Induction

→ showing that for any $n \geq 1$, there is a finite seq. of implications that starts w/ something known to be true and ultimately leads to showing $q(n)$ is true

→ specifically: 1.) begin justification by induction by showing $q(n)$ is true for $n=1$
2.) justify the inductive step is true for $n > k$

Example

→ Consider fibonacci function $F(n)$. We claim $F(n) < 2^n$.
→ base cases: $(n \leq 2)$. $F(1) = 1 < 2 = 2^1$ and $F(2) = 2 < 4 = 2^2$
→ induction: $(n > 2)$. Suppose our claim is true for all $n' < n$. Consider $F(n)$.
Since $n > 2$, $F(n) = F(n-2) + F(n-1)$.
Since both $(n-2)$ and $(n-1)$ are less than n, we apply inductive assumption to imply that $F(n) < 2^{n-2} + 2^{n-1}$, since
$2^{n-2} + 2^{n-1} < 2^{n-1} + 2^{n-1} = 2 \cdot 2^{n-1} = 2^n$

Loop Invariants

→ to prove statement L about a loop is correct, define L in terms of a series of smaller statements $L_0, L_1, \ldots, L_k$ where:
1.) initial claim (L_0) is true before the loop begins
2.) if L_{j-1} is true before iteration j, then L_j will be true after j
3.) final statement L_k implies desired statement (L) to be true

```
def find(S, val):   → return index j such that S[j] = val or -1 if nonexistent
  n = len(S)
  j = 0
  while j < n:
    if S[j] == val:
      return j
    j += 1
  return -1
```

claim at start:
→ L_j: val is not equal to any of the first j elements of S