A Programming Language

- Two variables
  - x, y
- Three operations
  - x++
  - x--
  - (x=0)? L1:L2;

L1: x++;
   y--;
   (y=0)?L2:L1
L2: ...

Fact: This is “equivalent to” to every PL!

Good luck writing quicksort
... or Windows, Google, Spotify!

So why study PL?

“A different language is a different vision of life”
- Federico Fellini
<table>
<thead>
<tr>
<th>Course Goals</th>
<th>Learn New Languages/Constructs</th>
</tr>
</thead>
</table>
| "Free your mind" | New ways to:
- describe
- organize
- think about computation

- Morpheus
Goal: Enable you to Program

- Readable
- Correct
- Extendable
- Modifiable
- Reusable

Goal: How to learn new PLs

No Java (C#) 15 (10) years ago

Learn the anatomy of a PL
- Fundamental building blocks
- Different guises in different PLs

Re-learn the PLs you already know
Goal: How to design new PLs

...“who, me?”

Buried in every extensible system is a PL
- Emacs, Android: Lisp
- Word, Powerpoint: Macros, VBScript
- Unreal: UnrealScript (Game Scripting)
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML ...

Choose Right Language

Enables you to choose right PL

“...but isn’t that decided by
- libraries,
- standards,
- and my boss?”
Yes.

Speaking of Right and Wrong...

My goal: educate tomorrow’s tech leaders & bosses, so you’ll make informed choices
Imperative Programming

WTF?

$x = x+1$

Imperative = Mutation
Imperative = Mutation

Bad!

Don’t take my word for it

Tim Sweeney (Epic, Creator of UNREAL)

“In a concurrent world, imperative is the wrong default”
Functional

No Assignment.
No Mutation.
No Loops.

OMG! Who uses FP?!

So, Who Uses FP?

Google
MapReduce

So, Who Uses FP?

Microsoft
Linq, F#
So, Who Uses FP?

Facebook

Erlang

Scala

Wall Street
(all of the above)

...CSE 130
Course Mechanics

cseweb.ucsd.edu/classes/sp12/cse130-a/

Nothing printed, everything on Webpage!

Peer Instruction/Clickers

• Make class interactive
  - Help YOU and ME understand what's tricky

• Clickers Not Optional
  - Cheap ones are fine
  - 5% of your grade
  - Respond to 75% questions

• Seating in groups (links on Piazza)

• Bring laptop if you have one
In Class Exercises

1. Solo Vote: Think for yourself, select answer

2. Discuss: Analyze Problem in Groups of 3
   - Practice analyzing, talking about tricky notions
   - Reach consensus
   - Have questions, raise your hand!

3. Group Vote: Everyone in group votes
   - Must have same vote to get points

4. Class-wide Discussion:

Requirements and Grading

- The good news: No Homework
- In-Class Exercises: 5%
- Midterm: 30%
- Programming Assignments (7-8): 30%
- Final: 35%

Grading on a curve. Two hints/rumors:
1. Lot of work
2. Don’t worry (too much) about grade

No Recommended Text

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

Suggested Homeworks

- On webpage after Thursday lecture
- Based on lectures, section of previous Tue, Thu
- Recommended, ungraded, HW problems are sample exam questions
- Webpage has first samples already
**Weekly Programming Assignments**

Schedule up on webpage

Due on **Friday 5 PM**

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

---

**Plan**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. FP,</td>
<td><strong>Ocaml</strong>,</td>
<td>4 weeks</td>
</tr>
<tr>
<td>2. OO,</td>
<td><strong>Scala</strong>,</td>
<td>4 weeks</td>
</tr>
<tr>
<td>3. Logic,</td>
<td><strong>Prolog</strong>,</td>
<td>1 week</td>
</tr>
</tbody>
</table>

---

**Weekly Programming Assignments**

**Unfamiliar languages**
+ **Unfamiliar environments**

**Start Early!**

---

**Weekly Programming Assignments**

**Scoring = Style + Test suite**

**No Compile, No Score**
Weekly Programming Assignments

Forget Java, C, C++ …
… other 20th century PLs

Don’t complain
… that Ocaml is hard
… that Ocaml is @!%@#

Immerse yourself in new language

It is not.

Free your mind.

Word from our sponsor …

- Programming Assignments done ALONE
- We use plagiarism detection software
  - I am an expert
  - Have code from all previous classes
  - MOSS is fantastic, plagiarize at your own risk
- Zero Tolerance
  - offenders punished ruthlessly
- Please see academic integrity statement
Why readability matters…

To Ask Me Questions?

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+1;
    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

Why readability matters…
!

sort=:(($:@(<#],[),(=#[),]$:@(#<])({~ ?@#))^: (1:<#))  //  Quicksort in J

Quicksort in Ocaml

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

```ocaml
let rec sort xs =  
   match xs 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 ML

2. Rewind

3. Slow motion  
   - Go over the pieces individually

ML: History, Variants

“Meta Language”
Designed by Robin Milner  
To manipulate theorems & proofs

Several dialects:  
- Standard ML (SML)  
  - Original syntax
- Objective Caml: (Ocaml)  
  - “The PL for the discerning hacker”  
    - State-of-the-art, extensive library, tool, user support
- F# (Ocaml+.NET) released in Visual Studio

ML’s holy trinity

- Everything is an **expression**
- Everything has a **value**
- Everything has a **type**
Interacting with ML

“Read-Eval-Print” Loop

Repeat:
1. System reads expression e
2. System evaluates e to get value v
3. System prints value v and type t

What are these expressions, values and types?

Base type: Integers

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2+2</td>
<td>4</td>
</tr>
<tr>
<td>2 * (9+10)</td>
<td>38</td>
</tr>
<tr>
<td>2 * (9+10) -12</td>
<td>26</td>
</tr>
</tbody>
</table>

Complex expressions using “operators”: (why the quotes?)
- +, -, *
- div, mod

Base type: Strings

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ab”</td>
<td>string</td>
</tr>
<tr>
<td>“ab” ^ “xy”</td>
<td>“abxy”</td>
</tr>
</tbody>
</table>

Complex expressions using “operators”: (why the quotes?)
- Concatenation ^

Base type: Booleans

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>1 &lt; 2</td>
<td>true</td>
</tr>
<tr>
<td>“aa” = “pq”</td>
<td>false</td>
</tr>
<tr>
<td>(“aa” = “pq”) &amp;&amp; (1&lt;2)</td>
<td>false</td>
</tr>
<tr>
<td>(“aa” = “aa”) &amp;&amp; (1&lt;2)</td>
<td>true</td>
</tr>
</tbody>
</table>

Complex expressions using “operators”:
- “Relations”: =, <, <=, >=
- &&, ||, not
**Type Errors**

Untypable expression is rejected
- No casting, No coercing
- Fancy algorithm to catch errors
- ML's single most powerful feature (why?)

**Complex types: Product (tuples)**

(2+3) || ("a" = "b")

“pq” ^ 9

(2 + "a")

**Complex types: Lists**

(2+2  , 7>8)

(int * bool)

(9-3,"ab"^^"cd", (2+2 , 7>8))

(6, “abcd”, (4,false))

(int * string * (int * bool))

- Triples,…
- Nesting:  
  - Everything is an expression
  - Nest tuples in tuples

**Complex types: Product (tuples)**

[(1,"ab"^^"cd") ; (3+4,"c")];

[(1,"ab") ; (7,"c")]

(int*string) list

[[1] ; [2;3]; [4;5;6]];  

(int list) list

- Unbounded size
- Can have lists of anything (e.g. lists of lists)
- but …
Complex types: Lists

All elements must have same type

Complex types: Lists

List operator “Cons” ::

Can only “cons” element to a list of same type

Complex types: Lists

List operator “Append” @

Can only append two lists ... of the same type

Complex types: Lists

List operator “head” hd

Only take the head a nonempty list
Complex types: Lists

List operator “tail” \( \text{tl} \)

Only take the tail of nonempty list \( \text{tl} \ [;] \);

Recap: Tuples vs. Lists?

What’s the difference?

- **Tuples:**
  - Different types, but fixed number:
    - pair = 2 elts
    - triple = 3 elts
  - (3, “abcd”), (int * string)
  - (3, “abcd”, (3.5, 4.2)), (int * string * (float * float))

- **Lists:**
  - Same type, unbounded number:
    - [3;4;5;6;7], int list

So far, a fancy calculator...

... what do we need next?