How To Debug

Steps			Resources
1	**Understand the System**	**Talk It Out** ● Find someone (or something) who doesn't know what you're doing and explain what you want your code to do. ○ What should your code do? ○ How will it do it? ○ Will this get you the results you want? Why? ○ What is it doing now? ○ How might you fix it? **Pseudo Code** ● Write out the goals of the project ● Write small helper functions to break these goals into obtainable parts	**GeeksforGeeks** https://www.geeksforgeeks.org/ Breaks down data structures and algorithms in multiple different programming languages **visuAlgo** www.visualgo.net Shows data structures and algorithms using animation **W3schools** https://www.w3schools.com Document different features or functions within different programing languages and allows you to run small snippets of code **Tutorialspoint** www.tutorialspoint.com/tutorialslibrary.htm Resources that document different features or functions within different programing languages **Stack Overflow** https://stackoverflow.com/ The reddit of debugging. Post a question or look at old questions and answers
2	**Identify the Problem**	**Isolate** ● Monitor output--where does it go wrong? ● Evaluate singular functions or sections of code ● Check your math--is your order of operations correct? ● Use print statements to evaluate singular variables during execution. Does every variable do what you think it does? **Use a Debugger** ● Set a breakpoint near where you think the code is broken ● Identify any relevant variables and make sure they are being kept track of ● Step through the code line by line as needed and make sure that your variables are changing in the ways you expect.	
3	**Form a Solution**	**Try Things Out** ● Rewrite code--start with simply modifying single lines to make small improvements ● Read documentation to find discrepancies between what you thought something did and what it is doing	

CREATED BY: Garnet Droppo, Olivia Coleman, Lori Torres, & Mark Rogers.

Using a Debugger

Breakpoints

- This is where you want your code to stop while you are debugging
- Often denoted by a red circle on the left side of the code

Steps

- Each time you take a step the next line in your code is run and the variables are updated
- Sometimes the “current” line is the one that will be run the next time you take a step
- When you reach one of the following you can step through or step over
    - `for/foreach` loop
    - `while/do while` loop
    - `if/else/elif` (conditional logic)
    - Function call “`function_name();`”

Stepping Through

- By stepping through you enter the for/while loop, if statement, or function call and can make sure that the code inside is working properly.
- This is usually done by continuing to use the regular method for stepping through the code in a debugger.

Stepping Over

- This is useful if you have a function call, loop, or conditional logic that you have already tested and would take a long time to step through.
- This runs all the code in the function call, loop, or conditional logic as it would when the debugger isn’t running and stops on the first line not inside the “skipped” code.

Reading your Variables

- A lot of modern IDEs (Integrated Development Environments, or where you are writing the code) will have a built in debugger that will automatically keep track of variables. You can find this information usually in a column on the left or right side of the window when you start debugging and run your code.
- If you are using a terminal based debugger you might have to print and keep track of your variables more manually. Below is some resources for specific debuggers:
    - https://u.osu.edu/cstutorials/2018/09/28/how-to-debug-c-program-using-gdb-in-6-simple-steps/ (GNU Debugger or GDB for C code)
    - https://www.tutorialspoint.com/jdb/jdb_quick_guide.htm (Java Debugger or JDB for Java code)

CREATED BY: Garnet Droppo, Olivia Coleman, Lori Torres, & Mark Rogers.