Unit 2 "Study Guide"

The main point of Unit 2 was to get you thinking about elements of programming that make some programs "better" than other programs. These are things that you might explicitly teach, but you are just as likely to teach through continually showing your students what good code looks like I went through all of our discussions, readings, notes, etc. and came up with the following list of "elements"

Elements of good code (not complete):

- Meaningful Names
  - Variables
  - Functions
- Appropriate use of White space
  - Vertical and Horizontal
  - Consistent use of formatting/white space
- Appropriate use of functions
  - To provide abstraction and/or modularization
  - To "chunk" code into appropriately sized segments
- Appropriate use of comments
  - Like Goldilocks, not to many, not too few
  - Most important when the code "meaning" is not obvious or when there are variables being set that might need to be modified in other applications
- Appropriate use of "style"
  - When and how you indent
  - Naming style for variables and constants
- Using loops appropriately to reduce/eliminate duplicate code
- Not writing unnecessary code
  - Examples include
    - Not defining variables that aren't needed/used
    - Not creating conditionals that check for things no longer possible
    - Explicitly checking for "==True" when using Boolean variables
- Placing code inside loops when it really belongs outside of loops
  - The example of this is calculating the average=total/count INSIDE of the loop when it really belongs outside of the loop
- Eliminating "dead code"

Questions I could ask that ask you to use/apply this knowledge

1. Explain why this element makes code good/bad to read?
2. Explain (give an example?) how you could teach this element to students
3. Consider this code example. List two good things? List two bad things? Explain why you say that.
4. Consider these two examples. Which is "better" and why.