Solve Before You Code

A programmer's guide to turning a problem into a working program

1. Understand the Problem

Before you touch the keyboard, step away from the computer. Read the problem carefully and more than once.
Ask yourself:
What is the problem really asking me to do?

Do this:

- Restate the problem in your own words.
- Make a sketch or diagram if it helps.
- Identify the requirements.
- Define what "solved" means.

Key takeaway:
Know the goal before you start building the solution.

2. Identify the Data

Every program works with information. Before coding, figure out what information you already have and what information the program still needs.

Ask yourself:

- What values are given?
- What values will the user need to provide?

Do this:

- List any given values.
- Choose clear variable names for those values.
- Identify any user inputs needed.
- Write the prompt the user will see for each input.
- Choose variable names for each input.

Key takeaway:
Good variable names help you think clearly.

3. Design the Logic

Now decide what the program must do with the data. This is where you plan the calculations, decisions, loops, and functions.

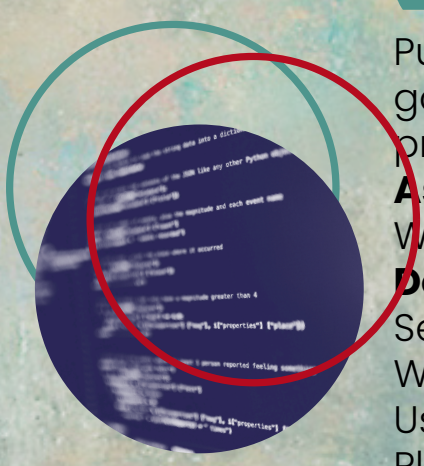
Ask yourself:
What steps does the program need to follow?

Do this:

- Write any formulas or calculations needed.
- Identify decisions the program must make.
- Plan any repeated steps that may need a loop.
- Decide whether functions are required.
- Name each function based on the task it performs.
- Identify any parameters each function needs.

Key takeaway:
Plan the thinking before writing the code.

4. Organize the Solution



Put the steps in the correct order before you begin coding. A good program follows a logical sequence from input to processing to output.

Ask yourself:
What needs to happen first, next, and last?

Do this:
Sequence all steps in the correct order.
Write pseudocode in plain English.
Use your chosen variable and function names.
Place the pseudocode as comments near the beginning of the program.

Key takeaway:
Pseudocode is the bridge between your plan and your program.

5. Code, Test, and Debug

Now turn your plan into code using the required programming language. Keep your pseudocode nearby and update it as needed.
Ask yourself:
Does the program meet every requirement?

Do this:
Write the code using your pseudocode as a guide.
Add comments to clarify confusing sections.
Run the program with different inputs.
Fix errors as you find them.
Check the requirements again.
Proofread spelling and output formatting.

Key takeaway:
A program is not finished until it works, meets the requirements, and is clear to the user.

Remember

Think first.
Plan clearly.
Code carefully.
Test everything.