

CSE 130 : Spring 2015

Programming Languages

Lecture 1: Hello, World!

Ranjit Jhala
UC San Diego



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

So why study PL ?

Programming language
shapes
Programming thought

Course Goals



“Free your mind”
-Morpheus

So why study PL ?

Language affects how:

- Ideas are expressed
- Computation is expressed

Learn New Languages/Constructs

Lucretio da Pisto
English version by
Hubert Thomas Martin

Wolfgang Amadeus Mozart

Overture

Andante

Allegro

A page of musical notation for Wolfgang Amadeus Mozart's Overture. The score is written for piano and includes various musical notations such as notes, rests, and dynamic markings. The tempo markings 'Andante' and 'Allegro' are visible.

New ways to:

- describe
- organize
- think about
computation

Goal: Enable you to Program



- Readable
- Correct
- Extendable
- Modifiable
- Reusable



I WANT YOU

Learn How To Learn

Goal: How to learn new PLs

No Java (C#) 15 (10) years ago
AJAX? Python? Ruby? Erlang? F#?...

Learn the **anatomy** of a PL

- Fundamental **building blocks**
- Different guises in different PLs

Re-learn the PLs you already know



I WANT YOU

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 ...

Enables you to choose right PL

“...but isn't that decided by

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

Yes.



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



Choose Right Language

Speaking of **Right** and **Wrong**...

Imperative Programming

$x = x + 1$

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”*



Don't take my word for it

John Carmack

Creator of FPS: Doom, Quake,...



Functional
Programming

Functional

No Assignment.

No Mutation.

No Loops.

OMG! Who uses FP?!

So, Who Uses FP ?

So, Who Uses FP ?

The Google logo, consisting of the word "Google" in its characteristic multi-colored font.

MapReduce



Microsoft®

Linq, F#

So, Who Uses FP ?

So, Who Uses FP ?

facebook



twitter

Erlang

Scala

So, Who Uses FP ?

So, Who Uses FP ?

**Wall Street
(all of the
above)**

...CSE 130

Course Mechanics

Peer Instruction (ish)

Mechanics

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

Nothing printed, everything on Webpage!

Peer Instruction/Clickers

- Make class interactive
 - Help YOU and ME understand whats 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:

No Recommended Text

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

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

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

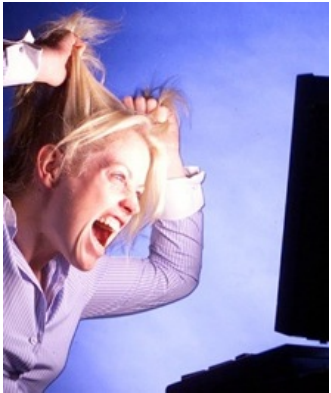
Start Early!

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



I WANT YOU

Free your mind.

Immerse yourself in new language

It is not.

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

```

while(mdis=ul){ /*showing values of the multiple between the limits*/
    printf("%10d",mdis); /*10 spaces for each value, right align*/

    if((mdis%2)!=1) /*Counting odds and evens*/
        even++;
    else
        odd++;
    rsum+=disMult; /*sum up all multiple values in the row*/
    counter++;
    disMult*=mult; /*set display value to next multiple value*/
    /*do not display a multiple value of zero*/
    if(disMult==0)
        disMult+=mult;
    /*display the sum of the row at the end of the row*/

    if(counter==5){
        printf("%10d\n",rSum);
        sub=rSum; /*add the sum of the row to the tot
        rSum=0; /*reset row sum for new row*/
        counter=1; /*reset counter for new row*/
    }

    /*display blank spaces to keep table nice*/
    while((counter<=5)&&(counter!=1)){
        printf(" ");
        counter++;
    }
    /*display the sum of the row at the end of the row*/
    if(counter==5){
        printf("%10d",rSum);
        sub+=rSum;
        rSum=0;
        counter=1;
    }

    /*display total number of odd and even multiples and the sum of all values*/
    printf("\n\nThere are %d odd and %d even numbers.\n",odd,even);
    printf("The sum of all numbers is: %d\n\n",sub); /*showing the sum*/
    /*reset values for next repetition*/
    even=0;
    odd=0;
    sub=0;
}

return 0;
}

```



I WANT YOU

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

```

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...

```

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

```

Quicksort in J

Say hello to 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

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

Plan (next 4 weeks)

1. Fast forward

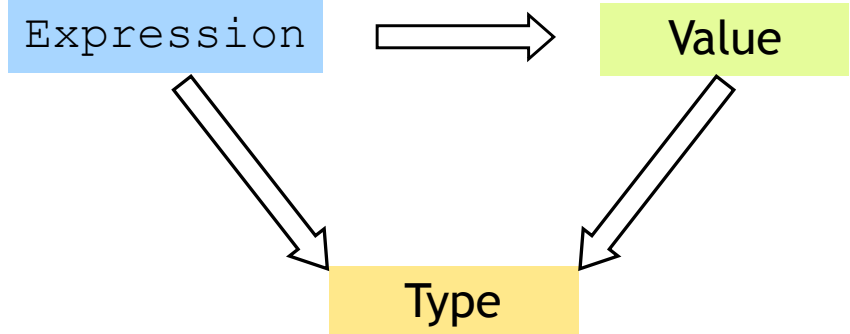
- Rapid introduction to what's in ML

2. Rewind

3. Slow motion

- Go over the pieces individually

ML's holy trinity



- Everything is an **expression**
- Everything has a **value**
- Everything has a **type**

Interacting with ML

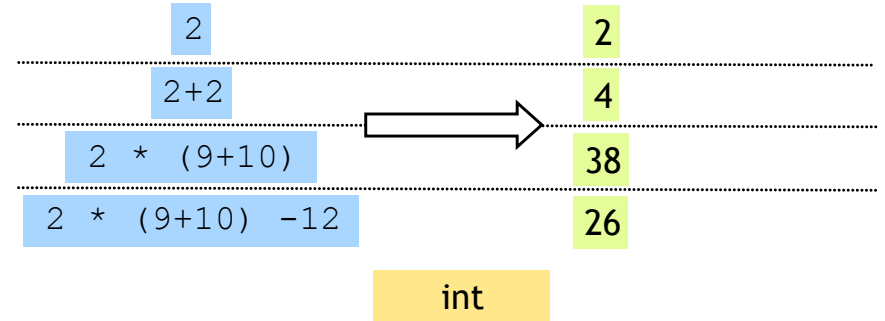
“Read-Eval-Print” Loop

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 ?

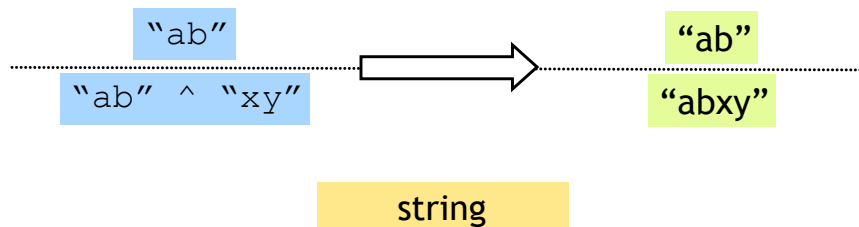
Base type: Integers



Complex expressions using “operators”:(*why the quotes ?*)

- +, -, *
- div, mod

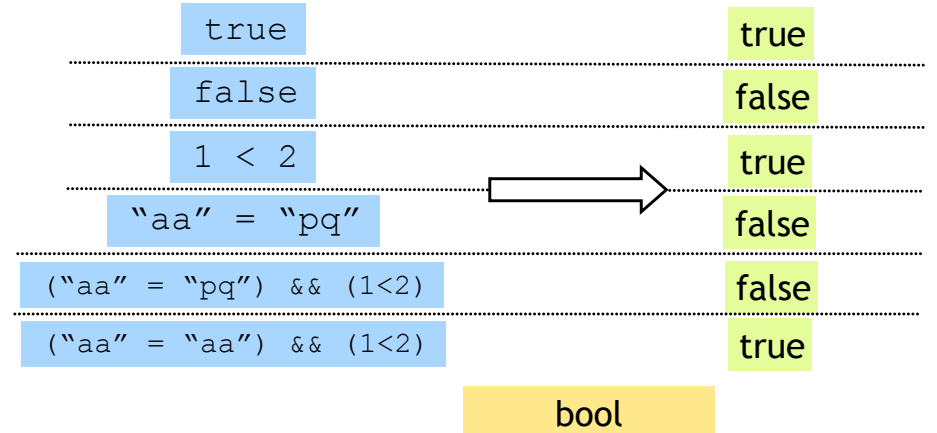
Base type: Strings



Complex expressions using “operators”:(*why the quotes ?*)

- Concatenation ^

Base type: Booleans



Complex expressions using “operators”:

- “Relations”: =, <, <=, >=
- &&, ||, not

Type Errors

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

`"pq" ^ 9`

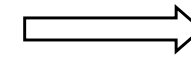
`(2 + "a")`

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+2 , 7>8);`



`(4,false)`

`int * bool`

Complex types: Product (tuples)

`(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: Lists

<code>[];</code>	<code>[]</code>	<code>'a list</code>
<code>[1;2;3];</code>	<code>[1;2;3]</code>	<code>int list</code>
<code>[1+1;2+2;3+3;4+4];</code>	<code>[2;4;6;8]</code>	<code>int list</code>
<code>["a";"b"; "c"^^"d"];</code>	<code>["a";"b"; "cd"]</code>	<code>string list</code>
<code>[(1, "a"^^"b"); (3+4, "c")];</code>	<code>[(1,"ab");(7,"c")]</code>	<code>(int*string) list</code>
<code>[[1]; [2;3]; [4;5;6]];</code>	<code>[[1];[2;3];[4;5;6]];</code>	<code>(int list) list</code>

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

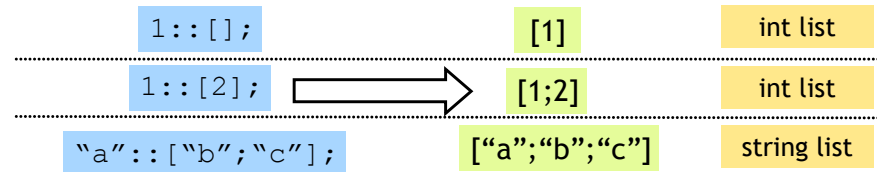
Complex types: Lists

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

All elements must have same type

Complex types: Lists

List operator "Cons" ::

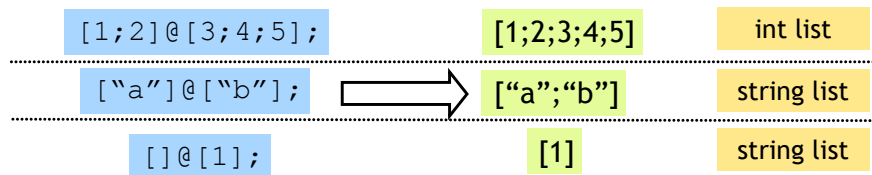


Can only "cons" element to a list of same type

```
1 :: ["b"; "cd"];
```

Complex types: Lists

List operator "Append" @

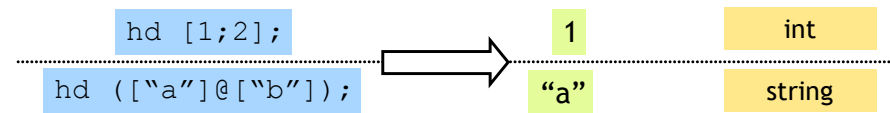


Can only append two lists `1 @ [2;3];`

... of the same type `[1] @ ["a"; "b"];`

Complex types: Lists

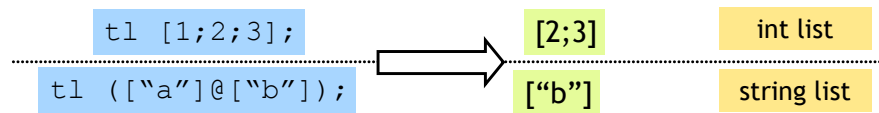
List operator "head" hd



Only take the head a nonempty list `hd [];`

Complex types: Lists

List operator “tail” `tl`



Only take the tail of nonempty list `tl [];`

So far, a fancy calculator...

... what do we need next ?

Recap: Tuples vs. Lists ?

What's the difference ?

- Tuples:

- Different types, but fixed number:

`(3, "abcd")` (int * string)

- pair = 2 elts

`(3, "abcd", (3.5, 4.2))` (int * string * (float * float))

- triple = 3 elts

- Lists:

- Same type, unbounded number:

`[3;4;5;6;7]` int list