Quick Ruby

AUBURN UNIVERSITY

Rodrigo Sardiñas

Topics

AUBURN

- Running Ruby File
- Variables
- Selection
  - Cases
- Iteration (while)
- Methods
- Classes

Running Ruby File

AUBURN

- From a terminal type in
  - ruby [file name].rb

Variables

AUBURN

- Data type is inferred, no need to declare type
- CONSTANTS begin with capitol letters
  - Customary to use ALL CAPS to indicate constant
- Global variables begin with $
- Instance variables begin with @
  - Similar to class variables but local to one instance of a class object
- Class variables begin with @@
  - Think static scoping in Java

Variables (Continued)

AUBURN

Default Access Modifier for Instance variable = Private

Accessors?

attr_accessor :ink_level

attr_accessor :paper_level

Selection

AUBURN

- Do not need () or semi-colons or colons

Example 1:

```
if 10 < 20 [then]
    Do this stuff
end
```

Selection

AUBURN

Example 2:

if 10 < 20; Do this stuff end

Selection

AUBURN

Example 3

```
if …
    do this
else
    do this instead
end
```

Example 4

```
if …
    do this
elsif
    do this instead
end
```

Cases

AUBURN

- Case statement does return a value, so it is possible, but not necessary, to save output

Example:

```
result = case value_to_test
    when match1 then result1 ← this gets saved into result
    when match2 then result2 ← “”
    when match3 ← does value_to_test equal this value?
    else return_this_instead
end
```

Cases

AUBURN

- Saving return value is optional, can just as easily simply execute statements similar to if statements
- No “then” statement

Example:

```
case my_var
    when my_var_equals_this_value
        do this stuff
    when my_var_equals_another_value
        do this stuff instead
    else
        do this stuff if no matches
    end
```

Iteration (While)

AUBURN

- Syntax looks like if statement

Example:

```
while test this condition
    body of while loop
    execute these statements
end
```

Iteration (For)

AUBURN

```
for some_loop_var in some_range
    do this stuff
end
```

Example (2 .. Inclusive, 3 … Exclusive):

```
    for i in 1…10
        puts “I am in a for loop”
    end
```

Methods

AUBURN

- Begin with “def” keyword, followed by function name (and any amount of optional parameters)

Example:

```
def function_name (opt_arg1, opt_arg2, …)
    do stuff here
    body of function
    [return some_value] ← this is optional
end
```

Methods

AUBURN

- Use like java function

Example:

```
function_name(1, 2, 3)
another_function_w_no_args()
value = this_func_returned_smth()
```

Classes

AUBURN

- Start with “class” keyword followed by desired class name (Class name must be CONSTANT)
- Define an initialize function for a constructor. Behaves as a normal constructor
- Class functions called via dot (.) notation
  - Class_name.function_name()
- Can manually create your own mutator and accessor functions, or use “attr_accessor” keyword
- Examples on next slides

Classes (Example part 1)

AUBURN

```
class My_made_up_class

    attr_accessor :instance_var ← make accessor
                                    and mutator for me

    def initialize(my_arg) ← our constructor
        @instance_var = my_arg
    end
```

Classes (Example part 2)

AUBURN

```
…
    A_CONSTANT = 10
    ANOTHER_CONSTANT = “a”

    def another_function
        return 10
    end

    def another_function2
        just do this stuff
    end

end
```

Classes (Example part 3)

AUBURN

load “path to file for My_made_up_class.rb”

```
class_object = My_made_up_class.new(10)
another_var = class_object.another_function
class_object.another_function2
```

Interactive Ruby (IRB)

AUBURN

- Interactive editor for
  - Dynamic code creating/testing
  - Debugging classes
  - Quickly testing syntax