

# Collections

## Curso de Ruby - Collections (Aula 4)



## Conteúdo

O conteúdo que será abordado na aula:

- Collection
- Manipulando Collections
  - Array
  - Hash
- Iterações

Nesta aula você aprenderá o que são collections e como manipulá-las.

## O que são Collections



Na programação, **collection** representa um conjunto de dados semelhantes em

uma única unidade.

**Ex:** Um amigo tem uma grande quantidade de livros sobre programação e a fim de guardá-los de forma organizada os colocou dentro de uma caixa com o rótulo “Livros de Programação”.

Note que nossa caixa é uma representação dos livros com conteúdo de Programação. Uma collection é exatamente isso, um local onde concentramos uma quantidade de itens semelhantes.

Agora, sempre que meu amigo precisar rever um livro de programação, ele saberá onde encontrá-lo!

Dois tipos de collections bastante utilizados na linguagem Ruby são Array e Hash, os quais conhecemos na segunda aula deste curso.

## Manipulando Collections

### Array

Existem várias maneiras de manipular arrays. Abaixo encontram-se algumas muito úteis para todo programador.

#### Criando um Array

1- Crie um array vazio.

Default

```
1 | irb
2 | estados = []
```

Collections podem ter zero ou mais elementos.

#### Adicionando itens

1- Insira um novo item ao array **estados**.

Default

```
1 | estados.push('Espírito Santo')
```

O push sempre irá adicionar itens de forma sequencial. ✓

2- Também é possível inserir vários elementos de uma só vez.

Default



```
1 | estados.push('Minas Gerais', 'Rio de Janeiro', 'São Paulo')
```

3- Veja o array **estados** com a instrução

Default

```
1 | puts estados
```

4- Para manter nossa coleção organizada em ordem alfabética ao inserir os itens 'Acre' e 'Amapá', devemos especificar que eles ocuparão as primeiras posições do array. Para isso contamos com o **insert**

Default

```
1 | estados.insert(0, 'Acre', 'Amapá')
```

Primeiro é passado o valor do índice onde a instrução será aplicada acompanhado por um ou mais itens a serem adicionados.

5- Exiba o array **estados**

Default

```
1 | puts estados
```

Os elementos Acre e Amapá tornaram-se os primeiros do array.

## Acessando elementos

Como já vimos na segunda aula, o item de um array pode ser acessado pelo valor de seu index.

1- Recupere o segundo elemento do array **estados**

Default

```
1 | estados[1]
```

Saiba que o primeiro elemento não inicia no índice 1, mas sim no 0.

2- Você também pode acessar índices através de intervalos

Default

```
1 | estados[2..5]
```



Retorna os itens dos índices 2, 3, 4 e 5

Utilizando números negativos conseguimos recuperar elementos a partir do último item do array, de forma regressiva. O número -1 representa o último elemento.

3- Adquira o penúltimo elemento de **estados**

Default

```
1 | estados[-2]
```

4- Também funciona com intervalos

Default

```
1 | estados[-3..-1]
```

5- Uma forma muito intuitiva e natural de recuperar o primeiro item é usar **first**

Default

```
1 | estados.first
```

6- Seguindo a mesma ideia, use **last** para o último

Default

```
1 | estados.last
```

## Obtendo informações

1- Para saber a quantidade de itens em um Array você pode utilizar qualquer uma destas duas instruções

Default

```
1 | estados.count
2 | estados.length
```

2- Descubra se o array está vazio

Default

```
1 | estados.empty?
```

O resultado será verdadeiro ou falso



3- verifique se um item específico esta presente

Default

```
1 | estados.include?('São Paulo')
```

Igual ao empty, também resulta um valor verdadeiro ou falso

Feedback



## Excluindo elementos

1- Remova um item através de seu índice

Default

```
1 | estados.delete_at(2)
```

2- Exclua o ultimo item do array **estados**

Default

```
1 | estados.pop
```

3- Para excluir o primeiro item faça

Default

```
1 | estados.shift
```

## Hash

A seguir veja exemplos importantíssimos sobre manipulação de Hashes

### Novo Hash

1- Crie um hash vazio

Default

```
1 | capitais = Hash.new
```

2- Um Hash também pode ser iniciado com vários pares de chave-valor

Default

```
1 | capitais = { acre: 'Rio Branco', sao_paulo: 'São Paulo' }
```

3- A chave de um Hash pode ser qualquer tipo de dado

Default

```
1 | hash = {1 => 'Chave do tipo inteiro', true => 'Chave do tipo booleano', [1,2,3] => 'Chave' }
```



## Adicionando itens

1- Adicione um novo item ao hash **estados**

Default

```
1 | capitais[:minas_gerais] = "Belo Horizonte"
```

2- Acesse a capital que acabamos de inserir utilizando sua chave

Default

```
1 | capitais[:minas_gerais]
```

De forma sucinta, a chave é o index de nossos itens

3- Para retornar todas as chaves de um hash

Default

```
1 | capitais.keys
```

4- Agora, todos os valores de um hash

Default

```
1 | capitais.values
```

## Exclusão

1- Remova um elemento chave-valor

Default

```
1 | capitais.delete(:acre)
```

## Obtendo informações

1- Descubra o tamanho do hash

Default

```
1 | capitais.size
```

2- Verifique se o Hash está vazio

Default

```
1 | capitais.empty?
```



## Iterações

Agora, você será apresentado a três novas estruturas de repetição utilizadas para trabalhar com **collections**

## Each

Percorre uma coleção de forma parecida ao **for**, porém, não sobrescrevendo o valor de variáveis fora da estrutura de repetição.

## Array

1- Adicione o seguinte código a um programa chamado **each\_array.rb**

Default

```
1 names = ['Joãozinho', 'Manoel', 'Juca']
2
3 name = 'Leonardo Scorza'
4
5 names.each do |name|
6   puts name
7 end
8
9 puts name
```

Ao executar o programa perceba que não foi alterado o valor da variável `name`, definida antes da estrutura de repetição.

## Hash

1- Crie um arquivo chamado **each\_hash.rb** com o seguinte código

Default

```
1 aulas = {'Aula 1 ' => 'liberada', 'Aula 2 ' => 'liberada', 'Aula 3 ' => 'liberada', 'Aul
2
3 aulas.each do |key, value|
4   puts "#{key} #{value}"
5 end
```

Em cada vez que a estrutura percorre o hash, o elemento atual é representado por `key` e `value`.

## Map

Cria um array baseando-se em valores de outro array existente.



1- Crie um arquivo chamado **map.rb**

Default

Feedback



```
1 array = [1, 2, 3, 4]
2
3 # \n é uma quebra de linha
4 puts "\n Executando .map multiplicando cada item por 2"
5 # .map não altera o conteúdo do array original
6 new_array = array.map do |a|
7     a * 2
8 end
9
10 puts "\n Array Original"
11 puts " #{array}"
12
13 puts "\n Novo Array"
14 puts " #{new_array}"
15
16 puts "\n Executando .map! multiplicando cada item por 2"
17 # .map! força que o conteúdo do array original seja alterado
18 array.map! do |a|
19     a * 2
20 end
21
22 puts "\n Array Original"
23 puts " #{array}"
24 puts ''
```

Como vimos neste exemplo, podemos forçar que o array original seja alterado utilizando map!

## Select

Realiza uma seleção de elementos presentes em uma collection através de uma condição pré definida. Traz como resultado somente os valores que passam nesta condição.

## Array

1- Crie um arquivo chamado **select\_array.rb**

Default

```
1 array = [1, 2, 3, 4, 5, 6]
2
3 selection = array.select do |a|
4     a >= 4
5 end
6
7 puts selection
```

A condição para que um item do array seja selecionado é que seu valor seja maior ou igual a 4.

## Hash

1- Crie um arquivo chamado **select\_hash.rb** ✓



Default

Feedback



```
1 hash = {0 => 'zero', 1 => 'um', 2 => 'dois', 3 => 'tres'}
2
3 puts 'Selecionando keys com valor maior que 0'
4 selection_key = hash.select do |key, value|
5     key > 0
6 end
7
8 puts selection_key
```

Veja que dentro de um Hash podemos fazer uma seleção por chave ou valor.

## Missões especiais

### Missão 1

Utilizando uma collection do tipo Array, escreva um programa que receba 3 números e no final exiba o resultado de cada um deles elevado a segunda potência.

### Missão 2

Crie uma collection do tipo Hash e permita que o usuário crie três elementos informando a chave e o valor. No final do programa para cada um desses elementos imprima a frase “Uma das chaves é \*\*\*\* e o seu valor é \*\*\*\*”

### Missão 3

Dado o seguinte hash:

Numbers = {a: 10, b: 30 2, c: 20, d: 25, e: 15}

Crie uma instrução que seleciona o maior valor deste hash e no final imprima a chave e valor do elemento resultante.

## Aula passada

Código da missão passada

Default

```
1 result = ''
2 loop do
3   puts result
4   puts 'Selecione uma das seguintes opções'
5   puts '1- Adicionar'
6   puts '2- Subtrair'
7   puts '3- Multiplicar'
8   puts '4- Dividir'
9   puts '0- Sair'
10  print 'Opção: '
11
```





```

11
12 option = gets.chomp.to_i
13
14 case option
15 when 1..4
16   print 'Digite o primeiro número: '
17   number1 = gets.chomp.to_i
18
19   print 'Digite o segundo número: '
20   number2 = gets.chomp.to_i
21   case option
22   when 1
23     result = "#{number1} + #{number2} = #{number1 + number2}"
24   when 2
25     result = "#{number1} - #{number2} = #{number1 - number2}"
26   when 3
27     result = "#{number1} * #{number2} = #{number1 * number2}"
28   when 4
29     result = "#{number1} / #{number2} = #{number1 / number2}"
30   end
31 when 0
32   break
33 else
34   result = 'Opção inválida'
35 end
36 # Comando que limpa o console
37 system "clear"
38 end

```

Obs: Não se preocupe caso seu código esteja diferente da solução apresentada. Na programação sempre existem várias formas de resolver o mesmo problema.

## DISCUSSION



*brunoaraujo* 4 MESES AGO

### Desafio M1, M2 e M3

```
result = ""
```

```
loop do
```

```
  puts "Potência"
```

```
  puts "Selecione um das seguintes opções"
```

```
  puts "1 - Calculo elevado a segunda Potencia (M1)"
```

```
  puts "2 - Impressão da chave e valor informado (M2)"
```

```
  puts "3 - Maior valor do hash (M3)"
```

```
  puts "0 - Sair"
```

```
  print "Opção: "
```

```
  option = gets.chomp.to_i
```

```
  i = 1
```

```
  case option
```

```
    when 1
```

```
      potencia = []
```





```
3.times do
  print "Digite o #{i}º numero: "
  potencia1 = gets.chomp.to_i
  potencia.push(potencia1)
  i += 1
end

potencia.each do |a|
  result = a ** 2
  puts "O resultado do número #{a} elevado a segunda potência é #{result}"
end

when 2
  hash = {}

  3.times do
    print "Informa a #{i}º chave: "
    chave = gets.chomp

    print "Informe o #{i}º valor: "
    valor = gets.chomp

    hash[chave] = valor
    i += 1
  end

  hash.each do |c, v|
    puts "Uma das chaves é #{c} e o valor é #{v}"
  end

  when 3
    numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}

    maiornumero = 0
    result = []

    numbers.each do |key, value|
      if value > maiornumero
        maiornumero = value
        result = [key, value]
      end
    end

    puts "A chave do maior número é #{result[0]} com o valor #{result[1]}"

    when 0
      break if option == 0
    else
```





REPL

```
result = "Opção não identificada"
end
end
```



*evertonlc* 8 MESES AGO

```
# CRIAR UMA CALCULADORA QUE PERMITA:
# SOMAR, SUBTRAIR, MULTIPLICAR E DIVIDIR
# COM A OPÇÃO DE SAIR

option = ""
result = 0

loop do
  puts 'Calculadora Padrão'
  puts 'Escolha dentre as opções abaixo.'
  puts '1 – Somar | 2 – Subtrair | 3 – Multiplicar | 4 – Dividir | 0 – Sair '
  print 'Digite o número correspondente; '
  command = gets.chomp.to_i

  if command == 1
    puts 'SOMA'
    print 'Digite o primeiro número: '
    numberOne = gets.chomp.to_i

    print 'Digite o segundo número: '
    numberTwo = gets.chomp.to_i

    option = 'soma'
    result = numberOne + numberTwo

  elsif command == 2
    puts 'SUBTRAÇÃO'
    print 'Digite o primeiro número: '
    numberOne = gets.chomp.to_i

    print 'Digite o segundo número: '
    numberTwo = gets.chomp.to_i

    option = 'subtração'
    result = numberOne - numberTwo

  elsif command == 3
    puts 'MULTIPLICAÇÃO'
    print 'Digite o primeiro número: '
    numberOne = gets.chomp.to_i
```



```
print 'Digite o segundo número: '
numberTwo = gets.chomp.to_i

option = 'multiplicação'
result = numberOne * numberTwo

elsif command == 4
puts 'DIVISÃO'
print 'Digite o primeiro número: '
numberOne = gets.chomp.to_i

print 'Digite o segundo número: '
numberTwo = gets.chomp.to_i

option = 'divisão'
result = numberOne / numberTwo

elsif command == 0
break
else
result = 'Valor inválido, tente novamente!'
end

system "clear"
puts "Resultado da #{option} é: #{result}"
puts ""
end
```

REPLY

*Tatiane Reis* 9 MESES AGO

# Missão 1

```
numeros = []

3.times do
print "Número: "
numeros.push(a=gets.chomp.to_i)
end

pot = numeros.map do |a|
a ** 2
end
```

```
puts " Números inseridos pelo usuário: #{numeros}"
puts " Segunda potência dos números: #{pot}"
```



Fernando Galvão 9 MESES AGO

## # Missões especiais

### # Missão 1

# Utilizando uma collection do tipo Array, escreva um programa que receba 3 números

# e no final exiba o resultado de cada um deles elevado a segunda potência.

# criando uma array vazio

```
array_num = []
```

# conferindo se o array está vazio

# para isso, podemos usar as funções ".length" e/ou ".count"

```
puts "
```

```
p array_num.length
```

```
p array_num.count
```

```
puts "
```

# podemos também usar a função ".empty?"

array\_num.empty? # o resultado será verdadeiro (true) ou falso (false)

```
puts "
```

# inserindo três dados numéricos no array

# para isso, utilizaremos a inserção via usuário/teclado

# e guardaremos nas variáveis "num\_1", "num\_2" e "num\_3"

```
print "Digite o primeiro número – inteiro e/ou decimal: "
```

```
num_1 = gets.chomp.to_f # a função ".to_f" transforma a string e um número float
```

```
print "Digite o segundo número – inteiro e/ou decimal: "
```

```
num_2 = gets.chomp.to_f
```

```
print "Digite o terceiro número – inteiro e/ou decimal: "
```

```
num_3 = gets.chomp.to_f
```

```
puts "
```

# iremos utilizar a função ".push" para carregar os dados numéricos digitados pelo usuário

# e também iremos organiza-los em ordem numérica crescente, utilizando a função ".sort" ✓

# e sobrescrevendo o array e imprimindo o seu resultado



```
p array_num = array_num.push(num_1, num_2, num_3).sort
puts "

puts "\nExecutando .map elevando a segunda potência"
# .map não altera o conteúdo do array original
new_array_num = array_num.map do |x|
  x ** 2
end
puts "

# imprimido os arrays – “Array Original” e “Novo Array”

puts "\nArray Original"
puts " #{array_num}"

puts "\nNovo Array"
puts " #{new_array_num}"
puts "

# outra solução

array = []

i = 1

1..3.times do
  print "Digite o #{i}º número: "
  array.push gets.chomp.to_i
  i += 1
end

array.each do |a|
  result = a ** 2
  puts "\n\tO resultado do número #{a} elevado a segunda potência é #
{result}"
end

# Missão 2

# Crie uma collection do tipo Hash e permita que o usuário crie três
elementos informando a chave e o valor.
# No final do programa para cada um desses elementos imprima a
frase “Uma das chaves é **** e o seu valor é ****”

# criando um novo Hash vazio

conta_cliente = Hash.new
puts "
```



```
# imprimindo o Hash criado

puts "conta_cliente = #{conta_cliente}" # não possui nenhum dado
puts "

# dentro do Hash "conta_cliente" iremos instanciar as chaves e sem
nenhum valor

p conta_cliente = {nome:", cpf:", renda:", cidade:", estado:"} #
imprimindo o Hash com suas chaves e sem valores
puts "

# agora iremos preencher cada chave com seu respectivo valor
conforme a instrução de entrada
# e iremos salvar cada valor na sua referida chave

print "Digite o seu nome: "
conta_cliente[:nome] = gets.chomp

print "Digite o seu cpf: "
conta_cliente[:cpf] = gets.chomp

print "Digite a sua renda mensal: "
conta_cliente[:renda] = gets.chomp.to_f

print "Digite o nome da cidade: "
conta_cliente[:cidade] = gets.chomp

print "Digite a sigla do estado: "
conta_cliente[:estado] = gets.chomp

puts "

# imprimindo o Hash depois de todas as chaves já com valores

p conta_cliente
puts "

# realizando iteração em cada chave e seu valor
# e imprimindo cada iteração

conta_cliente.each do |key, value|
  puts "\n\tUma das chaves é #{key} e o seu valor é #{value}\n"
end
puts "

# outra solução
```

```
hash = {} # instanciando um Hash vazio
```



5.times do # com a função .times , diminuimos seu ciclo de repetição em 3x

print 'Informe uma chave: ' # entrada do nome da chave via teclado

key = gets.chomp # guardando o nome da chave na variável chamada "key"

print 'Informe seu valor: ' # entrada do valor da chave via teclado

value = gets.chomp # guardando o valor na variável "value"

hash[key] = value # associando a chave ao valor

end

puts "

hash.each do |k, v|

puts "\nUma das chaves é #{k} e o seu valor é #{v}"

end

puts "

p hash

puts "

# Missão 3

# Dado o seguinte hash:

# numbers = {a: 10, b: 30 2, c: 20, d: 25, e: 15}

# Crie uma instrução que seleciona o maior valor deste hash e no final imprima a chave e valor do elemento resultante.

p numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}

puts "

maior\_valor = numbers.sort\_by {|key,value| value}.reverse.first

puts "Segue a chave com maior valor: #{maior\_valor}"

puts "

# outra solução

numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}

valor\_maior = 0

resultado = []

numbers.each do |key, value|

if value > valor\_maior

valor\_maior = value

resultado = [key, value]

end

end



```
puts "A chave com o maior valor é: \n\t#{resultado[0]}:#{resultado[1]}"
puts "
```

REPLY 

JOAO.MOREIRA 11 MESES AGO

**Desafio 3**

```
Numbers = {'a' => 10, 'b' => 30, 'c' => 20, 'd' => 50, 'e' => 15}
```

```
maior_value = Numbers.values.max
```

```
puts "O elemento de maior valor é #{Numbers.values.max}"
```

REPLY



JOAO.MOREIRA 11 MESES AGO

**Corrigindo o hash**

```
Numbers = {'a' => 10, 'b' => 30, 'c' => 20, 'd' => 25, 'e' => 15}
```

REPLY



JOAO.MOREIRA 11 MESES AGO

```
Numbers = {'a' => 10, 'b' => 30, 'c'
=> 20, 'd' => 25, 'e' => 15}
puts "A chave de maior valor é #
{Numbers.key(Numbers.values.max)} e o
elemento de maior valor é #
{Numbers.values.max}"
```

REPLY



RafaelSanzio 11 MESES AGO

```
Numbers = {'a' => 10, 'b' => 30, 'c' => 20, 'd' => 25, 'e' =>
15, 'f' => 79}
```

```
def maior(hash)
```

```
maior = hash.values.max
```

```
puts "KEY -> #{hash.key(maior)} VALUE -> #{maior}"
```

```
maior
```

```
end
```

```
maior(Numbers)
```

REPLY



Eulania Soares de Oliveira Costa 12 MESES AGO





```
num = [2,4,6]
```

```
num.each do |a|  
  a = a * 2  
  print " #{a},"  
end  
  
print "\n#{num}"
```

Feedback



REPLY



*KAUA RICARDO FRANCO COSTA* 12 MESES AGO

Não entendi absolutamente nada na parte do Map. Alguém me ajuda por favor?

REPLY



*Raquel Limá* 10 MESES AGO

Kaua, você ainda precisa de ajuda para entender? Se ainda tiver dúvidas, me manda uma mensagem.

REPLY



*Roseane Vieira Moreira* 1 ANO AGO

### Desafio 3

```
numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}  
puts numbers.sort_by {|key,value| value}.reverse.first  
p numbers.key(numbers.values.max)
```

REPLY



*Thiago* 1 ANO AGO

01

```
numeros = []  
  
3.times do |i|  
  puts "Digite o #{i+1} numero"  
  numeros[i] = gets.chomp.to_i  
end  
  
numeros_quadrado = numeros.map do |a|  
a*a  
end
```





```
print numeros_quadrado
-----

02
hash = {}

3.times do |i|
  puts "Digite a CHAVE do #{i+1} elemento"
  key = gets.chomp
  puts "Digite o VALOR do #{i+1} elemento"
  value = gets.chomp
  hash.merge!({key => value})
end

hash.each do |key, value|
  puts "Uma das chaves é '#{key}' e o seu valor é '#{value}'"
end
-----

03
menor = -99
numbers = {:A => 10, :B => 30, :C => 20, 😊 => 25, :E => 15}

numbers.each do |key, value|
  if value > menor
    menor = value
  end
end

selection_key = numbers.select do |key, value|
  value == menor
end

puts selection_key
```

REPLY

*Paula Isabela Cardoso Resende* 1 ANO AGO

count = 0

```
array = []

3.times do
  count += 1
  puts "Digite o #{count}º número:"
  array[count-1] = gets.to_i
end

newArray = array.map do |a|
```



```
a * a
end

print newArray
```



REPLY

*Eduardo* 2 ANOS AGO

```
loop do
  puts result
  puts 'Select one of the following options'
  puts '1 – Addition'
  puts '2 – Subtraction'
  puts '3 – multiplication'
  puts '4 – Division'
  puts '0 – Go out'

  option = gets.chomp.to_i

  if option == 1
    addition = one_number + second_number
    result = "the result of addiction is: #{addition}"
  elsif option == 2
    subtraction = one_number – second_number
    result = "the result of subtraction is: #{subtraction}"
  elsif option == 3
    multiplication = one_number * second_number
    result = "the result of multiplication is: #{multiplication}"
  elsif option == 4
    division = one_number / second_number
    result = "the result of division is: #{division}"
  elsif option == 0
    break
  else
    print 'Inavlid options'
  end
end

system "clear"
```



end

REPLY



*Simonsen Lucio da Silva* 2 ANOS AGO

3o. Desafio:

```
numbers = {A: 10, B:30, C: 20, D: 25, E:15}

maior_value = numbers.values.max

maior = numbers.select do |key, value|
  value >= maior_value
end

puts "O elemento de maior valor é #{maior}"
```

REPLY



*Gustavo Coelho* 2 ANOS AGO

#MISSÃO 1

```
ar = [2 , 4 , 3]

ar.each do |pot|
  puts pot**2
end

#MISSÃO 2

hash = Hash.new
loop do
  puts "0-sair ; 1-continuar"
  op = gets.chomp.to_i
  if op == 1
    puts "escreva uma chave: "
    key = gets.chomp
    puts "escreva um valor: "
    val = gets.chomp
    hash[key] = val
  else break
end
end

hash.each do |key, value|
  puts "Uma das chaves é #{key} e o seu valor é #{value}"
end
```



## #MISSAO 3

```
numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}
m = 0

numbers.each do |key, value|
  if value > m
    m = value
  end
end

numbers.each do |key, value|
  if value == m
    puts "Chave: #{key}\nValor: #{value}"
  end
end
```

REPLY

*Ideilson* 2 ANOS AGO

===Missão 1===

```
numbers = []
i = 1

puts "Digite três números:"
3.times do
  puts "Digite o #{i}º número"
  numbers.push(gets.chomp.to_i)
  i += 1 # i = i + 1
end

new_numbers = numbers.map do |number|
  number ** 2
end

puts "\nNúmeros digitados: #{numbers}\nNúmeros elevando a 2ª
potência: #{new_numbers}"
===Missão 2===
hash = {}
i = 1
puts "Digite 3x: "
3.times do
  puts "Digite #{i}ª chave e #{i}ª valor: "
  hash[gets.chomp] = gets.chomp
  i += 1
end
```



```
end

hash.each do |k, v|
  puts "Uma das chaves é: #{k} e o seu valor é : #{v}"
end

===Missão 3===

Numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}

max = 0
puts 'Selecionando chave que contém maior valor'
Numbers.select do |key, value|
  if value > max
    max = value
  end
end

Numbers.select do |k, v|
  if max == v
    puts ":{k}=>#{v}"
    break
  end
end
```

REPLY

Frankyston Lins 2 ANOS AGO

<https://gist.github.com/frankyston/b95a57f3e75252cdb973371983adbe>

REPLY



Flávio S Ferreira 2 ANOS AGO

### MISSÃO 1

```
numbers = []
i = 1
3.times do
  print "Digite o #{i}º número: "
  numbers.push(gets.chomp.to_i)
  i+=1
end
numbers_product = numbers.map do |number|
  number ** 2
end
puts "Array original: #{numbers}"
```







```
puts "Array com segunda potência: #{numbers_product}"
```

### MISSÃO 2

```
hash = {}
keys = []
values = []
count = 1
puts '### Criando hash ###'
3.times do
  print "\nDigite a #{count}ª chave: "
  keys[count-1] = gets.chomp
  print "Digite o #{count}º valor: "
  values[count-1] = gets.chomp
  hash[keys[count-1]] = values[count-1]
  count+=1
end
puts hash
```

### MISSÃO 3

```
numbers = {a: 10, b: 302, c: 20, d: 25, e: 15}
hash_result = {}
high_value = 0
high_key = ""
numbers.each do |key, value|
  if value > high_value
    high_value = value
    high_key = key
  end
end
puts "Chave: #{high_key} Valor: #{high_value}"
```

REPLY



Lucas Teixeira 2 MESES AGO

### MISSÃO 3

```
numbers = {a: 10, b: 302, c: 20, d: 25, e: 15}
hash_result = {}
high_value = 0
high_key = ""
numbers.each do |key, value|
  if value > high_value
```



```
high_value = value
high_key = key
end
end
puts "Chave: #{high_key} Valor: #{high_value}"
```

REPLY

*Arlei Oliveira* 2 ANOS AGO

Desculpe, mas ao colar o código, as formatações foram perdidas. Aqui no curso, é possível postar como código para não perder a formatação? Obrigado.

REPLY

*Leonardo Scorza* 2 ANOS AGO

Olá Arlei 😊

Você pode gerar um Gist para enviar <https://gist.github.com> o/  
Obrigado por acompanhar o curso.

REPLY

*Arlei Oliveira* 2 ANOS AGO

Bom dia. Segue a forma como fiz.  
Grato pelo curso, estou aprendendo bastante...

```
def le_hashes
  print 'Entre com um ID (key): '
  key = gets.chomp
  print 'Entre com o valor (value): '
  value = gets.chomp.to_i
  print "\n"
  return key, value
end

def imprime_hashes(valores)
  print "Imprimindo os valores armazenados...\n"
  valores.each do |key, value|
    print "\nA chave é: #{key} e o valor é: #{value}"
  end
  print "\n"
end
```





```
valores = Hash.new

puts 'Digite q em ID e 0 em value para sair'

loop do
  key, value = le_hashes
  #puts "Key lida: #{key}"
  #puts "Value lido: #{value}"
  if key == 'q'
    break
  end
  valores[key] = value
end

imprime_hashes(valores)
```

REPLY

*Robson* 2 ANOS AGO**Missões:**

```
1-
array = []
puts "Informe 3 Valores:"
print "Valor 1: "
array.push(gets.chomp.to_i)
print "Valor 2: "
array.push(gets.chomp.to_i)
print "Valor 3: "
array.push(gets.chomp.to_i)

#puts "#{array[0]}"

val1 = array[0] ** 2
puts "Potência do valor 1 #{val1}"

val2 = array[1] ** 2
puts "Potência do valor 2 #{val2}"

val3 = array[2] ** 2
puts "Potência do valor 3 #{val3}"

2-
puts 'Informe 3 Chaves e 3 Valores:'
print "Chave 1: "
key1 = gets.chomp.to_s
```



```
print "Valor 1: "
value1 = gets.chomp.to_s

print "Chave 2: "
key2 = gets.chomp.to_s
print "Valor 2: "
value2 = gets.chomp.to_s

print "Chave 3: "
key3 = gets.chomp.to_s
print "Valor 3: "
value3 = gets.chomp.to_s

hash = {key1 => value1, key2 => value2, key3 => value3}

hash.each do |key, value|
  puts "A Chave é #{key} e o Valor é #{value}!"
end
```

---

3-

Numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}

```
if Numbers.values[0] > Numbers.values[1]
  maior = Numbers.values[0]
else
  maior = Numbers.values[1]
end
```

```
if maior < Numbers.values[2]
  maior = Numbers.values[2]
elsif maior < Numbers.values[3]
  maior = Numbers.values[3]
elsif maior < Numbers.values[4]
  maior = Numbers.values[4]
end
```

```
print "Maior número é: #{maior} \n"
```

Se puder avaliar meus códigos seria ótimo, parabéns pelo conteúdo, até aqui tá top!!

REPLY



Keylane 2 ANOS AGO

1º DESAFIO



1ª resolução

puts "Digite 3 numeros"

n1 = gets.chomp.to\_i

n2 = gets.chomp.to\_i

n3 = gets.chomp.to\_i

numeros = [n1,n2,n3]

numeros.map! do |a|

a \*\* 2

end

puts "Resultado #{numeros}"

2º resolução:

puts "Digite 3 numeros para o programa poder funcionar: "

cont = 1

array = []

3.times do

puts "Digite o #{cont} numero"

array.push(gets.chomp.to\_i)

cont +=1

end

new\_array = array.map do |a|

a \*\* 2

end

puts "Array original #{array}"

puts "Novo array #{new\_array}"

2º DESAFIO

puts "Digite uma chave e o valor:"

key1 = gets.chomp.to\_i

value1 = gets.chomp.to\_i

puts "Digite uma chave e o valor:"

key2 = gets.chomp.to\_i

value2 = gets.chomp.to\_i

puts "Digite uma chave e o valor:"

key3 = gets.chomp.to\_i

value3 = gets.chomp.to\_i

hash = {key1 => value1, key2 => value2, key3 => value3}



```
hash.each do |key, value|
```

```
  puts "Uma das chaves é #{key} e o seu valor é #{value}"
```

```
end
```

### 3º DESAFIO

```
numbers = {'a' => 10, 'b' => 30, 'c' => 20, 'd' => 25, 'e' => 15}
```

```
x=0
```

```
k=0
```

```
chave=0
```

```
m=0
```

```
numbers.each do |key, value|
```

```
  if value >= x #maior
```

```
    x = value
```

```
    k = key
```

```
  end
```

```
  m = x
```

```
  if value < m
```

```
    m = value
```

```
    chave = key
```

```
  end
```

```
end
```

```
numbers.each do |key, value|
```

```
  if value < m
```

```
    m = value
```

```
    chave = key
```

```
  end
```

```
end
```

```
puts "A chave do maior valor é #{k} e o valor é #{x}"
```

```
puts "A chave do menor valor é #{chave} e o valor é #{m}"
```

[REPLY](#)

*Lucas Ponte e Silva* 2 ANOS AGO

Não sei se a missão 3 poderia ser feito assim,

segue:

```
hash = {10 => '0', 30 => '1', 20 => '2', 25 => '3', 15 => '4'}
```

```
puts 'Selecionando value com valor maior que 0'
```

```
selection_key = hash.select do |key, value|
```

```
  key > 29
```



```
end
```

puts "maior valor é #{selection\_key}"

REPLY



Paulo Rogerio 2 ANOS AGO

Pior que não ,pq as vezes vc pode ã saber o valores dentro da hash.

REPLY



Edward 3 ANOS AGO

#Desafio 1 – Utilizando uma collection do tipo Array, escreva um programa que receba 3 números e no final exiba o resultado de cada um deles elevado a segunda potência.

```
puts 'Exercício 1'
```

```
array = [2, 3, 4]
```

```
array.each do |elevado_segunda_potencia|
```

```
puts elevado_segunda_potencia**2
```

```
end
```

#Desafio 2 – Crie uma collection do tipo Hash e permita que o usuário crie três elementos informando a chave e o valor. No final do programa para cada um desses elementos imprima a frase "Uma das chaves é \*\*\*\* e o seu valor é \*\*\*\*"

```
puts 'Exercício 2'
```

```
hash = {0 => 'BackEnd', 1 => 'FrontEnd', 2 => 'FullStack'}
```

```
hash_tres_valores = hash.select do |key, value|
```

```
puts "Uma das chaves é #{key} e o seu valor é #{value}"
```

```
end
```

#Desafio 3 – Crie uma instrução que seleciona o maior valor deste hash e no final imprima a chave e valor do elemento resultante.

```
puts 'Exercício 3'
```

```
hash_numeros = {Letra_A: 10, Letra_B: 30.20, Letra_C: 20, Letra_D: 25, Letra_E: 15}
```

```
key = hash_numeros.sort{|a,b| a[1] b[1]}.last[0]
```

```
value = hash_numeros.sort{|a,b| a[1] b[1]}.last[1]
```

```
puts "A posição do hash de maior valor foi \n Chave: #{key} – Valor: #{value}"
```



REPLY

Karlysson Alves 3 ANOS AGO

```

numbers = { A: 10, B: 302, C: 20, D: 25, E: 15 }
max = numbers.values.max
numbers.each do |k, v|
  puts "OS dados do maior valor é chave[#{k}] valor[#{v}]" if v.to_i == max
end

```

Feedback



REPLY



Priscila Ferreira Bezerra 3 ANOS AGO

```

puts ""
puts "-----"
puts ""
# Missão 1
# Utilizando uma collection do tipo Array, escreva um programa que
# receba 3 números e no final exiba o resultado de cada um deles
# elevado a segunda potência.

arrayColection = [1, 2, 3, 4]

arrayClone = arrayColection.map do |a|
  a ** 2
end

puts "Original value #{arrayColection} !"
puts "Result is #{arrayClone} !"

puts ""
puts "-----"
puts ""

# Missão 2
# Crie uma collection do tipo Hash e permita que o usuário crie três
# elementos informando a chave e o valor. No final do programa para
# cada um desses elementos imprima a frase "Uma das chaves é **** e o
# seu valor é ****"

hashElement = {}

x = 1
while x <= 3
  print "Enter with the key: "
  keyValue = gets.chomp
  print "Enter with the value: "
  hashValue = gets.chomp
  hashElement[keyValue] = hashValue
end

```







```

x += 1
end
puts "#{hashElement}"

hashElement.each do |key,value|
puts "Uma das chaves é: #{key} e o seu valor é: #{value} "
end

puts ""
puts "-----"
puts ""

# Missão 3
# Dado o seguinte hash:
# Numbers = {a: 10, b: 30 2, c: 20, d: 25, e: 15}
# Crie uma instrução que seleciona o maior valor deste hash e no final
imprima a chave e valor do elemento resultante.

number = {a: 10, b: 30 , c: 20, d: 25, e: 15}

new_number = number.sort_by{|k, v| -v}
print "Original hash: #{number}"
puts ""
print "the biggest number in hash is: #{new_number.first}"
puts ""
print "the lowest number in hash is: #{new_number.last} "

```

REPLY



*Rodrigo de Melo Marcolino* 3 ANOS AGO

O ruby n possui reduce???

REPLY



*Denilson Silva* 3 ANOS AGO

```

array = [1,2,3]
hash = {1 => 'um', 2 => 'dois', 3 => 'tres' }
numbers = {A: 10, B: 30, C: 20, D: 25, E: 15}

array1 = array.map do |a|
a**2
end
print array1

puts "\n"

hash1 = hash.each do |key, value|

```





```
puts "UMA DAS CHAVES E #{key} E SEU VALOR #{value}"
end
print hash1
puts "\n"

print numbers.values.max
puts "\n"
```

REPLY

*Marcio* 3 ANOS AGO

olá professor, Seria possivel você olhar meu codigo, em especial a desafio 3 e mandar um feedback?

desafio1

-----

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

```
new_array = array.map do|a|
```

```
a**2
```

```
end
```

```
puts "\n array original"
```

```
puts "#{array}"
```

```
puts "\n Array elevado a segunda potência "
```

```
puts "#{new_array}"
```

desafio 2

-----

```
hash= Hash.new
```

```
3.times do
```

```
puts "digite uma chave para o hash: "
```

```
key = gets.chomp
```

```
puts "agora digite o valor para essa chave"
```

```
valor = gets.chomp
```

```
hash[key]=valor
```

```
end
```

```
hash.each do |key,value|
```

```
puts "uma das chaves é #{key} e o seu valor é #{value}"
```

```
end
```

desafio 3

-----



```
Numbers = {a: 10, b: 30 , c: 20, d: 25, e: 15}
```

```
max = 0
max_key = 0
Numbers.each do|key,value|
  if value > max
    max = value
    max_key = key
  end
end

puts " chave e valor do maior elemento, #{max_key} : #{max} "
```

REPLY

*Leonardo Scorza* 3 ANOS AGO

Opa e ai Marcio, beleza?

Legal cara, sua solução funciona o/

REPLY

*Marcio* 3 ANOS AGO

olá professor, estou fazendo o curso no window com o vs code, mas quando coloco acento nas palavras ele da esse tipo de erro:

```
irb(main):003:0> estados.push('Ceará')
```

Traceback (most recent call last):

```
3: from C:/Ruby26-x64/bin/irb.cmd:31:in ' '
```

```
2: from C:/Ruby26-x64/bin/irb.cmd:31:in load'
```

```
1: from C:/Ruby26-x64/lib/ruby/gems/2.6.0/gems/irb-
```

```
1.0.0/exe/irb:11:in ``
```

SyntaxError ((irb):3: invalid multibyte char (UTF-8)), como faço para corrigir?

REPLY

*Leonardo Scorza* 3 ANOS AGO

Bom dia Marcio, como vai?

A sua aspas simples está correta? (o certo é ', aqui no comentário está: ')

Valeu o/

REPLY

*Gabriel* 3 ANOS AGO



Gabriel Santos

## Desafio 3:

```
numeros = {A: 10, B: 30, C:20, D: 25, E:15}
teste = numeros.sort_by{|key, value| value}.to_h
puts "A chave do maior valor é #{teste.keys.last} e a maior valor é #{
{teste.values.last}."
```

Feedback



REPLY

William Medeiros Silva 3 ANOS AGO

## "Exercicio 1"

```
array = [1, 2, 3,]

# \n é uma quebra de linha
puts "\n Executando .map elevado a segunda potencia: "
# .map não altera o conteúdo do array original
new_array = array.map do |a|
  a ** 2
end

puts "\n Valor atual"
puts " #{array}"

puts "\n Nova Potencia"
puts " #{new_array}"
```

REPLY

William Medeiros Silva 3 ANOS AGO

## "Exercicio 2"

```
curso = { 0 => 'Ruby', 1 => 'Python', 2 => 'Javascript'}

#curso.each do |key, value|
#puts "#{key} #{value}"
#end

puts 'Selecionando o curso maior que 0'
selection_key = curso.select do |key, value|
  key > 0
end

puts selection_key
```

REPLY



## "Exercicio = 3 "

```
Numbers = {'a:' => 10, 'b:'=> 30, 'c:'=> 20, 'd:'=> 25, 'e:' => 15}

puts 'Selecionando o valor maior que 29'
selection_value = Numbers.select do |key, value|
  value > 29
end

puts selection_value
```

REPLY



Caio Duque 3 ANOS AGO

tenho uma duvida Scorza ou a algum que me ajude, como realizo um select numa string, tipo uma pesquisa de nome, podendo ser tanto pelo nome completo ou pelas 1 ou as duas primeiras letras destes possíveis nomes dentro de um array

REPLY



wmv123 3 ANOS AGO

Bom dia Caio, se eu entendi, fica mais fácil usando expressão regular, exemplo:

```
#Criando a string completa
string_completa = "Caio Duque"

#Var que armazenará a sua busca
string_busca = "Caio"

if !!string_completa.match("^#{string_busca}.*")
  puts "A palavra #{string_busca} existe!"
else
  puts "#{string_busca}, não encontrada"
end
```

Explicando:

Já deu pra ver que a busca é feita dentro do método match, do inglês "correspondente".

Sobre expressão regular (regex):

Explicado o conteúdo dentro de match("^#{string\_busca}.\*"), por partes.

o símbolo "^", , significa início do texto,

imagine se ele aparecesse, seria assim:

```
^TEXTO TEXTO TEXTO TEXTO TEXTO TEXTO
^TEXTO TEXTO TEXTO TEXTO TEXTO
^TEXTO TEXTO
```

Então se usar ele na busca, ele começa a procurar a partir do início do texto.

Sobre o “#{string\_busca}” é o parâmetro que deseja buscar e “#{ }” é a forma de concatenar a string dentro comando match()

o ponto “.”, em expressão regular significa qualquer caracter, ou seja a partir de “Caio. (Caio.ponto)”, procura-se qualquer coisa por conta do ponto e \* “.\*”

OBS: O “!” no if, se usado ele retorna true ou false, no teste de uma string por exemplo.

```
2.5.0 :096 > s = nil
```

```
=> nil
```

```
2.5.0 :097 > !!s
```

```
=> false
```

```
2.5.0 :098 > s = “Caio”
```

```
=> “Caio”
```

```
2.5.0 :099 > !!s
```

```
=> true
```

Com expressões regulares da pra fazer muita coisa.

Espero ter ajudado

Olhe isso: <https://ruby-doc.org/core-2.1.1/Regexp.html>

REPLY

Yuri Tavares 3 ANOS AGO

Missão 1

```
numeros = []
```

```
count = 1
```

```
3.times do |numero|
```

```
puts “Informe o numero #{count}!”
```

```
numero = gets.chomp
```

```
puts “Entendi #{numero}”
```



```
numeros <= 30
```



REPL

```
end
puts "O maior numero esta na chave #{bigger.keys} e valor é #
#{bigger.values}"
```



Moisés Tedeschi de Melo 3 ANOS AGO

### Questão 1

```
hash_simple = {}
3.times{
  print 'Informe uma chave: '
  chave = gets.chomp
  print 'Informe um valor: '
  valor = gets.chomp
  hash_simple[chave] = valor
}
hash_simple.each do |key, value|
  puts "Uma das chaves é: #{key} e o seu valor é: #{value}"
end

####
```

### Questão 2

```
numeros = []
3.times{
  print 'Entre com um número: '
  numeros.push(gets.chomp.to_i)
}
numeros.each do |numero|
  puts "\nResultado: #{numero**2}"
end

###
```

### Questão 3

```
numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}
puts 'Selecione o maior valor dentro de um intervalo "X"'

selecionar_max = numbers.select do |key, value|
  value == numbers.values.max
end

puts "O maior valor é: #{selecionar_max}"
```





Luan 3 ANOS AGO

Adorando o curso.

Segue minha calculadora:

```
result = "
```

```
loop do
```

```
puts result
```

```
puts '1 somar'
```

```
puts '2 subtrair'
```

```
puts '3 multiplicar'
```

```
puts '4 dividir'
```

```
print '5 sair'
```

```
option = gets.chomp.to_i
```

```
if option == 1
```

```
print ' numero 1 : '
```

```
numero_1 = gets.chomp.to_i
```

```
print 'numero 2 : '
```

```
puts numero_2 = gets.chomp.to_i
```

```
puts resultado = numero_1 + numero_2
```

```
puts result = "o total de #{numero_1 } Somado ao #{numero_2} é igual a  
#{resultado} "
```

```
elsif option == 2
```

```
print ' numero 1 : '
```

```
numero_1 = gets.chomp.to_i
```

```
print 'numero 2 : '
```

```
puts numero_2 = gets.chomp.to_i
```

```
puts resultado = numero_1 - numero_2
```

```
puts result = "o total de #{numero_1 } Subtração #{numero_2} é igual a  
#{resultado} "
```

```
elsif option == 3
```

```
print ' numero 1 : '
```

```
numero_1 = gets.chomp.to_i
```

```
print 'numero 2 : '
```

```
puts numero_2 = gets.chomp.to_i
```

```
puts resultado = numero_1 * numero_2
```

```
puts result = "o total de #{numero_1 } Multiplicado #{numero_2} é igual  
#{resultado} "
```





```
a #{resultado}

elsif option == 4
  print ' numero 1 : '
  numero_1 = gets.chomp.to_i
  print 'numero 2 : '
  puts numero_2 = gets.chomp.to_i
  puts resultado = numero_1 / numero_2
  puts result = "o total de #{numero_1 } Dividido #{numero_2} é igual a #{resultado} "

  if option >= 4
    end
  end
  break
end
```

REPLY

*José dos Santos Junior* 3 ANOS AGO**Missão 1**

```
array = []
print "Calcule qual quer numero elevado a 2 \n"
print "Coloque o primeiro numero a ser calculado: "
prnumero = gets.chomp.to_i
print 'Coloque o segundo numero a ser calculado: '
segnumero = gets.chomp.to_i
print 'Coloque o teceiro numero a ser calculado: '
ternnumero = gets.chomp.to_i

array.insert(0, prnumero)
array.insert(1, segnumero)
array.insert(2, ternnumero)

valor1 = array[0] ** 2
valor2 = array[1] ** 2
valor3 = array[2] ** 2

puts "O valor dos numeros #{prnumero} elevado a 2 é #{valor1} "
puts "O valor dos numeros #{segnumero} elevado a 2 é #{valor2} "
puts "O valor dos numeros #{ternnumero} elevado a 2 é #{valor3} "
```

**Missão 2**

hash = Hash.new



3.times do



```
print "informe a Key para Hash: "  
key = gets.chomp  
print "informe o Valor para Hash: "  
valor = gets.chomp  
hash[key] = valor  
end  
hash.each do |key, value|  
puts "Uma das chaves é #{key} e o seu valor é #{value}"  
end
```

Missão 3

```
numbers = {a: 10, b: 30, c: 20, d: 25, e: 15}  
  
selection_key = numbers.select do |key, value|  
value == numbers.values.max  
end  
  
selection_key.each do |key, value|  
puts "A Chave do maior Valor é #{key} e o Valor maior é #{value}"  
end
```

REPLY

*Jakeline* 3 ANOS AGO

```
array = []  
  
3.times do  
puts "Digite um número:"  
array.push(gets.chomp.to_i)  
end  
puts "Array criado: #{array}"  
  
# .map não altera o conteúdo do array original  
new_array = array.map do |a|  
a ** 2  
end  
puts "Elevando cada número do array a segunda potência: #  
{new_array}"
```

REPLY



*Iago Silva Vieira* 3 ANOS AGO

Desafio 01:

```
puts "Informe 3 números aleatórios para o Programa funcionar."
```





```
cont = 1
array = []
3.times do
  puts "\nDigite o #{cont}º Numero: "
  array.push(gets.chomp.to_i)
  cont += 1
end

puts "\nEste é o seu conjunto de numeros que será elevado à 2º
potência: #{array}"

sleep(2)

array.map! do |a|
  a**2
end

puts "Após a elevação, os seus numeros ficarão da seguinte forma: #
{array}"
```

---

Desafio 02:

```
puts "Informe 3 chaves e um valor para cada uma dessas chave: "

hash = {}
chave = nil
value = nil

3.times do |cont|
  print "\nInforme a #{cont+1}º a chave: "
  chave = gets.chomp

  print "\nInforme o #{cont+1}º valor: "
  value = gets.chomp

  hash[:"#{chave}"] = " #{value}"
end

hash.each do |key, value|
  puts "Uma das chaves é: #{key}. E o seu valor é: ' #{value}' "
end
```

```
system "clear"
```

---



Desafio 03:

```
numbers = {A: 10,B: 30,C: 20,D: 25,E: 15}
max_value = numbers.select do |key, value|
  value == numbers.values.max
end
puts "O valor é: #{max_value}"
```

---

REPLY



Aroldo 3 ANOS AGO  
Missao2

```
times = Hash.new
puts 'Digite o primeiro time: '
time1 = gets.chomp
puts 'Digite o seu estadio: '
estado1 = gets.chomp
puts 'Digite o segundo time: '
time2 = gets.chomp
puts 'Digite o seu estadio: '
estado2 = gets.chomp
puts 'Digite o terceiro time: '
time3 = gets.chomp
puts 'Digite o seu estadio: '
estado3 = gets.chomp

times = { time1 => estado1 , time2 => estado2 , time3 => estado3}

times.each do |key, value|
  puts "O seu time é #{key}, e o seu estadio é #{value}"
end
```

REPLY



Aroldo 3 ANOS AGO  
Missao\_1

```
”
array = []
puts "Digite o primeiro numero."
num1 = gets.chomp.to_i
array.push(num1)
```





```
puts 'Digite o segundo numero: '  
num2 = gets.chomp.to_i  
array.push(num2)  
puts 'Digite o terceiro numero: '  
num3 = gets.chomp.to_i  
array.push(num3)  
puts "=="  
puts 'Segue o array'  
puts "=="  
  
new_array = array.map do |a|  
  a ** 2  
end  
  
puts new_array "
```

REPLY

*Leiomanaluz* 3 ANOS AGO

```
hash = {0 => 'zero', 1 => 'um', 2 => 'dois', 3 => 'tres'}
```

```
puts 'Selecionando keys com valor maior que 0'  
selection_key = hash.select do |key, value|  
  key > 0  
end
```

```
puts selection_key
```

Copiei o código escrito abaixo do vídeo e coleí no meu projeto e mandei rodar.

Apresentou um outro erro:

```
hash_select.rb:8: syntax error, unexpected end-of-input, expecting  
keyword_end  
puts selection_key
```

REPLY

*Leiomanaluz* 3 ANOS AGO

Devido ao erro, não estou nem conseguindo fazer as missões : (

REPLY

*Leiomanaluz* 3 ANOS AGO



Estou com problemas, não consegui identificar, pois está

exatamente conforme está nos códigos deixados abaixo do vídeo.  
Poderia me auxiliar, o que está acontecendo?

Esse é o erro:

Traceback (most recent call last):

2: from hash.rb:3:in ' '

1: from hash.rb:3:in 'each'

hash.rb:4:in block in '': undefined method 'puts' for main:Object  
(NoMethodError)

Did you mean? puts

REPLY



Leonardo Scorza 3 ANOS AGO

Bom dia 😊

Me manda o seu código que está dando esse erro por favor.  
Possivelmente é um problema na indentação do código:

```
-----  
hash = {0 => 'zero', 1 => 'um', 2 => 'dois', 3 => 'tres'}  
  
puts 'Selecionando keys com valor maior que 0'  
selection_key = hash.select do |key, value|  
  key > 0  
end  
  
puts selection_key
```

REPLY

Leiomanaluz 3 ANOS AGO

Boa noite,

Copiei o código que está debaixo do vídeo, coleí  
no projeto, não alterei nada, nem indentação e  
mandei rodar. Apresentou isso:

```
hash_select.rb:8: syntax error, unexpected end-of-  
input, expecting keyword_end  
puts selection_key
```

REPLY

ASK QUESTION



