

CSE 130 : Fall 2011

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 language is “equivalent” to **every** PL!
Try writing quicksort ... or Windows, Google,
Spotify,... in this language.

So why study PL ?



*“A different language is a
different vision of life”*

- Federico Fellini

So why study PL ?

Programming language
shapes
Programming thought

So why study PL ?

PL characteristics affects how:

- Ideas are expressed
- Computation is expressed

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Course Goals



“Free your mind”
-Morpheus

Course Goals: Learn



Languages/Constructs

New ways to:

- describe
- organize
- think about computation

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Enables you to



Write software that's

- Readable
- Correct
- Extendable
- Modifiable
- Reusable



On the internet,
nobody knows you're
a dog...

...a Ruby Program
...a Scala Program
...an Erlang Program
...

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Enables you 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

Understand PLs you already know



Enables you to design new PLs

...“who, me ?”

Buried in **every extensible** system is a PL

- Emacs: Lisp
- Word, Powerpoint: Macros, VBScript
- Unreal: UnrealScript (Game Scripting)
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML ...

Tuesday, September 27, 2011

Tuesday, September 27, 2011

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

Tuesday, September 27, 2011

Mechanics

cseweb.ucsd.edu/classes/fa11/cse130-a/

Nothing printed, everything on Webpage!

Meetings:

- Lectures: [Ranjit Jhala](#), Tu-Th 6:30-7:50pm @ CENTER 212
- TAs: [Ming Kawaguchi](#), [Ross Tate](#)
- Tutor: [Ayelet Bitton](#)
- Office Hours: RJ (Th 2-4p)
 - Hours posted on web page (may change a bit)

Tuesday, September 27, 2011

Material

Outline:

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

No recommended Text:

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

Tuesday, September 27, 2011

Requirements and Grading

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

Grading on a curve. Two hints/rumors:

1. Lot of work
2. Don't worry (too much) about grade

Tuesday, September 27, 2011

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**

[PA #1](#) online, due 4/8, 5:00 PM

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Weekly Programming Assignments

Unfamiliar languages
+ Unfamiliar environments

Start Early!

Tuesday, September 27, 2011

Weekly Programming Assignments

Scoring = Style + Test suite

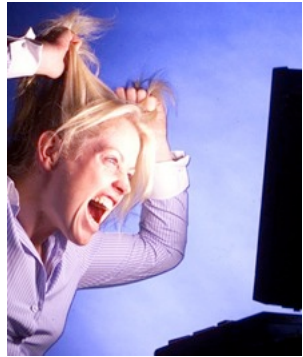
No Compile, No Score

Tuesday, September 27, 2011

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



Free your mind.

Tuesday, September 27, 2011

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

Tuesday, September 27, 2011

Tuesday, September 27, 2011

```
while(mdis%4){ /*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++;
        rsum=mdis;
        counter++;
        mdis+=mult;
    }
    if(mdis%2==1){
        odd++;
    }
    if(counter==5){ /*sumin at end of each row*/
        printf("\n");
        sum+=sum; /*running total sum by adding row sums*/
        rsum=0; /*reset row sum for next row*/
        counter=1; /*reset counter for next row*/
    }
    while((counter == 5) && (counter != 1)){ /*placing spaces to keep alignment*/
        printf(" ");
        counter++;
    }
    if(counter==5){ /*summing at end of the last row*/
        sum+=sum;
        counter = 1;
    }
    printf("\n\nThere are %d odd and %d even numbers.\n",odd,even); /*stating the
    printf("The sum of all numbers is: %d\n",sum); /*showing the sum*/
    /*reseting the value for next time running through progr
    even=0;
    odd=0;
    sum=0;
    }
    return 0;
}
```

Tuesday, September 27, 2011

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+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

Tuesday, September 27, 2011

Tuesday, September 27, 2011

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

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Plan (next 4 weeks)

1. Fast forward

- Rapid introduction to whats 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



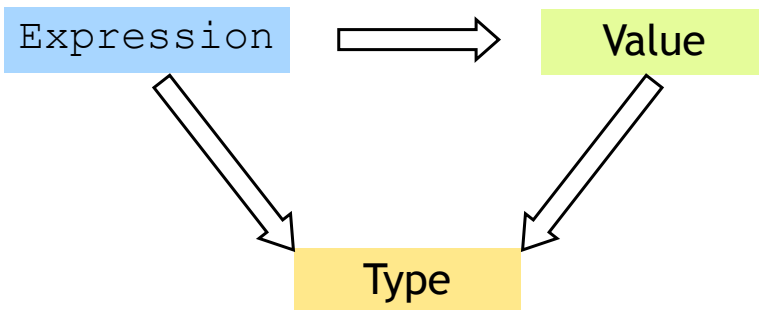
Several dialects:

- 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

Tuesday, September 27, 2011

Tuesday, September 27, 2011

ML's holy trinity



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

Interacting with ML

“Read-Eval-Print” Loop

Repeat:

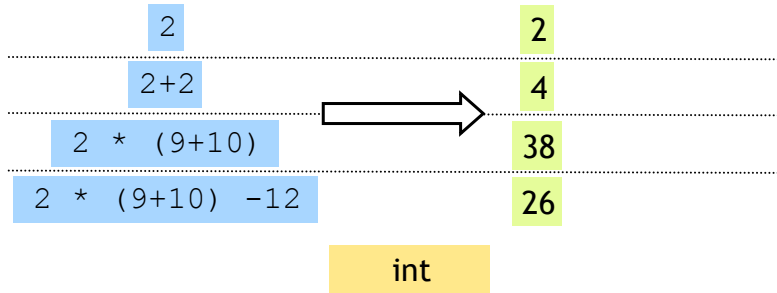
1. System reads expression **e**
2. System evaluates **e** to get value **v**
3. System prints value **v** and type **t**

What are these **expressions**, **values** and **types** ?

Tuesday, September 27, 2011

Tuesday, September 27, 2011

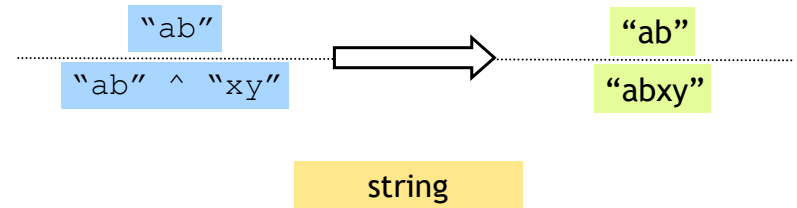
Base type: Integers



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

- `+`, `-`, `*`
- `div`, `mod`

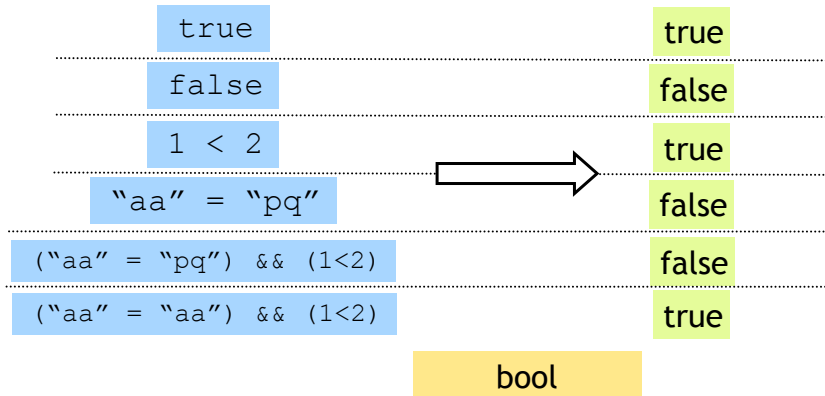
Base type: Strings



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`

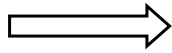
`(2 + "a")`

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) ;`

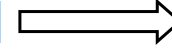


`(4,false)`

`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

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Complex types: Lists

`[] ;`

`[]`

`'a list`

`[1;2;3] ;`

`[1;2;3]`

`int list`

`[1+1;2+2;3+3;4+4] ;`

`[2;4;6;8]`

`int list`

`["a";"b"; "c"^^"d"] ;`

`["a";"b"; "cd"]`

`string list`

`[ (1, "a"^^"b") ; (3+4, "c") ] ;`

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

`(int*string) list`

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

`[[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

`[1; "pq"] ;`

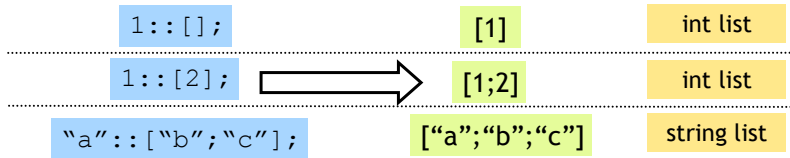
All elements **must** have same type

Tuesday, September 27, 2011

Tuesday, September 27, 2011

Complex types: Lists

List operator “Cons” `::`

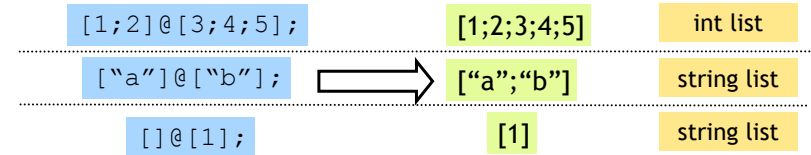


Can only “cons” element to a list of **same type**

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

Complex types: Lists

List operator “Append” `@`

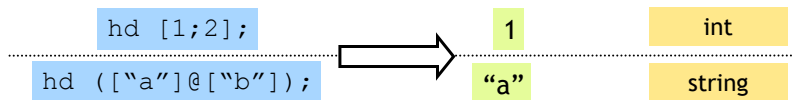


Can only append two lists `1 @ [2;3];`

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

Complex types: Lists

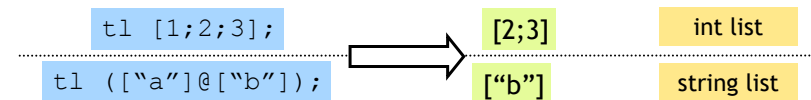
List operator “head” `hd`



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

Complex types: Lists

List operator “tail” `tl`



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

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

So far, a fancy calculator...

... what do we need next ?