Chapter 15 - Lazy Evaluation

→ Expressions are evaluated using a technique called lazy evaluation

1. Avoids doing unecessary evaluation
2. Ensures termination whenever possible
3. Supports programming with infinite lists
4. Allows programs to be modular.

→ Benefits

Ex: square n = n * n

= square (1 + 2)
↳ Apply first
= square 3
= 3 * 3
= 9

= square (1+2)
↳ Apply square first
= (1+2) * (1+2)
= 3 * (1+2)
= 3 * 3
= 9

Any way of evaluating the expressions gives the same result provided it terminates

→ Number of reductions (reducible expressions a.k.a redex)
→ 2 ways innermost / outermost evaluation

Innermost
square (1+2)
= square 3 ①
= 3 * 3 ②
= 9 ③
3 steps

Outermost
square (1+2)
= (1+2) * (1+2) ①
= 3 * (1+2) ②
= 3 * 3 ③
= 9 ④
4 steps

→ Pointers to indicate sharing of arguments can reduce outermost evaluation

square (1+2)

= * ⟶ 1+2 ①

= * ⟶ 3 ②

= 9 ③

→ Lazy evaluation: Outermost evaluation with sharing of arguments