Inductive reasoning

- How do we know that our recursive code will work?
- iterMul terminates because b is initially positive, and decrease by 1 each time around loop; thus must eventually become less than 1
- recurMul called with b = 1 has no recursive call and stops
- recurMul called with b > 1 makes a recursive call with a smaller version of b; must eventually reach call with b = 1

Mathematical induction

- To prove a statement indexed on integers is true for all values of n:
  - Prove it is true when n is smallest value (e.g. n = 0 or n = 1)
  - Then prove that if it is true for an arbitrary value of n, one can show that it must be true for n+1

Example

- 0 + 1 + 2 + 3 + … + n = (n(n+1))/2
- Proof
  - If n = 0, then LHS is 0 and RHS is 0*1/2 = 0, so true
  - Assume true for some k, then need to show that
    - 0 + 1 + 2 + … + k + (k+1) = ((k+1)(k+2))/2
    - LHS is k(k+1)/2 + (k+1) by assumption that property holds for problem of size k
    - This becomes, by algebra, ((k+1)(k+2))/2
  - Hence expression holds for all n >= 0

What does this have to do with code?

- Same logic applies

```
def recurMul(a, b):
    if b == 1:
        return a
    else:
        return a + recurMul(a, b-1)
```

- Base case, we can show that recurMul must return correct answer
- For recursive case, we can assume that recurMul correctly returns an answer for problems of size smaller than b, then by the addition step, it must also return a correct answer for problem of size b
- Thus by induction, code correctly returns answer