

Guile

Guile

- Today, we'll write some **Guile** together.

Guile

- Today, we'll write some **Guile** together.
- Guile is a dialect of **Scheme**, which is a dialect of Lisp.

Guile

Guile

- You can launch the Guile REPL by running `rlwrap guile` on thepond.

Guile

- You can launch the Guile REPL by running `rlwrap guile` on thepond.
- The Guile REPL doesn't natively support history or line editing.

Guile

- You can launch the Guile REPL by running `rlwrap guile` on thepond.
- The Guile REPL doesn't natively support history or line editing.
- `rlwrap` adds this functionality.

Recap: `car` and `cdr`

Recap: `car` and `cdr`

- The fundamental data structure of Lisp is the *cons cell* (i.e., the pair).

Recap: `car` and `cdr`

- The fundamental data structure of Lisp is the *cons cell* (i.e., the pair).
- We access the first element of a pair using `car`.

Recap: `car` and `cdr`

- The fundamental data structure of Lisp is the *cons cell* (i.e., the pair).
- We access the first element of a pair using `car`.
- We access the second element of a pair using `cdr`.

Recap: `car` and `cdr`

- The fundamental data structure of Lisp is the *cons cell* (i.e., the pair).
- We access the first element of a pair using `car`.
- We access the second element of a pair using `cdr`.
- We construct lists by chaining cons cells.

Lists

Guile provides a few ways to construct lists.

Lists

Guile provides a few ways to construct lists.

1 `list`

Lists

Guile provides a few ways to construct lists.

1 `list`

- e.g., `(list 1 2 3)`

Lists

Guile provides a few ways to construct lists.

1 `list`

- e.g., `(list 1 2 3)`

2 `'`, aka `quote`.

Lists

Guile provides a few ways to construct lists.

1 `list`

- e.g., `(list 1 2 3)`

2 `'`, aka `quote`.

- e.g., `'(1 2 3)`

Lists

Guile provides a few ways to construct lists.

1 `list`

- e.g., `(list 1 2 3)`

2 `'`, aka `quote`.

- e.g., `'(1 2 3)`

3 Repeated application of `cons`

Lists

Guile provides a few ways to construct lists.

1 `list`

- e.g., `(list 1 2 3)`

2 `'`, aka `quote`.

- e.g., `'(1 2 3)`

3 Repeated application of `cons`

- `(cons 1 (cons 2 (cons 3 #nil)))`

`list` vs. `'`

`list` and `'` are not the same!

`list` vs. `'`

`list` and `'` are not the same!

```
1 (list 1 2 (+ 2 1))
```

`list` vs. `'`

`list` and `'` are not the same!

```
1 (list 1 2 (+ 2 1))
```

```
  ■ → (1 2 3)
```

list vs. '

list and ' are not the same!

1 (list 1 2 (+ 2 1))

■ → (1 2 3)

2 '(1 2 (+ 2 1))

list vs. '

list and ' are not the same!

1 (list 1 2 (+ 2 1))

■ → (1 2 3)

2 '(1 2 (+ 2 1))

■ → (1 2 (+ 2 1))

define

define

- In Guile, we define functions using the `define` special form.

define

- In Guile, we define functions using the `define` special form.

e.g.,

```
(define (add3 x y z)
  (+ (+ x y) z))
```

lambda

lambda

- Sometimes, it's useful to define functions without giving them names.

lambda

- Sometimes, it's useful to define functions without giving them names.
- We call such functions “lambdas.”

lambda

- Sometimes, it's useful to define functions without giving them names.
- We call such functions “lambdas.”

e.g.,

```
(lambda (x y z)
  (+ (+ x y) z))
```

is equivalent to the `add3` function we saw in the previous slide.

Guile Cheat-Sheet

Guile Cheat-Sheet

- `#t` : true

Guile Cheat-Sheet

- `#t` : true
- `#f` : false

Guile Cheat-Sheet

- `#t` : true
- `#f` : false
- `#nil` , aka `'()` : the empty list

Guile Cheat-Sheet

- `#t` : true
- `#f` : false
- `#nil`, aka `'()` : the empty list
 - Note that `#f` and `#nil` are distinct in Guile!

5 tasks, together

5 tasks, together

- A function that doubles its input

5 tasks, together

- A function that doubles its input
- ...recursively :)

5 tasks, together

- A function that doubles its input
- ...recursively :)
- A function to double each item in a list

5 tasks, together

- A function that doubles its input
- ...recursively :)
- A function to double each item in a list
- A function to sum a list

5 tasks, together

- A function that doubles its input
- ...recursively :)
- A function to double each item in a list
- A function to sum a list
- A function to double each item in a list, then sum it

Lab Time

Your turn :)