What is Computer Science?

Study of
1. Algorithms: sequence of instructions that transforms inputs into desired outputs.
2. Data structures: information (data) and the organization of the data.
3. Automata: Device that can follow the instructions of an algorithm.

Problem Solving and Abstraction

Focus of this course.
— Problem Solving: Recognizing common patterns in data and using standard design techniques (design recipes) to solve new problems.
— Abstraction: Technique of suppressing details to make implementations simpler.

We will use functional programming languages known as Scheme, Racket, and Swindle.

Programming Languages

• Every programming language comes with 2 sub-languages:
  — a language of data types and
  — a language of operations.

Programs you write in a computer language combine data and operations in such a way that the inputs can be evaluated to a valid data type, the output.

Syntax and Semantics

• The syntax rules of a language specify the legal words and expressions of that language.

• The semantic rules of a language specify the meaning of legal words and expressions.
DrRacket IDE

- Integrated Developer’s Environment:
  1. Editor designed specifically for racket/scheme/swindle.
  2. Two main windows, top one for writing, evaluating and saving programs (Definitions Window) and bottom one for viewing output of program and evaluating temporary code samples (Interactions Window).
  3. Translator (Interpreter) that converts code into a form that can be run by a machine.
     - allows us to write, run, test, and save code.

Programs

- Programs consist of expressions:
  - either a valid data type that evaluates to itself or
  - something that starts with an open parenthesis and ends in a matching right parenthesis.

- Parenthesis are used to:
  - nest expressions
  - join expressions into functions

DrRacket Features

DrRacket:
- auto-indents expressions
  - indentation begins at left side of screen and indents one level (~ 3 spaces) per subexpression
  - when indentation does not return to left after you think you have closed the expression with a “)”, it often indicates a missing right parenthesis
  - matches open/close parenthesis by shading area between them
  - uses different colors to indicate program code, comments, and strings

Starting up DrRacket

After you open DrRacket, choose a language under the Language menu. Specifically, choose Swindle under “Other languages” and press OK. Press Run to make the choice fixed for that session.

The window is split into an upper part (the definitions window) and the bottom part (the interactions window).

Functions (a.k.a. procedures)

The Global Environment

When you start up DrRacket, about 200 primitive functions and values are written into a conceptual table that has one column for names and another for values. This table is called the Global Environment.

You do not have access to look at this table, but you can type a name you expect to be predefined in the interactions window, for example:
> +
#procedure+
> #p
#p: undefined;
cannot reference undefined identifier
The #procedure => tells us + is already defined. However, #p is an error.
The define special form is used to add more name/value pairs to the global environment.
The first keywords you will use are:

- `define` used to create variable names
- `lambda` literally means "function", in the mathematical sense
- `quote` shields its argument from evaluation
- `and` logical operator
- `or` logical operator
- `cond` multi-way decision statement
- `if` 2-way decision statement

The `define` special form is used to add name/value pairs to the global environment.

Syntax and Semantics

- The syntax rules of a language specify the legal words and expressions of that language.
- The semantic rules of a language specify the meaning of legal words and expressions.
  - The default evaluation rule is one type of semantic rule.
  - Each special form has its own semantic rules.
Racket Literals

A literal is an expression that evaluates to itself, i.e., the output
displayed is exactly the same (to the computer) as the input,
e.g.,

1. **numbers**: 1, 42, #i2.3344223210, ¾, 3.12560
2. **booleans**: true #t #true or false #false
3. **characters**: #\a #\A #\b #\B ...
4. **strings**: start and end with " (quotation marks) and can contain almost
any characters, including no characters (for the empty string "")
5. **quoted symbols**: non-blank sequence of characters preceded by a
quote: 'abc 'a 'a1 'cat
6. **quoted lists**: empty list (') or non-empty list

non-empty list: '(2 #t 4 "vassar") = containers for any data types
7. **primitive functions**: non-empty lists with lambda expressions