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!

So why study PL?

“*A different language is a different vision of life*”
- Federico Fellini

Good luck writing quicksort
... or Windows, Google, Spotify!
So why study PL?

Programming language shapes Programming thought

Language affects how:
• Ideas are expressed
• Computation is expressed

Course Goals
“Free your mind”
-Morpheus

Learn New Languages/Constructs
New ways to:
- describe
- organize
- think about computation
Goal: Enable you to Program

- Readable
- Correct
- Extendable
- Modifiable
- Reusable

Learn How To Learn

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

To Design New Languages
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

x = x+1

WTF?

x = x+1

Imperative = Mutation
Imperative = Mutation

Bad!

Don’t take my word for it

John Carmack
Creator of FPS: Doom, Quake,…

I am starting to remove op= operator overloads to discourage variable mutation.

Don’t take my word for it

Tim Sweeney (Epic, Creator of UNREAL)

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

Functional Programming

Gears of War 3
Functional Programming?

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?

Erlang

Scala

Wall Street
(all of the above)

...CSE 130
Course Mechanics

http://ucsd-progsys.github.io/cse130/

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
   + Reach consensus
   + Have questions, raise your hand!
3. **Group Vote**: Everyone in group votes
   + Must have same vote to get points
3. **Class Discuss**: Everyone in group votes
   • What was easy/hard?

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

1. FP, Ocaml, 4 weeks
2. OO, Scala, 4 weeks
3. Logic, Prolog, 1 week

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.

Immerse yourself in new language

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
To Ask Me Questions?

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

Why readability matters...

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

Quick sort in C

```c
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);
  }
}
```

Quick sort in C

Quick sort in J

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

Quick sort in J
Say hello to OCaml

```ocaml
code
let rec sort xs =
    match xs with
    | [] -> []
    | h::t ->
      let (l, r) = List.partition (fun x -> x <= 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

\[
\begin{align*}
2 & \quad 2 \\
2+2 & \quad 4 \\
2 \times (9+10) & \quad 38 \\
2 \times (9+10) -12 & \quad 26 \\
\end{align*}
\]

Complex expressions using “operators”:
- +, -, *
- div, mod

Base type: Strings

```
"ab" ^ "xy"  ->  "abxy"
```

Complex expressions using “operators”:
- Concatenation ^

Base type: Booleans

\[
\begin{align*}
\text{true} & \quad \text{true} \\
\text{false} & \quad \text{false} \\
1 < 2 & \quad \text{true} \\
"aa" = "pq" & \quad \text{false} \\
("aa" = "pq") && (1<2) & \quad \text{false} \\
("aa" = "aa") && (1<2) & \quad \text{true} \\
\end{align*}
\]

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

Unbounded size
• Can have lists of anything (e.g. lists of lists)
• but ...

Triples,…
• Nesting:
  - Everything is an expression
  - Nest tuples in tuples
Complex types: Lists

All elements **must have same type**

```
[1; "pq"];  
```

Complex types: Lists

List operator “Cons” `::`

```
1::[ ];  
1::[2];  
“a”: [“b”; “c”];  
[“a”; “b”; “c”];  
```

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

```
1::[“b”; “cd”];  
```

Complex types: Lists

List operator “Append” `@`

```
[1;2]@[3;4;5];  
[1;2;3;4;5] int list  

[“a”]@[“b”];  
[“a”; “b”] string list  

[]@[1];  
[1] string list  
```

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

```
1 @ [2;3];  
```

Complex types: Lists

List operator “head” `hd`

```
1 [1;2];  
1 [“a”];  
```

Only take the head a nonempty list

```
hd [];  
```
Complex types: Lists

List operator “tail” \( \texttt{tl} \)

\[
\text{tl } [1;2;3]; \quad [2;3]
\]
\[
\text{tl } ([“a”]@[“b”]); \quad [“b”]
\]

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

Recap: Tuples vs. Lists?

What’s the difference?

- **Tuples:**
  - Different types, but **fixed** number:
  - pair = 2 elts
  - triple = 3 elts
  
  \[
  (3, “abcd”) \quad \text{(int * string)}
  \]
  
  \[
  (3, “abcd”,(3.5,4.2)) \quad \text{(int * string * (float* float))}
  \]

- **Lists:**
  - **Same** type, **unbounded** number:
  
  \[
  [3;4;5;6;7] \quad \text{int list}
  \]

So far, a fancy calculator...

... what do we need next?