

CURSO DE RUBY PURO

OneBitCode

[INTRODUÇÃO DO CURSO 5](#_Toc18289799)

[Funcionamento do Curso 5](#_Toc18289800)

[Aulas do curso: 5](#_Toc18289801)

[INTRODUÇÃO AO RUBY 7](#_Toc18289802)

[História e características 7](#_Toc18289803)

[Por que aprender Ruby? 7](#_Toc18289804)

[Instalação 7](#_Toc18289805)

[Rbenv - RVM 7](#_Toc18289806)

[Codeanywhere 7](#_Toc18289807)

[Primeiro Hello World 7](#_Toc18289808)

[IRB 7](#_Toc18289809)

[Missões especiais 01 8](#_Toc18289810)

[CONCEITOS BÁSICOS 9](#_Toc18289811)

[Tipos de dados 9](#_Toc18289812)

[O que são 9](#_Toc18289813)

[Integer 9](#_Toc18289814)

[Float 9](#_Toc18289815)

[Boolean 9](#_Toc18289816)

[String 9](#_Toc18289817)

[Array 9](#_Toc18289818)

[9](#_Toc18289819)

[Symbol 9](#_Toc18289820)

[9](#_Toc18289821)

[Hash 10](#_Toc18289822)

[10](#_Toc18289823)

[Tipagem dinâmica 10](#_Toc18289824)

[Operadores Matemáticos 10](#_Toc18289825)

[Entrada/Saída 10](#_Toc18289826)

[Print: 10](file:///C:\Users\felip\StudiStation\Ruby\RubyPuro\ApostilaRubyPuro.docx#_Toc18289827)

[Convertendo para inteiro 10](#_Toc18289828)

[Missões especiais 02 10](#_Toc18289829)

[Estruturas de Controle 11](#_Toc18289830)

[Condicionais 11](#_Toc18289831)

[If 11](#_Toc18289832)

[Else 11](#_Toc18289833)

[Elsif 11](#_Toc18289834)

[Unless 11](#_Toc18289835)

[Case 12](#_Toc18289836)

[Iteração 12](#_Toc18289837)

[For 12](#_Toc18289838)

[Times 12](#_Toc18289839)

[While 12](#_Toc18289840)

[Loop (Do While) 13](#_Toc18289841)

[Missões especiais 03 13](#_Toc18289842)

[Collections 14](#_Toc18289843)

[O que são? 14](#_Toc18289844)

[Array 14](#_Toc18289845)

[Criando array 14](#_Toc18289846)

[Inserindo dados no final do array 14](#_Toc18289847)

[Inserindo dados no início do Array 14](#_Toc18289848)

[Acessando dados no Array 14](#_Toc18289849)

[Tamanho do Array 14](#_Toc18289850)

[Verificando se é/está vazio 14](#_Toc18289851)

[Verificando se tem um item 15](#_Toc18289852)

[Removendo itens do Array 15](#_Toc18289853)

[Hash 15](#_Toc18289854)

[Criando um Hash 15](#_Toc18289855)

[Definindo valores em um Hash 15](#_Toc18289856)

[Adicionando novos dados no Hash 15](#_Toc18289857)

[Pegando todas a chaves do Hash 15](#_Toc18289858)

[Acessando dados do Hash 15](#_Toc18289859)

[Tamanho do Hash 15](#_Toc18289860)

[Verificando se é vazio 16](#_Toc18289861)

[Removendo um valor do Hash 16](#_Toc18289862)

[Iterações 16](#_Toc18289863)

[Each 16](#_Toc18289864)

[Map 16](#_Toc18289865)

[Select 16](#_Toc18289866)

[Missões especiais 16](#_Toc18289867)

INTRODUÇÃO DO CURSO

Funcionamento do Curso

* Aulas teóricas e práticas;
* Missões especiais (tarefas) no final de cada aula;
* Projeto de conclusão ao final do curso;

Aulas do curso:

* Introdução ao Ruby
  + História e Características;
  + Instalação;
  + Seu primeiro “Hello World!”;
  + Ruby Irb;
* Conceitos Básicos
  + Tipos de dados;
  + Operadores matemáticos;
  + Strings;
  + Entrada/Saída;
* Estrutura de Controle
  + If/else/elif;
  + Unless;
  + Case;
  + Iteração;
  + For;
  + While;
  + Times;
  + Do while;
* Collections
  + Arrays;
  + Hashes;
  + Each;
  + Map;
  + Select;
* Métodos e Gems
  + Como criar métodos;
  + O que são e como usar Gems;
* POO 1 (Programação Orientada a Objetos)
  + Classes – pt1;
* POO 2
  + Classes – pt2;
  + Construtor
* Ruby Avançado
  + Blocks;
  + Lambda;
  + Namespacing;
  + Modules;
* Ruby Avançado II
  + Regex;
  + Math and time;
  + Methodos misssing;
  + Procs;
* Projeto final
  + Leitura de arquivos;
  + Chamadas web;
  + Web scraping;
  + Projeto final.

INTRODUÇÃO AO RUBY

História e características

* Criado na década de 90 pelo Japonês Yukihiro Matz Matsumoto;
* Lançado ao publico em 1995;
* Orientado a objetos;
* Tipagem dinâmica e forte;
* Interpretada;
* Expansão ligada a criação do Ruby on Rails;

Por que aprender Ruby?

* É uma linguagem produtiva e divertida;
* É fácil aprender;
* Tem uma grande comunidade;
* É usada por grandes empresas;
* Está presente em várias áreas;
* Top 10 nos índices de popularidades;

Instalação

Rbenv - RVM

Ferramenta para gerenciar várias versões do Ruby.

* + - * 1. <https://gorails.com/setup>;
        2. Sistema Operacional desejado;

Subsidiando o Linux no Windows – (<https://olhardigital.com.br/dicas_e_tutoriais/noticia/como-instalar-o-ubuntu-e-outras-distribuicoes-linux-no-windows-10/71845>).

Codeanywhere

Site onde pode usar o Ruby online. Também existe outros sites.

Primeiro Hello World

puts “Hello World”;

p “Hello World”;

puts(“Hello World”);

IRB

Irb é um Ruby executável dentro do próprio terminal, onde não se cria nenhum arquivo. Porém com isso, acaba sendo bem limitado.

Missões especiais 01

1. Instalar o Ruby na sua máquina ou preparar o Codeanywhere.
2. Exiba no console a seguinte mensagem: “Curso de Ruby do ONEBITCODE”.
3. Leia o seguinte artigo: Ruby (linguagem de programação) – Wikipédia, A enciclopédia livre. ( <https://pt.wikipedia.org/wiki/Ruby_(linguagem_de_programa%C3%A7%C3%A3o)> ).

CONCEITOS BÁSICOS

Tipos de dados

O que são

É o conceito que usamos para classificar dados, possibilitando a definição de regras para cada tipo. Com eles o Ruby sabe como lidar com os dados de nosso programa. Para sabermos o tipo usamos “.class” exemplo: var.class;

String

var = “Hello World”

var.class

Integer

Como na matemática, inteiro é o tipo de dado que representa o conjunto de números positivos, negativos e 0.

var = 10;

Float

Tipo que representa os números reais inexatos. De forma abreviada, são números decimais, números que contem ponto flutuante.

var = 3.5;

Boolean

Tipo de dado usado para informar a veracidade de algo. Possui apenas dois estados, sendo eles true que é uma instância da TrueClass e false que é uma instância da classe FalseClass.

var = true;

var = false;

String

Tipo que representa um texto. Um conjunto de letras, símbolos ou números. Pode ser definido de várias formas, porém, as mais comuns são dentro de aspas simples ou duplas.

var = “Hello World”

Array

Um tipo que nos permite armazenar uma lista ordenada de dados em único objeto. Para definir um array você deve utilizar colchetes. ([ ]).

var = [“Felipe”,”Mariangela”,”Carlos”,”Mauricio”];

var[1]

Mariangela

Symbol

Símbolo é um tipo de dados semelhantes a String, com a diferença de que ele é um objeto imutável. Duas strings idênticas podem ser objetos diferentes, mas um símbolo é apenas um objeto, ocupado sempre o mesmo espaço na memória. Um símbolo sempre é definido começando com dois pontos : seguido pelo seu nome

sim = :”Hello”;

sim.object\_id;

00000000 (id do objeto na memoria)

Hash

Tipo que representa uma coleção de dados organizados por chaves únicas e seus respectivos valores. Enquanto Arrays são definidos com colchetes. Hash são definidos com chaves ({ }).

var = {curso :”ADS”, língua :”PT-BR”, duracao\_Semestres = :”6”, local = :”Estacio”};

var[:curso];

ADS

Tipagem dinâmica

No Ruby não é preciso definir o tipo de dado antes de atribuir um valor à uma variável. O tipo é dinâmico, ou seja, ele é definido de acordo com o dado atribuído.

Operadores Matemáticos

* + ► Adição (10+10 = 20)
* - ► Subtração (20-10 = 10)
* \* ► Multiplicação (2\*5 = 10)
* / ► Divisão (20/4 = 5)
* % ► Modulo (9%4 = 1)
* \*\* ► Expoente (5\*\*2 = 25)

Entrada/Saída

Essas duas operações manipulam dado.

Com a diferença que a entrada ocorre quando o programa lê dados que podem ter sido recebidos de um teclado, de um arquivo, ou então de outro programa.

Já a saída é um dado produzido pelo programa que pode ser exibido em uma tela, enviado para um arquivo ou então para outro programa.

Print:

Uma diferença do print é que ele não pula linha automática como o puts.

#Output(Saida)

print (“Digite seu nome”);

#Input(Entrada)

name = gets.chomp;

#Saida usando puts

puts (“Hello #{name}!”);

Convertendo para inteiro

gets.chomp.to\_i;

Missões especiais 02

1. No IRB, crie todos os tipos especiais de dados mencionados na aula com valores diferentes como exemplo.
2. Crie um programa que receba o nome e idade de uma pessoa e no final exiba estes dois dados em uma única frase.
3. Crie um programa que receba dois números inteiros e no final exiba a soma, a subtração, a multiplicação, e a divisão entre eles.

Estruturas de Controle

Códigos que escrevemos em nossos programas para analisar dados e decidir algo a ocorrer. Divide-se em dois tipos:

* Condicional.
* Iteração.

Condicionais

Estrutura que executa um trecho do código dependendo do resultado da condição.

If

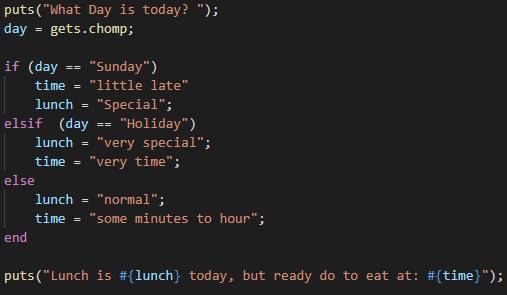
Verifica se uma condição é verdadeira (true), e a partir desse resultado determina se as instruções dentro de seu escopo serão ou não executadas.

Else

Informa o que fazer quando a verificação do if for falsa.

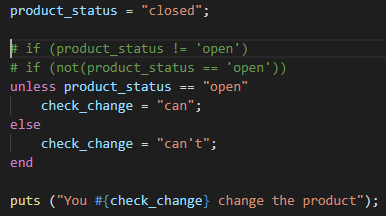
Elsif

Utilizado quando há necessidade de verificar mais de uma condição em um if.



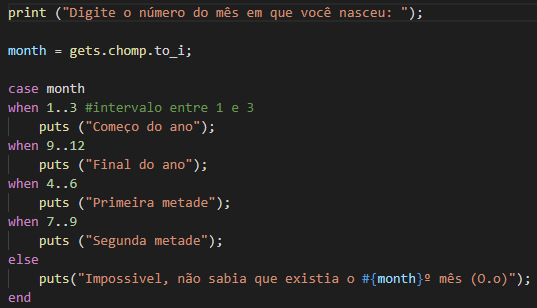
Unless

Enquanto o if é executado quando a condição for verdadeira, o unless ocorre de forma contrária. É executado apenas quando é falsa.



Case

Instrução muito parecida com o if. Ele se enquadra muito bem a situações com diversas condições.

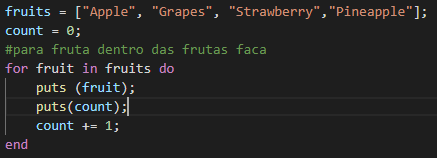


Iteração

Estrutura de controle que gerencia quantas vezes um trecho de código será executado.

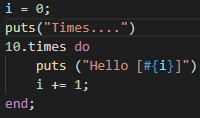
For

Usado para percorrer uma coleção de elementos.

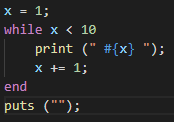


Times

Executa um trecho de código, por um determinado número de vezes.

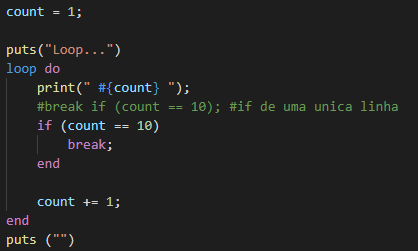


While

Instrução que repete o bloco de código enquanto sua condição for verdadeira.

Loop (Do While)

Cria um laço de repetição que só é parado quando instrução break for verdadeira.



Missões especiais 03

Utilizando as estruturas de iteração e de condição, crie uma calculadora que ofereça ao usuário a opção de multiplicar, dividir, adicionar ou subtrair números. Não se esqueça de também permitir que usuário feche o programa.

Collections

O que são?

Na programação, collection representa um conjunto de dados semelhantes em uma única unidade.

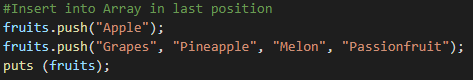
Fora: Um conjunto de livros agrupados, uns conjuntos numéricos, um conjunto de espécies, etc...

Array

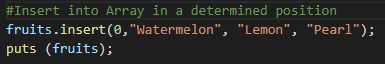
Criando array



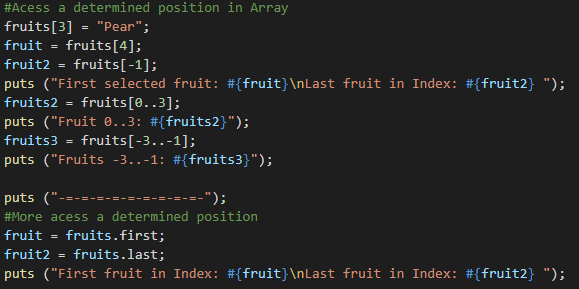
Inserindo dados no final do array



Inserindo dados no início do Array



Acessando dados no Array



Tamanho do Array



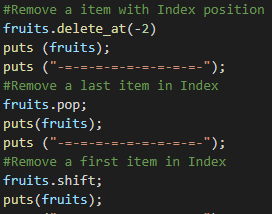
Verificando se é/está vazio



Verificando se tem um item



Removendo itens do Array



Hash

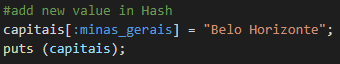
Criando um Hash



Definindo valores em um Hash



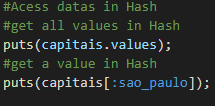
Adicionando novos dados no Hash



Pegando todas a chaves do Hash



Acessando dados do Hash



Tamanho do Hash



Verificando se é vazio



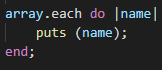
Removendo um valor do Hash



Iterações

Each

Percorre uma coleção de forma parecida ao For, porém, não sobrescreve o valor de variáveis fora da estrutura de repetição.

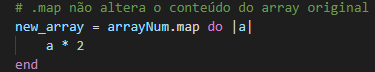




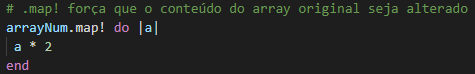
Map

Cria um Array baseando-se em valores de um Array já existente.

De maneira que mantém o conteúdo original intacto.

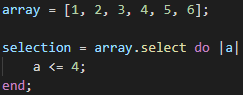


De maneira que alterar os valores do conteúdo original.



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.





Missões especiais 04

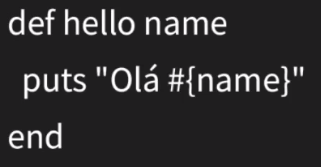
1. Utilizando uma collection do tipo Array, escreva um programa que receba 3 números e no final exiba o resultado de cada um deles elevado a segunda potência.
2. Crie uma collection do tipo Hash e permita que o usuário crie três elementos informando a chave e o valor. No final do programa para cada um desses elementos imprima a frase “Uma das chaves é \*\*\*\* e o seu valor é \*\*\*\*”
3. Dado o seguinte hash: Numbers = {a: 10, b: 30, 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

Métodos

O que são?

São uma forma organizada de instruções em um programa, permitindo que trechos de códigos sejam reutilizáveis.



Gems

Gem é um pacote que oferece funcionalidade a fim de resolver funcionalidades especificas de um programa Ruby.

Pense como o conceito de bibliotecas em outras linguagens de programação.

Instalando uma gem

Gem Install name

Gem Install os

Removendo uma Gem

Gem uninstall name

Gem uninstall os

Listando Gems

Gem list

Bundler

Agrega varias gems em um arquivo e depois instala todas elas.

Instalando

Gem install bundler

Criando

Criar arquivo chamado GemFile

Executando

Abrir pasta do GemFile, executar o seguinte comando:

bundle

Missões especiais 05

1. Crie um programa que possua um método que resolva a potência dado um número base e seu expoente. Esses dois valores devem ser informados pelo usuário.
2. Siga a documentação da gem CPF\_CNPJ para criar um programa que recebe como entrada um numero de CPF e em um método verifique se esse número é válido.