Why study PL ?

"A different language is a different vision of life"

- Fellini

- Hypothesis:

  Programming language shapes programming thought

- Characteristics of a language affect how ideas can be expressed in the language

So what does studying PL buy me?

Makes you look at things in different ways, think outside of the box

Knowing language paradigms other than traditional ones will give you new tools to approach problems, even if you are programming in Java

PL Dimensions

- Wide variety of programming languages
- How do they differ?
- along certain dimensions...
- What are these dimensions?

Dimension: Syntax

- Languages have different syntax
  - But the difference in syntax can be superficial
  - C# and Java have different syntax, but are very similar
- In this class, we have looked beyond superficial syntax to understand the underlying principles

Dimension: Computation model

- Functional: Lisp, OCaml, ML
- Imperative: Fortran, C, Python
- Object oriented: C++, Java, C#, Python
- Constraint-based: Prolog, CLP(R)

Dimension: Typing model

- Statically typed: Java, C, C++, C#, OCaml
- Dynamically typed: Lisp, Scheme, Perl, Smalltalk, Python

Dimension: Execution model

- Compiled: C, C++, OCaml
- Interpreted: Perl, Python, shell scripting PLs
- Hybrid: Java

Final words on functional programming

Advantages of functional progs

- Functional programming more concise
  "one line of lisp can replace 20 lines of C"
  (quote from http://www.ddj.com/dept/architect/184414500?pgno=3)

- Recall reverse function in OCaml:

```
let reverse l = fold (::) [] l
```

- How many lines in C, C++?

Can better reason about progs

- No side effects. Call a function twice with same params, produces same value

- As a result, computations can be reordered more easily
- They can also be parallelized more easily

Industry

- From the authors of map reduce: “Inspired by similar primitives in LISP and other languages”

  http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0003.html

- The point is this: programmers who only know Java/C/C++ would probably not have come up with this idea

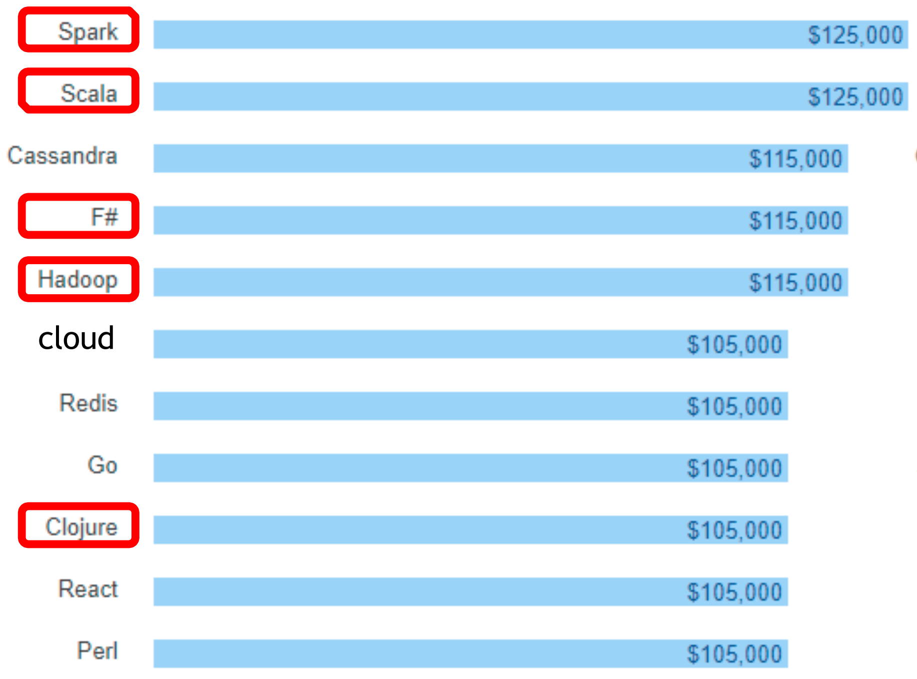
- Many other similar examples in industry

Industry

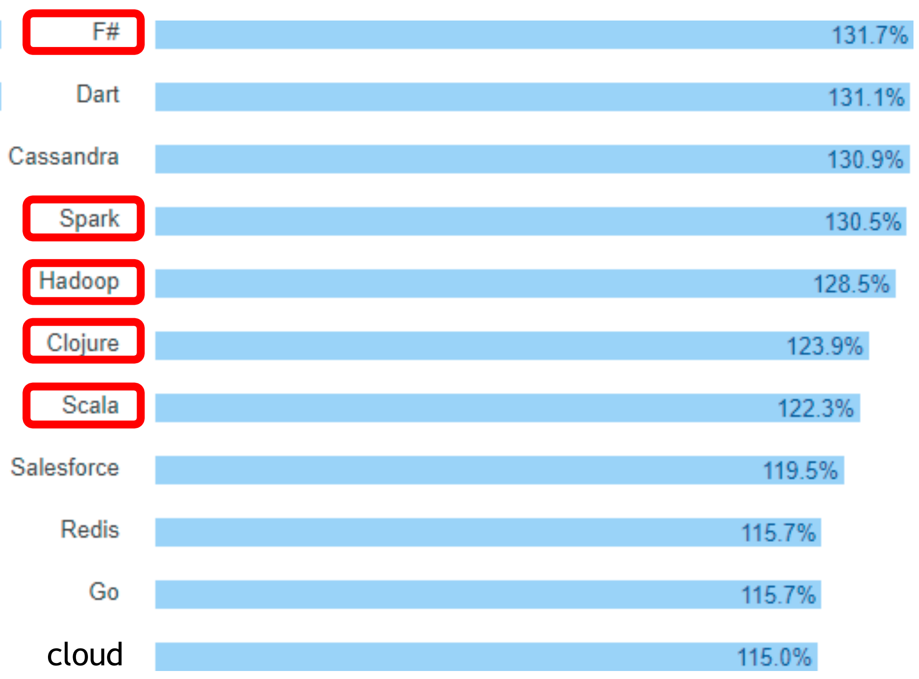
- Microsoft: F#, inspired by Ocaml
  https://channel9.msdn.com/blogs/pdc2008/tl11
- Jane Street Capital: uses Ocaml for their trading software
- Facebook: Infer program analysis tool implemented in OCaml, Sigma malware detection tool and its concurrency library implemented in Haskell
- Google: map reduce, influenced by FP
- Twitter: uses Scala for their back-end (Scala has roots in FP and OO)

Stack Overflow Survey

Top Paying by Language (self reported)


United States

Spark	$125,000
Scala	$125,000
Cassandra	$115,000
F#	$115,000
Hadoop	$115,000
cloud	$105,000
Redis	$105,000
Go	$105,000
Clojure	$105,000
React	$105,000
Perl	$105,000


World

F#	131.7%
Dart	131.1%
Cassandra	130.9%
Spark	130.5%
Hadoop	128.5%
Clojure	123.9%
Scala	122.3%
Salesforce	119.5%
Redis	115.7%
Go	115.7%
cloud	115.0%

🟥 : functional or heavily influenced by functional

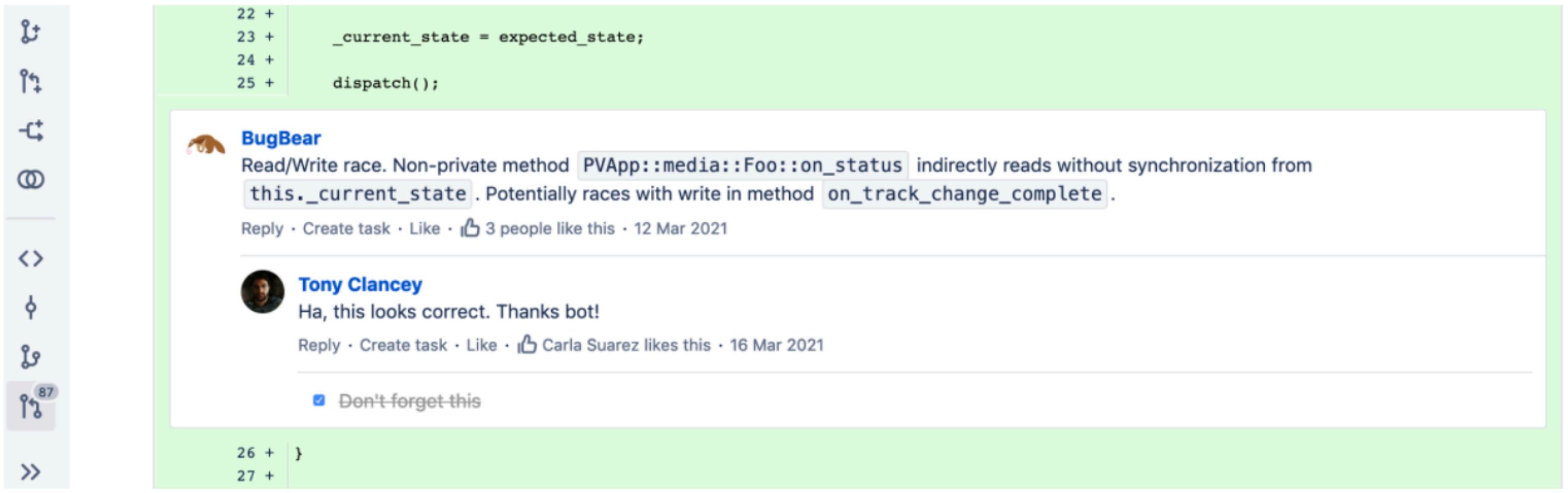
Final words on Constraint Logic Programming

Different way of thinking

- State constraints, and ask solver to get solution
- Very powerful paradigm: separates *constraint generation* from *constraint solving*
- You generate the constraints, and the used off-the-shelf solver

Industry

- Used in Watson, IBM’s Jeopardy-winning computer
- Used in Amazon’s automated-reasoning bot called BugBear for its Prime Video App
    - BugBear – code analyzers for C/C++, Java, and TypeScript



```
    _current_state = expected_state;

    dispatch();
```

BugBear
Read/Write race. Non-private method `PVApp::media::Foo::on_status` indirectly reads without synchronization from `this._current_state`. Potentially races with write in method `on_track_change_complete`.
Reply · Create task · Like · 3 people like this · 12 Mar 2021

Tony Clancey
Ha, this looks correct. Thanks bot!
Reply · Create task · Like · Carla Suarez likes this · 16 Mar 2021

☑ ~~Don't forget this~~

```
}
```

An example of BugBear in action (the names of the program functions and developers have been changed).

Industry

- Suppose we require that, in function F, the function `*open_resource*` should always be called before the function `*use_resource*`.

```
open_called_before_use(F) :-
    call_instruction(F:line1,@open_resource),
    call_instruction(F:line2,@use_resource),
    called_before(@open_resource,@use_resource).
```

- *called_before* imposes constraints on the shape of the so-called call graph.

Python

- Python has a very relaxed philosophy
  - if something "can be done" then it is allowed.

- Combination of dynamic types + everything is an object makes for very flexible, very intuitive code.

No static types

- No static type system to "prohibit" operations.
- No more of that OCaml compiler giving you hard-to-decypher error messages!
- And... No need to formally define the type system (although still need to define the dynamic semantics somehow)

Similarities to Ocaml

- Uniform model: everything is an object, including functions
- Can pass functions around just as with objects
- Supports functional programming style with map and fold

Industry

- Python is Everywhere!
    - Web Development: Instagram, Pinterest, Google incorporate Python in backend web development.
    - Data Science: Netflix uses scipy and numpy for numerical computing to manage user traffic.
    - Machine Learning: PyTorch and Tensorflow are essential Python libraries in ML systems.

OCaml/Python comparison

	OCaml	Python
PL paradigm	functional	OO/imperative
Basic unit	Expr/value	Objects/ instances
Types	statically	dynamicaclly
DataModel	env lookup	"pointers" to mutable objs

Dynamic vs. Static, OO vs. Func

	Statically typed	Dynamically typed
OO	Java	Python, Smalltalk
Functional	Ocaml, Haskell	Lisp/Scheme