**O Início**

**Conteúdo**

Nesta aula você aprenderá conceitos básicos de Ruby, importantes para o início de aprendizado em qualquer linguagem de programação.

Será abordado:

* Tipos de dados
* Operadores Matemáticos
* Entrada/Saída

**Tipos de dados**

Existem para classificar dados, possibilitando a definição de regras para cada tipo. Com eles o Ruby sabe como lidar com os dados de nossos programas.

Os principais tipos de dados são:

* **Integer**

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

1. Crie um objeto do tipo **Integer** atribuindo à uma variável o seguinte valor inteiro.

(variável é um local onde armazenamos dados que serão reutilizados).

irb

integer\_number = -20

2. Confirme que a variável realmente é do tipo inteiro executando.

|  |  |
| --- | --- |
| 1 | integer\_number.class |

* **Float**

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

1. Faça que um objeto tenha o tipo **float** atribuindo a ele um valor decimal.

(praticamente tudo no ruby é um objeto).

|  |  |
| --- | --- |
| 1 | float\_number = -20.05 |

2. Prove que o objeto realmente é do tipo **float** executando.

|  |  |
| --- | --- |
| 1 | float\_number.class |

* **Boolean**

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

(não se preocupe caso não saiba o que é uma classe, iremos abordar este assunto em uma próxima aula)

1. Inicie uma variável com tipo **boolean**da seguinte forma.

|  |  |
| --- | --- |
| 1 | ruby\_free\_course =  true |

2. Confirme que ela é uma instância da classe TrueClass executando.

|  |  |
| --- | --- |
| 1 | ruby\_free\_course.class |

* **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.

1. Crie um objeto do tipo **String**atribuindo à uma variável o seguinte texto.

|  |  |
| --- | --- |
| 1 | happiness =  "Programming with ruby" |

2. Para ter certeza que o objeto é uma string, rode

|  |  |
| --- | --- |
| 1 | happiness.class |

* **Array**

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

1. Crie um objeto do tipo **array**

|  |  |
| --- | --- |
| 1 | bitcode\_array = [ 0, 1, 2] |

2. Acesse a terceira posição do nosso array rodando

|  |  |
| --- | --- |
| 1 | bitcode\_array[2] |

3. Para nos certificarmos que bitcode\_array é do tipo Array execute

|  |  |
| --- | --- |
| 1 | bitcode\_array.class |

* **Symbol**

**Símbolo** é um tipo de dado semelhante 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, ocupando sempre o mesmo espaço na memória.

Um símbolo sempre é definido começando com dois pontos : seguido de seu nome.

1. Inicie um objeto do tipo **Symbol**atribuindo à uma variável o seguinte valor.

|  |  |
| --- | --- |
| 1 | onebit\_symbol = :ruby\_symbol |

2. Saiba qual posição da memória esse símbolo está salvo, rodando

|  |  |
| --- | --- |
| 1 | onebit\_symbol.object\_id |

3. Agora crie outra variável com o mesmo símbolo e note que ela aponta para o mesmo endereço na memória.

|  |  |
| --- | --- |
| 1  2  3 | second\_symbol = :ruby\_symbol  second\_symbol.object\_id  onebit\_symbol.object\_id |

4. Para confirmar que o objeto é do tipo **Symbol**execute

|  |  |
| --- | --- |
| 1 | onebit\_symbol.class |

* **Hash**

Tipo que representa uma coleção de dados organizados por chaves únicas e seus respectivos valores. Enquanto arrays são definidos com colchetes, hashes são definidos com chaves ‘{ }’

1. Crie um objeto do tipo **Hash** adicionando a uma variável o seguinte valor

|  |  |
| --- | --- |
| 1 | onebit\_hash = {course: 'ruby', language: 'pt-Br', locale: 'onebitcode.com'} |

2. Saiba onde encontrar o curso de ruby rodando

|  |  |
| --- | --- |
| 1 | onebit\_hash[:locale] |

3. Certifique-se que a variável é do tipo Hash executando

|  |  |
| --- | --- |
| 1 | onebit\_hash.class |

* **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.

Um exemplo é que podemos alterar o valor de uma variável diversas vezes e em cada uma delas notar que o tipo de dado também mudou.

1. Execute as seguintes linhas de comando

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | onebit\_dynamic = 2  onebit\_dynamic.class  onebit\_dynamic = "String type"  onebit\_dynamic.class  onebit\_dynamic = :onebit\_symbol  onebit\_dynamic.class |

**Operadores Matemáticos**

Para resolver operações matemáticas no ruby contamos com a seguinte lista de operadores aritméticos:

* **+** (Adição)
* **–** (Subtração)
* **\*** (Multiplicação)
* **/** (Divisão)
* **%** (Módulo)
* **\*\*** (Expoente)

**Resolvendo operações matemáticas**

1. Operação do tipo soma.

|  |  |
| --- | --- |
| 1 | -5 + 10 |

2. Subtração entre dois números

|  |  |
| --- | --- |
| 1 | 1 - 2 |

3. Multiplicação

|  |  |
| --- | --- |
| 1 | -2 \* -2 |

4. Operação do tipo divisão

|  |  |
| --- | --- |
| 1 | 10 / 2 |

5. Módulo entre a divisão de dois números. Será a sobra da divisão entre o primeiro pelo segundo.

|  |  |
| --- | --- |
| 1 | 10 % 3 |

6. Expoente

|  |  |
| --- | --- |
| 1 | 2\*\*3 |

7. As operações podem envolver mais de dois números e também mais de um operador aritmético.

|  |  |
| --- | --- |
| 1 | 2\*\*3 + 2 \* 3 |

Ressaltando que uma operação aritmética sempre terá apenas um resultado.

**Entrada/Saída**

Essas duas operações manipulam dados, 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 e a **saída** é um dado produzido pelo programa que pode ser exibido em uma tela, enviado para um arquivo ou então para outro programa.

1. Para fazer um programa com entrada e saída de dados, crie um arquivo chamado **onebit\_io.rb** e adicione o seguinte código.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # saída de dado  print 'Digite seu nome: '    # Recebendo uma entrada do teclado  name = gets.chomp    # saída utilizando puts  # use códigos ruby dentro de uma string com #{code}  puts "Hello #{name}!" |

Perceba que primeiro é exibido uma mensagem pedindo que a pessoa informe seu nome. Depois é criado uma variável chamada **name** que é igual a gets.chomp.

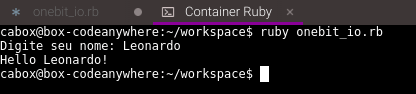
Mas afinal de contas o que é isso?

**gets** é um método que recebe como entrada um dado inserido pelo teclado. Como ele captura qualquer coisa precisamos do *.chomp* para que quando o enter for pressionado ele não crie uma quebra de linha.

Por fim realizamos uma nova saída de dado imprimindo uma saudação para a pessoa que informou seu nome.

2. Veja isso na prática rodando

|  |  |
| --- | --- |
| 1 | ruby onebit\_io.rb |

3. Resultado esperado  


O gets recebe os dados como String, mas podemos transformá-los em números inteiros e realizar operações matemáticas com eles como no exemplo a seguir.

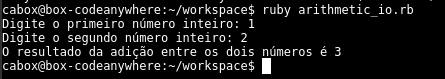
4. Crie um arquivo chamado **arithmetic\_io.rb** e adicione o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | print "Digite o primeiro número inteiro: "  # .to\_i transforma a string em um número inteiro  number1 = gets.chomp.to\_i    print "Digite o segundo número inteiro: "      number2 = gets.chomp.to\_i    addition = number1 + number2  puts "O resultado da adição entre os dois números é #{addition}" |

5. Execute o programa

|  |  |
| --- | --- |
| 1 | ruby arithmetic\_io.rb |

Depois de inserir dois valores inteiros, a saída é o resultado da adição entre os dois números.



**Missões especiais**

Preparei três missões especiais pra te deixar craque em todos assuntos que abordamos nesta aula.

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