CST1B10

The Structure and Interpretation of Computer Programs

Du Buqian | Maki's Lab

December 11 2021

The University of New South Wales

# Table of contents

Introduction

- In CST1B10, our codes will run in Windows.
- All codes will be written by Racket, a popular dialect of Scheme instead of MIT-Scheme.

Our Goal

Our goal is that students who complete this subject should

- have a good feeling for the elements of style and the aesthetics of programming
- have command of the major techniques for controlling complexity in a large system
- be capable of reading a 50-page-long program, if it is written in an exemplary style
- know what not to read, and what they need not understand at any moment
- feel secure about modifying a program, retaining the spirit and style of the original author

[1] Structure and Interpretation of Computer Programs, 2nd edition, MIT Press
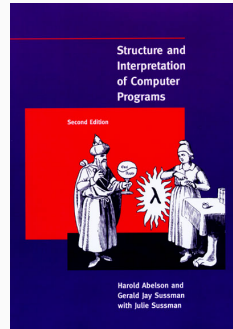
A Swift and Brutal Introduction to Racket

Saluton mondo!

Code:

```
(display "Saluton Mondo!")
```

- primitive expressions
- means of combination
- means of abstraction

Run the REPL(Read-Evaluation-Print-Loop) of Racket.

Then have fun!!!

Codes:

```
(+ 23  23) % => 46
(- 233 23) % => 210
(* 123 2)  % => 246
(/ 246 2)  % => 123
```

This sort of codes is called **Polish notion** or **Prefix notation**.

Some Concepts

Expressions representing numbers may be combined with an expression representing a primitive procedure (such as + or *) to form a compound expression that represents the application of the procedure to those numbers.

$$(+ \ 23 \ 23) \ \% \ => \ 46$$

In this simple code, '+' is a **operator**, and the other elements are called **operands**.

Delimit these expressions by parentheses to denote the procedure application which is called a **combination**.

Prefix notation is not very straightforward, but it is frequently shorter than infix notion. Because prefix notion can accommodate procedures that may take an arbitrary number of arguments. For example:

```
(+ 1 2 3 4 5 6 7 8 9 10) % => 55
```

if this was infix notion:

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
```

That is not adorable.

Racket allow you to write your code be nested like a Matryoshka doll.
For example:

```
(+ (* 2 (+ 2 3)) 3 5 (- 7 2)) % => 23
```

In Racket, we name a thing by

```
( define a 10)
```

Therefore, in this line of code, 'a' is associated with '2';

Absolutely, we also can name some symbols with other kinds of datas:

```
( define pi 3.1415926)
( define b 'b')
( define mit "Maki's Institute of Technology")
( define toh "猫头鹰魔法社")
```

"define" is one of the simplest means of **abstraction** in Racket.

When the programs in last pages is runing, they need some 'containers' to keep to track of the name-object pair. It is environment.