CSc 600-01 (Section 1)

Homework 5 - Introduction to Ruby

prepared by Ilya Kopyl

CSc 600 Homework 4 - Ruby Introduction

Ilya Kopyl

May 10, 2018

Homework is prepared in LaTeX with TeXShop editor (under GNU GPL).

1. Write a single Ruby demo program that illustrates the use of all main Ruby iterators (*loop*, *while*, *until*, *for*, *upto*, *downto*, *times*, *each*, *map*, *step*, *collect*, *select*, *reject*).

1.1 loop

```ruby
# loop repeatedly executes the block of code
# In the example below I tried to emulate the look of vi text editor:
def use_loop
  line_number = 1
  loop do
    print "#{line_number}\t"
    line = gets
    break if line =~ /^\:q!|\:wq/      # exit on either :q! or :wq
    line_number += 1
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_loop
1       Skepticism is a resting place for human reason
2       where it can reflect upon its dogmatic wanderings,
3       but it is no dwelling place for permanent settlement.
4       Simply to acquiesce in skepticism can never suffice
5       to overcome the restlessness of reason.:wq
=> nil
```

Depending on the existence and the location of the break statement inside the block, loop can be either a loop with exit at the top, with exit at the bottom, with exit in the middle, or with no exit at all, which would produce an infinite loop.

If no block is given, an enumerator is returned instead:

```
$ irb
irb(main):001:0> p loop
#<Enumerator: main:loop>
=> #<Enumerator: main:loop>
irb(main):002:0> puts loop
#<Enumerator:0x00007f813f09c140>
=> nil
```

1.2 while

```ruby
# while loop executes the code while condition is true:
def use_while
  # example of while with exit at the top:
  a = 0
  while a < 5 do
    p a
    a += 1
  end

  puts
  # example of while with exit at the bottom:
  i = 0
  while true
    puts "push #{i}"
    i += 1
    break if i >= 10
  end

  puts
  # example of while with exit in the middle:
  while true
    i -= 1
    break if i < 0
    puts "pop #{i}"
  end

  puts
  # example of while loop as an inline modifier:
  p a -= 1 while a > 0
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_while
0
1
2
3
4

push 0
push 1
push 2
push 3
push 4
push 5
push 6
push 7
push 8
push 9

pop 9
pop 8
pop 7
pop 6
pop 5
pop 4
pop 3
pop 2
pop 1
pop 0

4
3
2
1
0
=> nil
```

1.3 until

```ruby
# until loop executes the code while condition is false
def use_until
  a = 0
  until a > 4 do
    p a
    a += 1
  end

  puts
  # example of until loop as an inline modifier:
  p a -= 1 until a <= 0
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_until
0
1
2
3
4

4
3
2
1
0
=> nil
```

1.4 for

```ruby
def use_for
  for number in 1..5 do
    p number
  end

  puts
  # do is optional:
  for number in 1...5
    p number
  end
  puts

  # as an expression, for loop returns all the values it iterated over:
  p for number in 1...5 do end
  p for letter in 'a'..'z' do end
  p for number in [1, 2, 3, 4] do end
  p for letter in ['a', 'b', 'c', 'd'] do end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_for
1
2
3
4
5

1
2
3
4

1...5
"a".."z"
[1, 2, 3, 4]
["a", "b", "c", "d"]
=> ["a", "b", "c", "d"]
```

1.5 upto

```ruby
def use_upto
  # upto without block returns an iterator:
  p 5.upto(10)

  # upto with block returns the start value:
  p 5.upto(10) { |num| num }
  puts

  # upto can be written with inline block:
  5.upto(10) { |num| puts num }
  puts

  # or with multiline block:
  5.upto(10) do |num|
    p num
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_upto
#<Enumerator: 5:upto(10)>
5

5
6
7
8
9
10

5
6
7
8
9
10
=> 5
```

1.6 downto

```ruby
def use_downto
  # downto without block returns an iterator:
  p 10.downto(5)

  # downto with block returns the start value:
  p 10.downto(5) { |num| num }
  puts

  # downto can be written with inline block:
  10.downto(5) { |num| p num }
  puts

  # or with multiline block:
  10.downto(5) do |num|
    p num
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_downto
#<Enumerator: 10:downto(5)>
10

10
9
8
7
6
5

10
9
8
7
6
5
=> 10
```

1.7 times

```ruby
def use_times
  # if no block is given, an enumerator is returned instead:
  p 5.times

  # as an expression it would return the number of iterations:
  p 5.times { }
  puts

  # times can be written with inline block:
  x = 2
  5.times { x *= x }
  p x

  # or with multiline block:
  5.times do |num|
    print "#{num} "
    puts if num == 4      # the values are iterated from 0 to n-1
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_times
#<Enumerator: 5:times>
5

4294967296
0 1 2 3 4
=> 5
```

1.8 each

```ruby
def use_each
  # if no block is given, an enumerator is returned instead:
  p [1, 2, 3, 4, 5].each

  # with block it returns the initial collection:
  p [1, 2, 3, 4, 5].each { }
  puts
  array = ['a', 'b', 'c', 'd', 'e']

  # each with inline block:
  array.each { |char| puts char }

  # each with multiline block:
  array.each do |char|
    print "#{char} "
  end
  puts

  # an example of each_with_index:
  array.each_with_index do |char, index|
    puts "#{index}:\t#{char}"
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_each
#<Enumerator: [1, 2, 3, 4, 5]:each>
[1, 2, 3, 4, 5]

a
b
c
d
e
a b c d e
0:      a
1:      b
2:      c
3:      d
4:      e
=> ["a", "b", "c", "d", "e"]
```

1.9 map

```ruby
def use_map
  # if no block is given, an enumerator is returned instead:
  p [1, 2, 3].map
  puts

  # with empty block it returns an array filled with nil values:
  p [1, 2, 3].map { }
  puts

  # use case analogous to the use of map function in Scheme:
  # block is mapped to each element in the array
  # as an expression, map returns the modified array
  p [1, 2, 3].map { |x| x**x }
  p ['a', 'b', 'c'].map { |char| char * 3 }

  # it can also be chained with other enumerables:
  p 10.times.map { |item| item }
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_map
#<Enumerator: [1, 2, 3]:map>

[nil, nil, nil]

[1, 4, 27]
["aaa", "bbb", "ccc"]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
=> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1.10 step

```ruby
def use_step
  # if no block is given, an enumerator is returned instead:
  p 1.step(10)
  p 1.step(10, 2)

  # with empty block it returns the start value (i.e. 1)
  p 1.step(10) { }    # "identity function"
  puts

  # by default, it increments each values by 1
  # here it prints all iterated values and returns the first element to p
  p 1.step(10) { |num| print "#{num} "}     # i.e. 1 2 3 4 5 6 7 8 9 10 1

  # we can also set a different increment value:
  p 1.step(10, 2).map { |item| item }
  puts

  array = ['a', 'b', 'c', 'd', 'e', 'f']

  # the use of array.step method:
  (0...array.length).step(1).each do |index|
    print "#{array[index] * index}"
  end
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_step
#<Enumerator: 1:step(10)>
#<Enumerator: 1:step(10, 2)>
1

1 2 3 4 5 6 7 8 9 10 1
[1, 3, 5, 7, 9]

bccdddeeeeffffff=> 0...6
```

1.11 collect

```ruby
def use_collect
  # in no block is given, an enumerator is returned instead
  p [1, 2, 3, 4, 5].collect

  # with empty block it returns an array of nil values
  p [1, 2, 3, 4, 5].collect { }

  p [1, 2, 3, 4, 5].collect { |item| item }       # identity function

  puts
  # collect works the same way as map method:
  p [1, 2, 3, 4, 5].collect { |item| item ** 2 }   # returns the modified array
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_collect
#<Enumerator: [1, 2, 3, 4, 5]:collect>
[nil, nil, nil, nil, nil]
[1, 2, 3, 4, 5]

[1, 4, 9, 16, 25]
=> [1, 4, 9, 16, 25]
```

1.12 select

```ruby
def use_select
  # if no block is given, an enumerator is returned instead
  p [1, 2, 3].select

  # with empty block it returns an empty array
  p [1, 2, 3].select { }

  # returns only even numbers, by selecting all even values:
  p [1, 2, 3, 4, 5, 6, 7, 8].select { |num| num % 2 == 0 }
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_select
#<Enumerator: [1, 2, 3]:select>
[]
[2, 4, 6, 8]
=> [2, 4, 6, 8]
```

1.13 reject

```ruby
def use_reject
  # if no block is given, an enumerator is returned instead
  p [1, 2, 3].reject

  # with empty block it returns the initial array
  p [1, 2, 3].reject { }    #identity function
  puts

  # returns only odd numbers by rejecting all even values:
  p [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].reject { |item| item % 2 == 0 }
end
```

Result of the code execution:

```
$ irb -I . -r hw5_problem1.rb
irb(main):001:0> use_reject
#<Enumerator: [1, 2, 3]:reject>
[1, 2, 3]

[1, 3, 5, 7, 9]
=> [1, 3, 5, 7, 9]
```

2. Write Ruby recognizer methods *limited*? and *sorted*? that expand the Ruby class Array.

The expression `array.limited?(amin, amax)` should return *true* if $amin \leqslant a[i] \leqslant amax \ \forall i$.

The expression `array.sorted?` should return the following:

- 0 if the array is not sorted
- +1 if $a[0] \leqslant a[1] \leqslant a[2] \leqslant ... \leqslant a[n]$ (non-decreasing order)
- −1 if $a[0] \geqslant a[1] \geqslant a[2] \geqslant ... \geqslant a[n]$ (non-increasing order)

Show examples of the use of this method.

The answer is listed on pages 14 through 16.

Source code of the program:

```ruby
# extending class Array with methods limited? and sorted?,
# as well as 4 other auxiliary methods to showcase the former two
class Array
  def limited?(amin, amax)
    each { |item| return false if item < amin || item > amax }
    true
  end

  def sorted?
    return 1 if sort == self
    return -1 if sort.reverse == self
    0
  end

  def limited_print(amin, amax)
    print "Array #{self} is "
    print 'not quite ' unless limited?(amin, amax)
    puts "limited by #{amin} and #{amax}."
  end

  def limited_test(amin, amax)
    p limited?(amin, amax)
    limited_print(amin, amax)
    puts
  end
```

```ruby
  def sorted_print
    sort_value = sorted?
    print "array #{self} is "
    puts 'not sorted.' if sort_value.zero?
    puts 'sorted in non-increasing order.' if sort_value.equal?(-1)
    puts 'sorted in non-decreasing order.' if sort_value.equal?(1)
  end

  def sorted_test
    puts "Given array #{self}, sorted? returns #{sorted?},"
    print '=> '
    sorted_print
    puts
  end
end

array = [1, 2, 3, 4, 5]

array.limited_test(1, 5)
array.limited_test(0, 5)
array.limited_test(0, 4)
array.limited_test(1, 6)
array.limited_test(2, 6)

puts

array.sorted_test

array = [3, 3, 4, 5, 5]
array.sorted_test

array = [5, 5, 3, 3, 1]
array.sorted_test

array = [123, 123, 432, 12, 342]
array.sorted_test
```

The result of the program execution:

```
$ ruby hw5_problem2.rb
true
Array [1, 2, 3, 4, 5] is limited by 1 and 5.

true
Array [1, 2, 3, 4, 5] is limited by 0 and 5.

false
Array [1, 2, 3, 4, 5] is not quite limited by 0 and 4.

true
Array [1, 2, 3, 4, 5] is limited by 1 and 6.

false
Array [1, 2, 3, 4, 5] is not quite limited by 2 and 6.


Given array [1, 2, 3, 4, 5], sorted? returns 1,
=> array [1, 2, 3, 4, 5] is sorted in non-decreasing order.

Given array [3, 3, 4, 5, 5], sorted? returns 1,
=> array [3, 3, 4, 5, 5] is sorted in non-decreasing order.

Given array [5, 5, 3, 3, 1], sorted? returns -1,
=> array [5, 5, 3, 3, 1] is sorted in non-increasing order.

Given array [123, 123, 432, 12, 342], sorted? returns 0,
=> array [123, 123, 432, 12, 342] is not sorted.
```

3. Create a Ruby class *triangle* with initializer, accessors, and member functions for computing the *perimeter* and the *area* of arbitrary triangles. Also make a member function *test* that checks sides a, b, and c, and classifies the triangle as:

(1) equilateral,

(2) isosceles,

(3) scalene,

(4) right,

(5) not a triangle.

Right triangle can be either isosceles or scalene. Compute the perimeter and area only for valid triangles (verified by test). Show examples of the use of this class.

The answer is listed on pages 17 through 22.

Source code of the program:

```ruby
# Computing the perimeter and area of arbitrary triangles
class Triangle
  def initialize(side_a, side_b, side_c)
    @side_a = side_a
    @side_b = side_b
    @side_c = side_c
  end

  # There was an option to either pre-calculate perimeter and store it
  # in a separate instance variable, or to have it calculated every time
  # when perimeter method is called. I chose the latter to avoid having states
  # and to use the method as idempotent function. Otherwise I would have to
  # re-calculate perimeter each time any of the sides is changed.
  def perimeter
    side_a + side_b + side_c if triangle?
  end

  # do the sides a, b, c form a triangle?
  def triangle?
    @side_a + @side_b > @side_c &&
      @side_a + @side_c > @side_b &&
      @side_b + @side_c > @side_a
  end
```

```ruby
# are all sides equal?
def equilateral?
  @side_a == @side_b && @side_b == @side_c
end

# are at least two sides equal?
def isosceles?
  side_a == side_b || side_a == side_c || side_b == side_c
end

# are the triangle sides all unequal?
def scalene?
  !isosceles?
end

def right_triangle?
  return false if equilateral? || !triangle?

  hypotenuse = sides.max
  catheti = sides - [hypotenuse]

  # analogous to Scheme's (apply + (map square lst))
  sum_of_catheti_squares =
      catheti.map do |cathetus|
        cathetus**2
      end.reduce(:+)

  # isosceles right-angled triangles can not have sides with integer values
  # that's why I use approximation here:
  (hypotenuse**2 - sum_of_catheti_squares).abs < 1.0e-10
end

def sides
  [@side_a, @side_b, @side_c]
end
```

```ruby
# leveraging the properties of equilateral & isosceles triangles
# to compute their area;
# using Heron's formula to compute areas of any other triangles
def area
  if equilateral?
    0.5 * Triangle.find_isosceles_height(side_a, side_b) * side_a
  elsif isosceles?
    base, side = find_isosceles_base_and_side
    0.5 * Triangle.find_isosceles_height(base, side) * base
  elsif triangle?
    semiperimeter = 0.5 * perimeter
    Math.sqrt(semiperimeter *
                  (semiperimeter - side_a) *
                  (semiperimeter - side_b) *
                  (semiperimeter - side_c))
  end
end

def self.find_isosceles_height(base, side)
  half_base = 0.5 * base
  Math.sqrt((side**2) - half_base**2)
end

def find_isosceles_base_and_side
  if side_a == side_b
    [side_c, side_a]
  elsif side_a == side_c
    [side_b, side_a]
  else
    [side_a, side_b]
  end
end

def test
  return [5] unless triangle?
  return_values = []
  return_values << 1 if equilateral?
  return_values << 2 if isosceles?
  return_values << 3 if scalene?
  return_values << 4 if right_triangle?
end
```

```ruby
  def showcase
    p self
    puts "Test: #{test}"
    puts "Do sides #{sides} form a triangle? #{triangle?}"
    if triangle?
      puts "Perimeter of the triangle: #{perimeter}"
      puts "Area of the triangle: #{format('%.4f', area)}"
      puts "Is this a right triangle? #{right_triangle?}"
      puts "Is this a scalene triangle? #{scalene?}"
      puts "Is this an isosceles triangle? #{isosceles?}"
      puts "Is this an equilateral triangle? #{equilateral?}"
    end
    puts
    puts
  end

  private :find_isosceles_base_and_side
  attr_accessor :side_a, :side_b, :side_c
end

# right triangle
triangle = Triangle.new(3, 4, 5)
triangle.showcase

# scalene triangle
triangle.side_b = 12
triangle.showcase

# right triangle
triangle.side_a = 5
triangle.side_c = 13
triangle.showcase

# isosceles triangle
triangle = Triangle.new(10, 10, 2)
triangle.showcase
# equilateral triangle
triangle = Triangle.new(8, 8, 8)
triangle.showcase

# isosceles triangle consisting of two 3-4-5s
triangle = Triangle.new(5, 5, 8)
triangle.showcase
```

```ruby
# scalene triangle
triangle = Triangle.new(8, 15, 20)
triangle.showcase

# isosceles right triangle:
triangle = Triangle.new(1, 1, Math.sqrt(2))
triangle.showcase
```

The result of the program execution:

```
#<Triangle:0x00007fd47502dfe0 @side_a=3, @side_b=4, @side_c=5>
Test: [3, 4]
Do sides [3, 4, 5] form a triangle? true
Perimeter of the triangle: 12
Area of the triangle: 6.0000
Is this a right triangle? true
Is this a scalene triangle? true
Is this an isosceles triangle? false
Is this an equilateral triangle? false


#<Triangle:0x00007fd47502dfe0 @side_a=3, @side_b=12, @side_c=5>
Test: [5]
Do sides [3, 12, 5] form a triangle? false


#<Triangle:0x00007fd47502dfe0 @side_a=5, @side_b=12, @side_c=13>
Test: [3, 4]
Do sides [5, 12, 13] form a triangle? true
Perimeter of the triangle: 30
Area of the triangle: 30.0000
Is this a right triangle? true
Is this a scalene triangle? true
Is this an isosceles triangle? false
Is this an equilateral triangle? false


#<Triangle:0x00007fd47502c690 @side_a=10, @side_b=10, @side_c=2>
Test:
Do sides [10, 10, 2] form a triangle? true
Perimeter of the triangle: 22
Area of the triangle: 9.9499
Is this a right triangle? false
Is this a scalene triangle? false
Is this an isosceles triangle? true
Is this an equilateral triangle? false
```

```
#<Triangle:0x00007fd4730bf910 @side_a=8, @side_b=8, @side_c=8>
Test:
Do sides [8, 8, 8] form a triangle? true
Perimeter of the triangle: 24
Area of the triangle: 27.7128
Is this a right triangle? false
Is this a scalene triangle? false
Is this an isosceles triangle? true
Is this an equilateral triangle? true


#<Triangle:0x00007fd4730beba0 @side_a=5, @side_b=5, @side_c=8>
Test:
Do sides [5, 5, 8] form a triangle? true
Perimeter of the triangle: 18
Area of the triangle: 12.0000
Is this a right triangle? false
Is this a scalene triangle? false
Is this an isosceles triangle? true
Is this an equilateral triangle? false


#<Triangle:0x00007fd4730bde80 @side_a=8, @side_b=15, @side_c=20>
Test:
Do sides [8, 15, 20] form a triangle? true
Perimeter of the triangle: 43
Area of the triangle: 53.1972
Is this a right triangle? false
Is this a scalene triangle? true
Is this an isosceles triangle? false
Is this an equilateral triangle? false


#<Triangle:0x00007fd4730b7aa8 @side_a=1, @side_b=1, @side_c=1.4142135623730951>
Test: [2, 4]
Do sides [1, 1, 1.4142135623730951] form a triangle? true
Perimeter of the triangle: 3.414213562373095
Area of the triangle: 0.5000
Is this a right triangle? true
Is this a scalene triangle? false
Is this an isosceles triangle? true
Is this an equilateral triangle? false
```

4. Create a Ruby class *Sphere*. Each sphere is characterized by the instance variable radius. For this class create the initializer and the following methods:

- *area* − a method that returns the area of the sphere ($a = 4r^2\pi$)
- *volume* − a method that returns the volume of the sphere ($v = 4r^3\pi/3$)

Create the class *Ball* that inherits properties from the class *Sphere* and adds a new instance variable *color*. Then create the class *MyBall* that inherits properties from the class *Ball* and adds a new instance variable *owner*. Write the method *show* that displays the instance variables of the class *MyBall*. Show sample applications of the class *MyBall*.

The answer is listed on the page TBD.

Source code of the program:

Results of the program execution: