we would use a combination of print statements and the debugger to identify the exact cause and resolve the issue. If these methods did not work, then we would ask for help from a TA. (r72_DebugIssue)

'- Follow the error message and go over the problematic code (if applicable)\- If unable to solve, try online resources and look for similar issues (Piazza, stack overflow, ChatGPT)\- If issue is unknown, go to office hours and try to debug (r73_DebugIssue)

I usually would reread the specification as I found that to often be a source of our confusion and mistakes. Typically my workflow would be read > try > code > run > repeat over many iterations. If nothing would succeed, we would potentially try to write additional tests and eventually ask for help. (r74_DebugIssue)

read specification, try something new, code new code, run the code. if all of this still doesn't work, ask on piazza or office hours. (r75_DebugIssue)

After receiving feedback from Autograder, I would first check the specs for the "need to focus next" part again to see if anything was missing. If I noticed anything, I would try to write test and then implement that. If I cannot figure out what was missing, I would check with my partner for their opinion on this and also ask on the forum if needed. (r76_DebugIssue)

Consult documentation provided, checkpoint information, local tests debugging. In no particular order but generally in the above order, with local test debugging at all points. (r77_DebugIssue)

When encountering an issue, defect, or feedback, I tried to resolve the issue by looking at what tests were failing if any to determine the part of the code causing the issue, going to the section of the code those tests dealt with, running the code through the debugger, and debugging with print statements. I also looked back at the part of the specification that dealt with the part of code to make sure I hadn't misinterpreted something. If I still couldn't resolve it, then I reached out for help through Piazza or Office Hours. (r78_DebugIssue)

When I ran into issues, a lot of debugging and pair-programming was done to identify the issues at hand and try to resolve them. I would see if the auto-test/bot would provide any further information, otherwise, I would start reading errors and console logging issues where I might be running into the issue. From there, it's discussing solutions and trying to adapt code to fix these problems. (r79_DebugIssue)

Although I didn't encounter many issues, I generally was able to find issues by rereading the spec and relevant parts of my code, and writing targeted tests. (r80_DebugIssue)

If, for example, a test was failing, I would step through the code to see the object values as the test input moved through my code. This usually helped me see where in the code I was expecting it to do one thing, but it was actually doing another. In the case my code was doing what I was expecting it to do, I double checked that the test was written correctly. Generally, my issues were due to not anticipating all edge cases appropriately, or misinterpreting the desired behavior in the spec. (r81_DebugIssue)

Consulted a TA. Made more unit tests to catch edge cases. reran the 310 bot after we felt like we made solid progress (r82_DebugIssue)

Read specifications, code solutions, try tests(repeat until i am clueless) and then ask (piazza, friend, and partner) (r83_DebugIssue)

I mainly used unit tests to guide bug resolution. I would use the bug to narrow down what function/ code caused the bug, then use console logs to analyze the code behaviour. When all tests were passing and the behaviour seemed correct I would consult a TA. (r84_DebugIssue)

I would use the feedback from the bot and read through my code. If that didn't work, I would write better tests. If that didn't work, I'd ask a TA. (r85_DebugIssue)

I run the code and see the error messages that it gives me if there is an error or look at the feedback that the autograder gave. For feedback, I will typically go back to that portion of the code and review what I have written, making any changes that I think would improve it based on the feedback. For error messages, if it is obvious---such as syntax errors---I will fix it, and otherwise I will usually google the error message and check StackOverflow to assist in solving the error. (r86_DebugIssue)

Try finding solution online then go Ask Ta. (r87_DebugIssue)

I read the stack trace of the error so see which file and line the error came from. I also used the debugger of VScode to try and catch error. (r88_DebugIssue)

Almost every time I immediately ran to the debugger. Set a breakpoint around where it seemed the issue was occuring, and looked at the state of the program at that point. From there I would read the code and try to reason about what went wrong. (r89_DebugIssue)

I first put console logs, then debugged it based off the prompt and then went to office hours. (r90_DebugIssue)

I try to redo it on my own first, then consult OH if needed. (r91_DebugIssue)

Most of the issues faced by the bot were missing tests, so I mostly wrote extensive tests. So I would say, try, code, run, read, ask. (r93_DebugIssue)

First I would run through the code with the debugger. If that didn't identify the problem, I would look at the spec again and see if I missed something. Then I would try writing a new test. If that didn't work, I would ask my partner to take a look. (r92_DebugIssue)

I would first try to resolve the issue by making sure all my local unit and integration tests passed. More often than not the issue was not caught by my tests, so I would have to use the clues from the autograder to come up with more tests, and then if those new tests failed I would attempt to fix the code to get them to pass (r94_DebugIssue)

Depending on the type of issue I would do different things. If it was something 310 specific I would check piazza/spec/my partner, however if it was something to do with typescript/some code thing I would google. (r95_DebugIssue)

I mostly referred to the suggestion provided by the autograder on what part I needed to work on. Then, I would review my test suite and introduce a more robust code system. (r96_DebugIssue)

If I have an error, I usually try asking ChatGPT if it knows the possible source of error, or GitHub copilot. If it's a failing test, then I usually use print statements at different points to see where the issue starts, then I examine that section to make sure my code is correct. (r97_DebugIssue)

read specs, try print out some debut statements, modify the code, If still cannot fix it, ask groupmate and TA (r98_DebugIssue)

1) Read feedback from autograder or test failure\2) Tried to make changes in accordance to the feedback\3) If the change was obvious, went ahead and changed code\4) if not, asked ChatGPT and/or googled the problem (r99_DebugIssue)

When encountering an issue, the first thing I would do is look for and address any error messages. If no error messages output, I put print statements to try and determine where the issue is occurring. (r100_DebugIssue)

I would start reading the feedback, understand the potential problems of my code. Then, I will try to search on the google and find the relative resources and try different methods. If I cannot solve the problem after trying it several times, I would ask the ChatGBT or go to the office hour to solve this particular problems. (r101_DebugIssue)

If we did not get Proficient after submitting to autotest, attempted to find bugs or edge cases that were missed in our implementation based on the feedback from the autograder. We were able to fix all the problems we had in this way, and they were mostly minor things that we initially thought might be a problem, but decided to grade just in case to see (r102_DebugIssue)

Autotest failures are generally not very helpful as a lot of useful things are hidden. Local test failing indicates an issue - I will use breakpoints to find where something is not as expected, then pinpoint the issue and fix it. (r103_DebugIssue)

While encountering an issues, I tended to go back to the specification and see if I missed something which was usually the case. The order I did the actions mentioned above was code, try, run, read and ask. For feedbacks, since I had already done the order mentioned before I used to ask first and then repeat the cycle prior. (r104_DebugIssue)

Rereading specifications, Office hours, etc. (r105_DebugIssue)

Firstly we would read the feedback message, find the line number/function it occurred for, and read through our implementation to determine what the problem could be. We would then rerun our code with changes until the issue had been fixed. (r106_DebugIssue)

Sometimes the autograder would show a specific area to improve on, and then if I didn't have a unit test/e2e test for that case already, I would try to make one, and then step through the code in debug mode to find the issue. (r107_DebugIssue)

check piazza, create multiple solutions for a problem (r110_DebugIssue)

Run the test cases and read the error that is happening, then use the debugger to step through the code and figure out what going on. Or use the rubber duck method and explain your code to something. (r108_DebugIssue)

If I receive test failures from test cases or missing scores from the GitHub bot, I will review the relevant code, attempt to "console.log" the result, and identify the problem. If it is difficult to identify, I will consult Piazza or review the specification to verify if I misunderstood the problem initially. (r109_DebugIssue)

Add tests to catch the issue, look for and fix the bug (r111_DebugIssue)

I read the specs again to see if I missed anything like the error as string type for the most recent checkpoint, then try to code again. I ask if it's a problem I have been stuck on for many days. (r112_DebugIssue)

When I run into an error I see what the error is about first, then navigate to the section of code that I think is the cause. If its something I dont know then I search it up (r113_DebugIssue)

'- read spec\- re-run with more tests\- use chat-gpt and ask partner for help (r114_DebugIssue)

'- step through with debugger\- add console outputs \- google (StackOverflow)\- ask my partner\- chatGPT if I get really stuck (r115_DebugIssue)

Read error message (local or remote from autotest) -> console.log as needed -> try to fix -> ask partner / TA -> repeat (r116_DebugIssue)

I would read then code then run and if those don’t work I would ask a TA. (r117_DebugIssue)

I would first print to console to figure out exactly where the issue was coming from, then try to fix the problem if I knew how, searched for a solution on google otherwise (r118_DebugIssue)

I tried debugging using the built in debugger in IntelliJ, and checked the values of objects etc. at each step (r119_DebugIssue)

I tried resolving issues using logging and using the debugger to determine the cause of the issue. Then, I would ask my teammate or search online if I couldn't figure it out myself. (r120_DebugIssue)

Issues were identified through tests, then code was debugged step by step to find the cause of the issue. (r121_DebugIssue)

When autotest said my code was wrong, I read the spec and my tests, and my implementation\When my own tests failed, I read my tests and then my implementation, and then the spec to figure out which was right if there was an immediate discrepancy I could see. (r122_DebugIssue)

We first tried to review the code individually and check piazza to see if any other students were having the same problem. Then we would step through the issue using the debugger, often in a group debug session. Finally we would go to TA office hours. (r123_DebugIssue)

Our primary debugging tool was the use of console logs to asses what parts of the code were not running as expected. At first at a macro level, and then at a micro level. (r124_DebugIssue)

comment out all tests except for single one of interest. add console.logs everywhere in code to understand exactly what is happening. consult google if stuck on bug. (r125_DebugIssue)

identify why it doesn't work and fix that problem (r126_DebugIssue)

read the test output, isolate the failing test case, generate more similar test cases to isolate the failing code, run code using debugging/print statements to find where variable values are not correct, once failing code lines are identified fix the code and retest. (r127_DebugIssue)

if there was an issue, it was usually from misunderstanding the spec, so we would reread the spec to find the problem (r128_DebugIssue)

Read the spec, docs. Tried to come up with new edge cases. (r129_DebugIssue)

I mainly used the debugger provided by intellij. If I still was unable to resolve an issue, I would either use the autograder hints or discuss with my partner. (r130_DebugIssue)

I will read and go though my code first. Then if is not work, I will try to find issue using online sources. If all those not work, I will go to OH for help. (r131_DebugIssue)

We first created a test case to quickly reproduce the issue. Then, depending on the nature of the issue, we would either consult Piazza if we believe someone else has had similar problems or add logs to the code and run the debugger. During this term, we used the IntelliJ debugger extensively to check the state of different classes while the program was running. (r132_DebugIssue)

Usually running the code with the debugger was how these issues were resolved, reading and stepping through problematic portions (r133_DebugIssue)

i use debugger and break point to find out the exact issue, and then check online documentation if its api misuse issue (r134_DebugIssue)

We first read the feedback/defect given to us by the autotest in and attempt to narrow down where the particular issue arose. We spend some time writing additional test to your test suite to hopefully catch this bug. If we still can't fix the issue we consult our TA. (r135_DebugIssue)

I will first write more test to find where caused the problem, Then go back to that part of code and try to debug it. If I've tried 2 days without solving it, I'll ask. Either Google it or go to office hour. (r137_DebugIssue)

Order was mainly debugging, stepping through individual lines, googling and consulting partner/ TAs (r138_DebugIssue)

I would normally re-run test again… After accepting the fact that there is, in fact, an issue with the code, for bugs, I would normally insert a few console logs to locate the bug. On the other hand, for performance issues, I would typically break down my local test suite even more to understand which section of the code is causing the performance issue. After locating the issue, I would normally first try to think through my logic again and consult the spec to see if I missed anything from the spec. If I have already pinpoint the exact issue/error message, then I would search on google and/or piazza to look for similar issues and how it can be resolved. (r139_DebugIssue)

Thankfully I never had significant setbacks during this project. Majority of my questions were answered by the resources provided in the spec. (r140_DebugIssue)

I ran tests and tried to figure out where in the code the bug was. Then I would try to fix it. (r141_DebugIssue)

reread spec, code something, if still not working post on piazza. (r142_DebugIssue)

Locally, I use the debugger and test-writing is to find bugs. If that doesn't work, I would push the code and run @check. Then, I would go to office hours if all else fails (r143_DebugIssue)

My general approach to solving issues/bugs (typically feedback from the bot, or bugs) is to:\1. read the error message or suggestion from the 310 bot\2. look at the specific lines of code that caused the issue or bug\3. read through the code itself to spot any easy to fix bugs\4. if there are no obvious bugs, I will try to review the specification page to see if there is anything I missed\5. I will ask chatgpt if it sees anything wrong with it\6. I will either fix the bug or repeat the above steps until it is fixed (r144_DebugIssue)

A lot of feedback we received was missing tests, and thus we would always look to see what tests we could possibly add. (r145_DebugIssue)

We would first read the code again, then the spec to see if we are missing something, then check Piazza then write more tests and if all of those didn't help then asked. (r146_DebugIssue)

First I would try to reproduce the bug, then I would formulate some possible fixes and see shat impacts they had on the code. As a last resort, I would ask a TA or prof during office hours. (r147_DebugIssue)

first try to isolate the problem by figuring out what path the code takes and what values are in the local variables. sometimes its easier with print statements sometimes its easier with debugger. Then I come up with a theory and test it. If there is absolutely no explanation for the behavior even after that, then I check piazza. (r148_DebugIssue)

console logs to see what was being returned and how data was being sent (r149_DebugIssue)

given feedback, we first took into consideration the hint on what to work on. from there on, we checked our code again, reread the spec, wrote more local tests, and went to office hours in that relative order. (r150_DebugIssue)

First read feedback, then look in the code and identify which part might be responsible for the problem, then make some more internal tests and rewrite the code, then submit again, if nothing changed I usually went to the OH for help. (r151_DebugIssue)

Review the error logs and documentation for understanding the issue. Experiment with potential fixes in a controlled environment. Test the code to see if the issue is resolved. If not ask my project partner after. (r152_DebugIssue)

I would generally try to run and add print statement throughout my codebase to try to make an output. (r157_DebugIssue)

I used print statements to understand what was happening during the execution of my program, and then I would make slight changes and run it again to see if that changed anything. (r153_DebugIssue)

When encountering an issue (a bug for example) the first thing I would do is to try to log and see where / what the error is. Usually by doing this I am able to pin-point where the error is coming from and attempt to fix it. If this doesn't work then going on Piazza to search for the specific error is usually a good idea, or search online if the error is more rare. (r154_DebugIssue)

Debug with break points; debug with print statements; guess and check to find weak points in code (some parts seemed a bit sketchier and more prone to error than others); compare cases that work to cases that produce error to isolate cause; read online javascript and typescript documentation (like on mozilla, and typescript websites) or specific library documentation; ask partner (r155_DebugIssue)

Typically I wouldn't submit to autograder until I was sure all my tests were passing locally. Then I would run #c2 (or whatever checkpoint number it is) to get immediate feedback and see my marks. I would look at the smoke test results to see which areas need more work to and also read the additional feedback at the bottom. Sometimes the feedback at the bottom wasn't that helpful.. For example, it usually only tell me what I already knew from the smoke test (eg. smoke test gives me X on count, feedback says we suggest you work on count). I worked pretty systematically based on the smoke test, and if I couldn't find any bugs I would then ask the TAs for guidance. (r156_DebugIssue)

When encountering an issue or defect, I first tried to work with my partner to resolve the issue in general. If we both cannot figure out the issue, then I would look at piazza to see if anybody had the same issues as I did. If not then I would ask my TA during the scrum sessions (r158_DebugIssue)

try , read, code run ask (r160_DebugIssue)

Read: Where did the issue take place and what type is it. Try: Add more tests to see if some of them fails. Code: When the issue is identified, fix the corresponding piece of code. (r159_DebugIssue)

We will read our code again, and find out the function that has not passed. Then we will write more detailed local tests to help find the bugs. Then we will fix/improve our program, and run local tests again. (r161_DebugIssue)

Isolate and test the code separately. Start from the order of the code. (r162_DebugIssue)

I used an online typescript compiler to check what was wrong. Then, I created and ran tests. Then, I went to office hours to ask for help. (r163_DebugIssue)

I followed the traditional way: first read the error reports (either from auto grader or console), then debug the erroneous code if feasible. If the cause of the errors was spotted, I would propose a fix, possibly add a relevant test, and ensure the test passes before submitting the changes. If I could not identify the problem, I would seek help online or from my teammate. (r164_DebugIssue)

Identify: I first identified the root cause of the issue, whether it was a bug, a feature request, or feedback.\Research: I then researched relevant documentation, articles, or Stack Overflow posts to understand similar issues and potential solutions.\Trial and Error: Next, I attempted different solutions, either by tweaking existing code or writing new code to address the issue. \Testing: After implementing a potential solution, I thoroughly tested the code to ensure that it resolved the issue without introducing new bugs. \Consultation: If I was still stuck, I sought help from peers, mentors, or online communities to brainstorm solutions.\Iterate: Finally, I iterated on the solution based on feedback until the issue was fully resolved. (r165_DebugIssue)

I search online for relevant errors, also ChatGPT for possible solutions. (r166_DebugIssue)

Look at the error message, the code, then run the GitHub bot (r167_DebugIssue)

I didn't encounter many issues in the code. Development for each checkpoint (besides c0) usually happens under 3 hours. (r168_DebugIssue)

read the error message / issue feedback.\re-read specification to see if i missed anything \try writing tests to isolate potential problems\run new tests (and old tests for regression checks)\ask TA or check piazza if I'm still stuck (r169_DebugIssue)

I try to read through and think through the logic in my code to see what edge cases I may have missed, while also referring to the spec. I also go to Piazza and see if anyone else is experiencing a similar problem and what steps they may have to taken to resolve it. If I really can't debug the issue, I go to office hours to get help from the TAs. (r170_DebugIssue)

We made more unit tests, often designed to test the minutest details. We also checked the test environment to ensure each test was set up properly. We also used breakpoints to investigate which part of our code is returning incorrect results or generating an error. (r171_DebugIssue)

I would first look at the code related with hints given by auto test, and then search piazza or ask TA for help. (r172_DebugIssue)

We would try to create local tests and trace the code to find errors. It seldom happened that something from the autotest bot feedback wasn't already something we knew about or couldn't find in a relatively short time. (r173_DebugIssue)

When encountering an issue (like unexpected behaviour or weird error) I will first search up on Google or copy paste the error message to chatGPT and ask what could possibly be the problem. During front-end development the main source of errors is the import of libraries, using chatGPT and read documentations is extremely helpful. In previous checkpoint, when all the local test suite passed but the autobot suggest other way, I will head to the office hours and ask for help. (r174_DebugIssue)

I would write more unit tests to see which particular area of code that I am missing. If all my local tests are passing, I'll try to do #check and #checkpoint. Otherwise, I'll use the debugger to identify which line of code that I am wrong at (if the terminal does not return any error message) and also write more console.log statements all over my code to find out which function is returning an error. (r175_DebugIssue)

'-> re-check the test with #check to ensure it is correct\-> re-run the code (in the case of non-deterministic behaviour)\-> re-read my function code to ensure there is no silly mistake\-> Ask partner/piazza (r176_DebugIssue)

I re-read the spec to see if i missed anything. I looked through my tests to fix errors. I looked at my implementation of the methods and if possible ran a debugger. If I still had issues I'd ask my TA. (r177_DebugIssue)

in general, I look things up on Piazza and Google. I also use copilot a lot to fix bugs. (r178_DebugIssue)

Google, read spec, Google, ask project partner, Google more, office hours. (r179_DebugIssue)

In order: I tried to debug and clear issues, read the specs more, read piazza, asked my partner for help, asked the TA for help. (r180_DebugIssue)

intellij debugger, chatgpt, stackoverflow (r181_DebugIssue)

We read code, ran test suites, checked if stack overflow or some other online website had a potential solution, asked chatgpt to debug, run the autograder, and asked a TA for help. It was roughly in the same order listed from first to last. (r182_DebugIssue)

autobot feedback is not quite useful, since it is often too vague, but at least it points out where we code did not pass their tests. We would write more tests to see if our codes behave as expected (r183_DebugIssue)

I first thought of reasons that a test was failing or issue was happening then I would check and see if a test case was already covering my idea. If it was I would try and think of another one, if not then I would write a test case for it and check. If it failed I would then step through specific sections of my code to try and find what was happening and why then I would fix it. (r184_DebugIssue)

In order: read the error, understand the difference with expected vs. actual result, try to find what line(s) are causing it, iterate with making changes and running test cases that fail. (r185_DebugIssue)

I looked into which tests were failing and tried to figure out missing test cases in my own test suite. (r186_DebugIssue)

Google the issue. Check Piazza. Confer with partner. Try out some stuff on a piece of paper. Confer with an AI tool. Ask a human like a TA. (r187_DebugIssue)

Still don't know what specifically this question asks for (just like the quizzes and sometimes the content of the course, and even the instructors in class because it looked as someone forced them to talk about a topic for 1.5 hrs and they were trying their best - I don't blame them, the content is not that 'easy' to 'teach'; they did their best). I will still try my best to answer.\\I think I just used Piazza, however that wasn't really that helpful. For c1, the TA's helped a lot - and I mean it. They were really helpful, especially regarding using Promise.all(promisesArray) as the ESLint wouldn't allow nested for loops and so they helped with that and generally other questions like about persistence and caching etc... Other than that, I didn't really need the TA's or anyone/anything else's help from the teaching team. Although when there were code errors, I did consult the internet (StackOverflow and BingAI/ChatGPT) to understand the root of the problems. (r188_DebugIssue)

run the source code, read the feedback from Autobot also the feedback from local run, try to debug or set up a break point to figure out where is the bug, ask the Autobot again (r192_DebugIssue)

I tried to make sure that my code had lots of comments so I could pin point where it was failing\I would look at the type of error I was getting\If I didn't know the error I would use google or chatGPT to help me figure out what it meant.\If I still couldn't figure out what was wrong I would turn to stack overflow, google of chatGPT \if that didn't work I would go to my partner or a TA (r189_DebugIssue)

Read the spec again, add tests until I find a failing test case, fix the code so that the test case passes (r190_DebugIssue)

read, refractor, run, and if there are still issues, I will ask for help (r191_DebugIssue)

1. read the documentation 2. create tests 3. ask chat gpt (r193_DebugIssue)

I usually re-read the code slowly and made sure every line was written with my expected results. This would usually find any bugs. (r194_DebugIssue)

We reviewed the spec, wrote test cases, and narrowed down the cause of the issue until we found it. (r195_DebugIssue)

I would first check the logic of my code to see if they make sense in general, and then google search on the error message, which would usually direct me to StackOverflow. If that doesn't solve my problem, I will then use ChatGPT to see if there's any insight. (r196_DebugIssue)

I would first determine where the issue was. If the autotest feedback recommended an area for development, or that one of my tests passed the autotest but failed locally, I can determine that that area needed work on. \\If a test was failing, I would comapare the expected result with the actual result and try to debug the problem. I debug using the console or with the debugger. (r197_DebugIssue)

I would add test cases, add debug outputs, and change the code so it produces the correct behavior. (r201_DebugIssue)

Read the issue. Write unit tests to find where exactly the issue is. Fix the code that causes the issue. Rerun the whole thing to see if there are any other issues. (r199_DebugIssue)

When there was an issue, which was usually not getting full marks on the auto-feedback, I went back to my code related to the issue and ran tests again to make sure that there were no unknown issues. Then, I use the debugger to ensure that each part works and that all of the parts work together as intended. If I can not find any issues, then I assumed that the issue was related to a part that I did not work on and would tell the teammate who worked on that part to check. (r200_DebugIssue)

I tried manual tracing through the code. If this did not work, I tried using the debugger function. As a last resort, I used a chat bot to diagnose the issue. (r202_DebugIssue)

I usually used the debugger/print statements to find where the issue occurred and then read the code to see where the problems may lie. Then once I felt like I had a working fix I'd rerun the code to see if it now works (r203_DebugIssue)

Read, code, try, run, ask (r204_DebugIssue)

Usually I run it again to make sure what I thought was making a mistake was indeed wrong. Then I reread the code, add print statements and run again. Repeat until complete. (r205_DebugIssue)

When I get feedback, I first try to identify which test the issue relates too. Often the test is missing so I will try to create a test to emulate the issue/ defect. Once I have created the relevant test which is failing, I then go into the code to try to fix this. If I cannot recreate the detected issue in a test, I will ask for help. Usually this is the hardest part for me. Once the tests are accurate, I can code and run using a lot of console.log's to debug until success. (r206_DebugIssue)

I read any error messages. If I could not understand, I'd ask a TA (r207_DebugIssue)

I usually reread my code and keep trying for 3 hours. If 3 hours still can't solve, go to the office hour. (r208_DebugIssue)

Print statement debugging, used the debugger, asked the TA (r209_DebugIssue)

I'd add more tests first to isolate the issue. If that doesn't work, I'd consult piazza or office hours for help. (r210_DebugIssue)

I read spec first, then try change the code, after run the debug I try to fix the code itself, most time this works, when it not I ask TA (r211_DebugIssue)

Firstly run tests, try to debug, cos ult with partner, ask TA for help. (r212_DebugIssue)

First I read over the code and make sure I did not miss anything which can be very frustrating. Then I start printing statements at each stage and also print variables I am using to make sure everything is correct. I felt like this was the most useful and used strategy during this project. I used the debugger sometimes to go into more depth, and piazza was always helpful. (r213_DebugIssue)

Read where the error occurred, look through the code to see how that happened, use the debugger in the worst case (r214_DebugIssue)

write more local tests (r215_DebugIssue)

I first try to read over where things are failing and check the debugger to figure out where in my code things were going poorly. \Second I would google the issue and shop around for solutions and return to step 1 to test things. If I get too tired I got to step 3. \Third I would check Piazza (if relevant and more general)\Fourth I would ask ChatGPT why things aren't working.\Fifth I would ask a TA or groupmate.\\I usually find my solution by the second stage. (r216_DebugIssue)

google. try on online complier or Jupyter notebook. (r217_DebugIssue)

Met with the team mates, and researched for a while.. the TA helped a lot too (r218_DebugIssue)

reread the specification, then run and explain every line in the concerned area, use the debugger, and then finally ask for help. (r219_DebugIssue)

When we encounter issues and bugs, we usually try to get relevant information from the webpage of the projects and checkpoints. We also read manuals of different libraries and tools we got. And mostly, we made lots of changes to the code and test. If problems are not solved, we sometimes ask the TA during the lab or office hours. With their feedback, we did fix the bugs much faster in general. (r220_DebugIssue)

Actually no. The feedback for the 310bot is misleading. Such as in our C2, we got full in invalid query, but missing points in others part. We spent few days to debug the part that we missed the point. But actually the part we need to improve is invalid query. (r221_DebugIssue)

When encountered an issue, I first check over my code according to the feedback and try to reiterate it and write unit tests for it to see if it fixes the issue. If still no clue, I will head to OH. (r222_DebugIssue)

Read/research what the defect/feedback means, try fixing the code in whatever way I thought was best, and then request feedback again. (r223_DebugIssue)

Usually I ran it first. Then I went to debugging it and reading it at the same time. As a last resort I asked for help. (r224_DebugIssue)

look at piazza, review local tests to make sure all cases are covered and carefully analyze the code and track the execution to debug where the issue is. (r225_DebugIssue)

I put in print statements in places I think might cause issues. Sometimes of the values I think might cause issues. Then divide and conquer. (r227_DebugIssue)

I read the error, I go to the line in the code giving the error and think about what could be causing it. I print all variables I think could be an issue and then look at any that have an u expected value. I trace the code back to see why it has that value and fix the issue. I try this for several variables until I find the error (r226_DebugIssue)

We would pass it to copilot to check for anything obvious like mixed up variables or typos. If it was a deeper issue my partner and I would call and work backwards from where we think the bug start and everything that might be involved with it (r228_DebugIssue)

Read Instructions, Try More Tests, Look though code (r229_DebugIssue)

If it's an error I go straight to stackoverflow, but if it's an assertion error i'd try adding print statements between crucial steps first to see where the system is working in an unexpected way. (r230_DebugIssue)

There were more than several occasions where the code did not work as expected after completing chunks of code at a time. When faced with issues, our first solution attempt would be to debug the code. Sometimes just looking at the code would be enough to find an error. If not we used console.log statements to locate the error. If the error was obvious we would fix it, and if not, our next resort would be Piazza. A lot of the times students were facing the same error, and the instructor's or other students' comments would come in handy. If that does not work either, then both partners would start working on different versions of the same code (ie to implement the same feature that is not working) to try and give it a fresh start with two minds working on it. Our last resort, if nothing works would be to either attend office hours and get the TA's help or create an issue on GitHub as suggested by our TA. One of these solutions worked for all issues we encountered during the project, expect caching. (r231_DebugIssue)

Read -> Run -> Code -> Run (r232_DebugIssue)

First, I looked at the test case (for example) that was failing. I would look at the expected and actual outputs to gauge what part of the code this issue could be in. I would also run similar tests in the reference UI to understand what the expected behavior was supposed to be. I'd read through the sped again to see if I missed a portion or misunderstood something. If I couldn't figure it out then, I'd read through Piazza posts or ask a question myself to see what I was missing. (r233_DebugIssue)

Read spec again\Go on Piazza\Code more (r234_DebugIssue)

tried to read the error message, then figure out how that related back to my code. if i couldn't figure it out I asked a TA for assistance (r235_DebugIssue)

First, we tried to write some more tests to narrow the issue. We also tried to use the stepper to go through the code and see where the error was occurring, and then if we couldn't figure it out then we went to office hours. (r236_DebugIssue)

after encoountering an issue I would try and see mannual if any desired output is not matching. And then i would read the specification to look for if I am missing something. (r237_DebugIssue)

Read and understand error feedback, try to locate the general locations where error possibly occurred, reason through the execution flow, and test code separately (if needed) to check execution against expectation. (r238_DebugIssue)

I inserted print statements before and after potential failure points in my code to try to isolate where errors are occurring. If there are seemingly no errors in my code, I read the project specifications or Piazza posts. (r239_DebugIssue)

Return to the specifications, see if any nuances/ details were missed, write the appropriate tests, return to developing code until tests pass, try again and repeat if needed. (r244_DebugIssue)

Debug and go to office hours (r245_DebugIssue)

When I an encounter an issue, I generally provide my code to an AI chatbot like GPT along with a detailed description of my issue, and see if it can give me ideas of what's wrong. (r240_DebugIssue)

start by trying the code by myself, maybe search up questions on stack exchange or research the topic, talk to my partner (r241_DebugIssue)

Certainly I would never ask a generative AI for help. Nope. I'd never do that. (r242_DebugIssue)

When facing an issue, I typically start by thoroughly understanding the problem and its context. Then, I research relevant issues online to gather insights and potential solutions. If needed, I experiment with different approaches, test hypotheses, and debug code. Finally, do collaboration with my team member. (r243_DebugIssue)

First I would read the project specification pages to see if I could find a solution that I might have missed the first time reading through it. If I couldn't find anything and the issue was something related to libraries we used, I would try googling library documentation and reading other people's similar issues online (stack overflow). Sometimes the issues felt more specific to the 310 project, so I would look through Piazza to see if someone else ran into the same problem. If I still couldn't resolve the issue, I would write a Piazza question to see if other students or instructors could provide hints or assistance. If it was an issue with not understanding the implementations of my partner, I would talk to them about it and see if I could get more clarification or to make sure we were on the same page. (r246_DebugIssue)

I would attempt to debug it myself, but if I still had issues after an hour, I'd ask my partner and check piazza. Some problems just required me to create more specific tests and experiment with many, many console.log statements. (r247_DebugIssue)

run local test, code, then ask for help. (r248_DebugIssue)

Discuss with my partner first, then read the code and explain what it does. From there we'd come up with a solution and resolve (r254_DebugIssue)

I first look at the code to determine whether they make logical sense. If there is an error message, I would search up the error message in Google or ChatGPT. I would also seek the help from TAs. (r249_DebugIssue)

Usually used the debugger to try to figure out where the issue is. (r250_DebugIssue)

when encoumtering an issue, I oftne refered to looking up an error I encoutnered or using various debugging techniques to fix those issues. (r251_DebugIssue)

read -> code -> run(go back to first if things don't work well), in that order (r252_DebugIssue)

if a test would fail i would use glass box testing looking through the source code for any obvious errors. This would help as I printed the output after multiple functions to see if they were behaving correctly. If there was another error then i would use a debugging tool and step through the code. (r253_DebugIssue)

I'd first try whatever I can think of that might fix the issue. If that doesn't work then I'd start looking online for people who've faced the same issue and have potential solutions. (r255_DebugIssue)

Used local tests to identify issues. Reread spec and added more tests if local tests did not match autotest. Used debugging tools to find out why local tests are failing. (r256_DebugIssue)

'- pay attention to autotest feedback\- identify which tests were failing (both smoke tests and locally)\- re-examine and revise code related to the failing test(s) (r257_DebugIssue)

I'd check tool or code documentation, then try to narrow down on issues with more specific diagnostic tests and debugging. After doing this, I'd post on Piazza or email a TA. (r259_DebugIssue)

Assuming this issue comes from running the autograder (#c1, #c2, or #c3), I first checked the autograder report to see which parts supposedly had issues. Then, I looked at the local tests related to the area of interest, and made sure the covered every case. I would also check this with the specifications on the CPSC 310 website. Afterwards, I would try to look through the code and think about possible cases again, because it might be easier to see exactly what I missed there. (r258_DebugIssue)

most of the time I try to code some console.log() to see where is the issue and then I run the test , if I still havent figure out, I try to read the spec and ask a ta (r260_DebugIssue)

Read -> write more tests -> run -> repeat the process (r261_DebugIssue)

I usually would try to use console.log statements to find the bug, then I might consult piazza to see if someone had the same bug. If not, I would reference any similar problems on Stack Overflow and sometimes ChatGPT but most often I would try to use the online forums of others experiences. (r262_DebugIssue)

1) Look at the code\2) Try debugging by just looking at code and adding console.log statements\3) Then if everything is fine, looking at spec (r263_DebugIssue)

If I get stuck after trying for a certain period of time, I would sit down with my partner to do some peer programming. If we are both stuck, I would ask the TA. (r264_DebugIssue)

I tried to add mock values to see where our code would encounter a problem. (r265_DebugIssue)

run code again, read documentation (r266_DebugIssue)

Read test results in an attempt to isolate the issue. Then we would narrow down where exactly the issue arose before attempting to actively make changes to your implementation in the code. (r267_DebugIssue)

run debugger, parse code, talk to partner about it, ask code completion tools (r268_DebugIssue)

first by reading the error, then if its not suffice, I often just copy paste it to stack overflow to find the solution. (r269_DebugIssue)

The first thing I would do is read through the code I believe is associated with the problem one more time, and see if I can spot anything obvious to fix. If not, I start logging data or messages throughout the code, and try to diagnose where things are going wrong. Oftentimes, this leads me to fixing the bug eventually. However, on the few times where I could not figure it out, or if I was just getting tired debugging, I would call my project partner for some "rubber ducking", where I essentially speak out loud my issues and my thought processes to her, while she either listens, or offers her thoughts. Usually, running through my work with someone else helps clear things up and expose the bug I had been looking for. And finally, I go to my lab TA for any help if all else fails. (r270_DebugIssue)

The autograder feedback was generally very good. Feedback provided from running #check was very to work with, and the checkpoint commands helped me figure out what tasks I should work on next. However, there are some edge cases that the autograder does not catch, which was frustrating to deal with at times. (r271_DebugIssue)

I first try to read through my code to identify obvious issues, if I couldn't find it just from looking, I would use the debugger to step through the code and then maybe write more tests to target the issues. (r272_DebugIssue)

I mainly used my editor's debugger and verbose print statements to visualize code execution and catch errors (along with test automation errors). (r273_DebugIssue)

I would carefully read my code first to see if I can find any obvious error, if not I start running tests and set breakpoints to debug the code. If the problem is still not resolved, I will let the Autobot to give me some feedback. Lastly, I will ask for my partner or TA's help. (r274_DebugIssue)

When encountering an issue, I first try to reread my code and understand whats wrong. If I still cannot figure it out, I will go on piazza to search and ask for an answer. (r275_DebugIssue)

If I fail to fix or diagnose the solution with the debugger, I will usually read documentation first, then consult stackoverflow (r276_DebugIssue)

I first wrote code for the initial problem. I pushed to github and waited for the autograder to give me feedback. If all cases passed, i knew i was done and i can start cleaning up/ optimizing my code. If there was an error, i saw the feedback, re-read the specification for the section that i got wrong, re check my implementation and do debugging from there. If i still could not fix it, i went to Piazza in search for help. (r277_DebugIssue)

I would try to look online or on piazza for a solution or if anyone else had the same problem, and if I was still stuck I would go to office hours. (r278_DebugIssue)

The autotest feedback would be enough to indicate where the issues are occuring sometimes, so I'd just debug where it mentions the issues are. However, when this doesn't work and its obvious the problem is elsewhere, I'd go to office hours. (r279_DebugIssue)

Basically the example order given. I'd try to look it up online, try it, implement it, run it, then if it still didn't work, I would hold off on working on it until I could ask during an office hour (r280_DebugIssue)

I listed out possible causes for the bug and created tests to try an pinpoint the issue. (r281_DebugIssue)

I mainly used the debugger. (r282_DebugIssue)

Using debugger (r283_DebugIssue)

Step one was print statements, step two was debugging, step three was spending a while reviewing the code and requirements, and step 4 was asking. (r284_DebugIssue)

When I encounter an issue, defect, or receive feedback on my code, I typically follow a systematic approach to resolve it efficiently. Here's a general order of the steps I take:\\Understand the Issue: First, I make sure to thoroughly understand the problem. This involves reading the error message, understanding the context where the issue occurred, and replicating the problem. If it's feedback, I ensure I understand the expectations and the specifics of what needs to be improved.\Research: Next, I search for similar issues or feedback encountered by others. This often involves reading documentation, searching through forums like Stack Overflow, and reviewing relevant sections in textbooks or online tutorials. This helps me gauge if the issue is common and how others have resolved it. (r285_DebugIssue)

Read the default first, try to understand what it means, then locate the source of default and start debugging by inserting console.logs around the control flow. (r286_DebugIssue)

I use the debugger to step through the code. If that doesn't work, I sometimes resort to asking chatbots for their diagnosis or stepping through it manually myself. (r288_DebugIssue)