Local variables

- Set!
    - (set! <*name*> <*new-value*>)
    - Changes <name>
- Begin
    - (begin <*exp*$_1$> <*exp*$_2$> ... <*exp*$_k$>)
    - causes the expressions <*exp*$_1$> through <*exp*$_k$> to be evaluated in sequence and the value of the final expression <*exp*$_k$> to be returned as the value
- Example

```
(define balance 100)

(define (withdraw amount)
  (if (>= balance amount)
      (begin (set! balance (- balance amount))
             balance)
      "Insufficient funds"))
```

    - Set balance to a specified amount, then return balance
    - Or: use let and lambda to make one function
- Independent objects → local state variable
    - E.g. balance and counter
    - Changes as the user inputs different things
    - Only exists within the function
- Introducing assignment
    - Benefit: simplifies code
    - Cost: no simple model to interpret
- Use lambda when possible
- Setting counters is important
- Dispatch is the function used to put together all possible functions