Scheme Cheat Sheet

Emacs

alt-x

Called M-x in Emacs lingo. You type this and then type the name of a function. For example, hold alt, hit x, and then enter this in the prompt:

```
tetris
```

which runs a function that starts a game of tetris. Why tetris is included in Emacs I don't know.

Minibuffer

The area below the gray bar. This is where you see messages from Emacs and where you are prompted to enter information.

Open file

C-x C-f (control-x, control-f) After hitting it, you'll see a prompt in the minibuffer. If you type a filename, it will open the file. If you type a name that doesn't exist yet, it will open an empty buffer that you can type in. Once you save that buffer, a new file will be created with that name. So the open file function is also how you create new files.

Tutorial

Type `C-h t` to start the built-in tutorial. Highly recommended. Will take 2-5 hours.

Comic Strpis about Emacs

M-x butterfly
Emacs Learning Curve

Moving Around

Key Notation

`C-f` means hold down `control` and press `f` `M-f` means hold down `alt` and press `f` `S-f` means hold down `shift` and press `f` `C-M-f` means hold down `control` and `alt` and press `f`

Cancel key

Hit `C-g` to get out of sticky situations. Hit this key often.

.emacs file

Your Emacs configuration file is called `.emacs` and lives in your home folder. The file consists of code that is run when Emacs starts up.

What is Emacs?

An old text editor with a lot of features. Made by a cranky person who likes parrots.

Scheme

.scm

This is the file extension for Scheme, just like .docx is the file extension for Microsoft Word files. **shudder** We were using little.scm as the file to do our Scheme work.

Evaluation

The word for what the computer does with expressions. This can be a process of simplification—~(+ 2 2)~ will be evaluated as `4`, for example. In the case of functions, it can mean making them available for use.

Functions

There are three ways you can think of functions, in increasing order of correctness.

1. A way to make your computer do something.
2. A way to save some code for later. (Also known as a subroutine.)
3. Like in math, a thing that takes an input, does something with it, and returns an output. Think of a machine with a hole in the top and a slot in the bottom.

Use a function by making an expression (stuff between parentheses) that has the function name first and then the inputs after it. Like this:

#+BEGIN$_{\text{SRC}}$ (expt 5 2) => 25

#+END$_{\text{EXAMPLE}}$

`expt` is the exponent function. The first input is the base and the second input is the power. 5 to the power of 2. We call inputs to a function **arguments** or **parameters**.

Geiser

Geiser is just some Emacs code that lets us start and access Guile from inside Emacs.

Guile

Programming is kind of like making a house. First, you design something, then you actually build it. In programming, designs for big things like languages and protocols are called standards. The actual product of the design after it's built is the implementation. So, Scheme has a standard that defines the language in the abstract, but it also has different implementations, actual code that runs Scheme programs, that are in practice slightly different from each other. So far we're been using Guile, but there's also MIT Scheme and Chicken Scheme.

Running Scheme Code in Emacs

Open Emacs. (Finder > Emacs if you've downloaded the OSX version to your Applications folder)

Start the Scheme process. Hit alt-x. At the prompt, type

```
run-geiser
```

You'll see a prompt:

`Start Geiser for scheme implementation:`

Type `guile` at the prompt and hit enter. You should see a window appear with your Scheme process.

Run Scheme code. You can type directly into the REPL (the Scheme process) by typing at the prompt that looks like this:

```
scheme@(guile-user)>
```

Just type some Scheme and hit Enter.

The other way to run code is to run it from a .scm file (Scheme file). To do this, first make sure your REPL (guile process) is running. Open a .scm file. Type some Scheme into the file, then put your cursor at the end of an expression and type `C-x C-e`. The code should run and you'll see output, if any, in the minibuffer.

What is Lisp?

Lisp was a cool math notation that someone at MIT came up with. One of his grad students looked at it and was like, "Did you know this is basically a programming language?" The guy was like, "cool." They used it for AI research until the field of AI collapsed in on itself.

As a language it's interesting because code/procedures are the same as data. In Lisp's elemental form there really is no syntax, basically just expressions that look like lists. Unimaginatively, it stands for LISt Processing.

Lisps are basically lemurs in a world of monkeys. Almost all non-lisps are descended from a language called Algol by way of a language called C. Lisps are atavistic. Lisp had a ton of great ideas that Algol-derived languages continue to adopt at a rate of one every seven years or so.

See The Lisp Curse, Paul Graham on Lisp, and Why Lisp Failed. Lisp seems to be making a comeback—it's functional and functional programming is so hot right now. Also Clojure is a trendy new Lisp. Will it ever be as popular as Algol-derived languages? No, because path dependence.

What is Scheme?

It's a programming language that someone made at MIT to teach in his programming classes. The most famous book about it is SICP. That book is really hard but I will finish it some day. Scheme is a Lisp.

Scheme is pretty cool to learn because it's intentionally small and because there are good books written for it.