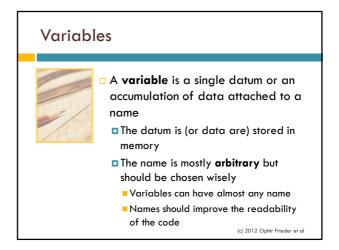
CHAPTER 3:
CORE PROGRAMMING ELEMENTS

Introduction to Computer Science Using Ruby



Variables in Ruby

- Use the format variable_name = value
- □ This format also **initializes** variable data

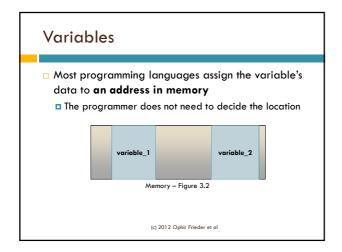
□ The **equal sign (=)** assigns the right-hand side to the variables in the left hand side

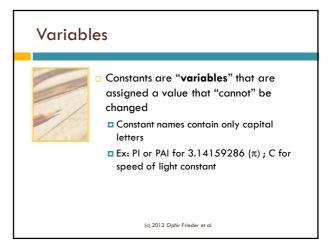
(c) 2012 Ophir Frieder et al

Common Standards for Variable Names

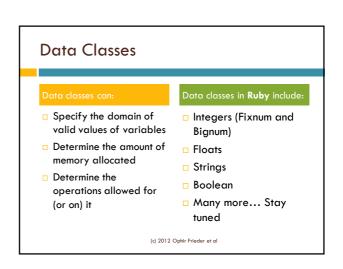
- Cannot start with an integer
 - Ex: bank1, not 1 bank
- Should avoid having special characters
 - Ex: money_spent, not
 \$_spent
 - Special characters have specific uses in many languages, including Ruby
 - (c) 2012 Ophir Frieder et al

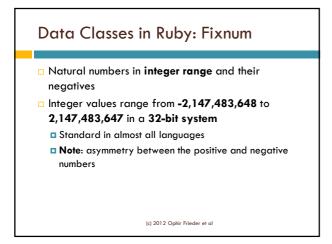
- Should explain the data they stand for
 - Ex: balance, not b
- Should complement the programming language style
 - Ex: check_balance, not checkBalance or checkbalance
 - Names with underscores match Ruby's style
 - Last two names are different because names are case sensitive

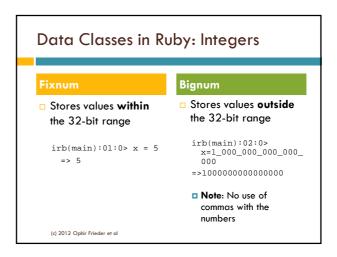


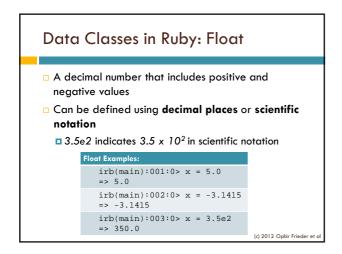


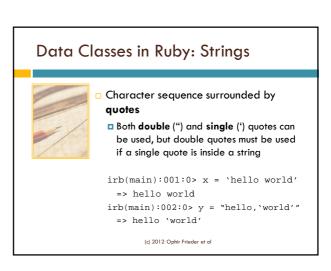
Data Classes Variables can represent words, numbers, and other entities depending on their data classes A data class indicates the properties of the data stored in a variable The nomenclature "Data Type" is used in non-object oriented languages The notion of "Class" has far more reaching meaning than "Type"











Basic Arithmetic Operators

- □ Used to perform mathematical operations
- Most are binary operators and require two operands

Symbol	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Power
Table 3.1	

Basic Arithmetic Operators



The modulus operator, %, is used to find the remainder when diving two integers

> irb(main):001:0> x = 5%2=> 1

> > (c) 2012 Ophir Frieder et al

Advanced Mathematical Functions

 Ruby's Math Module provides advanced mathematical functions, referred to as **Methods** (Table 3.2)

Method	Operation
sqrt()	Square Root
sin()	Sine
cos()	Cosine
tan()	Tangent
log()	Natural Log (In)
log10()	Log (Base 10)

Math Module methods are used in the following format:
 Math.Function_name(Value)

irb(main):001:0> x = Math.sqrt(16)

> 4 (c) 2012 Ophir Frieder et al

Use of Methods

- Ruby's Math Module provides advanced mathematical functions, referred to as Methods
- There is a way to include a whole module (like Math), without the need to specify it with every use

(c) 2012 Ophir Frieder et al

Input & Output: Direct Output

- The puts instruction displays text on the screen (i.e., standard out)
 - irb(main):001:0> puts "Hello World"
- □ Variables are displayed on the screen using puts
 - To use **puts** for a variable, enter the variable name without quotations

(c) 2012 Ophir Frieder et al

Input & Output: Input Using Variables

- □ The **gets** instruction stores values that are entered from the keyboard (i.e., standard input device)
- □ Its format is very similar to **puts**

irb(main):001:0> age_input = gets

- $\hfill\Box$ \hfill gets stops the program and waits for the user to type
 - Type the **input**, then press enter

(c) 2012 Ophir Frieder et al

Input & Output: Input Using Variables

- gets will store values as character strings
- □ To change the data from one class to another (i.e., a string into an integer), you need to explicitly perform a type (class) conversion, usually creating a new variable of the appropriate class

(c) 2012 Ophir Frieder et al

Input & Output: Conversion



gets will store character strings

irb(main):001:0> age_input = gets

- □ If you typed 19, age_input will be the string "19", NOT the number 19
- To convert "19" to 19, perform the following:

irb(main):002:0> age =
age_input.to_i

. to_i converts the contents of a variable to an integer

(c) 2012 Ophir Frieder et al

Common Programming Errors

□ Syntax errors refer to code that Ruby cannot

```
irb(main):001:0> x = 1 + "hello"
Type Error: String can't be coerced into Fixnum
 from (irb):1:in '+'
 from (irb):1
```

□ Ruby stops execution and tells the location where it had to stop

(c) 2012 Ophir Frieder et al

Common Programming Errors



Error messages can seem unrelated to the problem

irb(main):002:0> x = helloNameError: undefined local variable or method 'hello' for main: Object from (irb):2

□ Ruby assumed that hello was a variable since strings have quotes

(c) 2012 Ophir Frieder et al

Common Programming Errors

- Ruby cannot catch logic errors
 - The program runs, but the results are incorrect
- □ Logic errors are often harder to find because the error's location is not given
 - A common logic error involves integer division
 - Ruby performs integer division correctly, but many casual users expect a different result

irb(main):003:0> 5/2

A result of 2.5 may be expected, but it would not be an integer

(c) 2012 Ophir Frieder et al

Mixing Data Classes

- Ruby always tries to keep the same data class for all of its operands
- Ruby will convert data classes when it has different ones in the same arithmetic operation
- □ To get a decimal from the previous example, add a float or perform an explicit conversion

irb(main):003:0> 1.0*5/2 => 2.5

□ However, some data classes cannot be converted

Ruby will either create an error condition, or worse, produce an incorrect result irb(main):002:0> x
= "hello".to_i => 0

NOTE possible dependency!!!

(c) 2012 Ophir Frieder et al

Summary

- □ A **variable** is data attached to a name
- □ There are **common guidelines** to follow when creating variable names
- Constants are "variables" (really values) that never change
- Programs use various methods (operators and functions) available in each of the data classes to perform operations
- Ruby has many classes of operators and methods to perform math and other operations

(c) 2012 Ophir Frieder et o

puts The puts command is used to generate output on the screen (i.e., standard out) Three types of programming errors are syntax errors, logic errors, and type errors