Programação Funcional com Elixir

Controlando Fluxo com Funções

Controlando Fluxo com Funções

- Como você faria uma função para verificar qual o maior entre dois números? Em Ruby, seria algo assim...

```
def maior(n1, n2)
  if n1 > n2
    n1
  else
    n2
  end
end
```

Controlando Fluxo com Funções

- Como você faria uma função para verificar qual o maior entre dois números? Em Ruby, seria algo assim...

```
def maior(n1, n2)
  n1 > n2 ? n1 : n2
end
```

Controlando Fluxo com Funções

- Mas como faríamos a mesma coisa em Elixir, já que ainda não aprendermos condicionais? Ou seja, só utilizando funções?

Controlando Fluxo com Funções

- Em linguagens funcionais, na maioria das vezes usamos **funções** e **pattern matching** para controlar o fluxo, veja:

```
defmodule ComparaNumero do
    def maior(n1, n2) do
      check(n1 >= n2, n1, n2)
    end

    defp check(true, n1, _), do: n1
    defp check(false, _, n2), do: n2
end
```

Controlando Fluxo com Funções

- A primeira novidade é o **defp** que é quando queremos definir uma função privada:

```
defmodule ComparaNumero do
    def maior(n1, n2) do
      check(n1 >= n2, n1, n2)
    end

    defp check(true, n1, _), do: n1
    defp check(false, _, n2), do: n2
end
```

Controlando Fluxo com Funções

- A segunda é que o uso do resultado booleano de uma expressão passado para uma função permite escolher entre duas funções com assinaturas diferentes.

```
defmodule ComparaNumero do
    def maior(n1, n2) do
      check(n1 >= n2, n1, n2)
    end

    defp check(true, n1, _), do: n1
    defp check(false, _, n2), do: n2
end
```