Designing Loops with Invariants
Invariant

• An invariant: a property of the variables that is always true after an iteration of the loop
Computing $1 + 2 + 3 + 4 + \ldots + n$

$s = 0$
$i = 1$

while $i \leq n$:
    #invariant is true here
    $s += 1$
    $i += 1$
    #invariant is true here
#Invariant: $s = 1 + 2 + 3 + \ldots + (i-1)$
#When $i == (n+1)$, $s = (1 + 2 + 3 + \ldots + n)$
#After the while loop, $i$ does contains $(n+1)$, so $s$ is $(1 + 2 + 3 + \ldots + n)$
#(Why? $i$ stops being $\leq n$ when it becomes $(n+1)$)
Using Invariants

• Figuring out the stopping conditions for loops:
  • When should the loop stop? When \( i == (n+1) \), since at that point \( s \) contain the quantity that we’re looking for

• Figuring out how to update our variables:
  • How do we update \( i \) and \( s \) such that \( s = 1 + 2 + 3 + ... + (i-1) \)?

• More useful in more complicated scenarios
  • E.g., the repeated squaring example
  • Not that useful in simpler cases
More on Designing Loops

\[ s = 0 \]
\[ i = 1 \]

while \( i \leq n \):
    \[ s += 1 \]
    \[ i += 1 \]

# Very common pattern to accumulate the result in an accumulator variable, \( s \) in this case