Comptuer studies ruby notes

These are notes I took while I was learning Ruby. Comptuer studies ruby notes Instance of a class is a sub-category of that class. E. g. greyhound; dog. Every object has a class. Objects are instances of classes. Methods define what an object can do and properties describe it. Constants start with a capital letter, sometimes they are all caps e. g. INTEREST = 0. 012 #setting constant INTEREST to 1. 2% DOZEN = 12 #setting constant DOZEN to 12 Constants and variables store information in the memory for the duration of the use of the program.

Like RAM? Constants can’t be changed but variables can be reassigned based on certain properties and data. Different types of classes: ClassExample of Object Float6. 5 or 3. 9 Fixnum2 or 3 StringThisisastring or randomnumber34 ArrayMonday Tuesday Wednesday Thursday Friday Or January February March April HashToronto [Dion Phaneuf] Pittsburgh [Sidney Crosby] Washington [Alex Ovechkin] Or Haseeb [November 18] Humza [August 6] Hana [August 11] Range11.. 20 Or Hi.. Ho Float = decimal number Fixnum/Integer = whole number String = letters and numbers

Array = an ordered list, a couple of objects put together in one list that can also be accessed independently Create a new array or reset an old one by using this: array = Array. new You can also show individual variables by doing: array[number] e. g. subjects[3] you can also add objects to the array by: subjects [4] = ‘ SocialScience’ Hash = Similar to array but not in order, each object does not have a number. It’s based on key and value pairs. Like if you put five names and assigned each a birthday. They would go based on those pairs. It assigns the first name to the second. E. g: friends = Hash. ew friends['Andrea'] = 'July 22' friends['Mohammed'] = 'April 9' Range = A

sequence of values e. g. nums = 11.. 20 Setting variable: Fav_food= ‘ pizza’ #setting variable fav_food to pizza The quote marks around pizza identify that fav_food is a string value. Identifiers are the constant/variables names In order to change a variable’s value (not class) from one to another use the following To string: x. to_s To float: x. to_f To integer/fixnum: x. to_i Keep in mind, this only changes the variables value based on class properties, not its actual class Scope is where the variable can be accessed r seen within a program. Some are only used for a small task while others may be used for larger tasks and appear several times within the program. Constants’ scope depends on how often it is declared. If a constant is only declared within a class or module it’s scope is within that scope or module. However if it is declared outside of that class or module it’s scope is wider or “ global”. There four different variable scopes. Local variables are confined to the part of the program in which they are declared. If the variable is only declared within a method it is restricted to when that method is used or executed.

It can’t be used anywhere else in the program. (e. g. fav_food) Global variables can be declared anywhere in the program and are accessible from anywhere in the program. They are identified by a preceding “$” e. g. $fav_food. Global variables, however have to be used with extreme caution due to the fact that their values can be changed anytime in the program, sometimes by accidental or careless coding, these accidents can cause huge problems and are not easily fixable. Class variables are confined to a specific class but once all instances of the class are created the value of the variable is shared amongst all instances.

If the value is changed in one of the instances, it is changed in all of the instances. (e. g. @@fav_food) Instance variables are restricted to only certain instances of a class. If the value changes in one of the instances it stays the same in the others. (e. g. @fav_food) Commands to determine classes Either: puts variable. class or puts variable. kind_of? Class The first will tell you what class the variable is while the other will say true or false based on what class is inputted at the end of the line. Changing classes The easiest way is to just assign a new value to it.

Another way is to use the to_s, to_i etc. commands while also using the “=” assignment operator in order to change the object’s class permanently e. g. num = num. to_s you can do it without the “=” and the num variable before the equals sign in order to change it temporarily All math operations are the same except for two. Modulo (%) Divides and gives the remainder and exponent is (**). E. g: X = 6%4 X = 20 ** 2 There are also comparison operators such as: puts a == b #false as a and b are not equal puts a ! = b #true as a and b are not equal puts a < b #returns true as b is larger puts a