Notas de Elixir

Entrar al Shell de elixir con interactive elixir (iex)

C:\Users\HP>iex

←----------→

Las funciones de elixir siempre deben retornar algo.

Se pueden escribir múltiples expresiones, retornando siempre el último valor calculado.

iex(1)> 5 + 4; 5 + 1 → Retorna → 9

No significa que no se hayan calculado el valor, significa que el valor de sus resultados no fue guardado o retornado.

←----------→

Utilizar valores calculados anteriores en el shell

iex(1)> 5 + 4 → Retorna → 9

iex(2)> v1 → Retorna → 9 → Sin paréntesis

iex(3)> v(1) → Retorna → 9 → Con paréntesis

iex(4)> v(1) + 11 → Retorna → 20 → Operaciones con diferentes líneas de código

iex(5)> v(1) → Retorna → 20

←----------→

Para salir del Shell se puede mediante CTRL+C o escribiendo System.halt

←----------→

Variables

Elixir es un lenguaje de programación de tipado dinámico.

- **NO es necesario declarar de manera explícita una variable o su tipo de dato**
- **El tipo de dato se determina de acuerdo al valor contenido**
- **La asignación se conoce como fijación (binding)**
- **Cuando se inicializa una variable con un valor, la variable se fija con ese valor.**
- **Cuando pones un string este se fija como lista, porque los strings no existen como tal en erlang y elixir**

←----------→

Características de las variables.

- **El nombre de una variable siempre se inicia con una letra minúscula.**

- El guion bajo se usa para separar palabras (_).
- El signo de interrogación "?" se recomienda usar para variables booleanas.
- El signo de admiración "¡" se recomienda usar para variables warnings.
- La convención es usar solo letras, dígitos y subrayados.
- No es recomendable usar el formato camelCase.

←----------→

Inmutabilidad

– Los datos en Elixir son inmutables: su contenido no puede cambiarse.

– Las variables pueden ser refijadas (rebound) a un diferente valor

iex(1)> dia_semana = 5 → Retorna → 5 → Se fija el valor inicial

iex(2)> dia_semana → Retorna → 5 → Se verifica

iex(3)> dia_semana = 7 → Retorna → 7 → Se refija el valor inicial

iex(4)> dia_semana → Retorna → 7 → Se verifica el efecto de la refijacion

Cuando se refija una variable inmutable lo que hace es que en la memoria crea un nuevo valor para esa variable, el valor anterior se elimina. Cuando es mutable el valor que tiene una variable se modifica pero no se borra nada de la memoria.

←----------→

Módulos y Funciones

Módulos

Un archivo puede tener varios módulos.

Un módulo consta de varias funciones.

Cada función debe estar definida dentro de un módulo.

El modulo IO permite varias operaciones de (I/O)[Input/output]. La función puts permite imprimir un mensaje en la pantalla. Retorna un ":ok".

La sintaxis general es:

– NombreModulo.nombre_funcion(args). El nombre del módulo siempre se escribe en formato CamelCase iniciando con una letra mayúscula.

Un módulo puede estar dentro de un archivo. Un archivo puede contener varios módulos.

Se utiliza el constructor defmodule para la creación de los módulos

Dentro del módulo con el constructor def se crean las funciones.

Funciones

Una función siempre debe estar dentro de un módulo

Los nombres de funciones son igual que las variables:

- **El nombre de una función siempre se inicia con una letra minúscula.**
- **El guion bajo se usa para separar palabras (_).**
- **El signo de interrogación "?" se recomienda usar para funciones booleanas.**
- **El signo de admiración "¡" se recomienda usar para funciones warnings.**
- **La convención es usar solo letras, dígitos y subrayados.**
- **No es recomendable usar el formato camelCase.**

Tanto defmodule como def NO son palabras reservadas del lenguaje, son macros.

Función sin argumentos (Archivo 2-Helloworld).

Función con argumentos (Archivo 3-Area-Calculadora).

Las funciones pueden expresarse de manera condensada y anidada (Archivo 4).

Invocaciones internas de una función (Archivo 5).

Se pueden utilizar funciones privadas con el constructor defp y publicas con el constructor def (Archivo 6).

←----------→

Aridad (Arity) de funciones.

Es el nombre para el número de argumentos que una función recibe.

Una función se identifica por:

- **El módulo donde se encuentra.**
- **Su nombre.**
- **Su aridad (arity).**

Polimorfismo (sobrecarga)

Dos funciones con el mismo nombre pero con diferente aridad son dos diferentes funciones.

Ejemplos (Archivo 8).

←----------→

Argumentos por defecto

Se pueden especificar argumentos por defecto mediante el operador

Se puede utilizar cualquier combinación de argumentos por defecto

Ejemplo (archivo 9).

←----------→

Atributos de módulo

Existen los atributos en tiempo de compilación (Mientras están cargados).

Elixir permite el registro de atributos, que se almacenarán en el archivo binario.

- **@moduledoc**
- **@doc**

Sirven para documentar módulos y funciones.

Notas de Elixir

Shell de Elixir

-Cuando nos equivocamos en una expresión y no permite continuar el shell

-para salir del Shell se puede mediante CTRL+C o escribiendo System.halt

-Para pedir ayuda del Shell teclea "h"

Tipos de datos

• Elixir utiliza el mismo sistema de tipos de Erlang

Numeros

• Los números (numbers) pueden ser enteros o flotantes

Atoms

• Constantes literales nombradas

• es una constante cuyo nombre es su propio valor

• inician con : (dos puntos)

• seguidos de caracteres alfanuméricos y/o subrayados

• se pueden usar espacios en blanco si se ponen entre comillas

Nil

• similar al null de otros lenguajes

• Los átomos nil y false son tratados como valores falsos, mientras que todo lo demás es tratado como un valor de verdad.

• Esta propiedad es útil con los operadores de cortocircuito:

– || -> retorna la primera expresión verdadera

– && -> retorna la segunda siempre y cuando la primera lo sea también

– ! -> retorna la negación de la expresión sin importar el tipo de dato

• || -> retorna la primera expresión verdadera

Tuplas

• son como estructuras o registros

• permiten agrupar elementos fijos

Listas

• Manejo dinámico de datos

• Funcionan como listas enlazadas simples

Variables

• Elixir es un lenguaje de programación dinámico

– NO es necesario declarar de manera explícita una variable

– El tipo de dato se determina de acuerdo al valor contenido

– La asignación(=) se conoce como fijación (binding)

– Cuando se inicializa una variable con un valor, la variable se fija con ese valor.

°caracteristicas de las variables

-El nombre de una variable siempre inicia con un caracter alfabético en minúscula o caracter de subrayado (_)

– La convención es usar solo letras, dígitos y subrayados

– Pueden terminar con los carateres ? o !

– Por convención el ? se utiliza cuando la función retorna true o false

– El ! se utiliza generalmente en funciones que podrían provocar algún error en tiempo de ejecución

***ejemplos:**

***variable_valida**

***esta_variable_tambien_es_valida**

***esta_tambien_1**

***estaEsValidaPeroNoRecomendada**

***No_es_valida**

***nombre_valido?**

***claro_que_si!**

– Los datos en Elixir son inmutables: su contenido no puede cambiarse.

– Las variables pueden ser refijadas (rebound) a un diferente valor

Modulos y Funciones

• Un módulo consta de varias funciones

• Cada función debe estar definida dentro de un módulo

• El módulo IO permite varias operaciones de E/S (I/O),

• la función puts permite imprimir un mensaje en pantalla

• Se utiliza el constructor defmodule para la creación de los módulos

• Dentro del módulo con el constructor def se crean las funciones.

• Una función siempre debe estar dentro de un módulo

• Los nombres de funciones son igual que las variables

• Tanto defmodule como def NO son palabras reservadas del lenguaje, son macros

• Un módulo puede estar dentro de un archivo. Un archivo puede contener varios módulos.

• Reglas de los módulos

– Inicia con una letra mayúscula

– Puede consistir en caracteres alfanuméricos, subrayados y puntos (.).

Regularmente se usa para la organización jerárquica de los módulos.

• Se pueden utilizar funciones privadas con el constructor defp

Aridad (Arity) de funciones

• Es el nombre para el número de argumentos que una función recibe

• Una función se identifica por:

1. el módulo donde se encuentra
2. su nombre
3. su aridad (arity)

Polimorfismo (sobrecarga)

• Dos funciones con el mismo nombre pero con diferente aridad son dos diferentes funciones.

```
defmodule Rectangulo do
 def area(l) do
 l * l
 end
 def area(l1,l2) do
 l1 * l2
 end
end
```

Argumentos por defecto

• Se pueden especificar argumentos por defecto mediante el operador \\

```
defmodule Calculadora do
 def suma(n1,n2 \\ 0) do
 n1 + n2
 end
end
```

Alias

- Se puede utilizar alias como alternativa a import, permite hacer una referencia a un módulo con otro nombre

- Elixir permite el registro de atributos, que se almacenarán en el archivo binario.
  - @moduledoc
  - @doc
- Sirven para documentar módulos y funciones