Scala

MAS REU
Trinity University
Summer 2010
Introductions

- Dr. Mark C. Lewis
- Ph.D. in Astrophysics and Planetary Science
- M.S. in Computer Science
- Research in field of simulation
  - Particular focus on Saturn's rings
- Strong interest in programming languages
- Love to code!
Why Scala?

- Future parallel languages
  - X10, Fortress, Chapel
- Twitter back end
- Better Java than Java
- Highly expressive for compact code
- Static typing for reliable code
What is Scala?

- Stands for “Scalable Language”
- Object-functional
- Static typing with local type inference
- Supports programming in the small and programming in the large.

Go to www.scala-lang.org.
Scala REPL

- Just run scala
- Read-Evaluate-Print Loop
- Familiar to functional and some scripting languages
- Test little bits of code
- See how things work
- Let's do it!
Everything is an Object

- Purely Object-Oriented: Everything in Scala is an Object
- Improved uniformity in programming.
- Operators as methods.
- There is no static. Object declarations create singleton objects that serve same role better.
Functional Nature

- Functions are first-class values.
- Higher order functions.
- Other stuff
  - Expressions are statements and control structures are expressions (except while)
  - Function literals, Pass-by-name, Lazy values
  - Curried functions
  - Partially applied functions
- Tuples!!!
Declarations

- **val** – creates a variable that can't be changed. It is like a final value in Java.
- **var** – creates a variable's value.
- **def** – creates a named function/method.
- Types after name separated by colon.
- Specifying types is generally optional for val and var. Required for arguments in def and recommended for return type.
- **val/var** with tuples/patterns
Collections

- Scala has an extensive collections library.
- Array and List are the most common.
- Also has things like sets and maps.
- Subpackages for mutable and immutable. Arrays are mutable. Lists are immutable.
- Generics specified with types in square brackets.
- Shortcuts for making them.
Normal Methods

- Scala collections have very rich interfaces.
  - ++, drop, dropRight, grouped, head, init, isEmpty, last, max, min, mkString, size, slice, sliding, tail, take, takeRight, zip

- These are just the normal methods on Iterable. The list gets longer with other subtypes.
Functional Methods

- All the collection types have a set of higher order methods that operate on the elements. 
  - collect, count, dropWhile, exists, filter, find, flatMap, foldLeft, foldRight, forall, foreach, groupBy, partition, reduceLeft, reduceRight, scanLeft, scanRight, span, takeWhile
- Again, these are on Iterable.
- Most collections also work as partial functions.
Function Literals

- Can be defined as literals.
  - Put argument list before => followed by logic.
  - Type of arguments can often be inferred.
  - \( x \Rightarrow x \times x \)

- Abbreviated syntax
  - If each argument appears only once and they appear in order, you can use underscores and leave out the argument list and arrow.
  - \_<5
Conditionals

- If expression
  - Looks like what you are used to, but it is an expression with a value.

- Match (simple and patterns)
  - Like a switch, but far more powerful
  - Yields a value
  - Can match types (no generics)
  - Can match patterns
Loops

- **For loop**
  - Iterates over Iterable[T]
  - Shortcuts for counting
  - Expression using yield
  - Multiple iterators, conditionals, intermediate values

- **While and do-while loops** are just like what you are used to in other languages.
Try/Catch

- Much like Java.
- Catch is structured like match with cases for different types of exceptions.
- Produces a result value.
- Has finally.
- No checked exceptions.
- Now you know all the Scala control structures.
Scripting

- You can write scripts in Scala by putting code in a file that ends in .scala and running it with the scala command.
- This is great for simple programs or to glue together other elements.
- Can be used to glue together Java programs because Scala calls Java seamlessly.
Classes

- Similar to Java with a few differences.
  - Use var, val, and def for members/methods
  - Methods can be operator symbols
  - Constructor code put directly into class
  - Arguments passed into class
  - Inheritance uses extends and with
  - Abstract types
  - Visibility is public by default
Traits Not Interfaces

- Traits allow data and method definitions.
- No arguments to traits.
- Ambiguity of multiple inheritance dealt with by resolution order.
- Tends to lead to rich interfaces instead of thin interfaces.
Imports and Packages

- Packages really nested in Scala.
- Use _ instead of * to import everything in a scope.
- Import statements don't have to be at the top of a file.
- Normal Scala import can function like Java static import.
- Import selections, renames, hide with {}.
Visibility

- Public by default.
- Private and protected modifiers.
- Can specify scope with square brackets.
- Can specify private[this] so only current object can access.
Other Language Stuff

- Triple quote strings
- Partial functions
- Case classes
- Extractors
- Covariant and contravariant types
- Structural Types
- XML
- Single quote symbols, Backquotes
Other Library Stuff

- XML
- Actors
- Parsers
- Regular expressions
- Math
- Concurrency
Reference Material

- The API is at the main Scala site.
- Daily Scala
  - http://daily-scala.blogspot.com
- Scala Cheat Sheet
  - http://anyall.org/scalacheat/