

CS 3520/6520  
Programming Languages

Fall 2023

Instructors: **Ben Greenman**  
**Matthew Flatt**

TA: **Alec Mills**

# CS 3520/6520 Programming Languages

# CS 3520/6520 Programming Languages

**a survey course:**



an object-oriented language



a functional language



a logic language

# CS 3520/6520 Programming Languages

**Not a survey course:**



an object-oriented language



a functional language



a logic language

# CS 3520/6520 Programming Languages

This course is about programming language **concepts**

lexical scope	closures	recursion
$\lambda$ -calculus	objects	classes
continuations	eager and lazy evaluation	
state	type checking	polymorphism
soundness	type inference	subtyping
compilation	garbage collection	

... especially **functional programming** concepts

*use one language, implement many languages*

# CS 3520/6520 Programming Languages

This course is about programming language **concepts**

- To help you understand new programming languages
- To make you a better programmer in any language

## Course Details

See syllabus in Canvas

*In person, livestreamed and recorded via Zoom*

Formal prerequisite: CS 3500

Informal prerequisite: more programming experience than that

### Grading:

- Weekly homework (55%)
- Two mid-term exams (30%)
- Extended final homework (10%)
- Online quizzes (5%)

Late policy for homework: up to 48 hours, two automatic “free lates”

## Lectures are Online

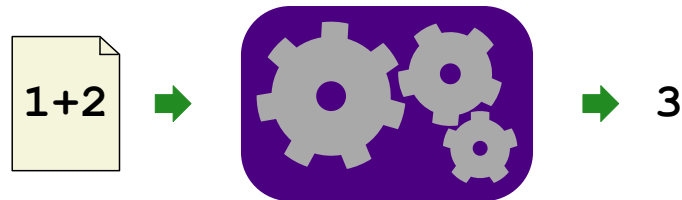
All slide presentations are online

- **Watch the videos before class**
- **Take the quiz before class**
  - $\geq 60\%$  over semester  $\Rightarrow 100\%$
  - no late quizzes
- **Meet as a class for more examples and homework solutions**
  - a.k.a. “recitation”
  - guideline: no new material introduced in class
  - will need in-class volunteers



# Interpreters

- Learn concepts by implementing **interpreters**



new concept  $\Rightarrow$  new interpreter

We'll always call the language that we implement **Moe**, even though the language keeps changing

**Moe** = successor to Curly

## Racket and Shplait

- **Implement interpreters using **Shplait**, a variant of **Racket****

Historically: **Lisp** ⇒ **Scheme** ⇒ **Racket** ⇒ **Rhombus** ⇒ **Shplait**

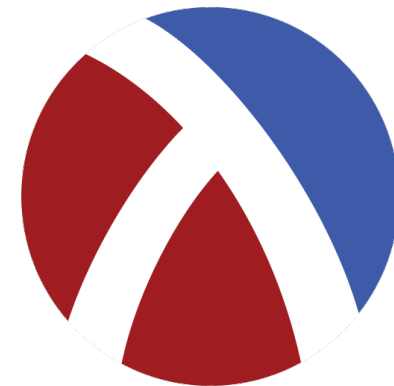
## Racket and Shplait

- **Implement interpreters using **Shplait**, a variant of **Racket****

Historically: **Lisp** ⇒ **Scheme** ⇒ **Racket** ⇒ **Rhombus** ⇒ **Shplait** ← **OCaml**

### **Racket** is

- a programming language
- a language for creating programming languages



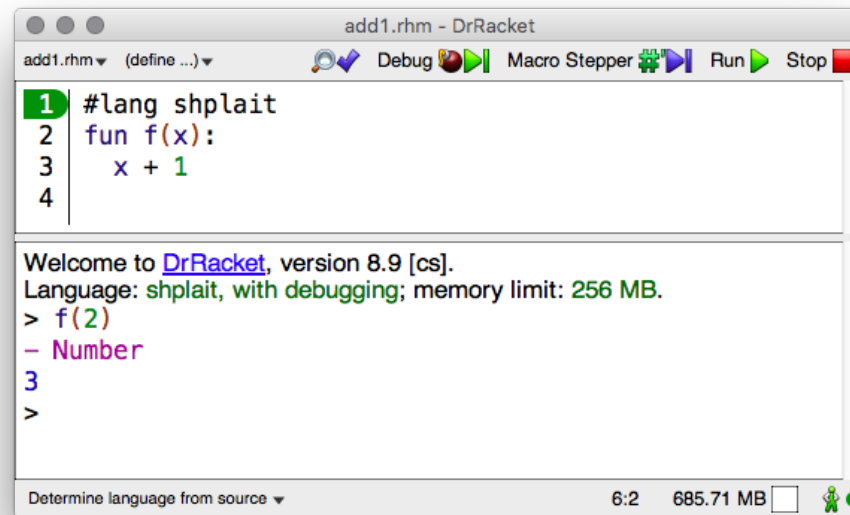
... including **Shplait**

Sh = *Shrubbery*, a notation

PLAI = *Programming Languages: Application and Interpretation*, a textbook

t = *types*, a la ML

# DrRacket



The screenshot shows the DrRacket IDE window titled "add1.rhm - DrRacket". The editor contains the following code:

```
1 #lang shplait
2 fun f(x):
3   x + 1
4
```

The output window displays the following text:

```
Welcome to DrRacket, version 8.9 [cs].
Language: shplait, with debugging; memory limit: 256 MB.
> f(2)
- Number
3
>
```

The status bar at the bottom shows "Determine language from source", "6:2", and "685.71 MB".

# Preview: Shplait Tutorial

`http://docs.racket-lang.org/shplait`

v8.9

## Shplait

The Shplait language syntactically resembles the [Rhombus](#) language, but the type system is close to that of [ML](#). For a quick introduction, see the [tutorial section](#) or the [tutorial videos](#).

```
#lang shplait
```

```
package: shplait
```

### 1 Tutorial

#### 1.1 Getting Started

## Preview: Shplait Notation

`f(x)`

`1+2`

`1+2*3`

`s=6`

`f(x)=x+1`

`{ x < 0: -1`  
`x = 0: 0`  
`x > 0: 1`

`f(x)`

`1 + 2`

`1 + 2 * 3`

`def s = 6`

`fun f(x):`  
`x + 1`

`cond`

`| x < 0: -1`  
`| x == 0: 0`  
`| x > 0: 1`

## Preview: Shplait Data

- Numbers and strings

obvious

```
1 -42 "Hello, World!"
```

- Booleans

straightforward

```
#true #false
```

- Symbols

unusual

```
#'apple #'def
```

## Preview: Shplait Quoted Code

- Single quote `'` instead of string `"`

convenient

```
'x'
```

```
'x + 1'
```

```
'fun f(x) :  
  x + 1'
```



## Preview: Shplait Datatypes

```
type Shape
| circle(radius :: Int)
| rectangle(width :: Int,
            height :: Int)

fun area(s):
  match s
  | circle(r): 3 * r * r
  | rectangle(w, h): w * h

check: area(circle(2))
      ~is 12
check: area(rectangle(4, 5))
      ~is 20
```

## Preview: Interpreters

See `lambda.rhm`

Example **Shplait** program:

```
type Value
| intV(n :: Int)
| closV(arg :: Symbol,
        body :: Exp,
        env :: Env)
```

Example **Moe** program:

```
3 * 4 + 8
```

Example **Moe** program as a **Shplait** value:

```
'3 * 4 + 8'
```

## Datatype and Function Shapes Match

```
type Shape
| circle(radius :: Int)
| rectangle(width :: Int,
            height :: Int)
| adjacent(left :: Shape,
           right :: Shape)

fun area(s):
  match s
  | circle(r): 3 * r * r
  | rectangle(w, h): w * h
  | adjacent(l, r): area(l)
                    + area(r)

check: area(circle(2))
      ~is 12
check: area(rectangle(4, 5))
      ~is 20
check: area(adjacent(circle(2), rectangle(4, 5)))
      ~is 32
```

## Datatype and Function Shapes Match

```
type Shape
| circle(radius :: Int)
| rectangle(width :: Int,
            height :: Int)
| adjacent(left :: Shape,
           right :: Shape)

fun area(s):
  match s
  | circle(r): 3 * r * r
  | rectangle(w, h): w * h
  | adjacent(l, r): area(l)
                    + area(r)

check: area(circle(2))
      ~is 12
check: area(rectangle(4, 5))
      ~is 20
check: area(adjacent(circle(2), rectangle(4, 5)))
      ~is 32
```

## Datatype and Function Shapes Match

```
type Shape
| circle(radius :: Int)
| rectangle(width :: Int,
            height :: Int)
| adjacent(left :: Shape,
           right :: Shape)

fun area(s):
  match s
  | circle(r): 3 * r * r
  | rectangle(w, h): w * h
  | adjacent(l, r): area(l)
                    + area(r)
```

```
check: area(circle(2))
      ~is 12
```

```
check: area(rectangle(4, 5))
      ~is 20
```

```
check: area(adjacent(circle(2), rectangle(4, 5)))
      ~is 32
```

# Course Outline

- Functional programming
- Interpreters
- State
- Control
- Compilation and GC
- Objects and classes
- Types
- Macros and more

## Rest of Today

- Take “Syllabus” quiz
- Watch “Shplait Tutorial” videos (~30 minutes)
- Take “Shplait Tutorial” quiz

*Quizzes due by the end of the day*

# Homework 0

- Create handin account
- Shplait warm-up exercises

*Due Friday, August 25*