

# COMPARAÇÃO DAS LINGUAGENS PYTHON E RUBY

## Declaração/atribuição de valores a variáveis

Não tem diferença, com exceção dos valores booleanos. Em Python eles começam com letra maiúscula (True e False), enquanto em Ruby eles começam com letra minúscula (true e false).

```
python Copy code  
  
nome = "João"    # String  
idade = 25       # Inteiro  
altura = 1.75    # Float  
ativo = True     # Booleano
```

```
ruby Copy code  
  
nome = "João"    # String  
idade = 25       # Inteiro  
altura = 1.75    # Float  
ativo = true     # Booleano
```

## Expressões aritméticas com os 4 operações

Também é a mesma coisa. Contudo, em Ruby, a operação de divisão (/) retorna um valor inteiro se a divisão for entre dois inteiros, e não um valor float float Python faz.


### adição

```
python Copy code  
  
a = 5 + 3    # Resultado: 8  
b = "Olá" + " Mundo" # Resultado: "Olá Mundo" (concatenação)
```

```
ruby Copy code  
  
a = 5 + 3    # Resultado: 8  
b = "Olá" + " Mundo" # Resultado: "Olá Mundo" (concatenação)
```


## subtração

python

 Copy code

```
a = 10 - 4 # Resultado: 6
```


ruby

 Copy code

```
a = 10 - 4 # Resultado: 6
```


## multiplicação

python

 Copy code

```
a = 7 * 3 # Resultado: 21
b = "Oi" * 3 # Resultado: "OiOiOi" (repetição de string)
```


ruby

 Copy code

```
a = 7 * 3 # Resultado: 21
b = "Oi" * 3 # Resultado: "OiOiOi" (repetição de string)
```


## divisão

python

 Copy code

```
a = 8 / 2 # Resultado: 4.0 (sempre float)
b = 9 / 2 # Resultado: 4.5 (float mesmo se resultado não for inteiro)
```

ruby

 Copy code

```
a = 8 / 2 # Resultado: 4 (inteiro se ambos forem inteiros)
b = 9 / 2 # Resultado: 4 (divisão inteira, sem conversão automática)
c = 9.0 / 2 # Resultado: 4.5 (conversão para float)
```

## Um comando condicional (if/else, switch/case)

### Python

- A sintaxe usa `if`, `elif` e `else`.
- Os blocos são definidos pela indentação (não há `{}` ou `end`).

```
python Copy code  
  
idade = 20  
  
if idade < 18:  
    print("Menor de idade")  
elif idade == 18:  
    print("Tem exatamente 18 anos")  
else:  
    print("Maior de idade")
```

### Ruby

- Usa `if`, `elsif` e `else`.
- Blocos são delimitados por `end`.

```
ruby Copy code  
  
idade = 20  
  
if idade < 18  
  puts "Menor de idade"  
elsif idade == 18  
  puts "Tem exatamente 18 anos"  
else  
  puts "Maior de idade"  
end
```

## Um comando de repetição (for, while, do while, repeat until)

A única diferença do comando **while** no Python e no Ruby é que no Ruby ele termina com um **end**.

```
python                                                                    Copy code

contador = 1
while contador <= 5:
    print(contador)
    contador += 1
```

```
ruby                                                                    Copy code

contador = 1
while contador <= 5
  puts contador
  contador += 1
end
```

## Expressões lógicas com E e OU

Em Python temos o **and** e o **or**. Embora também temos essas mesmas palavras reservadas em Ruby, quando escritas dessa forma, estamos escrevendo um operador lógico de baixa precedência. Então, em Ruby, o certo seria **||** (para "ou") e **&&** (para "e").

## Printando algo na tela

No Python temos o **print(variável)**, **print("texto")** e **print('texto')**. Enquanto no Ruby temos o **puts variável**, **puts "texto"** e **puts 'texto'**.

## Declaração e chamada de função com parâmetros posicionais

Em Python, para declarar uma função devemos inserir um **def** seguido do **nome da função** e **abrindo parêntese** para colocar seus **parâmetros**, depois vem os **dois pontos**, lembrando que os parâmetros são opcionais. A função encerra com um

**return** para indicar **algo retornado na função**, o return só não é obrigatório se a função não retornar nada.

```
python Copy code

# Definindo a função
def saudacao(nome, idade):
    return f"Olá, {nome}! Você tem {idade} anos."

# Chamando a função
resultado = saudacao("Ana", 25)
print(resultado)
```

Já em Ruby, é a mesma coisa, mas **não precisa ter dois pontos**, a **função termina com end** e também **não precisa do return**.

```
ruby Copy code

# Definindo a função
def saudacao(nome, idade)
    "Olá, #{nome}! Você tem #{idade} anos."
end

# Chamando a função
resultado = saudacao("Ana", 25)
puts resultado
```