Why Study Programming Languages?
Imperative Programming

Imperative programming is about
• modifying mutable variables
• using assignments
• and control structures such as if-then-else, loops, break, continue, return

The most common way to understand imperative programs is as instruction sequences for a von Neumann computer.
Imperative Programs and Computers

There is a strong correspondence between
- Mutable variables $\approx$ memory cells
- Variable dereferences $\approx$ load instructions
- Variable assignment $\approx$ store instructions
- Control structures $\approx$ jumps

**Problem:** Scaling up! ("von Neumann" bottleneck)

How can we avoid conceptualizing programs on this low-level of abstraction?
Moore's Law
Scaling Up

• In the end, pure imperative programming is limited by the "Von Neumann" bottleneck
  – One tends to conceptualize data structures and algorithms in terms of low-level machine instructions
  – Difficult to scale up in the new area of multicore and distributed computing systems

• We need other techniques for defining high-level programming abstractions
Example: Polynomials in Mathematics

• The sum of two polynomials are defined by laws such as
  \[(a \cdot x + b) + (c \cdot x + d) = (a+b) \cdot x + (b+d)\]

• But there is no operator to change a coefficient while keeping the polynomial the same

    Polynomial p = new Polynomial(a, b);
    p.setCoefficient(0, 42);
The Goals of this Course

Study the fundamental principles of programming languages from a fresh perspective so that

• You can think about computation abstractly, making you better programmers overall

• You can pick up new languages quickly because you thoroughly understand their building blocks

• You can decide which language is best suited for a given task
How will be study Programming Languages?

Not like a trip to the zoo...
Programming Paradigms

Main programming paradigms:
• imperative programming
• functional programming
• logic programming

Orthogonal:
• object-oriented programming
How will be study Programming Languages?

More like a study of Anatomy
Programming Paradigms

Main programming paradigms:
- imperative programming
- functional programming
- logic programming

Orthogonal:
- object-oriented programming

Focus of this course
Questions this course will answer

• **Language Constructs:**
  How can we think about computation and algorithms?
  – algebraic datatypes and pattern matching
  – higher-order functions
  – recursion and continuation passing
  – state and monads
  – objects and classes

• **Language Foundations:**
  How can we represent and reason about programs?
  – syntax and semantics
  – induction
  – types

• **Language Implementations:**
  How can we realize programming languages?
  – interpreters
How will we answer these questions?

• In-class lectures and discussions
  – Lectures to introduce topics and techniques

• Course project: an interpreter for JavaScript
  – Written in Scala
  – Split up into interpreters of increasing complexity
  – Teams of 2 students
But Why?

http://xkcd.com/1312/

http://xkcd.com/1270/
Why build interpreters?

• "I am much more interested in developing..."
  – a dating app
    (surprisingly popular among master's students)
  – a molecular dynamics analysis toolkit
  – anything, as long as it's related to Big Data
  – ...

Why build interpreters?

• They are complex and intricate programs

• But they are not too complex

• They allow us to learn programming language constructs and techniques by implementing these constructs and techniques
Why Scala?

• Scala fuses the **object-oriented** and **functional programming** paradigms
• This is a general trend in languages that are used in industry (see Java 8, C# 6.0, Swift, ...)
• Scala goes furthest in this trend
• It has perhaps the most avant-garde and advanced features of these languages
• Many of these features are extremely useful for writing interpreters
Why Scala?
Why JavaScript?

Brendan Eich – inventor of JavaScript and former CEO of the Mozilla Foundation
Why JavaScript?

• JavaScript is the assembly language of the Web
• It is an interpreted language (although, modern interpreters incorporate JIT/static compilation)
• It combines functional and object-oriented features
• We will focus on a well-behaved subset (reference implementation: node.js)
More Details on Course
Important Dates

• Class meetings
  – Tuesdays and Thursdays, 11am-12:15pm

• Office hours
  – Wednesdays, 3-4pm in WWH 407
  – or by appointment

• Midterm exam
  – Recitation: Friday, Oct 21, 2-3:50pm
  – Exam: Tuesday, Oct 25, 11am-12:15pm

• Final exam
  – Recitation: Friday, Dec 16, 2-3:50pm
  – Tuesday, Dec 20, 10am-11:50am
Resources

Class notes (self-contained, no need to buy a textbook)

Recommended Textbooks:

• Programming in Scala

• Essentials of Programming Languages

There are many Scala resources available on the web for free.
Tools

• ScalaIDE (based on Eclipse)
  – integrated development environment

• sbt
  – command line build tool

• ScalaTest
  – unit testing framework
Expectations

• Class meetings are an integral part of this course
  – You really should attend
  – This is one of the few classes where I will be using slides

• The course home pages (including NYU Classes) are an important part of this course
  – lists important dates
  – course notes
  – homework assignments
  – further reading, links to useful materials
Grading

• 50% for homework assignments
  – weekly
  – alternating between written and programming assignments

• 20% for midterm exam

• 30% for final exam
Late Submission

• Late submissions will be graded with a 10% penalty per day after the deadline.

• No solutions will be accepted one week after the deadline.
A Cautionary Tale
A Cautionary Tale (cont.)

• Karl Theodor zu Guttenberg
  – Used to be secretary of defense in Germany, extremely popular
  – Forced to resign because most of his PhD thesis was plagiarized
    • 94.4% of all pages, 63.8% of all text lines
  – Some choice quotes
    • “The allegation that my thesis is plagiarized is absurd”
    • “I did not consciously or deliberately cheat”
    • “I personally wrote this dissertation”
Rules

• You must submit all assignments on your own
  – Though you should work on them with your partner
• You should do the assignments in groups of 2 students
  – But not with more students or with other groups
• You should help other students and groups on specific technical issues
  – But you must acknowledge such interactions
Introduce yourself in a few minutes

≠
How to get started?

• Subscribe to the class mailing list
  – by tonight

• Find a peer with whom you will work together

• Get ScalaIDE and sbt running on your laptop
  – see notes on today's class in NYU classes syllabus for instructions