## Deeper

- What does it mean to say that code is maintainable?
  - *Given a program, it is relatively easy to understand its structure and find the module responsible for a function.*
  - *Given a module, it is relatively easy to understand what the module does and how each routine contributes to that.*
  - *Given a routine it is relatively easy to understand, what the variables are used for and how the code works.*
  - *Given a problem or enhancement, it is relatively easy to make the required changes without breaking the program.*
  - *Given a set of changes, it is easy to test and be confident that the program works correctly.*



## Deeper

- How can architecture make the code easier to understand?
  - *If the architecture is easily understood, we will understand much of what each component does before we even start looking at the code, and the code will just fill in details.*
  - *If the architecture is complex and subtle, we will not have a clear notion of what each component is supposed to do, and will be trying to figure it out as we study the code.*



## Deeper

- How can specifications make the code easier to understand?
  - *The code for a class or routine implements the specifications for that class or routine.*
  - *If the specifications are simple, clear, and complete, then we will have a very good understanding of what a routine does before we start reading it.*
  - *If the specifications are vague or confusing, we won't know what the code does until after we have studied it.*



## Deeper

- How can design make the code easier to understand?
  - *High cohesion and low inter-routine coupling mean that we can understand each routine in isolation.*
  - *If called functions are well abstracted and named, we will have a good sense of what they do even if we have not yet read them.*
  - *Simple algorithms are more easily understood, as are the implications of changes to them.*



## Deeper

- How can a standard order and format for the elements of a module improve its understandability?
  - *This is analogous to consistency of spatial placement and visual metaphors in graphical user interfaces. Readers will be able to recognize key elements of the module based on their location and appearance, without having to actually read and understand the code.*



## Deeper

- What is McConnell's Theorem of Formatting?
  - *Good visual layout shows the logical structure of the program.*

## Deeper

- Why is consistent indentation important?
  *Because it enables us to reliably infer structure from indentation.*
- What is the problem with excessive indentation?
  *If ¾ of the screen is indentation white-space we will find that almost every line needs a continuation, and this detracts from ease of reading.*



## Deeper

- How do parentheses and white space make expressions more easily understood?
  *Precedence rules are not designed for clarity, but to eliminate ambiguities. Decomposing an expression based solely on precedence rules is intellectual work.*
  *By breaking large and complex expressions into hierarchically structured compositions of easily recognized terms, we greatly simplify the mental parsing process.*



## Deeper

- Why is it important to have a standard indentation policy for continuation lines?
  *so that it is visually obvious that they are continuations of the previous line.*
  *so that it is visually obvious where the next statement begins.*



## Deeper

- **if** statements would seem to be "simpler" than conditional expressions. How can more subtle/complex expressions be more understandable?
  *(a) Because they involve far fewer characters (and lines).*
  *(b) Because they clearly indicate that we are choosing a value rather than altering our control flow – and this is a meaningful distinction.*



## Deeper

- When does creating intermediate variables simplify code, and when does it complicate it?
  *If they replace a more complex expression <u>and</u> represent an intuitively reasonable value, intermediate variables improve readability.*
  *If it is merely a piece of a more complex expression, that has no intrinsic meaning, introducing intermediate variables could actually make the expression harder to understand.*



## Deeper

- Why is it a poor idea to comment relatively straight forward code?
  *If the code is truly simple, and the variable and routine names have been well chosen, it should speak for itself. Additional comments would add clutter for little additional clarity.*
  *If the code isn't quite simple enough to speak for it self, simplifying it or making it more obvious would probably have a better pay-off than writing descriptive comments.*
  *The best way to ensure that code is <u>correct</u> is to make it <u>obvious</u>.*

## Deeper

- How do module, class, and routine pre-ambles contribute to readability?
  - *They are bold visual markers that quickly draw our eyes to the start of each class or routine.*
  - *They provide a brief description of the parameters, function, and return values, enabling us to better understand code that uses this method/routine.*
  - *They provide a synoptic overview of the class or routine that follows, giving us the background to better understand the code.*



## Deeper

- Why or when would we need to label an else statement?
  - *If there are deeply nested IF statements, it may not be immediately obvious which IF an ELSE corresponds to.*
- Why or when would we need to label a break target?
  - *If there are deeply nested blocks and the end of a block is not nearby, it may not be immediately obvious where the BREAK will take us to.*



## Deeper

- Why might it be important to include copyright or license information in a module?
  - *To protect intellectual property, or comply with terms of licensing agreements.*
- Why might it be important to include identity and version information in a module?
  - *So that we know what version of what module we are looking at.*



## Deeper

- Why would information for CAD tools embedded in comments?
  - *Because this information it is directly associated with parts of the code, but is not meant to be processed by the compiler.*
  - *Comments (usually with special syntax) will be ignored by the compiler, but recognized by CAD software.*



## Deeper

- How can coding standards affect productivity?
  - *They encourage layout and commenting practices that improve code readability, making structure and meanings more readily apparent.*
  - *They can reduce start-up time:*
    - *people will recognize familiar notational conventions*
    - *won't have to spend time learning new conventions*
  - *They can reduce wasted effort:*
    - *confusion resulting from misunderstood code*
    - *time spent reformatting code to meet personal tastes*
    - *eliminate aesthetics-based non-reuse*
  - *They can enable the building & use of automated tools*