

Top-down design

Example adapted from Zelle

As you start working on more complex programs, it helps to have guiding principles to help deal with what might seem to be a daunting task. One of the most useful principles is called **top-down design**.

I think this is best understood by working through an example.

Let's consider the scenario of making a program that simulates two players playing a number of games of the sport racquetball. I'm adapting this example from Zelle Chapter 9.

Basic rules of racquetball

- Game is similar to tennis and volleyball. Players hit a ball with a racquet back and forth in a court.
- To start the game, one of the players puts the ball into play—this is called **serving**.
- The players then alternate hitting the ball to keep it in play; this is a **rally**. The rally ends when one of the players fails to hit a legal shot.
- The player who misses the shot loses the rally.
 - If the loser is the player who served, then no points are won or lost: the other player now serves the ball.
 - If the server wins the rally, a point is awarded. Players can only score points if they are serving.
- The first player to reach 15 points wins the game.

Problem

The problem is: Let's say Player A plays Player B in a championship of n total games. *How many of the games would we expect Player A and Player B to win in the championship?*

This will of course depend on how good the players are at winning points. Let's say that Player A has a history of winning a point on their serve a proportion of time `probA` (number between 0 and 1). We also know that Player B has a history of winning a point on their serve a proportion of time `probB`.

Pseudocode

Simulating the game might seem daunting, but let's break down the problem into high level tasks that need to get done for us to write the program.

Let's write the steps as comments in `main` function.

1. We print an introduction to users, instructing them what the program does.
2. We ask the user to enter the probability that player A wins at a serve, the probability that player B wins at a serve, and the number of games in the championship.
3. Simulate n games of racquetball using `probA` and `probB`.
4. Output a written summary of the number of games won by each player.

- Writing out an outline of steps we need to complete in a program in English rather than exact code is called **pseudocode**.
- Programmers often write pseudo code in comments to plan out code to write in the future.
- I often use pseudocode in class examples to break down what tasks I need to follow when live coding.