Currently this is simply a list, I hope to categorize and organize these in the near future.

- Read the compiler/interpreter messages carefully .. they will often give you some idea about what is wrong., and where i.e., which line of source code might be causing trouble.

- The compiler/interpreter may be off in its guess as to where the error occurred. Generally, if it's not the line it indicates, look at the statement above it. The real error will never be below the line indicated.

- Always work from the top to bottom. Often (really quite frequently) fixing an error above will eliminate a slew of errors below.

- Talk to someone (or yourself). By trying to explain the problem, frequently you will discover it (because verbalizing it forces you to think about, and outline your steps)

- Can you reliably reproduce the problem? If so cut the code down to the smallest possible segment that shows the problem, doing this will often reveal the real problem.

- Learn to read the programming documentation available to you, man pages, API specs, etc. Get familiar with the index of your text book. Look for examples on-line.

- BEFORE you can do any useful coding, you have to SOLVE the problem. Do you have an algorithm (ie series of steps) that will accomplish your task? If so, write them out in pseudo code.

- Learn to use a debugger. While print statements are a true-and-tried method, and always will work in a pinch, time spent learning to use a debugger is time well spent. If you have extra time, find out if there is a profiling tool available. It may become helpful when you try to speed up your code. Careful not to obfuscate your code in order to gain 0.00000001% speed up.

If you have some methods/strategies/tricks that work well for you, please let me know.
esmail@franklin.edu