Level 4–2

Control Flow

The cond Statement

Transferring Between Accounts

We'll write a function to transfer money between accounts.

Transfer Depends on Validation

The **validation** for a transfer involves the amount transferred and the hour of the day.

```
defmodule Account do
  def transfer_amount(from_account, to_account, amount) do
    hourOfDay = DateTime.utc_now.hour

    if !valid_transfer?(amount, hourOfDay) do
      {:error, "Invalid Transfer"}
    else
      perform_transfer(from_account, to_account, amount)
    end
  end
  ...
end
```

Part of Elixir's standard library

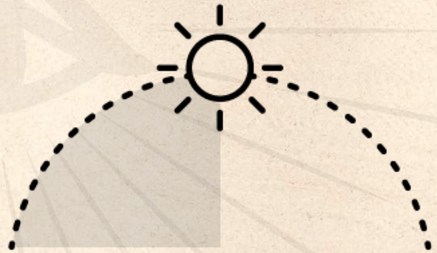
Defined elsewhere in this module

The Logic for the valid_transfer? **Function**

The amount allowed to be transferred depends on the time of the day.

Morning ***(before noon)***
No more than $5000

Afternoon ***(before 6pm)***
No more than $1000



Evening ***(after 6pm)***
No more than $300

* MIXING IT UP *
with
ELIXIR

And the Nested if Statements Attack Again!

We could implement this using nested if statements... but we've been there before, remember?

```
...
def valid_transfer?(amount, hourOfDay) do
  if hourOfDay < 12 do
    amount <= 5000
  else
    if hourOfDay < 18 do
      amount <= 1000
    else
      amount <= 300
    end
  end
end
...
```

Valid code, but hard to read and maintain!

The cond Statement

The cond statement checks multiple **conditions** and finds **the first one** that evaluates to *true*.

```
...
def valid_transfer?(amount, hourOfDay) do
  cond do
    hourOfDay < 12 -> amount <= 5000
    hourOfDay < 18 -> amount <= 1000
    true -> amount <= 300
  end
end
...
```

Block runs when condition is true

condition to be checked

Catch all when none of the previous conditions are true

Running the Transfer

The Account.transfer_amount **function is now complete!**

```
Account.transfer_amount(112233, 445566, 150.50)
```

→ `{:ok, 150.50}`

```
Account.transfer_amount(112233, 445566, 980)
```

→ `{:ok, 980}`

Can't transfer this much after 12pm

```
Account.transfer_amount(112233, 445566, 1500)
```

→ `{:error, "Invalid Transfer"}`

To case or to cond?

We use case for **matching** on multiple **patterns:**

```
case File.read(filename) do
  { :ok, content } -> "Content: #{content}"
  { :error, type } -> "Error: #{type}"
end
```

We use cond for **checking** multiple **conditions:**

```
cond do
  hourOfDay < 12 -> amount <= 5000
  hourOfDay < 18 -> amount <= 1000
  true -> amount <= 300
end
```