COMP 302 Winter 2016 Lecture 1

Prakash Panangaden\textsuperscript{1}

\textsuperscript{1}School of Computer Science
McGill University

McGill University, Montréal, January 2016
Welcome to COMP 302

- My name: Prakash Panangaden
Welcome to COMP 302

- My name: Prakash Panangaden
- How should you address me?
Welcome to COMP 302

- My name: Prakash Panangaden
- How should you address me?
- Prof. Panangaden,
Welcome to COMP 302

- My name: Prakash Panangaden
- How should you address me?
  - Prof. Panangaden,
  - not Prof. Prakash!!
Welcome to COMP 302

- My name: Prakash Panangaden
- How should you address me?
- Prof. Panangaden,
- **not** Prof. Prakash!!
- Prakash (no title) is fine also
Welcome to COMP 302

- My name: Prakash Panangaden
- How should you address me?
  - Prof. Panangaden,
  - **not** Prof. Prakash!!
- Prakash (no title) is fine also
- as is Sir.
Introduction

Course Title

Official Title

Programming Languages and Paradigms
TAs

1. Giulia (pronounced Julia) Alberini
2. Stefan Knudsen
3. Ira Kones
4. Patricia Olson
5. David Thibodeau
TAs

1. Giulia (pronounced Julia) Alberini
2. Stefan Knudsen
3. Ira Kones
4. Patricia Olson
5. David Thibodeau
6. Carl Patenaude-Poulin [TEAM Mentor]
Course Administration

1. Lecture notes, assignments and solutions will be posted there.
2. Office hours: Mondays, Fridays 1:30 to 3:00
3. Office location: Room 105N McConnell
4. I will set up a MyCourses page with
   - links to the course web site
   - discussion boards
   - lecture recordings
Grading

- 6 assignments: 40% Submitted through an automated system using myCourses.
Grading

- 6 assignments: 40% Submitted through an automated system using myCourses.
- 1 in-class midterm: 10% of your grade,
Grading

- 6 assignments: 40% Submitted through an automated system using myCourses.
- 1 in-class midterm: 10% of your grade,
- Final exam: 50% of your total grade.
- Cheat sheets for exams: no other notes, no books, no calculators, phones, laptops, smart watches, Google glasses, mirrors or magic owls.
Paradigms

Official definition

a distinct concept or **thought pattern**

1. Functional programming: higher-order, polymorphically typed (F#)
2. Imperative programming (F#)
3. Object-oriented programming: inheritance and subtyping (Java)
4. Stream programming (Scheme)
5. Concurrent programming (in theory)
6. Reactive programming (in theory)
What languages will we use?

- Answer 1: F#, Java, Scheme

Anyone who describes this course as "Programming in F#" does not get it!
What languages will we use?

- Answer 1: F#, Java, Scheme
- Answer 2: not important!
What languages will we use?

- Answer 1: F#, Java, Scheme
- Answer 2: not important!
- Anyone who describes this course as “Programming in F#” does not get it!
Topics

1. Recursion
Course contents

Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
Course contents

Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
5. Higher-order functions
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
5. Higher-order functions
6. Updatable data: references
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
5. Higher-order functions
6. Updatable data: references
7. Environments and binding
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
5. Higher-order functions
6. Updatable data: references
7. Environments and binding
8. Closures and objects
Topics

1. Recursion
2. how to think about it, not how it is implemented with stacks!
3. Inductively defined types and structures
4. Operational semantics
5. Higher-order functions
6. Updatable data: references
7. Environments and binding
8. Closures and objects
9. Some other topics
Topics II

1. Types, typing rules
Topics II

1. Types, typing rules
2. Type inference, polymorphism
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm

---

They are NOT the same thing!
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm
5. Subtyping and inheritance
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm
5. Subtyping and inheritance
6. they are NOT the same thing!
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm
5. Subtyping and inheritance
6. They are **NOT** the same thing!
7. Scheme: the Wild West
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm
5. Subtyping and inheritance
6. they are **NOT** the same thing!
7. Scheme: the Wild West
8. Stream programming
Topics II

1. Types, typing rules
2. Type inference, polymorphism
3. Interpreters, parsers and compilers
4. Object oriented paradigm
5. Subtyping and inheritance
6. They are NOT the same thing!
7. Scheme: the Wild West
8. Stream programming
9. Concurrent programming
We all speak natural language(s) with varying degrees of precision and accuracy.
The role of language

- We all speak natural language(s) with varying degrees of precision and accuracy.
- But sloppy use of language causes confusion in mathematics.
The role of language

- We all speak natural language(s) with varying degrees of precision and accuracy.
- But sloppy use of language causes confusion in mathematics.
- In mathematical discussions we need to use precise mathematical notation.
The role of language

- We all speak natural language(s) with varying degrees of precision and accuracy.
- But sloppy use of language causes confusion in mathematics.
- In mathematical discussions we need to use precise mathematical notation.
- The so-called “popular” books are more confusing than the real thing.
The role of language

- We all speak natural language(s) with varying degrees of precision and accuracy.
- But sloppy use of language causes confusion in mathematics.
- In mathematical discussions we need to use precise mathematical notation.
- The so-called “popular” books are more confusing than the real thing.
- Computers are **completely unforgiving**!!
The role of language

- We all speak natural language(s) with varying degrees of precision and accuracy.
- But sloppy use of language causes confusion in mathematics.
- In mathematical discussions we need to use precise mathematical notation.
- The so-called “popular” books are more confusing than the real thing.
- Computers are completely unforgiving!!
- In order to train our minds we need to learn to speak carefully.
We all speak natural language(s) with varying degrees of precision and accuracy.

But sloppy use of language causes confusion in mathematics.

In mathematical discussions we need to use precise mathematical notation.

The so-called “popular” books are more confusing than the real thing.

Computers are completely unforgiving!!

In order to train our minds we need to learn to speak carefully.

Every sentence has to be constructed with care.
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”; sloppy but we usually understand.
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”; sloppy but we usually understand.
- “I would like eggs and bacon or sausages.” Ambiguity
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”: sloppy but we usually understand.
- “I would like eggs and bacon or sausages.” Ambiguity
- “Dr. Lex Luthor is a former alumni of Gotham State.”
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”; sloppy but we usually understand.
- “I would like eggs and bacon or sausages.” Ambiguity
- “Dr. Lex Luthor is a former alumni of Gotham State.”
- Clearly does not know what “alumnus” means nor what is the singular form. I saw this in a newspaper article.
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”; sloppy but we usually understand.
- “I would like eggs and bacon or sausages.” Ambiguity
- “Dr. Lex Luthor is a former alumni of Gotham State.”
- Clearly does not know what “alumnus” means nor what is the singular form. I saw this in a newspaper article.
- “Hey Prof. I wanna go to an all-night rave tonight, what are you gonna cover tomorrow?”
English sloppiness that drives me crazy

- Writing “can not” instead of “cannot”: causes a real mistake in meaning.
- Mistaking “that” and “which”; sloppy but we usually understand.
- “I would like eggs and bacon or sausages.” Ambiguity
- “Dr. Lex Luthor is a former alumni of Gotham State.”
- Clearly does not know what “alumnus” means nor what is the singular form. I saw this in a newspaper article.
- “Hey Prof. I wanna go to an all-night rave tonight, what are you gonna cover tomorrow?”
- Clear, but drives me insane.
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
We are not studying a dozen languages to pad out your CV.

- **Linguistics**: syntax and semantics.
- **Syntax**: what is a correctly formed sentence/program?
- **Semantics**: what does a sentence mean? (natural)
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
- Commonly written in manuals,
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
- Commonly written in manuals,
- but they are generally vague and ambiguous.
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
- Commonly written in manuals,
- but they are generally vague and ambiguous.
- We will show you (a bit of) formal semantics
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
- Commonly written in manuals,
- but they are generally vague and ambiguous.
- We will show you (a bit of) *formal* semantics
- and typing rules.
What is the study of language?

- We are not studying a dozen languages to pad out your CV.
- Linguistics: syntax and semantics.
- Syntax: what is a correctly formed sentence/program?
- Semantics: what does a sentence mean? (natural)
- Semantics: what does a program do when you run it?
- Commonly written in manuals,
- but they are generally vague and ambiguous.
- We will show you (a bit of) formal semantics
- and typing rules.
The role of abstraction

- Our only technique for handling complexity.
The role of abstraction

- Our only technique for handling complexity.
- Conceptualization without reference to specific instances.
The role of abstraction

- Our only technique for handling complexity.
- Conceptualization without reference to specific instances.
- Isolating the fundamental, essential issues without irrelevant details.
The role of abstraction

- Our only technique for handling complexity.
- Conceptualization without reference to specific instances.
- Isolating the fundamental, essential issues without irrelevant details.
- When designing a data structure: where it will be used is not important.
The role of abstraction

- Our only technique for handling complexity.
- Conceptualization without reference to specific instances.
- Isolating the fundamental, essential issues without irrelevant details.
- When designing a data structure: where it will be used is not important.
- When using a data structure: details of the implementation are not important.
The role of abstraction

- Our only technique for handling complexity.
- Conceptualization without reference to specific instances.
- Isolating the fundamental, essential issues without irrelevant details.
- When designing a data structure: where it will be used is not important.
- When using a data structure: details of the implementation are not important.
- Thinking about all the details is not a virtue.
The abstraction principle

“Every significant piece of functionality should be implemented in just one place in the source code. Where similar functions are carried out by distinct pieces of code, it is generally beneficial to combine them into one by abstracting out the varying parts.” — Benjamin C. Pierce
“Software engineering is all about abstraction. Every single concept, construct and method is entirely abstract. Of course, it does not feel that way to most software engineers. But that’s my point. The main benefit that they got from the mathematics they learned in academia was the experience of rigorous reasoning with purely abstract objects and structures.” — Keith Devlin.
The basic pieces

1. Values
The basic pieces

1. Values
2. Names
The basic pieces

1. Values
2. Names
3. Variables (Locations)
Components of a programming language

The basic pieces

1. Values
2. Names
3. Variables (Locations)
4. Expressions
The basic pieces

1. Values
2. Names
3. Variables (Locations)
4. Expressions
5. Commands
Components of a programming language

Higher-level pieces

1. Combination mechanisms:
   1. control-flow constructs
   2. combinators
Higher-level pieces

1. Combination mechanisms:
   1. control-flow constructs
   2. combinators
2. Parametrized expressions = functions
Components of a programming language

Higher-level pieces

1. Combination mechanisms:
   1. control-flow constructs
   2. combinators

2. Parametrized expressions = functions

3. Parametrized commands = procedures (methods)
Components of a programming language

Higher-level pieces

1. Combination mechanisms:
   1. control-flow constructs
   2. combinators

2. Parametrized expressions = functions

3. Parametrized commands = procedures (methods)

Components of a programming language

Types

1. Classify values
Types

1. Classify values
2. and expressions
Components of a programming language

Types

.Classify values

2. and expressions

3. and functions and procedures
Components of a programming language

Types

1. Classify values
2. and expressions
3. and functions and procedures
4. in order to *restrict* what can be expressed.
Types

1. Classify values
2. and expressions
3. and functions and procedures
4. in order to restrict what can be expressed.
5. Give up expressive power for
Types

1. Classify values
2. and expressions
3. and functions and procedures
4. in order to restrict what can be expressed.
5. Give up expressive power for
6. guarantees of good behaviour.
What to understand

1. Names: binding and scoping
What to understand

1. Names: binding and scoping
2. Evaluation rules: expressions $\rightarrow$ values
What to understand

1. Names: binding and scoping
2. Evaluation rules: expressions $\rightarrow$ values
3. Typing rules,
What to understand

1. Names: binding and scoping
2. Evaluation rules: expressions $\rightarrow$ values
3. Typing rules,
4. which may not be exclusive.