CIT 483/583 Lab01

Fall 2018

# Instructions

Open and save this file in any MS word-compatible format as Lab01\_*Firstname*\_*Lastname*.<ext> and place your answers in that document. Do a **Save-As** and retain all of my content. Keep the document safe in case your submission fails, or you discover an error prior to the due date and wish to re-submit. Submit your document to the Lab01 dropbox in Canvas. The due date and any other pertinent information are noted in the Canvas item.

This lab should be completed on students.cs.nku.edu as a reference implementation and as a means of testing your answers. It can be done using IRB only, or you may create .rb files to test some or all of your code, but do not submit .rb files (programs).

**Place the answers in or immediately following each question and make sure your answers stand out from the questions by using a different font color.**

1. Type in the *class* and the *id* of the object represented by the following literals using the class and object\_id methods. [You may also use \_\_id\_\_ as an alternative to object\_id.]

Example output for the literal 99 as a guideline: Class: Fixnum, Object ID: 199

* 1. 3
  2. 3
  3. 3.14
  4. 3.14
  5. :test
  6. :test
  7. "test"
  8. "test"
  9. ['Hello', 'World']
  10. ['Hello', 'World']
  11. {'Hello' => 'World'}
  12. {'Hello' => 'World'}

1. What does the previous question demonstrate with regard to the identity of Fixnums, Floats, and Symbols as compared to Strings, Arrays, and Hashes?
2. Equality and equivalence are not the same concepts in Ruby. Type in the evaluation of the following expressions (true or false)
3. 3.equal?(3)
4. 3.eql?(3)
5. (3.14).equal?(3.14)
6. 3.14 == 3.14
7. '3'.equal?('3')
8. '3' == '3'
9. '3'.eql?('3')
10. :test.equal?(:test)
11. "test".equal?("test")
12. "test".eql?("test")
13. ['Hello', 'World'].equal?(['Hello', 'World'])
14. ['Hello', 'World'].eql?(['Hello', 'World'])
15. {'Hello' => 'World'}.equal?({'Hello' => 'World'})
16. {'Hello' => 'World'}.eql?({'Hello' => 'World'})
17. For any two objects a and b, does a.equal?(b) being true imply anything about a.eql?(b) ?
18. Rewrite the following block of code in three different ways and show the code and the result. Note that an alternate way to write "not x < 5" is !(x < 5), but you need the parentheses due to operator precedence. [x should be initialized to 3 for testing, but you may also want to set it to 5 for an additional test case to make sure your logic is correct.]

x = 3

if not x < 5

puts "#{x} is greater than or equal to (>=) 5"

end

1. Rewrite it as a 3-line block using unless instead if and not.
2. Rewrite it in one line using a trailing if and not ("suffix" if).
3. Rewrite it using a trailing unless.
4. Rewrite the following block of code in three different ways and show the code and the result. HINT: If the loop does not explicitly require you to update the counter, don't do it. If you do need to manage the counter, don't forget to set it back to zero for testing

x = 0

while x < 5

print "#{x} "

x += 1

end

puts() # blank line after the loop

1. Rewrite it using until instead of while
2. Rewrite it as a for loop with a Range literal for counting
3. Rewrite it using a loop..do structure and a break statement with an appropriate trailing conditional expression (break if ...)
4. Rewrite the code below using a case statement and show the result using grade = 89 as the test value.

grade = 89

if grade >= 93

"A"

elsif grade >= 83

"B"

elsif grade >= 73

"C"

else

"Try again!"

end

1. Consider the class Person below. Type in the definition and create an instance of a Person with the object reference p. [If you use IRB, just capture the result of creating the object, but if you test with a standalone ruby program, you would need to add a statement puts p.inspect to see the result.]

class Person

@name

end

1. Make the following additions/modifications and test them.
   1. Add a getter and setter for @name using the naming conventions discussed in class. In other words, manually write the methods as opposed to using attr\_accessor or a similar shortcut.
   2. Instantiate an object of type Person and use the getter to see what value @name is initially given. Then, use the "sugared" form of the setter to set the name to "Tim" and use the getter again to display the name.
2. Which of the attr\_ shortcuts would you use to automatically generate getters and/or setters for @name as specified below? (Just write the attr\_ expression, not the full class definition.)
   1. A read-only attribute (getter only)
   2. A write-only attribute (setter only)
   3. A read-write attribute with both a getter and setter
3. Show the getter and setter methods that would be generated by the following class definition.

class Test

attr\_accessor :property

end

1. Suppose that a class Derived is defined as shown below.

class Derived < Test

end

* 1. If we create an object reference d as d = Derived.new ,

is d.property = "derived" a legal statement?

* 1. Run d.methods.grep(/prop/) and show the result.

Where do the definitions of these two methods originate?

1. Change the definition of Derived as given below and answer the following questions, based on the documentation of the Comparable module @ <https://ruby-doc.org/core-2.3.7/Comparable.html> and your own test cases. If you need to review the "spaceship" operator, see <https://ruby-doc.org/core-2.3.7/Object.html#method-i-3C-3D-3E>. Essentially, it is used to implement <, >, <+, and so on.

class Derived < Test

include Comparable

end

* 1. Suppose d = Derived.new. Both d == d and d < d are legal expressions, but what is being compared at this point?
  2. Create Derived object d1 and d2 and try to compare them. What happens?

d1 = Derived.new

d2 = Derived.new

d1 < d2

* 1. Add the <=> method to the Derived class, following the example in the Ruby docs, using property as the value to be compared. Then, try the d1 < d2 test case again.

d1 = Derived.new

d2 = Derived.new

d1 < d2

Why is it false? (Think about the value of property if the setter is never invoked.)

* 1. Now, use the setter to give d1 and d2 explicit values for property and re-test.

d1.property = "a"

d2.property = "z"

d1 < d2

Explain the result.

* 1. Set the d1 and d2 properties to some other values and re-test.

d1.property = "z"

d2.property = "a"

d1 < d2

Explain the result.