Procedures

What Makes Them?

- Expressions
- Composites of expressions
- Abstraction
REPL

- Read-Evaluate-Print Loop
- Started with Lips Interpretive Environment, but several languages now support it
- Easy to evaluate expressions
- Helps to learn a language
- Helps to experiment and build code

Expressions

- Expressions are evaluated and produce a result
- A number is a primitive expression
- Combinations: You can use operators to create other expressions
  - $4 + 2$ [or in LISP (+ 4 2)]
Variables

- Identified by a name
- Has values
- Define vs. Assign

Procedures

- Composed of expressions
- Compound Procedures—identified by a name and represents an operation
- Applying a function—you evaluate a procedure by sending arguments for the parameters
Order of Evaluation

- Applicative Order
  - Operators and parameters evaluated before procedure is evaluated
- Normal Order
  - Operators and parameters are evaluated only after the procedure is fully expanded

Applicative Order

- def square(x) = x * x
- square(2 + 3) = square(5) = 5 * 5 = 25
- Efficient
- Can remove duplicate computations
Normal Order

- def square(x) = x * x
- square(2 + 3) = (2 + 3) * (2 + 3) = 5 * 5 = 25
- Result is same as applicative order
- Not as efficient
- However, lazy evaluation can come in handy at times
- May eliminate the need for some computations or may push it just in time

Decomposing Procedures

- It is easier if we decompose procedures into smaller procedures
- Easy to comprehend
- Easy to explain
- Easy to express and maintain
Finding square root

- Successive Refinement

\[
\text{sqrt} \quad \rightarrow \quad \text{sqrtSuccessive}
\]
\[
\text{goodEnough} \quad \text{improveGuess}
\]
\[
\text{Square} \quad \text{abs} \quad \text{avg}
\]

[Structure and Interpretation of Computer Programs by Abelson, Sussman, and Sussman]

Abstraction and Encapsulation

- Procedures should abstract and encapsulate the details
- They should tell you what they provide and hide the details of how they do it
- User of a procedure should not be forced to know the details of the implementation
- User should be conveniently use the procedure
Formal Parameters & Binding

- The names chosen for formal parameters should not affect the choice of names by user of a procedure.
- The formal parameters are called bound variables and a procedure binds to its formal parameters.
- Local variables are only visible within procedures.
- These are not affected by outside procedures and do not affect outside procedures.

Dependency

- Procedures tend to depend on other procedures.
- In fact they are encourage due to decomposition.
- This, however, increases coupling.
- If you modify the name of a procedure your procedure depends on, you will affect your procedures implementation.
Block Structure

- You can eliminate this issues with dependency and yet achieve decomposition using a block structure.
- This is nesting of a procedure within another procedure.

Lexical Scoping

- You don’t have to pass parameters repeatedly to nested procedures.
- They can use the parameters (formal and local) defined in the nesting block.
- This is called lexical scoping.