1 | Intro

Let's get started making our own configuration file! Make the file `~/.emacs.d/init.el` (remember, you can do that with `C-x C-f` and typing in the filename). Emacs runs this file whenever you start it.

Emacs is configured in *Emacs Lisp*: so let's quickly review some important parts of Lisp that make it a tad unusual:

- Everything is *array-based* in Lisp, so instead of calling a function like `function(arg1, arg2)`, you call it like `(function arg1 arg2)`
- Lisp is *functional*, so code does not execute sequentially by default unless for certain circumstances. Code is mostly written by composing functions (nesting function calls).
- Lisp has a lot of single quotes scattered around: these are not strings but *symbols* that refer to a function/variable by its name. A lot of interfaces in Emacs want the *name* of a function rather than the function itself.
  - Example: `'(chicken 1)` would not call the function `chicken` but just return what you literally typed out: that is, `'(chicken 1)`.

Let's get started by configuring our first thing! Let's remove those annoying headerbars and scrollbars. `setq` is a function we're gonna use a lot: it sets the value of a variable (which are used to express most of the options in Emacs).

```
(tool-bar-mode -1) ;; This calls the function tool-bar-mode and turns it off by passing -1
(menu-bar-mode -1) ;; Same but for menubar
(scroll-bar-mode -1) ;; Same but for scroll bar
(setq use-dialog-box nil) ;; Set the variable use-dialog-box to nil, turning off GUI popups
```

Next we'll banish the horror of `custom.el` to the abyss.

```
(setq custom-file "/dev/null")
```

Emacs has a tendency to litter autosave files everywhere, so let's put them all in one place:

```
(setq backup-directory-alist `(("." . "~/.saves")))
(setq backup-by-copying t)
```

We can then load `package.el` and open ourselves to the Emacs universe!

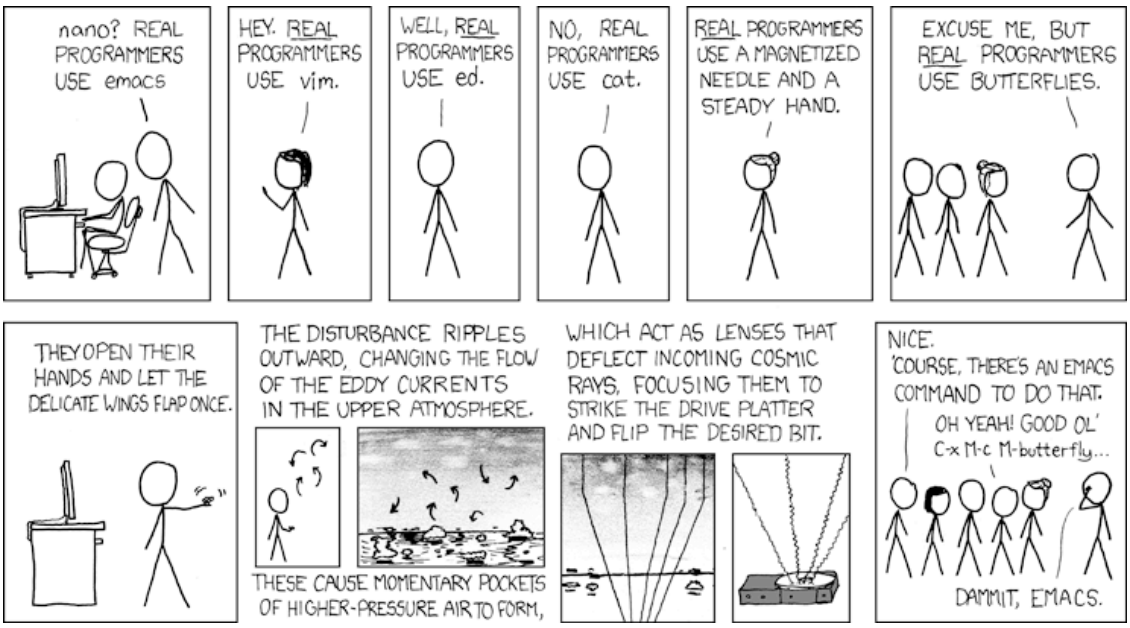
```
(require 'package)
```

Also we'll set the theme to be `tango`, but you can change it with `M-x load-theme`.

```
(load-theme 'tango)
```

2 | Other Usecases

2.1 | **Binding Keys**



Let's harness the power of the butterfly (although with a shorter key combo)

```
(global-set-key (kbd "C-x M-c") 'butterfly)
```

Now, after evaluating that, use the key combo `C-x M-c` and unleash Emacs' true potential!

2.2 | **Hooks**

Let's say you want to harness the power of the butterfly whenever you open an Emacs Lisp buffer (**don't actually do this**, since that'll be a bit too much to handle). We can use a *hook*, which allows you to run custom code before/after important events, allowing for a large amount of extensibility. Hooks are actually variables, so if you ever want to search for one, good ol' `C-h v` will help.

```
(add-hook 'emacs-lisp-mode-hook 'butterfly)
```

This could be use for many other things, like enabling a linter whenever you open a Python buffer.

3 | **More Lisp**

Here's some more complicated examples of plain Emacs Lisp that may not be directly useful just yet.

```
(if (> 2 1)
    (message "hello")
  (message "goodbye"))

(defun my-function (arg1 arg2)
  (message "look!")
  (message "sequential!")
```

```
  (message arg2)
  (message "whee!"))
(my-function "hello" "there")
```

You also might see something like `:blah` around, and they're usually for specifying arguments. They're just upgraded string literals, don't worry about them too much.