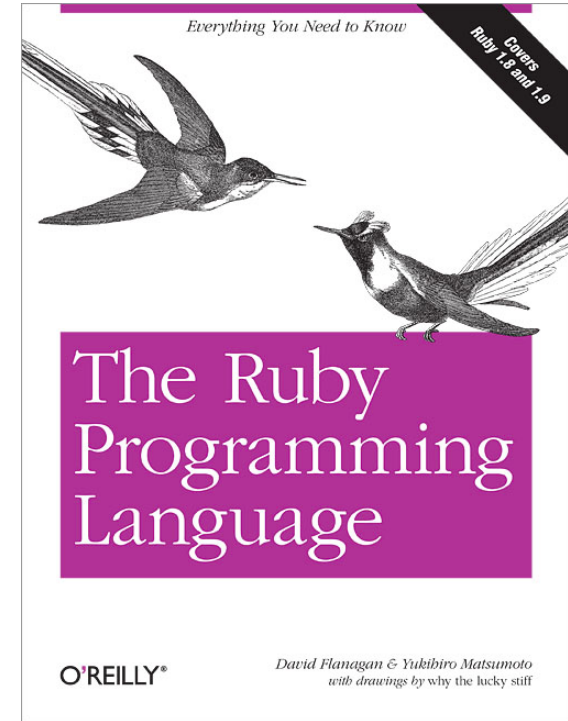


CMSC 330: Organization of Programming Languages

Introduction to Ruby

Ruby

- ▶ An *object-oriented, imperative, dynamically typed (scripting) language*
 - Similar to Python, Perl
 - Fully object-oriented
- ▶ Created in 1993 by Yukihiro Matsumoto (Matz)
 - “Ruby is designed to make programmers happy”
- ▶ Adopted by **Ruby on Rails** web programming framework in 2005
 - a key to Ruby’s popularity



Static Type Checking (Static Typing)

- ▶ **Before** program is run
 - Types of all expressions are determined
 - Disallowed operations cause **compile-time error**
 - Cannot run the program
- ▶ Static types are often **explicit (aka manifest)**
 - Specified in text (at variable declaration)
 - C, C++, Java, C#
 - But may also be inferred – compiler determines type based on usage
 - OCaml, C#, Rust, and Go (limited)

Dynamic Type Checking

- ▶ **During** program execution
 - Can determine type from run-time value
 - Type is checked before use
 - Disallowed operations cause **run-time exception**
 - Type errors may be latent in code for a long time
- ▶ Dynamic types are ***not* manifest**
 - Variables are just introduced/used without types
 - Examples
 - **Ruby**, Python, Javascript, Lisp
 - **Note**: Ruby v3 adds support for static types, mixed with its native dynamic ones. We'll discuss this more, later in the course.

Static and Dynamic Typing

- ▶ Ruby is dynamically typed, C is statically typed

```
# Ruby
x = 3
x = "foo" # gives x a
          # new type
x.foo    # NoMethodError
          # at runtime
```

```
/* C */
int x;
x = 3;
x = "foo"; /* not allowed */
/* program doesn't compile */
```

Tradeoffs?

Static type checking

More work for programmer (at first)

Catches more (and subtle) errors at compile time

Precludes some correct programs

More efficient code
(fewer run-time checks)

Dynamic type checking

Less work for programmer (at first)

Delays some errors to run time

Allows more programs
(Including ones that will fail)

Less efficient code
(more run-time checks)

Java: *Mostly* Static Typing

- ▶ In Java, types are mostly checked statically

```
Object x = new Object();
```

```
x.println("hello"); // No such method error at compile time
```

- ▶ But sometimes checks occur at run-time

```
Object o = new Object();
```

```
String s = (String) o; // No compiler warning, fails at run time
```

```
// (Some Java compilers may be smart enough to warn about above cast)
```

Quiz 1: Get out your clickers!

- ▶ True or false: This program has a type error

```
# Ruby
x = "hello"
y = 2.5
y = x
```

- A. True
- B. False

Quiz 1: Get out your clickers!

- ▶ True or false: This program has a type error

```
# Ruby  
x = "hello"  
y = 2.5  
y = x
```

- A. True
- B. **False**

Quiz 2

- ▶ **True** or **false**: This program has a type error

```
/* C */  
void foo() {  
    int a = 10;  
    char *b = "hello";  
    a = b;  
}
```

- A. True
- B. False

Quiz 2

- ▶ **True** or **false**: This program has a type error

```
/* C */  
void foo() {  
    int a = 10;  
    char *b = "hello";  
    a = b;  
}
```

- A. **True**
- B. **False**

Control Statements in Ruby

- ▶ A **control statement** is one that affects which instruction is executed next

- While loops
- Conditionals


```
i = 0
while i < n
  i = i + 1
end
```

```
if grade >= 90 then
  puts "You got an A"
elsif grade >= 80 then
  puts "You got a B"
else
  puts "You're not doing so well"
end
```

What is True?

- ▶ The **guard** of a conditional is the expression that determines which branch is taken

```
if grade >= 90 then
  ...
```



Guard

- ▶ **True:** anything except
 - false
 - nil
- ▶ Warning to C programmers: **0 is not false!**

Quiz 3: What is the output?

```
x = 0
if x then
  puts "true"
elsif x == 0 then
  puts "== 0"
else
  puts "false"
end
```

- A. Nothing – there's an error
- B. "false"
- C. "== 0"
- D. "true"

Quiz 3: What is the output?

```
x = 0
if x then
  puts "true"
elsif x == 0 then
  puts "== 0"
else
  puts "false"
end
```

- A. Nothing – there's an error
- B. "false"
- C. "== 0"
- D. "true"

x is neither **false** nor **nil** so the first guard is satisfied