# **Métodos e Gems**

#### Conteúdo

O conteúdo que será abordado na aula:

* **Métodos**
  + O que são?
  + Trabalhando com métodos
    - Como criar?
    - Parâmetros
    - Retorno
* **Gems**
  + O que são?
  + Trabalhando com gems
    - Como instalar uma gem
    - Novas Funcionalidades
    - Desinstalar uma gem
    - Informação
    - Bundler

#### Métodos

### O que são?

Método é uma forma de organizar funções específicas de um programa.  Caso necessário permite a reutilização de código, ou seja, evita escrever o mesmo código diversas vezes.

### Trabalhando com métodos

### Como criar?

1- Crie seu primeiro método em um arquivo chamado **method.rb**

|  |  |
| --- | --- |
| 1  2  3  4  5 | def talk   puts 'Olá, como você está?'  end    talk |

Para definir um método utiliza-se a palavra reservada **def**seguida pelo nome do método. Depois é escrito um conjunto de expressões, e por fim, a palavra **end**determina o término do método.

Para executar um método basta apenas escrever o seu nome.

### Parâmetros

1- Substitua o código de **method.rb** por

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | def talk(first\_name, last\_name)   puts "Olá #{first\_name} #{last\_name}, como você está?"  end    first\_name = 'Leonardo'  last\_name = 'Scorza'    talk(first\_name, last\_name) |

Um método pode depender de um ou mais parâmetros para realizar determinada tarefa.

O exemplo define um método que precisa do primeiro e segundo nome para exibir uma frase. Tendo ciência disso, deve-se passar dois argumentos toda a vez que o método talk for chamado.

2- Crie um arquivo chamado **transito.rb** com o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | def signal(color = 'vermelho')   puts "O sinal está #{color}"  end    signal    color = 'verde'  signal(color) |

Definindo um valor padrão ao parâmetro, ele torna-se opcional.

Um método pode ser invocado mais de uma vez, em diferentes áreas do programa. Assim seu código está sendo reutilizado.

Quer mudar o código? Altere apenas o método!

### Retorno

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | def compare(a, b)   a > b  end    a = 1  b = 2    result = compare(a, b)    puts "O resultado da comparação é '#{result}'" |

O retorno de um método ruby é sempre o resultado de sua última instrução.

#### Gems

### O que são?

Gem é um pacote que oferece funcionalidades a fim de resolver uma necessidade específica de um programa Ruby. Pense como o conceito de **biblioteca** em outras linguagens de programação.

### Trabalhando com gems

### Como instalar uma gem

1- Para instalar uma **gem**execute em seu terminal

|  |  |
| --- | --- |
| 1 | gem install os |

Você instalou a gem **os.**Ela é uma biblioteca para você extrair informações sobre o PC que você está usando (exp: Sistema operacional, quantidade de cores e etc).

**Novas Funcionalidades**

1- Crie um arquivo chamado **os.rb** e adicione o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | require 'os'    def my\_os   if OS.windows?       "Windows"     elsif OS.linux?       "Linux"     elsif OS.mac?       "Osx"     else       "Não consegui identificar"   end    end    puts "Meu PC possui #{OS.cpu\_count} cores, é #{OS.bits} bits e o sistema operacional é #{my\_os}" |

No começo do arquivo, o require ‘gem\_name’ carrega os arquivos da gem. Isso possibilita a declaração de códigos com funcionalidades da biblioteca.

obs: Para saber como utilizar uma gem consulte sua documentação.

**Desinstalar uma gem**

1- Para desinstalar uma gem utilize o comando

|  |  |
| --- | --- |
| 1 | gem uninstall gem\_name |

Ex:

|  |  |
| --- | --- |
| 1 | gem uninstall os |

**Informação**

1- Liste todas as gems instaladas na máquina rodando

|  |  |
| --- | --- |
| 1 | gem list |

**Bundler**

Para ter controle sobre as dependências de um projeto contamos com uma ferramenta que procura e instala gems chamada Bundler.

1- Crie um projeto chamado **first\_project**

|  |  |
| --- | --- |
| 1  2 | mkdir first\_project  cd first\_project |

2- O bundler também é uma gem. Para instalá-lo rode

|  |  |
| --- | --- |
| 1 | gem install bundler |

3- Salve a lista de gems do projeto em um arquivo chamado **Gemfile**

|  |  |
| --- | --- |
| 1  2  3 | source 'https://rubygems.org'    gem 'os' |

Na primeira linha é definido onde o bundle deve procurar pelas gems.

Depois é listado as dependências do projeto.

4- Instale estas gems com o comando.

|  |  |
| --- | --- |
| 1 | bundle install |

Assim, qualquer pessoa envolvida no projeto pode facilmente instalar suas dependências.

### Missões especiais

### Missão 1

Crie um programa que possua um método que resolva a potência dado um número base e seu expoente. Estes dois valores devem ser informados pelo usuário.

### Missão 2

Siga a documentação da gem cpf\_cnpj para criar um programa que recebe como entrada um número de cpf e em um método verifique se este número é válido.