CS450 (section 2)

High Level Languages

UMass Boston Computer Science

Monday, September 11, 2023

Logistics

- HW 0, part 1: **past due**
- HW 0, part 2 due: **Sun 9/17 11:59 pm EST**
- Course web site:

  `https://www.cs.umb.edu/~stchang/cs450/f23`

- Add / drop ends tomorrow

Today’s Theme

I don’t know what _____ is …

… and at this point, I’m too afraid to ask!

(don’t be this person)

Racket

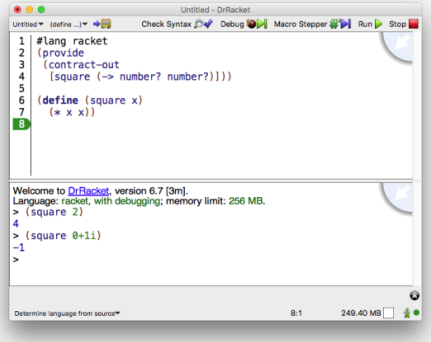
Web: `racket-lang.org`

Download: `download.racket-lang.org`

- Linux: `https://launchpad.net/~plt/+archive/ubuntu/racket`

Version: **8.6+**

IDE: **DrRacket** (easiest)

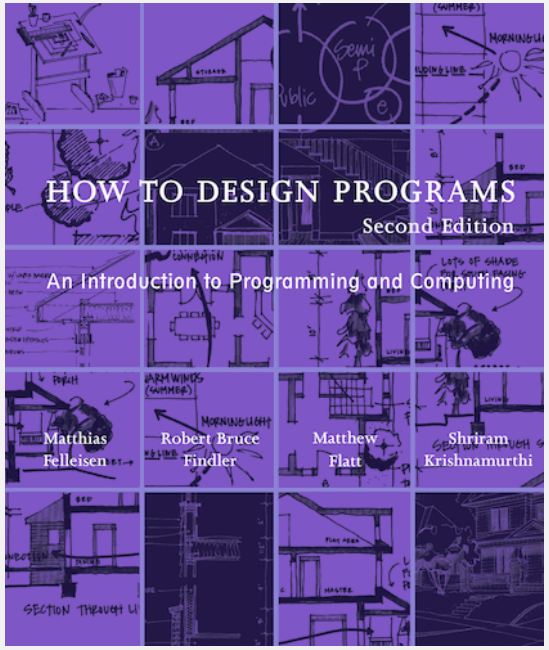


Docs: `docs.racket-lang.org`

forum:

- Hw help: **piazza**
- General: `racket.discourse.group`

Textbook: How to Design Programs 2nd ed.



htdp.org

Git and Github

Did you …

- Create `Github account`?
- Install `git client`?
  - GUI or command line ok
- Learn basic `git` commands?
  - `Clone, push, pull, fork, branch`

Other Logistics

- Piazza
  - Ask all hw questions here (not over email)
  - Faster response
  - Helps other students
- Gradescope
- Read course web page?

`https://www.cs.umb.edu/~stchang/cs450/f23`

Racket (Very) Basics

- File extension: `.rkt`
- First line: `#lang racket`
- syntax: **s-expressions**
- Comments:
  - `;` line
  - `#;` s-expression
- Identifiers:
  - everything except: `( ) [ ] { } " , ' ` ;`
  - And not: `| \ #`
  - **Case sensitive!**

Racket Basics

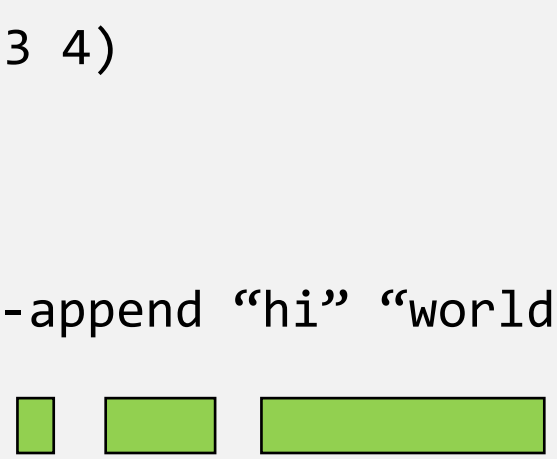
- Function call: **prefix notation** (fn name first)
  - Easier to write multi-arity functions

```
(+ 1 2 3 4)
```

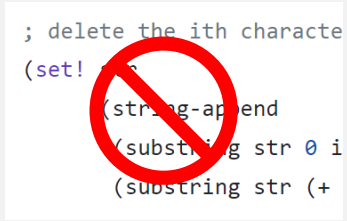
- (fundamental) programming model: **arithmetic**
  - But **not just numbers!**

```
(string-append “hi” “world”)
```

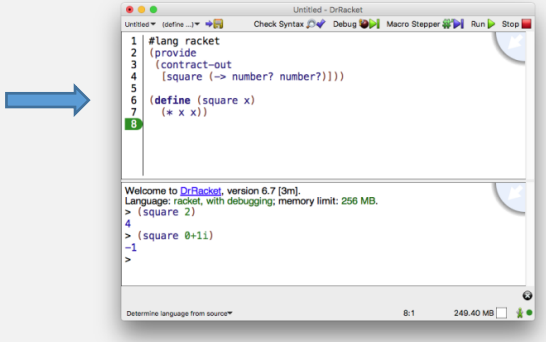
```
(above                    )
```



  - **No statements!**



- Use the **REPL** (“interactions”) for initial testing!



Style

- Critical for writing readable code, e.g.

Google Style Guides

Every major open-source project has its own style guide: a set of conventions (sometimes arbitrary) about how to write code for that project. It is much easier to understand a large codebase when all the code in it is in a consistent style.

"Style" covers a [illegible] bles" to "never use exceptions." This [illegible] If you are modifying a project that or[illegible] apply to that project.

- AngularJS S[illegible]
- Common Li[illegible]
- C++ Style Guide
- C# Style Guide
- Go Style Guide
- HTML/CSS Style Guide
- JavaScript Style Guide
- Java Style Guide
- Objective-C Style Guide
- Python Style Guide

Airbnb JavaScript Style Guide() {

A mostly reasonable approach to JavaScr[illegible]

Microsoft | Learn Documentation Training Certifications Q&A Code Sampl

.NET Languages Features Workloads APIs Resources

Learn / .NET / C# guide / Fundamentals /

Common C# code conventions

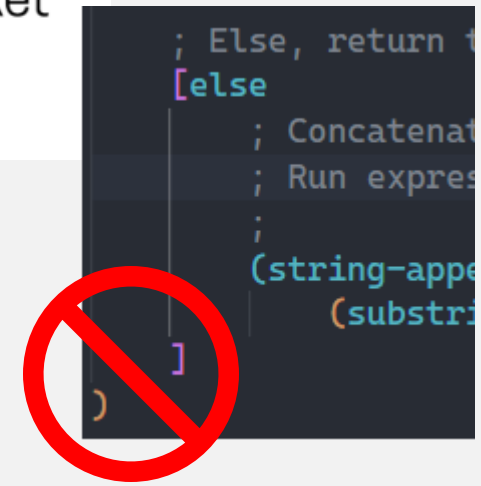
A code standard is essential for maintaining code readability, consistency, and collaboration within a development team. Following industry practices and established

Style: This Class

https://docs.racket-lang.org/style/index.html

Racket

How to Program Racket: a Style Guide



A few tips

- Closing parens do not get their own line
- code width 80-100 columns
- Use dashes to separate multi-word identifiers:
  - `string-append`
- Use `DrRacket` auto-indenter

Style: Git commits

- Git commits must also be **readable** (concise and informative)

How to Write a Git Commit Message

Commit messages matter. Here's how to write them well.

The seven rules of a great Git commit message

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
6. Wrap the body at 72 characters
7. Use the body to explain *what* and *why* vs. *how*

A properly formed Git commit subject line s
complete the following sentence:

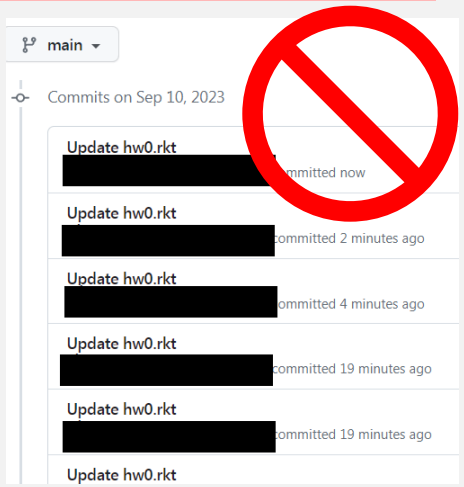
- If applied, this commit will *your subject line here*

For example:

- If applied, this commit will *refactor subsystem X for readability*
- If applied, this commit will *update getting started documentation*
- If applied, this commit will *remove deprecated methods*
- If applied, this commit will *release version 1.0.0*
- If applied, this commit will *merge pull request #123 from user/branch*

Notice how this doesn't work for the other non-imperative forms:

- If applied, this commit will ~~*fixed bug with Y*~~
- If applied, this commit will ~~*changing behavior of X*~~
- If applied, this commit will ~~*more fixes for broken stuff*~~
- If applied, this commit will ~~*sweet new API methods*~~



Code “demo”

Check-In Quiz 9/11

(see gradescope)

(due right before midnight)