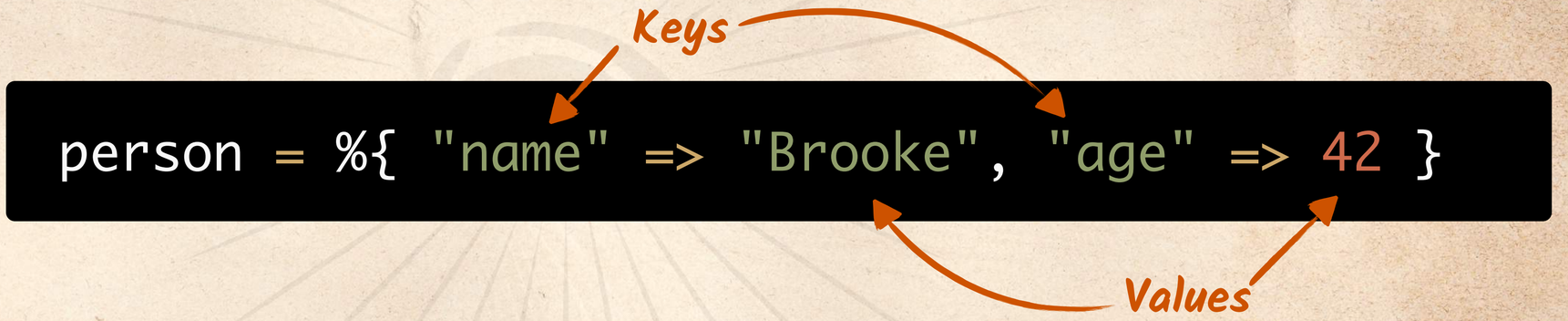
Level 3–3

Tuples & Maps

Maps

Using Maps for Structures With Named Fields

We use curly braces with the percent sign `%{}` to create maps, a collection of key-value pairs commonly used to represent a **structure with named fields.**



Reading Maps With Map.fetch and Map.fetch!

The Map module from Elixir's standard offers a set of functions for working with maps.

Map.fetch **returns a tuple when key is present**

```
Map.fetch(person, "name")
```

```
{:ok, "Brooke"}
```

...and the :error **atom when it's not.**

```
Map.fetch(person, "banana")
```

```
:error
```

Map.fetch! **returns a value when key is present**

```
Map.fetch!(person, "name")
```

```
"Brooke"
```

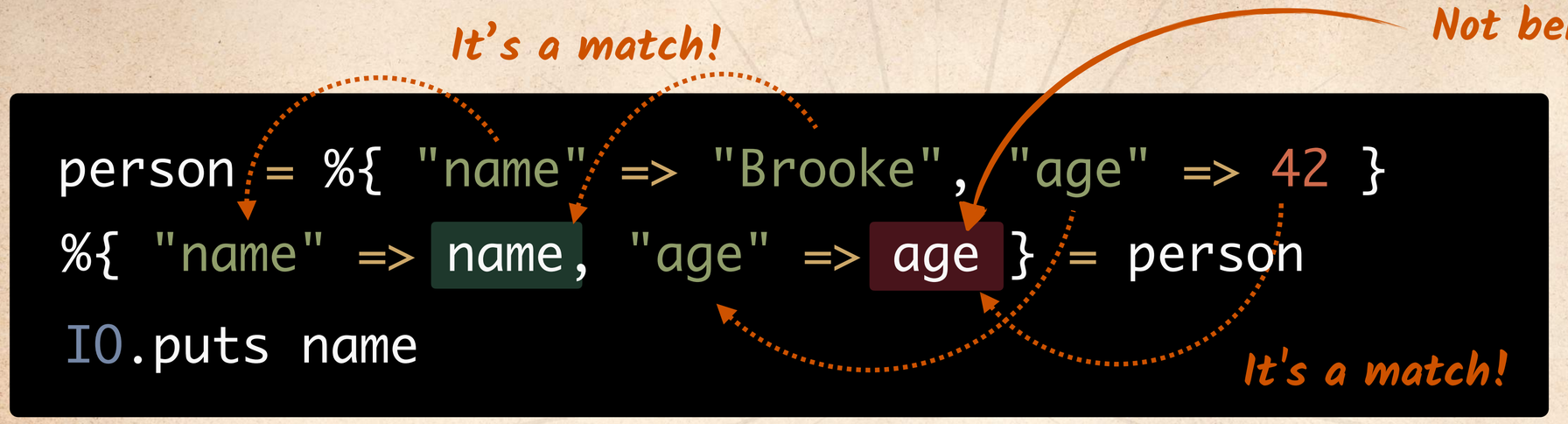
...and raises an error when it's not.

```
Map.fetch!(person, "banana")
```

```
** (KeyError) key "banana" not found in: %{"name" => "Brooke,
     "age" => 42}
    (elixir) lib/map.ex:164: Map.fetch!/2
```

Reading Maps With Pattern Matching

We can also use **pattern matching** to read values from a map.




```
warning: variable age is unused

Brooke
```

Warnings will NOT stop programs from running, but it's best not to have them.

Matching Portions of a Map

Unlike tuples, with maps we can pattern match only **the portion** we are interested in.

...other keys are ignored.

```
person = %{ "name" => "Brooke", "age" => 42 }
%{ "name" => name } = person
IO.puts name
```

Only reads the value for the name key on the map...

```
Brooke
```

```
person = [{:name, "Booke"}, {:age, 42}]
[{:name, name}] = person
IO.puts name
```

List of tuples do not support partial match

```
** (MatchError) no match of right hand
   side value: [name: "Booke", age: 42]
```

* MIXING IT UP *
with
ELIXIR

Advanced Pattern Matching With Maps

Even deeply nested keys in maps can be read using pattern matching.

Nested keys

```
person = %{ "name" => "Brooke",
            "address" => %{ "city" => "Orlando", "state" => "FL"}}

%{ "address" => %{ "state" => state }} = person

IO.puts "State: #{state}"
```

```
State: FL
```

Match on portion of the nested keys

Keyword Lists or Maps?

Here's a quick summary to help pick the appropriate data type.

When to use keyword lists?

```
Account.balance(transactions,
  currency: "dollar", symbol: "$")
```

To pass optional values to functions.

When to use maps?

```
person = %{ "name" => "Brooke", "age" => 42 }
%{ "name" => name } = person
```

To represent a structure as a key-value storage.