Vamos a ver cómo el operador = en Elixir es en realidad un operador de coincidencia y cómo usarlo para establecer patrones dentro de las estructuras de datos. También, aprenderemos sobre el operador de pin ^ usado para acceder a valores previamente vinculados.

## El operador match

Hemos usado el operador = un par de veces para asignar variables en Elixir:

```
iex> x = 1
1
iex> x
1
```

Pero en Elixir, el operador = en realidad se llama operador de coincidencia. Veamos por qué:

```
iex> x = 1
1
iex> 1 = x
1
iex> 2 = x
** (MatchError) no match of right hand side value: 1
```

Observe que 1 = x es una expresión válida y coincide porque los lados izquierdo y derecho son iguales a 1. Cuando los lados no coinciden, se genera un MatchError.

Una variable solo se puede asignar en el lado izquierdo del =.

```
iex> 1 = unknown
** (CompileError) iex:1: undefined function unknown/0
```

Como no hay una variable desconocida previamente definida, Elixir asumió que estaba intentando llamar a una función llamada unknown/0, pero esa función no existe.

# Pattern Matching

El operador de coincidencia no solo se utiliza para comparar valores simples, sino que también es útil para desestructurar tipos de datos más complejos. Por ejemplo, podemos combinar patrones en tuplas:

```
iex> {a, b, c} = {:hello, "world", 42}
{:hello, "world", 42}
iex> a
:hello
iex> b
"world"
```

Se producirá un error de coincidencia de patrón si los lados no pueden coincidir, por ejemplo, si las tuplas tienen tamaños diferentes:

```
iex> {a, b, c} = {:hello, "world"}
** (MatchError) no match of right hand side value: {:hello, "world"}
```

Y también al comparar diferentes tipos:

```
iex> {a, b, c} = [:hello, "world", 42]
** (MatchError) no match of right hand side value: [:hello, "world", 42]
```

Más interesante aún, podemos coincidir en valores específicos. El siguiente ejemplo afirma que el lado izquierdo solo coincidirá con el lado derecho cuando el lado derecho es una tupla que comienza con el átomo :ok:

```
iex> {:ok, result} = {:ok, 13}
{:ok, 13}
iex> result
13
```

```
iex> {:ok, result} = {:error, :oops}
** (MatchError) no match of right hand side value: {:error, :oops}
```

Podemos emparejar patrones en listas:

```
iex> [a, b, c] = [1, 2, 3]
[1, 2, 3]
iex> a
1
```

Una lista también admite coincidencias en su propia cabeza y cola:

```
iex> [head | tail] = [1, 2, 3]
[1, 2, 3]
iex> head
1
iex> tail
[2, 3]
```

Similar a las funciones hd/1 y tl/1, no podemos hacer coincidir una lista vacía con un patrón de cabeza y cola:

```
iex> [head | tail] = []
** (MatchError) no match of right hand side value: []
```

La [cabeza | el formato tail] no solo se usa en la coincidencia de patrones, sino también para anteponer elementos a una lista:

```
iex> list = [1, 2, 3]
[1, 2, 3]
iex> [0 | list]
[0, 1, 2, 3]
```

La coincidencia de patrones permite a los desarrolladores desestructurar fácilmente los tipos de datos, como tuplas y listas.

# El operador pin

Utilice el operador de pin ^ cuando desee comparar patrones con el valor de una variable existente en lugar de volver a vincular la variable:

```
iex> x = 1
1
iex> ^x = 2
** (MatchError) no match of right hand side value: 2
iex> {y, ^x} = {2, 1}
{2, 1}
iex> y
2
iex> {y, ^x} = {2, 2}
** (MatchError) no match of right hand side value: {2, 2}
```

Como hemos asignado el valor de 1 a la variable x, este último ejemplo también podría haberse escrito como:

```
iex> {y, 1} = {2, 2}
** (MatchError) no match of right hand side value: {2, 2}
```

Si una variable se menciona más de una vez en un patrón, todas las referencias deben unirse al mismo patrón:

```
iex> {x, x} = {1, 1}
{1, 1}
iex> {x, x} = {1, 2}
** (MatchError) no match of right hand side value: {1, 2}
```

En algunos casos, no le importa un valor particular en un patrón. Es una práctica común vincular esos valores al guión bajo, _. Por ejemplo, si solo nos importa el encabezado de la lista, podemos asignar la cola para subrayar:

```
iex> [head | _] = [1, 2, 3]
[1, 2, 3]
iex> head
1
```

La variable _ es especial porque nunca se puede leer. Si intenta leer de él da un error de compilación:

```
iex> _
** (CompileError) iex:1: invalid use of _. "_" represents a value to be ignored in a pattern and cannot be
used in expressions
```

Aunque la coincidencia de patrones nos permite construir construcciones potentes, su uso es limitado. Por ejemplo, no puede realizar llamadas a funciones en el lado izquierdo de una coincidencia. El siguiente ejemplo no es válido:

```
iex> length([1, [2], 3]) = 3
** (CompileError) iex:1: cannot invoke remote function :erlang.length/1 inside match
```