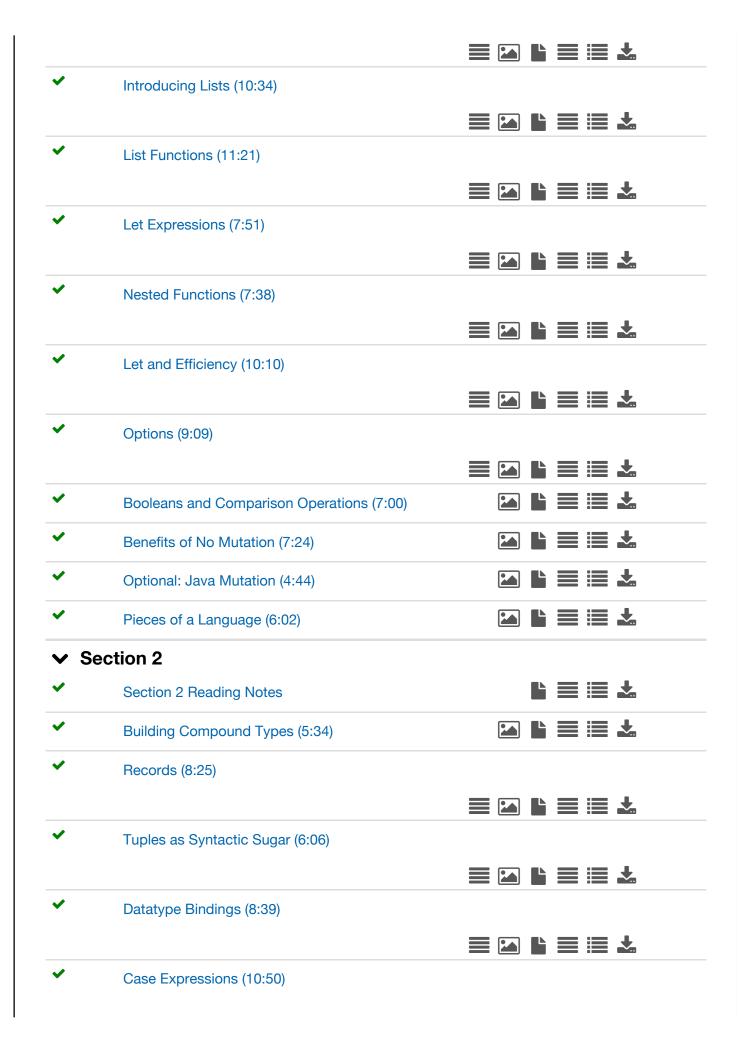
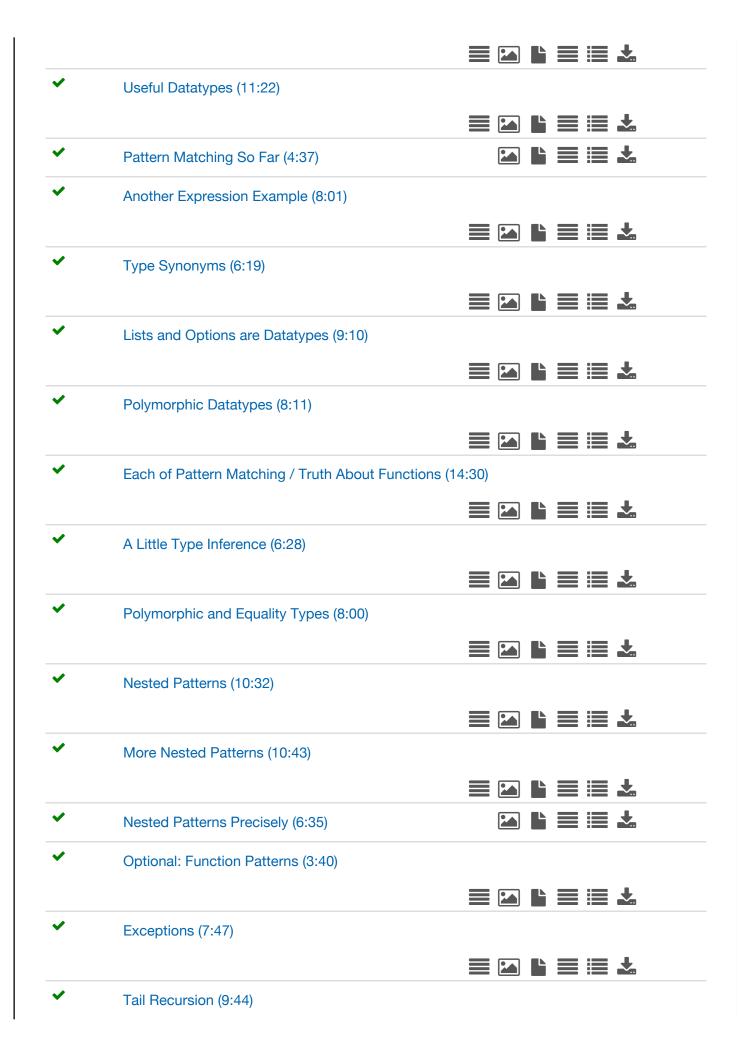
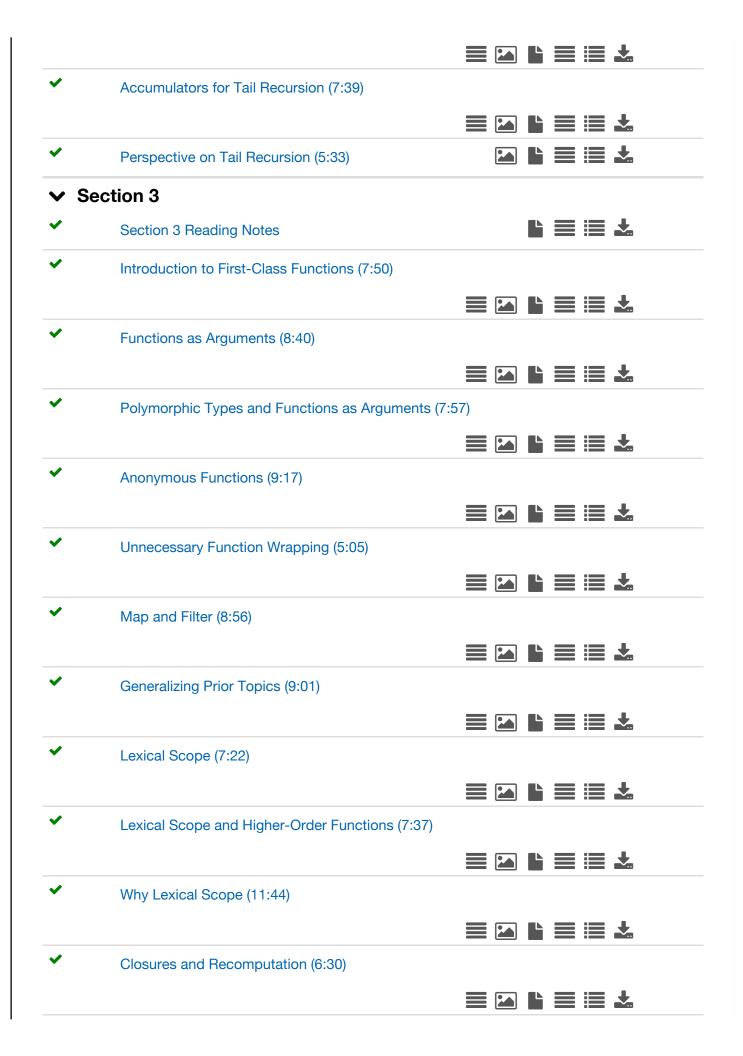
Having trouble viewing lectures? Try changing players. Your current player format is html5. Change to flash.

| ∨ Se | ection 0 | | | | |
|-------------|---|--------------|--|----------|----------|
| / | Course Introduction: About the Course (12:23) | 2 | | | ₹. |
| | Course Introduction: About Programming Langua | iges (12:04) | | | |
| | | | | | ↓ |
| | Grading Policy (12:59) | 2 | | | ₹ |
| • | Software Installation Introduction (1:22) | | | | ₹. |
| / | Emacs installation (2:27) | | | | 1 |
| / | SML installation (1:47) | | | | <u>†</u> |
| • | SML Mode installation (3:52) | | | | <u>+</u> |
| ✓ Se | ection 1 | | | | |
| / | Section 1 Reading Notes | | | | + |
| / | ML Variable Bindings and Expressions (14:32) | | | | |
| | | | | | ↓ |
| | Rules for Expressions (9:13) | 2 | | | ↓ |
| / | The REPL and Errors (12:10) | | | | |
| | | | | = | <u>+</u> |
| | Shadowing (6:49) | | | | |
| | | | | | ₹. |
| • | Functions Informally (7:37) | | | | |
| | | | | | <u>+</u> |
| / | Functions Formally (8:56) | 2 | | | ₹. |
| | Pairs and Other Tuples (9:16) | | | | |



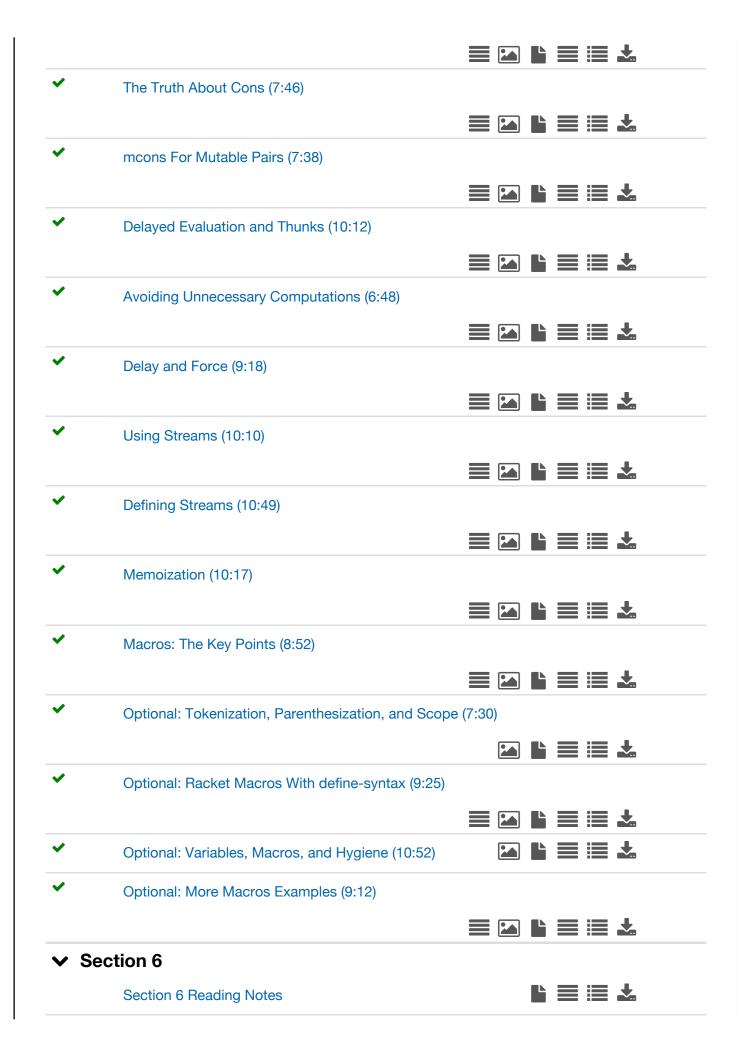




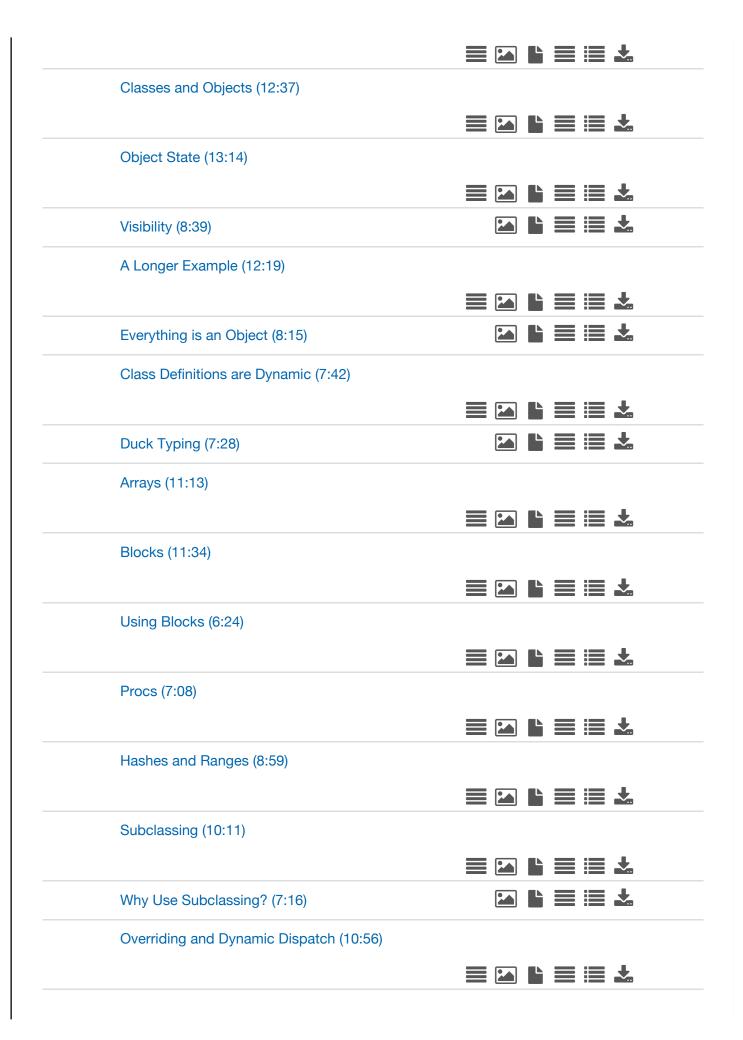
| ~ | Fold and More Closures (11:28) | | | | | | | |
|-------------|---|---|----------|---|---|---|------------|--|
| | | | 2 | | | | <u>+</u> | |
| ✓ | Closure Idiom: Combining Functions (9:18) | | | | | | | |
| | | | | | | | * | |
| ~ | Closure Idiom: Currying (10:32) | _ | | | _ | | | |
| | | | 2 | | | | * | |
| ~ | Partial Application (9:41) | _ | | | _ | | | |
| | | | | | | | * | |
| ~ | Currying Wrapup (6:40) | _ | | | _ | | | |
| | | | | | | | <u>~</u> . | |
| ~ | Mutable References (8:07) | | | | | | | |
| | | | ^ | | | | <u>+</u> | |
| ~ | Closure Idiom: Callbacks (8:25) | | | | | | | |
| | | | * | | | | <u>+</u> | |
| ✓ | Standard-Library Documentation (7:00) | | <u>*</u> | L | | | <u>+</u> | |
| ✓ | Optional: Abstract Data Types With Closures (11:14) | | | | | | | |
| | | | ^ | | | | <u>+</u> | |
| ✓ | Optional: Closure Idioms Without Closures (4:48) | | | | | | | |
| | | | | | | | <u>+</u> | |
| ~ | Optional: Java Without Closures (12:08) | | | | | | | |
| | | | 2 | | | | <u>+</u> | |
| ✓ | Optional: C Without Closures (10:56) | | | | | | | |
| | | | | | | | <u>+</u> | |
| ∨ Op | otional: Course Motivation | | | | | | | |
| ~ | Optional: Course-Motivation Introduction (5:56) | | | | | | <u>+</u> | |
| ~ | Optional: Why Study General PL Concepts? (10:30) | | 2 | L | | | <u>+</u> | |
| ~ | Optional: Are All PLs the Same? (6:51) | | 2 | L | | ≡ | <u>+</u> | |
| ~ | Optional: Why Functional Languages? (11:15) | | 2 | L | | | <u>+</u> | |

| | Optional: Why ML, Racket, and Ruby? (12:34) | |
|------------|---|--|
| ∨ S | ection 4 | |
| | Section 4 Reading Notes | |
| ~ | Section Introduction (1:45) | |
| ~ | What is Type Inference (5:37) | |
| | | |
| ~ | ML Type Inference (6:09) | |
| | | |
| ~ | Type Inference Examples (10:27) | |
| | | |
| ~ | Polymorphic Examples (10:52) | |
| • | | |
| • | Optional: The Value Restriction and Other Type-Ir | |
| ~ | Mutual Recursion (9:45) | |
| | mataar riccarcieri (cric) | |
| ~ | Modules for Namespace Management (6:25) | |
| | | |
| ~ | Signatures and Hiding Things (7:02) | |
| | | |
| ~ | A Module Example (11:06) | |
| | | |
| ~ | Signatures for Our Example (11:03) | |
| | | |
| • | Signature Matching (4:03) | |
| ~ | An Equivalent Structure (6:38) | |
| | | |

| ~ | Another Equivalent Structure (9:01) | | | | | |
|-------------|---|------------|----------|-------------|-----|----------|
| | | | * | = :: | ≣ 4 | |
| ✓ | Different Modules Define Different Types (3:32) | | | | | |
| | | = (| 2 | | ≣ 4 | L |
| ✓ | Equivalent Functions (8:41) | (| * | | ≣ 4 | L |
| ~ | Standard Equivalences (10:01) | [| * | | ≣ 4 | L |
| ✓ | Equivalence Versus Performance (6:00) | [| * | | ≣ 4 | L |
| ∨ Se | ction 5 | | | | | |
| ~ | Section 5 Reading Notes | | | | ≣ 4 | L |
| ✓ | Introduction to Racket (8:23) | | | | | |
| | | | * | | ≣ 4 | L |
| ~ | Racket Definitions, Functions, Conditionals (10:14) | | | | | |
| | | | * | | ≣ 4 | |
| ~ | Racket Lists (9:07) | | | | | |
| | | | * | | ≣ 4 | L |
| • | Syntax and Parentheses (8:36) | [| | | ≣ 4 | L |
| ✓ | Parentheses Matter! (Debugging Practice) (10:50) | | | | | |
| | | | * | | ≣ 4 | L |
| ✓ | Dynamic Typing (9:36) | | | | | |
| | | | | | ≣ 4 | L |
| ~ | Cond (8:52) | | | | | |
| | | | * | | ≣ 4 | L |
| ~ | Local Bindings (13:11) | | | | | |
| | | | | | ≣ 4 | L |
| ~ | Toplevel Bindings (4:54) | | | | | |
| | | | | | ≣ 4 | L |
| ~ | Mutation with set! (8:28) | | | | | |



| ~ | Section Introduction (2:08) | 2 | | . |
|-------------|---|----------|------------|------------|
| ~ | Datatype-Programming in Racket Without Structs (13:21) | | | |
| | | | = | Ł |
| ~ | Datatype-Programming in Racket With Structs (9:34) | | | |
| | | | | ₹ |
| ~ | Advantages of Structs (8:13) | | | ₹ |
| ~ | Implementing Programming Languages (10:07) | | | ₹. |
| ~ | What Your Interpreter Can and Cannot Assume (13:47) | | | |
| | | | | <u>+</u> |
| ~ | Implementing Variables and Environments (6:29) | | = | ₹. |
| ~ | Implementing Closures (6:32) | | | ± |
| ~ | Optional: Are Closures Efficient? (9:03) | | | L . |
| ~ | Racket Functions As "Macros" For Interpreted Language (9:15 | 5) | | |
| | | | | ↓ |
| ~ | ML Versus Racket (8:53) | | | L |
| ~ | What is Static Checking? (10:00) | | | <u>.</u> |
| ~ | Soundness and Completeness (9:46) | | | <u>+</u> |
| ~ | Weak Typing (9:39) | | | L . |
| ~ | Static Versus Dynamic Typing, Part One (9:40) | | | |
| | | | = . | ↓ |
| ~ | Static Versus Dynamic Typing, Part Two (13:47) | | | <u>t</u> |
| ~ | Optional: eval and quote (7:58) | | | |
| | | | = | <u>+</u> |
| ∨ Se | ction 7 | | | |
| | Section 7 Reading Notes | | | L . |
| | Introduction to Ruby (10:08) | | | |
| | | | | |



| Method-Lookup Rules, Precisely (11:31) |
|---|
| |
| Dynamic Dispatch Versus Closures (9:38) |
| |
| Optional: Dynamic Dispatch Manually in Racket (15:56) |
| |
| ✓ Section 8 |
| Section 8 Reading Notes |
| OOP Versus Functional Decomposition (12:51) |
| |
| Adding Operations or Variants (11:08) |
| |
| Binary Methods with Functional Decomposition (7:18) |
| |
| Double Dispatch (14:46) |
| |
| Optional: Multimethods (6:36) |
| Multiple Inheritance (10:18) |
| |
| Mixins (11:52) |
| |
| Interfaces (7:33) |
| Optional: Abstract Methods (8:51) |
| Subtyping From the Beginning (10:43) |
| The Subtype Relation (8:20) |
| Depth Subtyping (8:50) |
| Optional: Java/C# Arrays (9:11) |
| Function Subtyping (11:17) |
| |

| | Subtyping for OOP (11:44) | | | <u>+</u> | |
|---------------|---|----------|---|----------|--|
| | Generics Versus Subtyping (8:16) | 2 | | <u>+</u> | |
| | Bounded Polymorphism (8:05) | | | <u>+</u> | |
| | | | | | |
| ∨ Cou | se Wrapup | | | | |
| ∨ Coui | rse Wrapup Optional: Course Content Wrapup (9:43) | | = | <u></u> | |