```
yuans-MacBook-Pro:1.2 yuan$ irb
irb(main):001:0> a = [1,2,3,4,5]
=> [1, 2, 3, 4, 5]
irb(main):002:0> a = [1, "love", 'a', 2.4, [1,2], "strings"]
=> [1, "love", "a", 2.4, [1, 2], "strings"]
irb(main):003:0> a[0]
=> 1
irb(main):004:0> a[1]
=> "love"
irb(main):005:0> b = [ 2,3,4]
=> [2, 3, 4]
irb(main):006:0> a & b
=> []
irb(main):007:0> a = [1,2,3,4,5]
=> [1, 2, 3, 4, 5]
irb(main):008:0> b
=> [2, 3, 4]
irb(main):009:0> a & b
=> [2, 3, 4]
irb(main):010:0> a | b
=> [1, 2, 3, 4, 5]
irb(main):011:0> a = [1,2,3,4,5]
=> [1, 2, 3, 4, 5]
irb(main):012:0> b = [5,6,7,9]
=> [5, 6, 7, 9]
irb(main):013:0> a | b
=> [1, 2, 3, 4, 5, 6, 7, 9]
irb(main):014:0> a + b
=> [1, 2, 3, 4, 5, 5, 6, 7, 9]
irb(main):015:0> a - b
=> [1, 2, 3, 4]
irb(main):016:0> a << b
=> [1, 2, 3, 4, 5, [5, 6, 7, 9]]
irb(main):017:0> [1,2].class
=> Array
irb(main):019:0> [1,2].push(3,4,5)
=> [1, 2, 3, 4, 5]
irb(main):020:0> [1,2].pop
=> 2
irb(main):021:0> [1,2,4,5,6].pop(3)
=> [4, 5, 6]
irb(main):022:0> [1,2,4,5,6].length
=> 5
irb(main):023:0> %w(i love  you)
=> ["i", "love", "you"]
irb(main):024:0> %w(a b c).combination(2).to_a
=> [["a", "b"], ["a", "c"], ["b", "c"]]
irb(main):025:0> %w(a b c).permutation(2).to_a
```

```
=> [["a", "b"], ["a", "c"], ["b", "a"], ["b", "c"], ["c", "a"], ["c",
"b"]]
irb(main):026:0> x = %w(a b c).permutation(2).to_a
=> [["a", "b"], ["a", "c"], ["b", "a"], ["b", "c"], ["c", "a"], ["c",
"b"]]
irb(main):027:0> x[2]
=> ["b", "a"]
irb(main):028:0> a = [1, 2, [3,4]]
=> [1, 2, [3, 4]]

irb(main):030:0> a.flatten
=> [1, 2, 3, 4]
irb(main):031:0> a
=> [1, 2, [3, 4]]
irb(main):032:0> b = a.flatten
=> [1, 2, 3, 4]
irb(main):033:0> a
=> [1, 2, [3, 4]]
irb(main):034:0> b = a.flatten!
=> [1, 2, 3, 4]
irb(main):035:0> a
=> [1, 2, 3, 4]
irb(main):036:0> 5.even?
=> false
irb(main):037:0> a
=> [1, 2, 3, 4]
irb(main):038:0> a.reverse
=> [4, 3, 2, 1]
irb(main):039:0> b = a.reverse
=> [4, 3, 2, 1]
irb(main):040:0> a
=> [1, 2, 3, 4]
irb(main):041:0> b = a.reverse!
=> [4, 3, 2, 1]
irb(main):042:0> a
=> [4, 3, 2, 1]
irb(main):043:0> a
=> [4, 3, 2, 1]
irb(main):044:0> a.shift
=> 4
irb(main):045:0> a
=> [3, 2, 1]
irb(main):046:0> a.shift(2)
=> [3, 2]
irb(main):047:0> a
=> [1]
irb(main):048:0> a = [1,2,3,4,5]
=> [1, 2, 3, 4, 5]
irb(main):049:0> a.slice(2)
```

```
=> 3
irb(main):050:0> a
=> [1, 2, 3, 4, 5]
irb(main):051:0> a.slice(2..4)
=> [3, 4, 5]
irb(main):052:0> [1,2,3,4,3,3,2,2].uniq
=> [1, 2, 3, 4]
irb(main):053:0> b = [1,2,3,4,3,3,2,2].uniq
=> [1, 2, 3, 4]
irb(main):054:0> a = [1,2,3,4,3,3,2,2]
=> [1, 2, 3, 4, 3, 3, 2, 2]
irb(main):055:0> b = a.uniq
=> [1, 2, 3, 4]
irb(main):056:0> a
=> [1, 2, 3, 4, 3, 3, 2, 2]
irb(main):057:0> b = a.uniq!
=> [1, 2, 3, 4]
irb(main):058:0> a
=> [1, 2, 3, 4]
irb(main):059:0> a = %w#i love you#
=> ["i", "love", "you"]
irb(main):060:0> #w(i love you)
=> nil
irb(main):061:0> %w(i love you)
=> ["i", "love", "you"]
irb(main):062:0> %w/i love you/
=> ["i", "love", "you"]
irb(main):063:0> %w@i love you@
=> ["i", "love", "you"]
irb(main):064:0> %w(i love you)
=> ["i", "love", "you"]
irb(main):065:0> %w(i love you).join
=> "iloveyou"
irb(main):066:0> %w(i love you).join(",")
=> "i,love,you"
irb(main):067:0> "i love you".split("")
=> ["i", " ", "l", "o", "v", "e", " ", "y", "o", "u"]
irb(main):068:0> "i love you".split
=> ["i", "love", "you"]
irb(main):069:0> "i,love,you".split(",")
=> ["i", "love", "you"]
irb(main):070:0> [1,2,3].each do |x|
irb(main):071:1*   puts x
irb(main):072:1> end
1
2
3
=> [1, 2, 3]
irb(main):073:0> [1,2,3].map do |x|
irb(main):074:1*    x*2
irb(main):075:1> end
=> [2, 4, 6]
irb(main):076:0> def my_method
irb(main):077:1>   return "love"
irb(main):078:1> end
=> :my_method
irb(main):079:0> my_method
=> "love"
irb(main):080:0> def my_method
irb(main):081:1>    "love"
irb(main):082:1> end
=> :my_method
irb(main):083:0> my_method
=> "love"
irb(main):084:0> def my_method
irb(main):085:1>    a = 3 + 4
irb(main):086:1> end
=> :my_method
irb(main):087:0> my_method
=> 7
irb(main):088:0> a
=> ["i", "love", "you"]
irb(main):089:0> a = [1,2,3]
=> [1, 2, 3]
irb(main):090:0> a.map {|x| x*2}
=> [2, 4, 6]
irb(main):091:0> a
=> [1, 2, 3]
irb(main):092:0> a.map! {|x| x*2}
=> [2, 4, 6]
irb(main):093:0> a
=> [2, 4, 6]
irb(main):094:0> a = %w^i love you^
=> ["i", "love", "you"]
irb(main):096:0> a.map {|word| word.capitalize}
=> ["I", "Love", "You"]
irb(main):097:0> a.map {|word| word.upcase}
=> ["I", "LOVE", "YOU"]
irb(main):098:0> a
=> ["i", "love", "you"]
irb(main):099:0> a.map! {|word| word.upcase}
=> ["I", "LOVE", "YOU"]
irb(main):100:0> a
=> ["I", "LOVE", "YOU"]
irb(main):101:0> a = [1,2,3,4,5,6,7,8]
=> [1, 2, 3, 4, 5, 6, 7, 8]
irb(main):102:0> a.select {|x| x.even?}
=> [2, 4, 6, 8]
```

```
irb(main):103:0> a.select {|x| x%2 == 0}
=> [2, 4, 6, 8]
irb(main):104:0> a.select {|x| true}
=> [1, 2, 3, 4, 5, 6, 7, 8]
irb(main):105:0> a.select {|x| false}
=> []
irb(main):106:0> a
=> [1, 2, 3, 4, 5, 6, 7, 8]

irb(main):108:0> a.select {|n| n.even?}.map {|x| x/2}
=> [1, 2, 3, 4]
irb(main):109:0>
irb(main):110:0> a.inject {|sum, x| sum + x}
=> 36
irb(main):111:0> a
=> [1, 2, 3, 4, 5, 6, 7, 8]
irb(main):112:0> a.inject(10) {|sum, x| sum + x}
=> 46
irb(main):113:0> a = %w!i love you!
=> ["i", "love", "you"]
irb(main):114:0> longest = a.inject do |current, word|
irb(main):115:1*    current.length > word.length ? current : word
irb(main):116:1> end
=> "love"
irb(main):117:0> a = %w!i love hate you!
=> ["i", "love", "hate", "you"]
irb(main):118:0> longest = a.inject do |current, word|
irb(main):119:1* current.length > word.length ? current : word
irb(main):120:1> end
=> "hate"
irb(main):121:0> longest = a.inject do |current, word|
irb(main):122:1* current.length >= word.length ? current : word
irb(main):123:1> end
=> "love"
irb(main):124:0> a
=> ["i", "love", "hate", "you"]
irb(main):125:0> a.each_index do |index|
irb(main):126:1*  puts index
irb(main):127:1> end
0
1
2
3
=> ["i", "love", "hate", "you"]
irb(main):128:0>
irb(main):129:0>
irb(main):130:0> h = {"firstname" => 'yuan', "lastname" => "wang"}
=> {"firstname"=>"yuan", "lastname"=>"wang"}
irb(main):131:0> h.class
```

```
=> Hash
irb(main):132:0> %w( yuan wang)
=> ["yuan", "wang"]
irb(main):133:0> a = %w( yuan wang)
=> ["yuan", "wang"]
irb(main):134:0> a[1]
=> "wang"
irb(main):135:0> h["lastname"]
=> "wang"
irb(main):136:0> h = {:firstname => 'yuan', :lastname => "wang"}
=> {:firstname=>"yuan", :lastname=>"wang"}
irb(main):137:0> :firstname.class
=> Symbol
irb(main):138:0> firstname = 4
=> 4
irb(main):139:0> quit
```

```ruby
class Test
  def mymethod
    puts "testing symbol"
  end
end

obj=Test.new()

obj.mymethod

m = :mymethod

obj.method(m).call
```

```
irb(main):017:0> h.has_key?("a")
=> true
irb(main):018:0> h1 = {"a"=>1, "b"=>2, "c"=>3}
=> {"a"=>1, "b"=>2, "c"=>3}
irb(main):019:0> h2 = {"a"=>1, "b"=>4, "d"=>3}
=> {"a"=>1, "b"=>4, "d"=>3}
irb(main):020:0> h1.merge(h2)
=> {"a"=>1, "b"=>4, "c"=>3, "d"=>3}
irb(main):021:0> h2.merge(h1)
=> {"a"=>1, "b"=>2, "d"=>3, "c"=>3}
irb(main):022:0> h1 = {"a"=>1, "b"=>2, "c"=>3}
=> {"a"=>1, "b"=>2, "c"=>3}
irb(main):023:0> h1.delete_if {|key, value| value >=2}
=> {"a"=>1}
irb(main):024:0> h1.keep_if {|key, value| value >=2}
```

```
=> {}
irb(main):025:0> h1
=> {}
irb(main):026:0> h1 = {"a"=>1, "b"=>2, "c"=>3}
=> {"a"=>1, "b"=>2, "c"=>3}
irb(main):027:0> h1.keep_if {|key, value| value >=2}
=> {"b"=>2, "c"=>3}
irb(main):028:0>
```

function call by value (we will talk about more complicate cases later)

```
yuans-MacBook-Pro:1.2 yuan$ irb
irb(main):001:0>
irb(main):002:0> def myfunc(x)
irb(main):003:1>    x = 4
irb(main):004:1> end
=> :myfunc
irb(main):005:0> var = 1
=> 1
irb(main):006:0> myfunc(var)
=> 4
irb(main):007:0> var
=> 1
irb(main):008:0> "string".class
=> String
irb(main):009:0> "string".each do |x|
irb(main):010:1* puts x
irb(main):011:1> end
Traceback (most recent call last):
        2: from /usr/local/Cellar/ruby/2.5.3_1/bin/irb:11:in `<main>'
        1: from (irb):9
NoMethodError (undefined method `each' for "string":String)
irb(main):012:0> "string".each_char do |x|
irb(main):013:1* puts x
irb(main):014:1> end
s
t
r
i
n
g
=> "string"
irb(main):015:0> %q{this is a single quote string}
=> "this is a single quote string"
irb(main):016:0> %q{this is a single quote /n string}
=> "this is a single quote /n string"
irb(main):017:0> %Q{this is a single quote /n string}
```

```
=> "this is a single quote /n string"

irb(main):020:0> %q{this is a single quote string #{a}}
=> "this is a single quote string \#{a}"
irb(main):021:0> %Q{this is a single quote string #{a}}
=> "this is a single quote string 4"
irb(main):022:0> %{this is a single quote string #{a}}
=> "this is a single quote string 4"
irb(main):023:0> 'this is a single quote string #{a}}'
=> "this is a single quote string \#{a}}"
irb(main):024:0> "this is a single quote string #{a}"
=> "this is a single quote string 4"
irb(main):025:0> %/this is a single quote string #{a}/
=> "this is a single quote string 4"
irb(main):026:0>
```

HERE DOCUMENT

```
#!/usr/bin/ruby
#this is a comment
puts "i #{def x(s)
              s
           end
          x('love')
         }  you"
=begin
this is a multiple line comments
=end

#here document
x=<<ENDOFTHEDOC
this is a test
   this is a test
      this is a test
        this is a test
ENDOFTHEDOC

puts x

class Class1
  puts("this is class 1")
end

class Class2
  puts("this is class 2")
end
```

```ruby
class Box
  def initialize(height=1, width=2, length=3)
    @height = height
    @width = width
    @length = length
  end

  def show_height
    @height
  end

  def volume
    @height * @width * @length
  end

  def scale(x)
  end
end

box1 = Box.new()
```

```
i love  you
this is a test
   this is a test
      this is a test
        this is a test
this is class 1
this is class 2
1
```

to run program:

```
$ ruby myprog.rb
```

to make the program “executable”

add #!/usr/bin/ruby
as the first line.

then change mode:

```
$ chmod u+x myprog.rb
```

assignment statement

```
yuans-MacBook-Pro:1.2 yuan$ irb
irb(main):001:0> a = b = 5
=> 5
irb(main):002:0> a
=> 5
irb(main):003:0> b
=> 5
irb(main):004:0> a = 1,2,3,4
=> [1, 2, 3, 4]
irb(main):005:0> a, b = 1,2,3,4,5
=> [1, 2, 3, 4, 5]
irb(main):006:0> a
=> 1
irb(main):007:0> b
=> 2
irb(main):008:0> a, b = 1,2
=> [1, 2]
irb(main):009:0> a
=> 1
irb(main):010:0> b
=> 2
irb(main):011:0> a
=> 1
irb(main):012:0> b
=> 2
irb(main):013:0> a, b = b, a
=> [2, 1]
irb(main):014:0> a
=> 2
irb(main):015:0> b
=> 1
irb(main):016:0> a, b, c = 1..3
=> 1..3
irb(main):017:0> a
=> 1..3
irb(main):019:0> b
=> nil
irb(main):020:0> c
=> nil
irb(main):021:0> a, b, c = 1
=> 1
irb(main):022:0> a, b, c = *(1..3)
=> [1, 2, 3]
```

```
irb(main):023:0> a
=> 1
irb(main):024:0> b
=> 2
irb(main):025:0> c
=> 3
irb(main):026:0> a, b = [1,2,3], [4,5,6]
=> [[1, 2, 3], [4, 5, 6]]
irb(main):027:0> a
=> [1, 2, 3]
irb(main):028:0> b
=> [4, 5, 6]
irb(main):029:0> a, b = *[1,2,3], [4,5,6]
=> [1, 2, 3, [4, 5, 6]]
irb(main):030:0> a
=> 1
irb(main):031:0> b
=> 2
irb(main):032:0> a, *b = 1,2,3,4,5
=> [1, 2, 3, 4, 5]
irb(main):033:0> a
=> 1
irb(main):034:0> b
=> [2, 3, 4, 5]
irb(main):035:0> a, *b, c = 1,2,3,4,5,6
=> [1, 2, 3, 4, 5, 6]
irb(main):036:0> a
=> 1
irb(main):037:0> b
=> [2, 3, 4, 5]
irb(main):038:0> c
=> 6
```

function parameters is like variables in assignment statement
so you can define variable length method like this:

```
irb(main):039:0> def mymethod(a, *b, c)
```