

# CSE 130-230 : Fall 2017 Programming Languages

Sorin Lerner  
UC San Diego



## Hi! My name is Sorin



## Why study PL ? (discussion)

## 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

## Course Goals



*"Free your mind"*  
-Morpheus

- You will learn several new
- languages and constructs
  - ways to describe and organize computation

Yes, you *can* do that in Java/Assembly but ...

## So what does studying PL buy me?

Enables you to create software that is

- Readable
- Correct
- Extendable
- Modifiable
- Reusable

## So what does studying PL buy me?

Will help you learn new languages

- There was no Java (C#) 15 (10) years ago
- Will learn the **anatomy** of a PL
- Fundamental **building blocks** of languages reappear in different guises in different languages and different settings
- Re-learn the languages you already know

## So what does studying PL buy me?

Enables you to design new languages  
“who, me?”

Buried inside any **extensible** system is a PL

- Emacs: E-Lisp
- Word, Powerpoint: VBScript
- Quake: QuakeC
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML...

## So what does studying PL buy me?

Enables you to design new languages  
“who, me?”

Companies develop general purpose PLs/paradigm!

- Google: MapReduce
- Mozilla: Rust
- RedHat: Ceylon
- Nvidia: CUDA

## So what does studying PL buy me?

Enables you to better choose the right language

“but isn’t that decided by

- libraries,
- standards,
- and my boss ?”

Yes. Chicken-and-egg.



My goal: educate tomorrow’s tech leaders & bosses  
So you’ll make considered, informed choices

## 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?

## PL Dimensions (discussion)

## 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, will look beyond superficial syntax to understand the underlying principles

## Dimension: Computation model

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

## Dimension: Memory model

- Explicit allocation-deallocation: C, C++
- Garbage collection: Smalltalk, Java, C#
- Regions: safe versions of C (e.g. Cyclone)
  - allocate in a region, deallocate entire region at once
  - more efficient than GC, but no dangling ptrs

## Dimension: Typing model

- Statically typed: Java, C, C++, C#
- Dynamically typed: Lisp, Scheme, Perl, Smalltalk
- Strongly typed (Java) vs. weakly typed (C, C++)

## Dimension: Execution model

- Compiled: C, C++
- Interpreted: Perl, shell scripting PLs
- Hybrid: Java
- Is this really a property of the language?  
Or the language implementation?
- Depends...

## So many dimensions

- Yikes, there are so many dimensions!
- How to study all this!
  
- One option: study each dimension in turn
  
- In this course: explore the various dimensions by looking at a handful of PLs

## Course material

### Outline:

1. Functional, OCaml, 4 weeks
2. OO, Python, 4 weeks
3. Logic, Prolog, 1 weeks

### No recommended Text:

- Online lecture notes
- Resources posted on webpage
- Pay attention to lecture and section!

## Course Mechanics

[www.cs.ucsd.edu/classes/fa17/cse130-a/](http://www.cs.ucsd.edu/classes/fa17/cse130-a/)  
(Google “Sorin Lerner”, follow “Teaching Now”)

Nothing printed, everything on Webpage!

Piazza: [sign-up using link on web page](#)

TAs: [See web page](#)

Tutors: [See web page](#)

## Requirements and Grading

- Prog. Assignments (7): 30%
  
- Midterm (only cheat-sheet): 35%
  
- Final (only cheat-sheet): 35%

## Weekly Programming Assignments

Schedule up on webpage

### Deadline Extension:

- Four “late days”, used as “whole unit”
- 5 mins late = 1 late day
- Plan ahead, **no other extensions**

PA #1 online, due Sep 6th

## Academic Integrity

- Programming Assignments done **ALONE**
  
- We use plagiarism detection software
  - Have code from **all previous classes**
  - Have code from public repos
  - MOSS is fantastic at finding plagiarism
  - Make your repo private, or you will be found responsible
  
- **Cases referred to AI office**
- See <https://ucsd-pl.github.io/cse-130-230/fa17/grading.html>

## Weekly Programming Assignments

Unfamiliar languages  
+ Unfamiliar environments

**Start Early!**

## Weekly Programming Assignments



**Forget** Java, C, C++ ...  
... other 20<sup>th</sup> century PLs

**Don't complain**  
... that Ocaml is hard  
... that Ocaml is @!#@%

## Immerse yourself in new language



*Free your mind.*

## Enough with the small talk

?

## Say hello to OCaml

```
void sort(int arr[], int beg, int end){
  if (end > beg + 1){
    int piv = arr[beg];
    int l = beg + 1;
    int r = end;
    while (l != r-1){
      if(arr[l] <= piv)
        l++;
      else
        swap(&arr[l], &arr[r--]);
    }
    if(arr[l]<=piv && arr[r]<=piv)
      l=r;
    else if(arr[l]<=piv && arr[r]>piv)
      (l++; r--);
    else if (arr[l]>piv && arr[r]<=piv)
      swap(&arr[l++], &arr[r--]);
    else
      r=l-1;
    swap(&arr[r-], &arr[beg]);
    sort(arr, beg, r);
    sort(arr, l, end);
  }
}
```

Quicksort in C

```
let rec sort l =
  match l with [] -> []
  | (h::t) ->
    let (l,r)= List.partition ((<=) h) t in
    (sort l)@h::(sort r)
```

Quicksort in OCaml

## Why readability matters...

```
sort= (( $: @ (<#[]), (= #[]), $: @ (>#[])) ( { ~ ? @ # } ) ^: ( 1 : < # )
```

Quicksort in J

## Say hello to OCaml

---

```
let rec sort l =
  match l with [] -> []
  | (h::t) ->
    let (l,r)= List.partition ((<=) h) t in
    (sort l)@h::(sort r)
```

Quicksort in OCaml

## Plan (next 4 weeks)

---

### 1. Fast forward

- Rapid introduction to what's in OCaml

### 2. Rewind

### 3. Slow motion

- Go over the pieces individually