

João Gonçalves

Video 3.2

Forms of Pattern Matching



In this video, we are going to take a look at...

- Types of pattern matching
- Binary types
- Leveraging pattern matching

$$x = 1$$

Pattern Matching

Pattern Matching Revisited

`[1, 2, 3]` = `[1, 2, 3]`

Pattern Matching Revisited

$[x, y, z] = [1, 2, 3]$

The diagram illustrates the pattern matching process. Three arrows originate from the variables x , y , and z in the pattern $[x, y, z]$ and point to the corresponding values 1 , 2 , and 3 in the list $[1, 2, 3]$. The arrow from x to 1 is a curved arrow, the arrow from y to 2 is a straight arrow, and the arrow from z to 3 is a curved arrow.

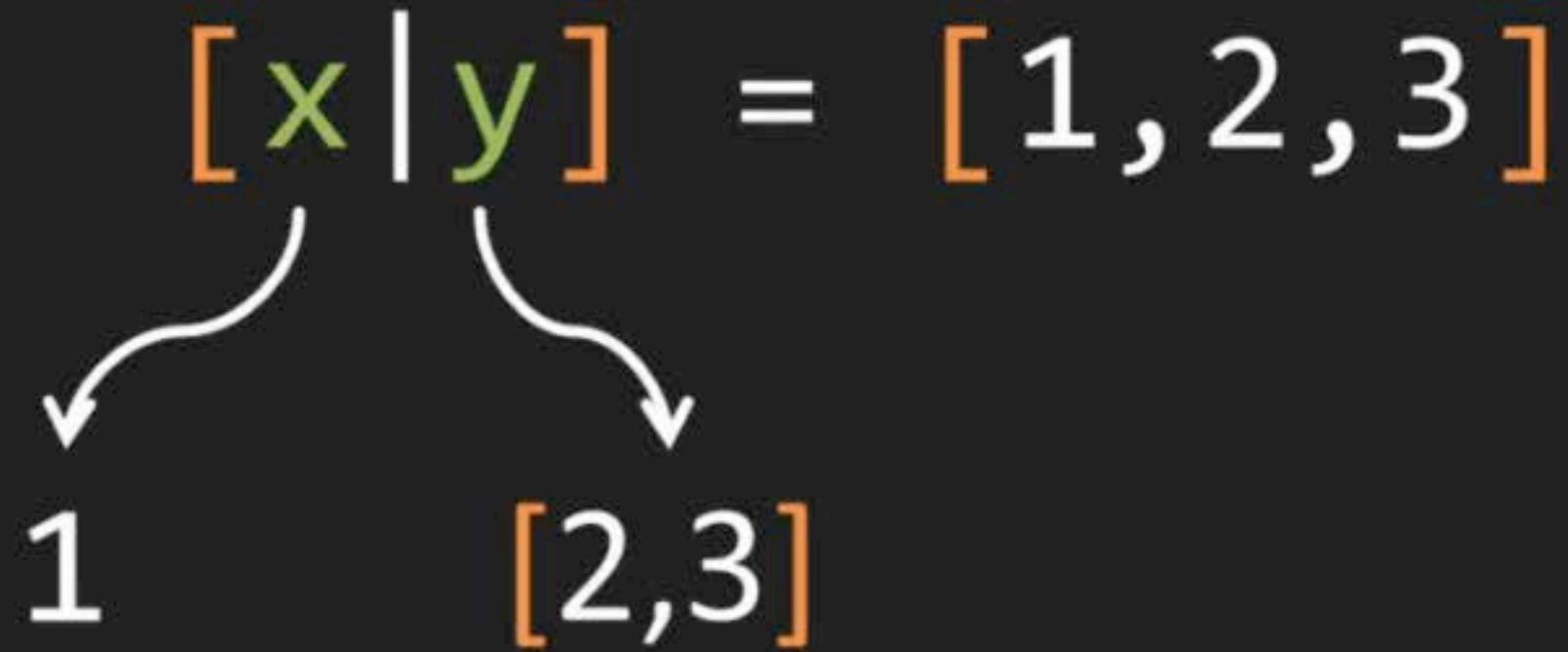
Pattern Matching Revisited

`[x, 2, 3]` = `[1, 2, 3]`

Pattern Matching Revisited

$[x | [2, 3]] = [1, 2, 3]$

Pattern Matching Revisited



Pattern Matching Revisited

`{:ok, result}` = `{:ok, 10}`

Pattern Matching Revisited

```
{name, age} = {"Francis", 30}
```

Pattern Matching Revisited

`{name, age}` = `{"Francis", 30}`

Pattern Matching Revisited

`{name, _} = {"Francis", 30}`



The underscore matches anything
and is an unreadable variable

Pattern Matching Revisited

```
%{name: name} = %{name: "Francis", age: 30}
```

Pattern Matching Revisited

`%{name: name} = %{name: "Francis", age: 30}`



Matches as long as the key is present
on the right hand side

A New Type

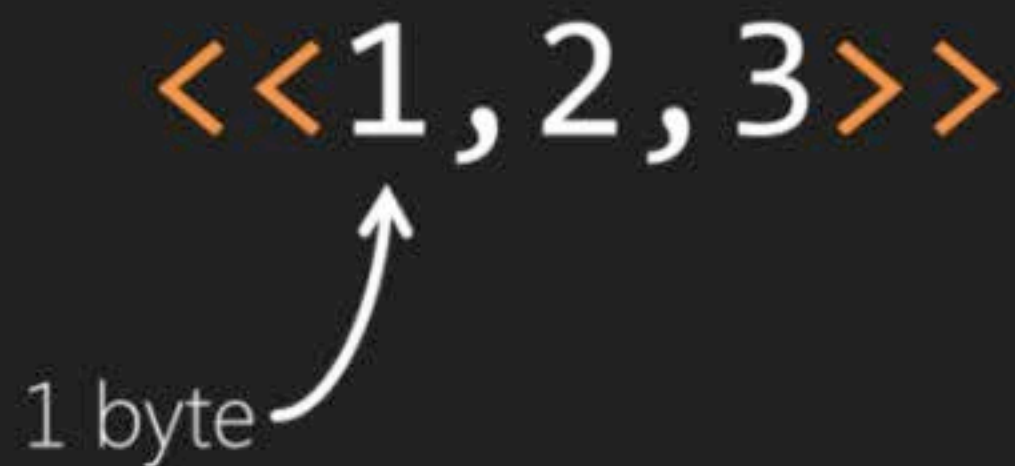
- Binary list

<<1, 2, 3>>

A New Type

<<1, 2, 3>>

1 byte

A diagram illustrating a nested list structure. The text "<<1, 2, 3>>" is displayed in white, with the opening and closing angle brackets of the inner list (<1, 2, 3>) highlighted in orange. Below the first element '1', the text "1 byte" is written in white, with a white curved arrow pointing upwards to the first orange bracket of the inner list.

A New Type

<<1, 2, 3>> <> <<4>>



concatenation

A New Type


<<1,2,3,4>>

A New Type

`<<1::size(16)>>`



Binaries can be tagged with
a size attribute



A New Type

<<65, "broad">>

"Abroad"

A New Problem

FIF (Fictitious Image Format)



A New Problem

FIF (Fictitious Image Format)

<<

```
0xCAFE::16,  
width::16,  
height::16,  
pixel_size,  
image_data::binary
```

>>

A New Problem

FIF (Fictitious Image Format)

<<

0xCAFE::16,

width::16,

height::16,

pixel_size,

image_data::binary

>>

Can only be used at
the end of the pattern

A New Problem

FIF (Fictitious Image Format)

```
<<
```

```
0xCAFE::16,
```

```
width::16,
```

```
height::16,
```

```
pixel_size,
```

```
image_data::binary
```

```
>> = << ... >>
```


Summary

- Discussed pattern matching versus assignment
- Explored pattern matching in complex types
- Discovered binaries