Ruby Homework

CSC 600-01 Programming Languages

Spring 2017

Emilio Quiambao

5 / 10 / 2017

1. Ruby conditional statements

```
#     CSC-600 HW4, RUBY DEMO
#     EMILIO QUIAMBAO
#     MAY 1, 2017

#     NUMBER (1)
#     CONDITIONAL STATEMENTS

puts "(1)   Ruby conditional statements"
puts "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
puts ""


# if-then
puts "if-then"
x1 = 1;
y1 = 1;

if x1 == y1 then puts "x and y are 1"
end
puts ""


# if-else
puts "if-else"
x2 = 1
y2 = 2

if x2 == y2
      puts "x and y are equal"
else
      puts "x and y are not equal"
end
puts ""

# if-elsif-else
puts"if-elsif-else"
x3 = 0
y3 = 0
z3 = 1

if x3 + y3 == 1
      puts "x and y add to 1"
elsif x3 + z3 == 1
      puts "x and z add to 1"
else
      puts "x cant find the one"
end
puts ""

# unless, unless-else
puts "unless, unless-else"
x4 = 3
y4 = 2
```

```
z4 = 1

unless x4 > y4 && y4 > z4
      puts "x y z are not decreasing"
else
      puts "x y z are decreasing"
end
puts ""

# if/unless modifiers
puts "if/unless modifiers"
x5 = 0
y5 = 0

x5 = 1 if x5 == 0
y5 = 1 if y5 == 1
puts "x is #{x5} and y is #{y5}"
puts""

# case, case with selector
puts "case, case with selector"
year = 2020
whereAreWe = case year
      when 1980 .. 1989 then "80's"
      when 1990 .. 1999 then "90's"
      when 2000 .. 2017 then "present"
      when 2017 .. 3000 then "future"
end

puts "We are in the #{whereAreWe}, the year #{year}!"
```

Output:

```
(1)     Ruby conditional statements
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

if-then
x and y are 1

if-else
x and y are not equal

if-elsif-else
x and z add to 1

unless, unless-else
x y z are decreasing

if/unless modifiers
x is 1 and y is 0

case, case with selector
We are in the future, the year 2020!
```

2. Ruby loops and iterators

```
#     CSC-600 HW4, RUBY DEMO
#     EMILIO QUIAMBAO
#     MAY 1, 2017

#     NUMBER (2)
#     LOOPS AND ITERATORS

puts "(2)   Ruby loops and iterators"
puts "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
puts ""

# loop-do
puts "loop-do"
x1 = 0
loop do
      x1 += 1
      puts x1
      if(x1 == 3)
            puts "BOOM!"
            break
      end
end
puts""

# while-do
puts "while-do"
x2 = 0
while x2 < 5 do
      puts "x = #{x2}"
      x2 += 1
end
puts""


# until-do
puts "until-do"
x3 = 5
until x3 <= 0 do
      puts "launching in #{x3}..."
      if x3 == 1
            puts "Blast off!"
      end
      x3 -= 1
end
puts""

# while/until modifiers
puts "while/until modifiers"
x4 = 1
y4 = 2
(puts"I will say this 5 times"; x4+=1) while x4 <= 5
(puts"I will say this 2 times"; y4-=1) until y4 == 0
puts""
```

```
# for-n, for-in-do
puts "for-n, for-in-do"
for i in 1..3
      puts i
end
puts""

# upto and downto iterations
puts "upto and downto iterations"
1.upto(3) {print "hello\n"}
3.downto(1) {|n| print n}
puts ""
puts ""

# times (implicit/explicit counter)
puts "times (implicit/explicit counter)"
4.times do |n|
      print "and #{n}! "
end
puts ""
puts ""

# each, each_with_index
puts "each, each_with_index"
myCards = ["Charizard", "Pikachu", "Mewtwo"]
myCards.each{|c| print c, " "}
puts""
myCards.each_with_index{|c, index| print"#{index+1} #{c}"}
puts""
puts""

# map
puts "map"
array = [1, 2, 3, 4 ,5]
array2 = array.map{|a| a*2}
puts array2
puts""

# step
puts "step"
(5..30).step(5) do |n|
      print n
      print "\n"
end
```

Output:

```
(2)     Ruby loops and iterators
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

loop-do
1
2
3
BOOM!

while-do
```

```
x = 0
x = 1
x = 2
x = 3
x = 4

until-do
launching in 5...
launching in 4...
launching in 3...
launching in 2...
launching in 1...
Blast off!

while/until modifiers
I will say this 5 times
I will say this 5 times
I will say this 5 times
I will say this 5 times
I will say this 5 times
I will say this 2 times
I will say this 2 times

for-n, for-in-do
1
2
3

upto and downto iterations
hello
hello
hello
321

times (implicit/explicit counter)
and 0! and 1! and 2! and 3!

each, each_with_index
Charizard Pikachu Mewtwo
1 Charizard2 Pikachu3 Mewtwo

map
2
4
6
8
10

step
5
10
15
20
25
30
```

3. Mean and standard deviation

```
#   CSC-600 HW4, RUBY DEMO
#   EMILIO QUIAMBAO
#   MAY 1, 2017

#   NUMBER (3)
#   MEAN AND STANDARD DEVIATION

def mean_sigma(v)

  #summation of all elements in array
  sum = 0.0
  v.each do |element|
    sum = sum + element
  end

  #mean of the array
  mean = (sum / v.length)

  #standard deviation
  sigma = 0.0
  v.each do |element|
    sigma = sigma + (element - mean) ** 2.0
  end
  sigma = (sigma / v.length) ** (1.0/2.0)

  return sum, mean, sigma

end

arr = [1, 2, 3]
puts "The sum is #{mean_sigma(arr)[0]}."
puts "The mean is #{mean_sigma(arr)[1]}."
puts "The standard deviation is #{mean_sigma(arr)[2]}."
puts ""

arr2 = [9, 12, 36, 100]
puts "The sum is #{mean_sigma(arr2)[0]}."
puts "The mean is #{mean_sigma(arr2)[1]}."
puts "The standard deviation is #{mean_sigma(arr2)[2]}."
puts ""

arr3 = [-1, -1, -1, -1]
puts "The sum is #{mean_sigma(arr3)[0]}."
puts "The mean is #{mean_sigma(arr3)[1]}."
puts "The standard deviation is #{mean_sigma(arr3)[2]}."
```

Output:

```
The sum is 6.0.
The mean is 2.0.
The standard deviation is 0.816496580927726.

The sum is 157.0.
The mean is 39.25.
The standard deviation is 36.601741761834234.

The sum is -4.0.
The mean is -1.0.
The standard deviation is 0.0.
```

4. Sorting function

```
#     CSC-600 HW4, RUBY DEMO
#     EMILIO QUIAMBAO
#     MAY 1, 2017

#     NUMBER (4)
#     SORTING FUNCTION

def sort(v)
      arr = v.clone

      loop do
            s = false
            (arr.length-1).times do |i|
                  if arr[i] > arr[i+1]
                        arr[i], arr[i+1] = arr[i+1], arr[i]
                        s = true
                  end
            end
            break if not s
      end

      return arr
end
puts sort([-3,0,3,2,-2,-1,1])
```

Output:

```
-3
-2
-1
0
1
2
3
```

5. Triangle class

```
#     CSC-600 HW4, RUBY DEMO
#     EMILIO QUIAMBAO
#     MAY 1, 2017

#     NUMBER (5)
#     TRIANGLE CLASS


class Triangle
      def initialize(a, b, c)
            @a = a
            @b = b
            @c = c
      end

      def perimeter
            @perimeter = @a + @b + @c
      end

      def area
            @area = ( ((@a+@b+@c)/2.0) * (((@a+@b+@c)/2.0) - @a) * (((@a+@b+@c)/2.0)
- @b) * (((@a+@b+@c)/2.0) - @c) ) ** (1.0/2.0)
      end

      def test
            unless @a + @b > @c and @a + @c > @b and @b + @c > @a
                  return "not a triangle"
            end

            if @a == @b and @a == @c and @b == @c
                  return "equilateral"
            end

            if    @a == @b or @a == @c or @b == @c
                  return "isosceles"
            end

            if @a != @b and @b != @c and @a != @c
                  return "scalene"
            end

            if  @a**2 + @b**2 == @c**2 or @a**2 + @c**2 == @b**2 or @b**2 + @c**2 ==
@a**2
                  return "right"
            end

            return "ERROR"

      end
end

#isosceles
triA = Triangle.new(3,4,4)
```

```
puts "Triangle A:"
puts "The perimeter is #{triA.perimeter}"
puts "The area is #{triA.area}"
puts "The triangle is #{triA.test}"
puts ""

#scalene
triB = Triangle.new(4,3,5)
puts "Triangle B:"
puts "The perimeter is #{triB.perimeter}"
puts "The area is #{triB.area}"
puts "The triangle is #{triB.test}"
puts ""

#equilateral
triC = Triangle.new(3,3,3)
puts "Triangle C:"
puts "The perimeter is #{triC.perimeter}"
puts "The area is #{triC.area}"
puts "The triangle is #{triC.test}"
puts ""

#not a triangle
triNot = Triangle.new(1,3,2)
puts "Invalid Triangle:"
puts "The perimeter is #{triNot.perimeter}"
puts "The area is #{triNot.area}"
puts "The triangle is #{triNot.test}"
puts ""
```

Output:

```
Triangle A:
The perimeter is 11
The area is 5.562148865321747
The triangle is isosceles

Triangle B:
The perimeter is 12
The area is 6.0
The triangle is scalene

Triangle C:
The perimeter is 9
The area is 3.897114317029974
The triangle is equilateral

Invalid Triangle:
The perimeter is 6
The area is 0.0
The triangle is not a triangle
```

6. Recognizer methods: limited and sorted

```
#     CSC-600 HW4, RUBY DEMO
#     EMILIO QUIAMBAO
#     MAY 1, 2017

#     NUMBER (6)
#     RECOGNIZER METHODS: LIMITED AND SORTED

class Array
      def limited?(amin, amax)
            check = true
            self.each do |i|
                  if !(amin <= i && i <= amax)
                        check = false
                        break
                  end
            end
            return check
      end

      def sorted?
            self.each_with_index do |x, i|
                  return true if i == self.length-1
                  break if x > self[i+1] && i != self.length-1
            end
            return false
      end
end

arr1 = [1, 2, 3]
arr2 = [-5, 0, 14]
arr3 = [900, 1, 4, 51, 39]

puts arr1.limited?(0,5)             #true
puts arr1.limited?(5, 10)           #false
puts arr1.sorted?                   #true
puts

puts arr2.limited?(-100, 100)       #true
puts arr2.limited?(0, 0)            #false
puts arr2.sorted?                   #true
puts

puts arr3.limited?(-99, 901)        #true
puts arr3.limited?(5, 10)           #false
puts arr3.sorted?                   #false
puts
```

Output:

```
true
false
true

true
false
true

true
false
false
```