CMSC 330: Organization of Programming Languages

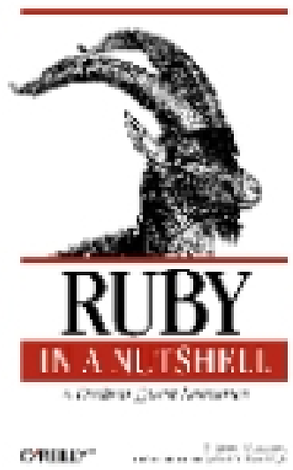
Ruby and Regular Expressions

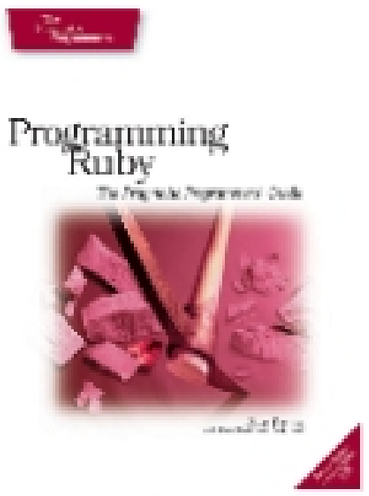
Introduction

- Ruby is an *object-oriented, imperative scripting language*
  - "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language."
  - "I believe people want to express themselves when they program. They don't want to fight with the language. Programming languages must feel natural to programmers. I tried to make people enjoy programming and concentrate on the fun and creative part of programming when they use Ruby."

  – *Yukihiro Matsumoto ("Matz")*

Books on Ruby





- Earlier version of Thomas book available on web
  - See course web page

Applications of Scripting Languages

- Scripting languages have many uses
  - Automating system administration
  - Automating user tasks
  - Quick-and-dirty development
- Major application:

  Text processing

Output from Command-Line Tool

```
% wc *
     271      674     5323 AST.c
     100      392     3219 AST.h
     117     1459   238788 AST.o
    1874     5428    47461 AST_defs.c
    1375     6307    53667 AST_defs.h
     371      884     9483 AST_parent.c
     810     2328    24589 AST_print.c
     640     3070    33530 AST_types.h
     285      846     7081 AST_utils.c
      59      274     2154 AST_utils.h
      50      400    28756 AST_utils.o
     866     2757    25873 Makefile
     270      725     5578 Makefile.am
     866     2743    27320 Makefile.in
      38      175     1154 alloca.c
    2035     4516    47721 aloctypes.c
      86      350     3286 aloctypes.h
     104     1051    66848 aloctypes.o

...
```


Climate Data for IAD in August, 2005

```
================================================================================
1   2   3   4   5  6A  6B    7    8   9  10  11  12  13   14  15   16   17  18
                                         AVG MX 2MIN
DY MAX MIN AVG DEP HDD CDD  WTR  SNW DPTH SPD SPD DIR MIN PSBL S-S WX    SPD DR
================================================================================

 1  87  66  77   1   0  12 0.00  0.0    0  2.5  9 200   M    M   7 18    12 210
 2  92  67  80   4   0  15 0.00  0.0    0  3.5 10  10   M    M   3 18    17 320
 3  93  69  81   5   0  16 0.00  0.0    0  4.1 13 360   M    M   2 18    17 360
 4  95  69  82   6   0  17 0.00  0.0    0  3.6  9 310   M    M   3 18    12 290
 5  94  73  84   8   0  19 0.00  0.0    0  5.9 18  10   M    M   3 18    25 360
 6  89  70  80   4   0  15 0.02  0.0    0  5.3 20 200   M    M   6 138   25 210
 7  89  69  79   3   0  14 0.00  0.0    0  3.6 14 200   M    M   7 1     16 210
 8  86  70  78   3   0  13 0.74  0.0    0  4.4 17 150   M    M  10 18    23 150
 9  76  70  73  -2   0   8 0.19  0.0    0  4.1  9  90   M    M   9 18    13  90
10  87  71  79   4   0  14 0.00  0.0    0  2.3  8 260   M    M   8 1     10 210
...
```


Raw Census 2000 Data for DC

```
u108_S,DC
    ,
    000,01,0000001,572059,72264,572059,12.6,572059,572059,572059,0,0,0,0,5
    72059,175306,343213,2006,14762,383,21728,14661,572059,527044,158617,34
    0061,1560,14605,291,1638,10272,45015,16689,3152,446,157,92,20090,4389,
    572059,268827,3362,3048,3170,3241,3504,3286,3270,3475,3939,3647,3525,3
    044,2928,2913,2769,2752,2933,2703,4056,5501,5217,4969,13555,24995,2421
    6,23726,20721,18802,16523,12318,4345,5810,3423,4690,7105,5739,3260,234
    7,303232,3329,3057,2935,3429,3326,3456,3257,3754,3192,3523,3336,3276,2
    989,2838,2824,2624,2807,2871,4941,6588,5625,5563,17177,27475,24377,228
    18,21319,20851,19117,15260,5066,6708,4257,6117,10741,9427,6807,6175,57
    2059,536373,370675,115963,55603,60360,57949,129440,122518,3754,3168,22
    448,9967,4638,14110,16160,165698,61049,47694,13355,71578,60875,10703,3
    3071,35686,7573,28113,248590,108569,47694,60875,140021,115963,58050,21
    654,36396,57913,10355,4065,6290,47558,25229,22329,24058,13355,10703,70
    088,65737,37112,21742,12267,9475,9723,2573,2314,760,28625,8207,7469,73
    8,19185,18172,1013,1233,4351,3610,741,248590,199456,94221,46274,21443,
    24831,47947,8705,3979,4726,39242,25175,14067,105235,82928,22307,49134,
    21742,11776,211,11565,9966,1650,86,1564,8316,54,8262,27392,25641,1751,
    248590,115963,4999,22466,26165,24062,16529,12409,7594,1739,132627,1167
    0,32445,23225,21661,16234,12795,10563,4034,248590,115963,48738,28914,1
    9259,10312,4748,3992,132627,108569,19284,2713,1209,509,218,125
```


A Simple Example

- Let's start with a simple Ruby program

ruby1.rb:

```
# This is a ruby program
x = 37
y = x + 5
print(y)
print("\n")
```

```
% ruby -w ruby1.rb
42
%
```

Language Basics

comments begin with #, go to end of line

variables need not be declared

```
# This is a ruby program
x = 37
y = x + 5
print(y)
print("\n")
```

line break separates expressions (can also use ";" to be safe)

no special main() function or method

Run Ruby, Run

- There are three ways to run a Ruby program
  - ruby -w *filename* – execute script in *filename*
    - tip: the -w will cause Ruby to print a bit more if something bad happens
  - irb – launch interactive Ruby shell
    - can type in Ruby programs one line at a time, and watch as each line is executed

```
irb(main):001:0> 3+4
=> 7
irb(main):002:0> print("hello\n")
hello
=> nil
```


Run Ruby, Run (cont'd)

- Suppose you want to run a Ruby script as if it were an executable

```
#!/usr/local/bin/ruby -w
print("Hello, world!\n")
```

  - ./filename # run program
  - The first line ("shebang") tells the system where to find the program to interpret this text file
  - Must chmod u+x *filename* first
    - Or chmod a+x *filename* so everyone has exec permission
  - Warning: Not very portable
    - Depends on location /usr/local/bin/ruby

Explicit vs. Implicit Declarations

- Java and C/C++ use explicit variable declarations
  - variables are named and typed before they are used
    - int x, y; x = 37; y = x + 5;
- In Ruby, variables are implicitly declared
  - first use of a variable declares it and determines type
    - x = 37; y = x + 5;
      - x, y exist, will be integers