- **Always Have a Plan**
    - **Importance of Planning:** Like military operations, having a plan for coding is essential, even if the plan changes. Planning helps understand capabilities and organizes efforts.
    - **Setting Goals:** Break the problem into smaller goals to track progress and maintain motivation.
- **Restate the Problem**
    - **Reframe for Clarity:** Restating can make a problem clearer and easier. It helps ensure you understand the problem and can identify simpler ways to solve it.
    - **Communication:** Restating also helps in confirming your understanding with others, such as supervisors or teammates.
- **Divide the Problem**
    - **Simplify with Subdivision:** Breaking a problem into smaller parts often makes each part easier to solve than tackling the whole at once.
    - **Sequential Processing:** Handle parts sequentially to manage complexity.
- **Start with What You Know**
    - **Leverage Existing Knowledge:** Begin with familiar aspects of the problem to build momentum and confidence.
    - **Build from Familiar Ground:** Using known skills and techniques first can provide insights into solving the more challenging parts.
- **Reduce the Problem**
    - **Simplify by Modification:** Alter constraints or simplify the problem to a version you can solve, then apply those insights to the original problem.
    - **Pinpoint Difficulties:** Reduction helps identify specific areas where more help or knowledge is needed.
- **Look for Analogies**
    - **Identify Similar Problems:** Find similarities with previously solved problems to apply known solutions or techniques.
    - **Build a Knowledge Base:** As you solve more problems, recognizing analogies becomes easier, improving problem-solving speed and skill.
- **Experiment**
    - **Controlled Testing:** Hypothesize, test, and observe results to gain information and refine your understanding.
    - **Debugging and Learning:** Experimenting with code helps clarify behavior, especially when working with unfamiliar APIs or libraries.