Design of Routines
It is okay to have small functions

- Self-documenting code
- Avoid duplication
  - eases maintenance
Cohesion

- **Strive for Functional Cohesion**
  - A routine that performs exactly one operation
  - Usually documentable with clear method name

- **High cohesion correlated with low fault rates / fix costs**

- **Fixing low cohesion?**
  - Break offending method into cohesive pieces
  - Call the pieces in turn
So how to choose a routine length?

- Let cohesion and complexity be your guide
  - Pull out highly cohesive routines
  - Limit the number of paths through the code
Parameters

- logical ordering (e.g., input-modify-output)
- consistency
- use all of the parameters
- limit number of parameters to ~7
- put status or error parameter last
- don't use input parameters as working parameters
Go Fish

Each player gets five cards. If you are dealt a four of a kind, or get four of a kind during game play, those cards are removed from your hand, and you get a point.

Moving clockwise, players take turns asking a specific player for a given rank of card. If someone asks you for a rank that you have, the cards are taken from your hand. If you do not have any cards of that rank, your opponent must “go fish”, taking one new card from the pile of cards.

When it’s your turn, select a player you think might have a needed card. Pick one card from your hand of the desired rank. If the player has the desired card, he or she must pass it over. If not, you must “go fish”. If you get the card you asked for, you get to go again.

If you run out of cards and there are still cards left, you get five free cards.

Play continues until all hands are empty and there are no more cards to draw from. The winner is the player with the most points at the end of the game.