1. Organizational trust
   1. Valuable data
      1. Having access to it (and not losing it)
      2. Having competitors getting access to it
   2. Reputation
   3. Being able to adapt to future needs
   4. False data
2. Government Trust
   1. Risks
   2. Laws
3. Programming
   1. Selection
   2. Iterations
   3. Variables
   4. Functions
   5. Arrays (& ArrayLists)
   6. Recursion
   7. Big Ideas
      1. Abstraction
         1. Why?
            1. Save time (Coding/Debugging)
            2. Easier to understand
         2. Forms of abstraction
            1. Blocks

Arguments

* + - * 1. Variables
        2. Lists
      1. Techniques
         1. Identify areas

Duplication

Things too long

Get overwhelmed

* + - * 1. Find abstraction idea
        2. Implement the solution
    1. Design first
       1. What is it?
          1. Removing ideas from thought to physical concrete form
       2. Why do it?
          1. Prevent unwanted changes
          2. Remember ideas for later
          3. Fix problems early
          4. Communicate with others easier
          5. Make ideas clearer
    2. Recursion
       1. Components
          1. Base case(s)
          2. Make it smaller
          3. Combine solution
       2. Why and when?
          1. Simple and clean
          2. If problem lends itself to it
       3. Why not and when to avoid it
          1. Can be expensive in terms of time
          2. HARD TO UNDERSTAND
    3. Precision == good
       1. Computer has no common sense
       2. Computer does exactly what you tell it to do
       3. If you don’t understand exactly what you want, you can’t tell the computer 🡪 it won’t figure it out
    4. Can’t force it
       1. The computer only has certain tools
       2. Can’t force it to do something else (easily)
       3. Look for the perfect match 🡪 good enough ISN’T good enough
  1. What else is there?
     1. Quality debugging
     2. Quality testing
     3. Data structures
     4. Better abstraction and additional abstraction techniques