2017-Fall

COMP 311: Functional Programming (Fall 2017)

Instructors	Dr. Nick Vrvilo (2) Dr. Corky Cartwright	**Graduate TAs**	• Yao-Hsiang Yang • Weixi Zhu
		Undergraduate TAs	• Nate Kim
Lectures	MEL 251 (Mechanical Laboratories)	**Lecture Times**	4pm–5:15pm TR
Course Email	{nick.vrvilo,cork}@rice.edu	**Online Discussion**	Piazza – Rice Comp 311

Description

This class provides an introduction to concepts, principles, and approaches of functional programming. Functional programming is a style of programming in which the key means of computation is the application of functions to arguments (which themselves can be functions). This style of programming has a long history in computer science, beginning with the formulation of the Lambda Calculus as a foundation for mathematics. It has become increasingly popular in recent years because it offers important advantages in designing, maintaining, and reasoning about programs in modern contexts such as web services, multicore programming, and distributed computing. Course work consists of a series of programming assignments in the Scala programming language and various extensions.

Grading, Honor Code Policy, Processes, and Procedures

Grading will be based on your performance on weekly programming assignments. All work in this class is expected to be your own, and you are expected not to post your solutions or share your work with other students, even after you have taken the course. Please read the Comp 311 Honor Code Policy for more details on how you are expected to work on your assignments. There will also be a final exam, as described in the syllabus.

All students will be held to the standards of the Rice Honor Code, a code that you pledged to honor when you matriculated at this institution. If you are unfamiliar with the details of this code and how it is administered, you should consult the Honor System Handbook. This handbook outlines the University's expectations for the integrity of your academic work, the procedures for resolving alleged violations of those expectations, and the rights and responsibilities of students and faculty members throughout the process.

Accommodations for Students with Special Needs

Students with disabilities are encouraged to contact me during the first two weeks of class regarding special needs. Students with disabilities should also contact Disabled Student Services in the Ley Student Center and the Rice Disability Support Services .

General Information

Course Syllabus
Homework Submission Guide

Office Hours	<table><tr><td>Nick</td><td>Available after class</td><td>-</td><td>-</td></tr><tr><td>Corky</td><td>Wednesday Tuesday, Thursday</td><td>2pm–4pm 9:15am–10:30am</td><td>DCH 3110 DCH 3104</td></tr><tr><td>TAs</td><td>Check Piazza</td><td>-</td><td>-</td></tr></table>
Textbooks	There is no required textbook. We will follow the pedagogic approach of "How to Design Programs" but in a typed context. We will also draw material from a variety of sources, including: • Felleisen, Findler, Flatt, Krishnamurthi. "How to Design Programs." MIT Press 2001. • Harold Abelson, Gerald Jay Sussman, Julie Sussman, "The Structure and Interpretation of Computer Programs." MIT Press 1985. • Odersky, Spoon, Venners. "Programming in Scala." Artima Press 2012. • Chiusano and Bjarnason. "Functional Programming in Scala." Manning Publications Co. August 2014. • Coursera: Functional Programming Principles in Scala by Martin Odersky. • edX: FP101x: Introduction to Functional Programming by Erik Meijer. • Okasaki. "Purely Functional Data Structures." Cambridge University Press. New York, NY. 1999. • The Apache Spark website. • Why is functional programming important ? by Corky Cartwright
Online Videos	• *Working Hard to Keep it Simple*, by Martin Odersky • *Growing a Language*, by Guy L. Steele, Jr. • *What to Leave Implicit,* by Martin Odersky • *Impromptu: A Lightweight, dependently-typed async framework for Scala*, by Jon Pretty
Development Environment	• The DrScala Pedagogic IDE is recommended for all homework assignments in this course. • You may also use IntelliJ IDEA. See the setup instructions for working with Scala projects in this course.

Lecture Schedule (Subject to Change Without Notice)

Conditional Functions on Ranges, Point Values, and Compound Datatypes

Semantics of Type Checking, Binary Methods, Abstract Datatypes

For Expressions, Monads, The Environment Model of Reduction

Call-by-Name, Environment Model of Type Checking, Generative Recursion

Week	Day	Date	Topic	Work Assigned	Work Due
1	Tu	Aug 22	Overview, Motivation		

	Th	Aug 24	What are Types, Core Scala	Hwk 0	
2	Tu	Aug 29	—		
	Th	Aug 31	—		
3	Tu	Sep 05	Doubles, Programming with Intention, The Design Recipe		
	Th	Sep 07	Functions on Ranges, Point Values, Compound Datatypes		
4	Tu	Sep 12	Methods, Grading, DrScala		
	Th	Sep 14	Static Type Checking, Abstract Datatypes	Hwk 1	
5	Tu	Sep 19	Abstract Datatypes 2, Recursively Defined Types		
	Th	Sep 21	Recursively Defined Types 2, Functions as Values		
6	Tu	Sep 26	1st-Class Functions, Imports, Variable & Named Args		
	Th	Sep 28	Exceptions, String Formatting, Generic Types	Hwk 2	Hwk 1
7	Tu	Oct 03	Covariance and Contravariance, ...		
	Th	Oct 05	... Currying, Fold, Flatmap, and For Expressions		
8	Tu	Oct 10	MIDTERM RECESS		
	Th	Oct 12	Type Hierarchy, Overrides, Exceptions, Operators	Hwk 3	Hwk 2
9	Tu	Oct 17	For Expressions, Monads, The Environment Model		
	Th	Oct 19	Scala Collections Classes, Traits		
10	Tu	Oct 24	Generative Recursion		

	Th	Oct 26	Strategies for Generative Recursion	Hwk 4	Hwk 3
11	Tu	Oct 31	Accumulators		
	Th	Nov 02	Functional Data Structures		
12	Tu	Nov 07	Streams, State, Mutation		
	Th	Nov 09	Mechanical Proof Checkin, The Curry-Howard Isomorphism	~~Hwk 5~~	Hwk 4
13	Tu	Nov 14	The State Monad		
	Th	Nov 16	Additional Scala Features, Extractors, Parser Combinators	Hwk 5	
14	Tu	Nov 21	More Parser Combinators, Actors and Concurrency*	Hwk 6 (Optional)	
	Th	Nov 23	THANKSGIVING		
15	Tu	Nov 28	Functional Distributed Computing*		
	Th	Nov 30	Videos: *What to Leave Implicit* and *Impromptu*		Hwk 5 & 6
16	Tu	Dec 05	Study Days (no classes)		
	F	Dec 08	Final Exam (DH 1064 at 2pm)		

* Lectures slides not yet updated from last year are marked with an asterisk.