Level 2

The Pipe Operator

Making Function Calls Easier to Read

TRY ELIXIR

The Problem With Multiple Nested Calls

When thinking purely in terms of functions, a common mistake is writing **too many** nested function calls. This can make code hard to read.

```
f(x)
```

This reads fine.

```
g(f(x), y)
```

This is a little hard to read, but not too bad.

```
j(i(h(g(f(x),y),z)))
```

This is just silly.
Don’t do this. Ever.

TRY ELIXIR

It's Hard to Read "From the Inside Out"

account.exs

```
defmodule Account do
  def balance(initial, spending) do
    interest(discount(initial, 10), 0.1)
  end

  def discount(total, amount) do
  end

  def interest(total, rate) do
  end
end
```

No need for module name...

...because functions belong to the same module

Must be read starting from inner function

Temp Variables Are Not the Best Solution

```
defmodule Account do
  def balance(initial, spending) do
    interest(discount(initial, 10), 0.1)
  end
  ...
end
```

Same code, but using temporary variables

```
defmodule Account do
  def balance(initial, spending) do
    discount_amount = discount(initial, 10)
    interest_amount = interest(discount_amount, 0.1)
    interest_amount
  end
end
```

1- First, we read this expression on the right...

2- Next, read this here on the left...

3- Then back to this on the right...

4- And, finally, back to this on the left.

The Pipe Operator

The pipe operator `|>` takes the output from the expression on the left and passes it as the first argument to the function call on the right.

```
defmodule Account do
  def balance(initial, spending) do
    discount(initial, 10) |> interest(0.1)
  end
  ...
end
```

Easier to read from left to right

Simliar to Unix pipes!

```
$ ls | sort
```

The Unix pipe operator

Return value from this function call...

...becomes first argument to this function call.

```
discount(initial, 10) |> interest( ,0.1)
```

Piping Multiple Functions

When piping through a handful of functions, it is a good practice to use new lines.

```elixir
defmodule Account do
  def balance(initial, spending) do
    discount(initial, 10) |> interest(0.1) |> format("$")
  end
  ...
end
```

Lines that are too long can become hard to maintain

```elixir
defmodule Account do
  def balance(initial, spending) do
    discount(initial, 10)
    |> interest(0.1)
    |> format("$")
    ...
  end
end
```

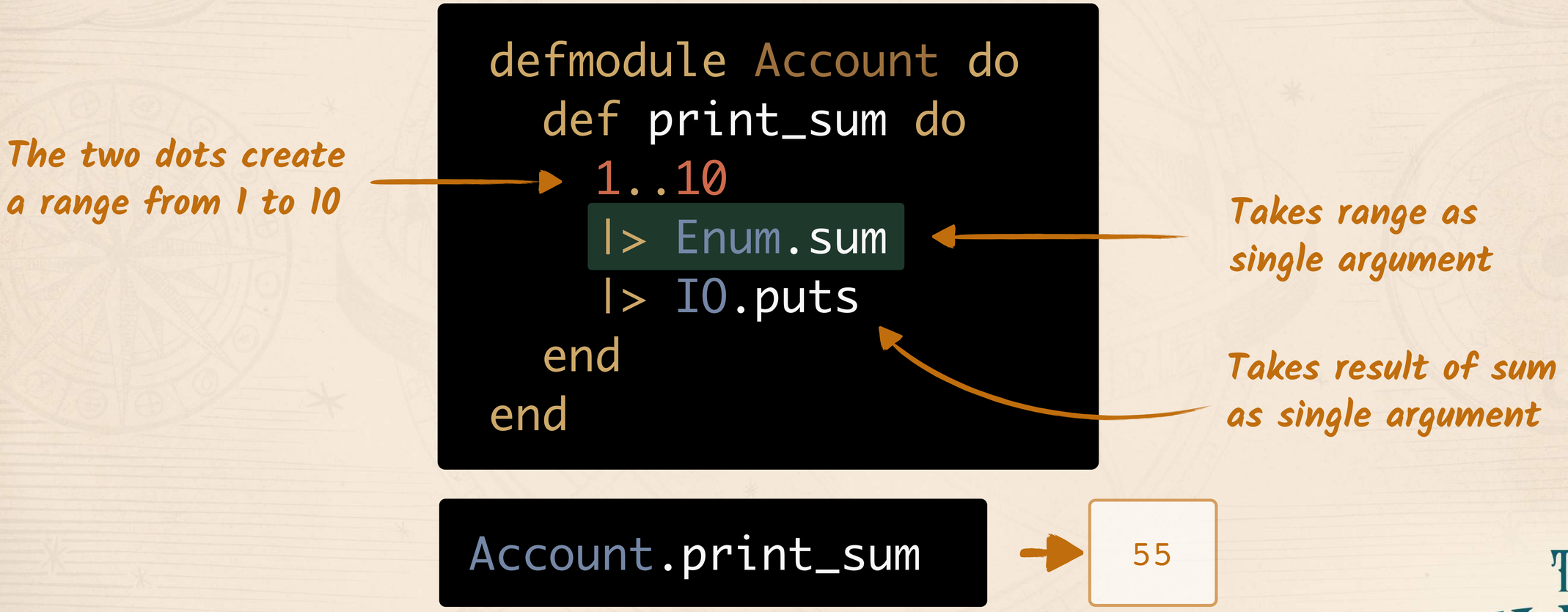
Easier to read as multiple lines

Can add more function calls in the future

TRY ELIXIR

Piping to Elixir Functions

The Enum.sum **function from Elixir's standard library returns the sum of all individual elements passed as argument.**



TRY ELIXIR