PRADITA University

IF140303-Web Application Development

Session-02: Elixir's Pattern Matching

PRU/SPMI/FR-BM-18/0222

Alfa Yohannis

Recap: Function Documentation and Arity

- In Elixir, the function name followed by a slash and number indicates the arity, or the number of arguments the function takes.
- `function name/number` notation helps in distinguishing between different functions with the same name but different arities.
- Example: `greet/0`, `generate_pool/0`, `contains?/2`, `distribute/2`.

Introduction to Advanced Concepts

PRADITA University

- This session covers more advanced Elixir concepts within the context of a Lottery module.
- We'll explore how to save and load data, and how to create more complex operations like creating a randomized hand from the lottery pool.

Saving the Lottery Pool to a File

PRADITA University

```
1       def save_pool(pool, filename) do
2       binary = :erlang.term_to_binary(pool)
3       File.write(filename, binary)
4       end
```

- `save_pool/2` saves the lottery pool to a specified file.
- Converts the pool to binary using `:erlang.term_to_binary/1`.
- Writes the binary data to the file with `File.write/2`.

Loading the Lottery Pool from a File

PRADITA University

```
1      def load_pool(filename) do
2      case File.read(filename) do
3      {:ok, binary} -> :erlang.binary_to_term(binary)
4      {:error, _reason} -> "That file does not exist"
5      end
6      end
```

- `load_pool/1` loads the lottery pool from a specified file.
- Reads the binary data from the file using `File.read/1`.
- Converts the binary data back into the original list with `:erlang.binary_to_term/1`.

Creating a Randomized Hand

PRADITA University

```elixir
def create_hand(draw_size) do
Lottery.generate_pool()
|> Lottery.randomize()
|> Lottery.distribute(draw_size)
end
```

- `create_hand/1` generates a randomized hand from the lottery pool.
- Combines multiple operations: generating the pool, shuffling, and distributing.
- Utilizes the pipe operator (`|>`) for chaining functions together.

The Pipe Operator (|>)

PRADITA University

- The pipe operator allows for chaining function calls in a clear and readable manner.
- It passes the result of the expression on the left as the first argument to the function on the right.
- Enhances code readability, especially when working with multiple transformations or operations.

Recap: Functional Programming Concepts

- Functional programming emphasizes immutability, pure functions, and higher-order functions.
- Functions can be passed as arguments, returned as results, and assigned to variables.
- Recursion is preferred over loops for iteration.

Summary

PRADITA University

- We explored advanced Elixir concepts by continuing the development of the `Lottery` module.
- We learned how to save and load data, create a randomized hand, and utilize the pipe operator.
- These concepts further enhance our understanding of functional programming and Elixir's capabilities.