Lab 1
164 Advanced Web Programming
Spring 2023
Yuan Wang

A. Basic practice

check out the following code in irb:

```
puts 11.even?
puts 11.odd?
puts 11.class
puts 12345678901234567890123456789O.class
puts 11.next
puts 11.succ


12.9.ceil
(-12.9).ceil
-12.9.abs
12.9.floor
12.9.to_i
12.9.to_int
10 ** 2
12.9.round
12.4.round
3.14159.round(2)
4.14159.round(4)


a = 10

a.times { |x|  puts x}
a.times { |x|  print x}

use a.times to calculate  1+2+3+4+5+6+7+8+9+10

a.upto(20) {|x| puts x}
a.upto(1) {|x| puts x}

b = 10..20

b.first
b.last




b.each { |x| puts x}
b.each do |x|
```

```
   puts x
end

puts ‘it’s a wonderful year’    # how to change this to print:  it’s a wonderful year

puts “it’s a wonderful year”

puts %q/it’s a wonderful year/

puts %q/i spent #{a} years to get this degree/
puts %Q/i spent #{a} years to get this degree/
puts %/i spent #{a} years to get this degree/


‘i am  ‘+ a.to_s +  ‘years old’

“i am#{a}years old”
“i am ””#{a}”” years old”

“i am” << a.to_s << “ years old”

“cat” <=> “car”

“dog” <=> “fog”

print each char in“abcdefghijklmnopqrstuvwxyz” in a separate line


["apple", “banana”, ”orange”].include?("cherry")

[“Hello”, “from”, “the”, “other”, “side”].join
[“Hello”, “from”, “the”, “other”, “side”].join(“ “)
[“Hello”, “from”, “the”, “other”, “side”].join(“-“)



str = “capital”

str.upcase
str.capitalize
str.capitalize!
str.upcase!



aBcDeFg.swapcase
aBcDeFg.swapcase!
```

```
arr = %w{d b e f z h a l a b e a z m}
arr.shuffle
arr
arr.shuffle!
arr
arr.slice(4)
arr
arr.slice!(4)
arr
arr.sort
arr
arr.sort!
arr
arr.uniq
arr
arr.uniq!
arr
arr.reverse
arr
arr.reverse!
arr
```

```
snowy_owl = { "type" => "Bird", "diet" => "Carnivore", "life_span" => "12 years" }
puts snowy_owl[“type"]
snowy_owl["weight"] = “0.5 ounces"
puts snowy_owl
puts snowy_owl.keys
puts snowy_owl.values

print each key/value pairs of snowy_owl
print each key of snowy_owl
print each value of snowy_owl
```

B. Write code:

1. Given a hash of family members, with keys as the title and an array of names as the values, use Ruby's built-in select method to gather only immediate family members' names into a new array.

```
family = {  uncles: ["bob", "joe", "steve"],
            sisters: ["jane", "jill", "beth"],
            brothers: ["frank","rob","david"],
            aunts: ["mary","sally","susan"]. }
```

2. Given the array:

```
words =  ['demo', 'none', 'tied', 'evil', 'dome', 'mode', 'live',
          'fowl', 'veil', 'wolf', 'diet', 'vile', 'edit', 'tide', 'flow', 'neon']
```

Write a program that prints out groups of words that are anagrams. Anagrams are words that have the same exact letters in them but in a different order. Your output should look something like this:

["demo", "dome", "mode"]
["neon", "none"]
(etc)

3. Write a method that can take variable length of parameters, and in this method, print out information about number of parameters, and value of each parameters.

4. Following is a code to call a method

```
def mysterious_total(subtotal, tax, discount)
  subtotal + tax - discount
end
```

You can see that when calling the method, we have no idea what the meaning of the parameters of "mysterious_total(100, 10, 5)" are without looking at the definition.

a. to make it more clear, change the above code so that "keyword parameters" are used. also provide the default values for each parameters.

b. another advantage of keyword parameters, is you can switch order of the parameter without affecting behavior of the method.

   try to switch order of the method call and see the result.

c. use double splat ** in method definition. a ** argument will be a hash that contain any uncollected keyword parameters passed to the method

5. Write a class called MyGreeter.

   This class can initialize a name list, it will have a say_hello method, this method will print “hello” to every name in the name list

   it will also have a say_bye method, this method will print “bye” to every name in the name list

for example

obj = MyGreeter.new([“john”, “ken”, “ivy”, "amy", "wen"])

obj.say_hello
will output:

hello john
hello ken
hello ivy

obj.say_bye
will output:

bye john
bye ken
bye ivy