Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)_____

О т ч ет

по лабораторной работе № 6

Название лабораторной работы: Циклы, Enumerator, Enumerable, Lambda-выражения и блоки

Дисциплина: Языки Интернет-программирования

Студент гр. ИУ6-31Б _____ Корнеев К.А.
(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Малахов Д.В.
(Подпись, дата) (И.О. Фамилия)

Москва, 2022

11 Вариант

Цель работы: научиться использовать циклы, в том числе Enumerator и Enumerable, labmda-выражения и блоки, узнать их различия

Задания:

Часть 1

Решить задачу, организовав итерационный цикл с точностью $\xi = 10^{-3}, 10^{-4}$. Вычислить длину кривой, определяемой функцией $y = ln\ x$ при $x \in [1, 2]$. Определить, как изменяется число разбиений при изменении точности.

Часть 2

Решить предыдущее задание с помощью Enumerable или Enumerator.

Часть 3

Составить метод minmax, отыскивающую $x \in [a, b]$, для которого функция $y = f(x)$ принимает максимальное и минимальное значение с точностью $0,01$. В основной программе использовать этот метод для математических функций $y = \dfrac{x-1}{x+2}; x \in [0, 2]$ и $y = sin(\dfrac{x}{2} - 1), x \in [-1, 1]$.

Реализовать вызов метода двумя способами: в виде передаваемого lambda-выражения и в виде блока.

Часть 1

lab6_1_func.rb

```ruby
# frozen_string_literal: true

# rubocop:disable Naming/MethodParameterName
def func(x)
  Math.sqrt(1 + (1 / x)**2)
end

def compute(hip, count, border)
  i1 = 0
  count.times do |i|
    i1 += hip * func(border + hip * i)
  end
  i1
end

def integral(eps, n = 1, a = 1.0, b = 2.0)
  i1 = 1
  i2 = 0
  while i1 - i2 > eps
    i1 = compute((b - a) / n, n, a)
    i2 = compute((b - a) / (n *= 2), n, a)
  end
  [i1, n / 2]
end
# rubocop:enable Naming/MethodParameterName
```

Ограничения рубокопа на имена переменных можно опустить для математической функции на переменную x и для данной в задании переменной n

lab6_1_test.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_1_func'
require 'rspec'

describe 'Integrals' do
  context 'When first testing integral values' do
    it 'should say its a good calcultaing' do
      expect(integral(10**-3)[0]).to be_between(1.22, 1.24).inclusive
    end
  end

  context 'When second testing integral values' do
    it 'should say its a good calcultaing' do
      expect(integral(10**-4)[0]).to be_between(1.22, 1.24).inclusive
    end
  end
end
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ
ТЕРМИНАЛ
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6/1$ rspec lab6_1_test.ruby
..

Finished in 0.00362 seconds (files took 0.06125 seconds to load)
2 examples, 0 failures
```

lab6_1.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_1_func'

p integral(10**-3)
p integral(10**-4)
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ
ТЕРМИНАЛ
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6/1$ ruby "/h
[1.2231761563869896, 128]
[1.2221608433295241, 1024]
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6/1$
```

Часть 2

lab6_2_func.rb

```ruby
# frozen_string_literal: true

# rubocop:disable Naming/MethodParameterName
def func(x)
  Math.sqrt(1 + (1 / x)**2)
end

def compute(hip, count, border)
  i1 = 0
  count.times do |i|
    i1 += hip * func(border + hip * i)
  end
  i1
end

def integral(n, a = 1.0, b = 2.0)
  Enumerator.new do |yielder|
    loop do
      yielder << [compute((b - a) / n, n, a), compute((b - a) / (n *= 2), n, a), n]
    end
  end
end
# rubocop:enable Naming/MethodParameterName
```

Ограничения рубокопа на имена переменных можно опустить для математической функции на переменную x и для данной в задании переменной n

lab6_2_spec.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_2_func'
require 'rspec'

describe 'Integral' do
  before :each do
    @int = integral(1)
  end

  context 'When first testing integral values' do
    it 'should say its a good calcultaing' do
      eps = 10**-3
      @int = @int.take_while { |x| x[0] - x[1] > eps }.to_a
      expect(@int[-1][1]).to be_between(1.22, 1.24).inclusive
    end
  end

  context 'When second testing integral values' do
    it 'should say its a good calcultaing' do
      eps = 10**-4
      @int = @int.take_while { |x| x[0] - x[1] > eps }.to_a
      expect(@int[-1][1]).to be_between(1.22, 1.24).inclusive
    end
  end
end

```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ
ТЕРМИНАЛ
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$ rspec 2/lab6_2_spec.rb
..

Finished in 0.00359 seconds (files took 0.06371 seconds to load)
2 examples, 0 failures

kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$
```

lab6_2.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_2_func'
eps = 10**-3
int = integral(1)
int = int.take_while { |x| x[0] - x[1] > eps }.to_a
puts [int[-1][1, 2]]

eps = 10**-4
int = integral(1)
int = int.take_while { |x| x[0] - x[1] > eps }.to_a
puts [int[-1][1, 2]]

```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ
ТЕРМИНАЛ
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$ ruby "/home/kiryu/VUZ/WPL/labo
1.2231761563869896
128
1.2221608433295241
1024
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$
```

lab6_3_func.rb

```ruby
# frozen_string_literal: true

# rubocop:disable Naming/MethodParameterName
def minmax(a, b, &function)
  min = max = yield a
  while a < b
    f = function.call(a)
    max = f if f > max
    min = f if f < max
    a += 0.01
  end
  [min, max]
end
# rubocop:enable Naming/MethodParameterName
```

Ограничения рубокопа на имена переменных можно опустить для заданных переменных a, b

lab6_3.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_3_func'

func = proc { |x| x + 2 != 0 ? (x - 1) / (x + 2) : nil }
lambda_func = ->(x) { Math.sin(x / 2 - 1) }

puts 'minmax of (x - 1) / (x + 2) in [0,2] with block'
p minmax(0.0, 2.0, &func)
puts
puts 'minmax of sin(x / 2 - 1) in [-1,1] with labmda'
p minmax(-1.0, 1.0, &lambda_func)

```

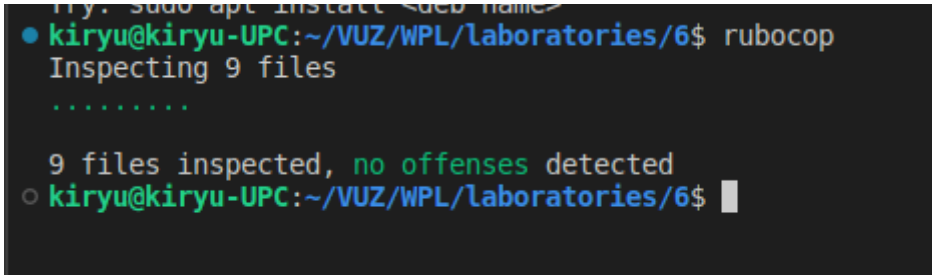
```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ
ТЕРМИНАЛ
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$ ruby "/home/kiryu/VUZ/WPL
minmax of (x - 1) / (x + 2) in [0,2] with block
[-0.5, 0.24812030075188]

minmax of sin(x / 2 - 1) in [-1,1] with labmda
[-0.9974949866040544, -0.4838074403239595]
kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$
```

lab6_3_spec.rb

```ruby
# frozen_string_literal: true

require_relative 'lab6_3_func'
require 'rspec'

func = proc { |x| x + 2 != 0 ? (x - 1) / (x + 2) : nil }
lambda_func = ->(x) { Math.sin(x / 2 - 1) }

describe 'Find min/max of functions' do
  context 'When first testing minmax of y = (x - 1)/(x + 2)' do
    it 'should say its a finding min well' do
      f = minmax(0.0, 2.0, &func)[0]

      expect(f).to be_between(-0.51, -0.49).inclusive
    end

    it 'should say its a finding max well' do
      f = minmax(0.0, 2.0, &func)[1]

      expect(f).to be_between(0.247, 0.25).inclusive
    end

    context 'When second testing minmax of sin(x / 2 + 1)' do
      it 'should say its a finding min well' do
        f = minmax(-1.0, 1.0, &lambda_func)[0]

        expect(f).to be_between(-1, -0.98).inclusive
      end

      it 'should say its a finding max well' do
        f = minmax(-1.0, 1.0, &lambda_func)[1]

        expect(f).to be_between(-0.5, -0.48).inclusive
      end
    end
  end
end
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ

ТЕРМИНАЛ

kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$ rspec 3/lab6_3_spec.rb
....

Finished in 0.00263 seconds (files took 0.06112 seconds to load)
4 examples, 0 failures

kiryu@kiryu-UPC:~/VUZ/WPL/laboratories/6$
```

Проверка rubocop



Вывод: в данной лабораторной работе я научился использовать циклы, в том числе Enumerator и Enumerable, labmda-выражения и блоки, узнл их различия