Chapter 14: Organizing Straight-Line Code

Statements That Must Be in a Specific Order

- The easiest sequential statements to order are those in which the order counts
    - Data = readData()
    - Results = calculateSomething()
    - Print(results)
- The underlying concept is that of **dependencies**
- Guidelines for managing these are
    - Organize code so dependencies are obvious
    - Make initializers specific routines
    - Name routines so that dependencies are obvious
        - Remember, if the name is bad, so is the routine
    - Use routine parameters to make dependencies obvious
    - Document unclear dependencies with comments
    - Check for dependencies with assertions or error-handling code

Statements Whose Order Doesn't Matter

- May encounter cases in which it seems the order of a few statements or blocks of code doesn't matter at all
    - One statement doesn't depend on or logically flow into another
- BUT, ordering affects readability, performance and maintainability
- **Principle of Proximity = Keep related actions together**

Making Code Read from Top to Bottom

- General principle is make the program read from top to bottom rather than jumping around
- Keep print statements near operations that generate them
    - Instead of saving up print statements till very end
    - (Just one example)

Grouping Related Statements

- Put related statements together
    - Related because
        - Operate on same data
        - Perform similar tasks
        - Depend on each others being performed in order
- Good way to check this is to draw boxes around related statements
    - If done right, should be nested rectangles with none overlapping
    - If boxes overlap, reorganize code
- Once regrouped related statements, you may find theyre strongly related and have no meaningful relationship to the statements before or after them
    - In this case, refactor strongly related statements into their own routines