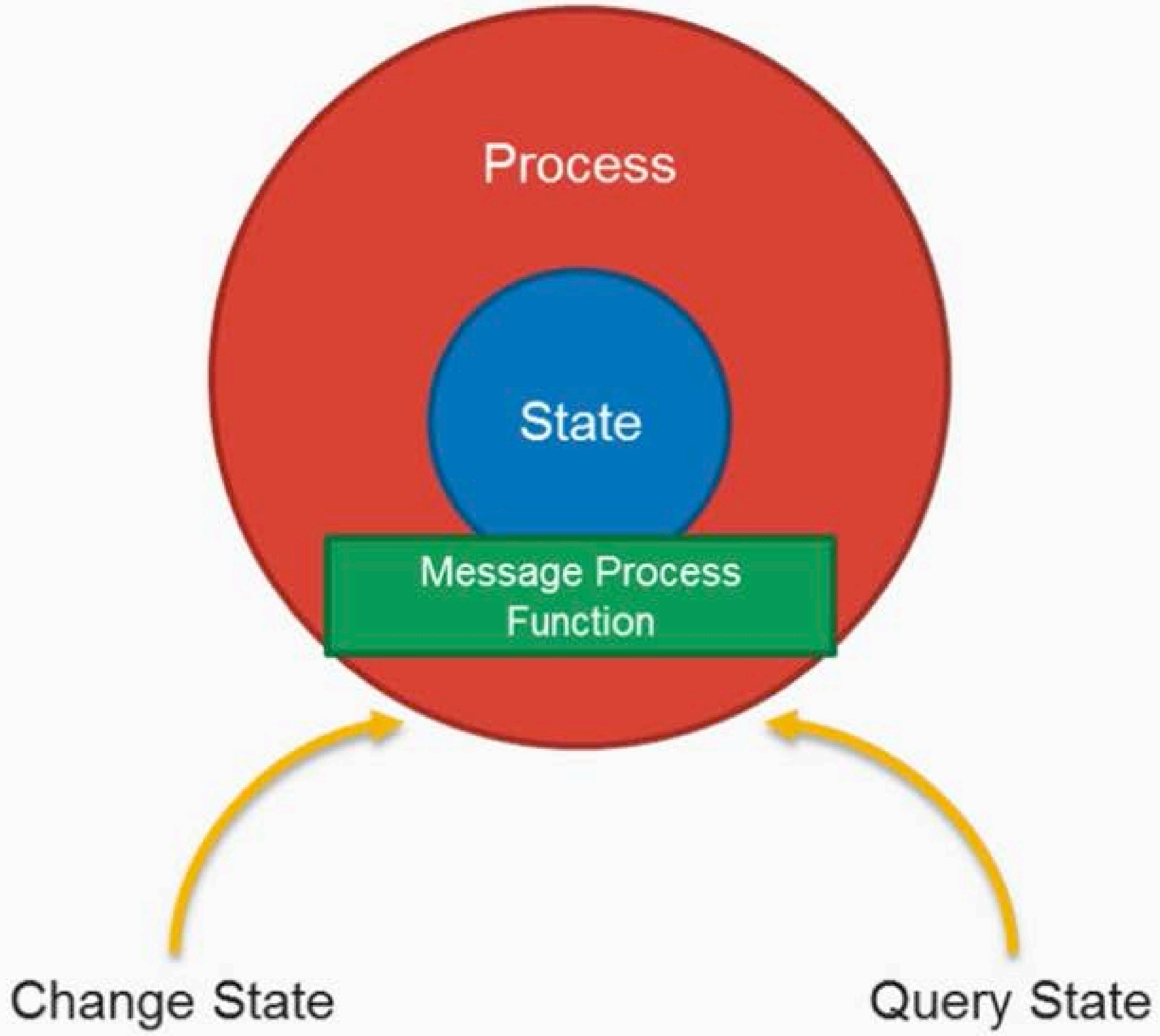
OTP Feature Set — GenServer

In this Video, we are going to take a look at…

- Generic server abstraction
- Implementing a GenServer
- Using a GenServer

GenServer

The OTP behavior for building generic server processes.



GenServer

```elixir
defmodule Counter do
  use GenServer

  def start_link() do
    GenServer.start_link(__MODULE__, 0)
  end

  def inc(server, x) do
    GenServer.cast(server, {:inc, x})
  end

  def dec(server, x) do
    GenServer.cast(server, {:dec, x})
  end

  def get(server) do
    GenServer.call(server, {:get})
  end

  def handle_cast({:inc, x}, counter) do
    {:noreply, counter + x}
  end

  def handle_cast({:dec, x}, counter) do
    {:noreply, counter - x}
  end

  def handle_call({:get}, _from, counter) do
    {:reply, counter, counter}
  end
end
```

GenServer

```elixir
defmodule Counter do
  use GenServer

  def start_link() do
    GenServer.start_link(__MODULE__, 0)
  end

  def inc(server, x) do
    GenServer.cast(server, {:inc, x})
  end

  def dec(server, x) do
    GenServer.cast(server, {:dec, x})
  end

  def get(server) do
    GenServer.call(server, {:get})
  end

  def handle_cast({:inc, x}, counter) do
    {:noreply, counter + x}
  end

  def handle_cast({:dec, x}, counter) do
    {:noreply, counter - x}
  end

  def handle_call({:get}, _from, counter) do
    {:reply, counter, counter}
  end
end
```

Public API

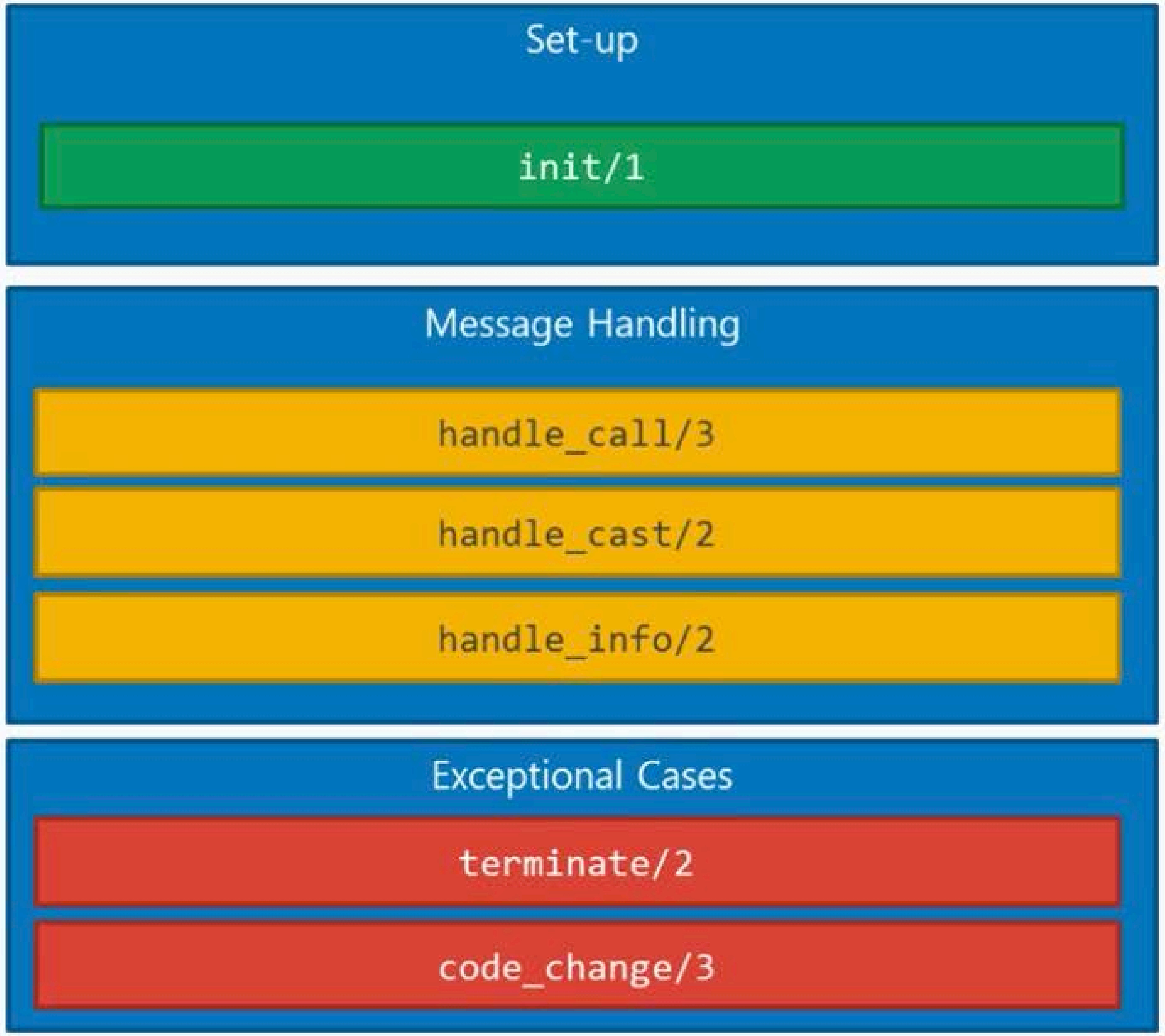
Hide the implementation (GenServer) from the calling parties

Callbacks

Implement the GenServer behavior and business logic

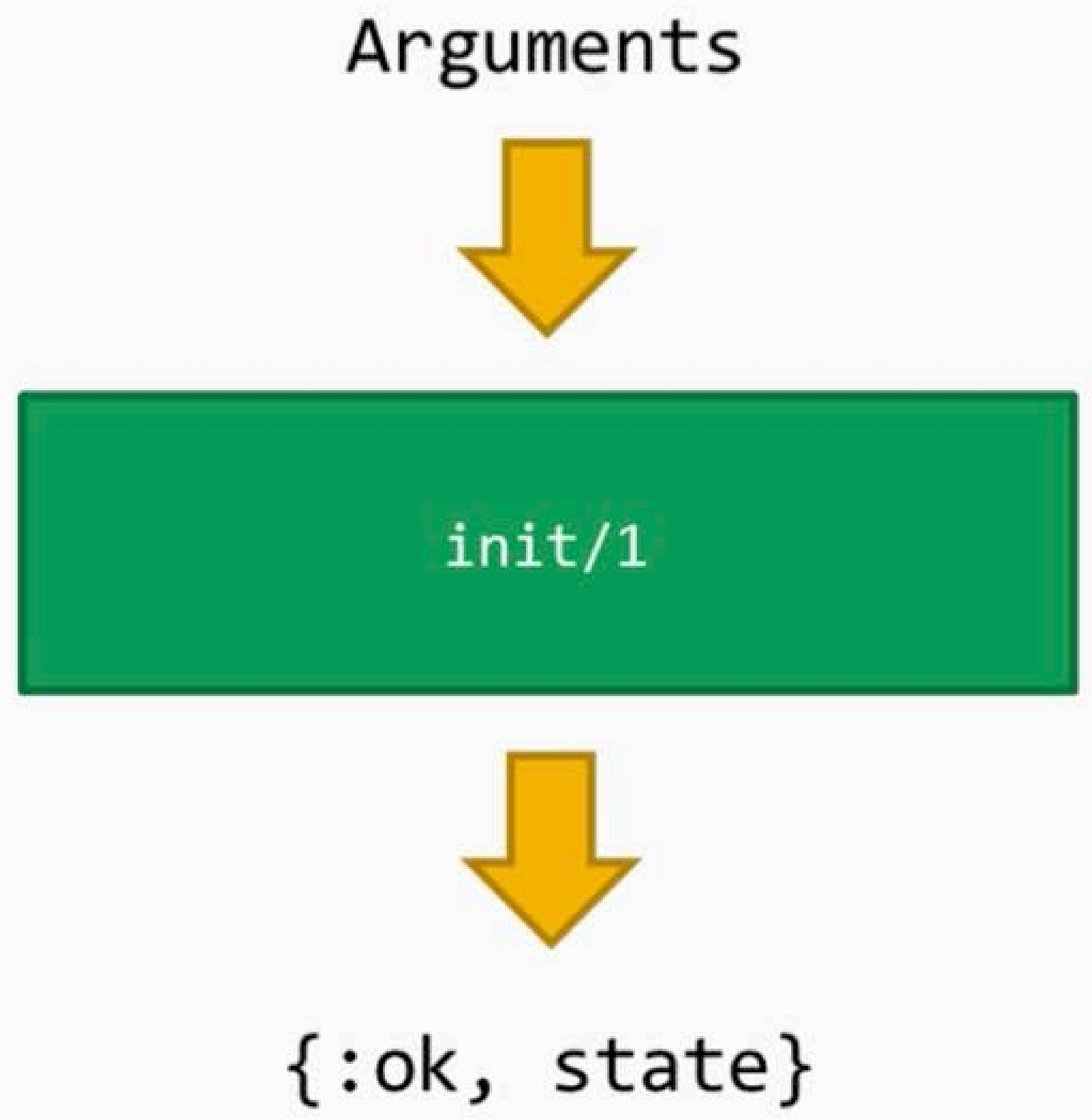
Callbacks

Implement the behavior of the GenServer and respond to events.



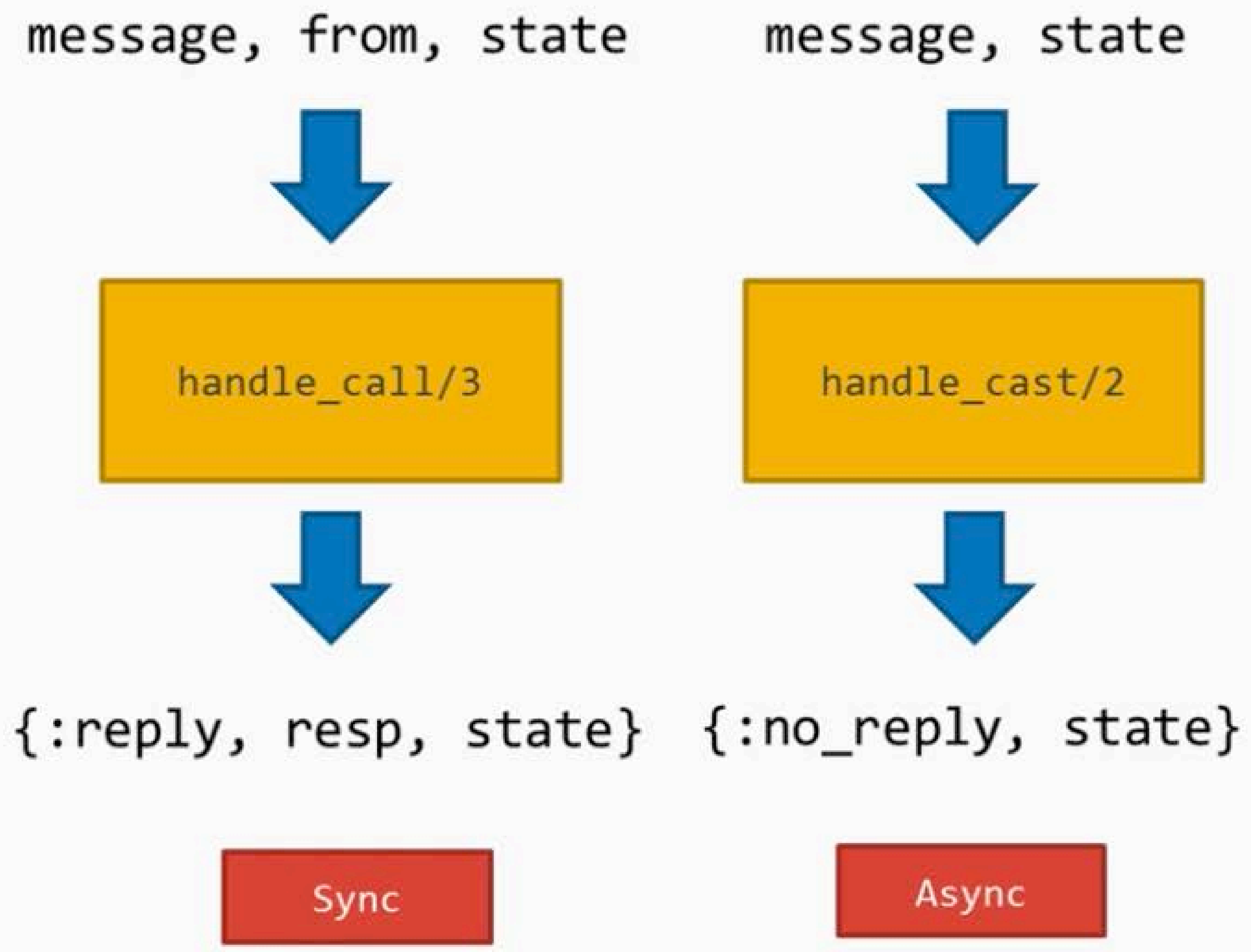
Initialization

Sets up the state for the GenServer.



Message Handling

Process messages coming to the process and handle replies.

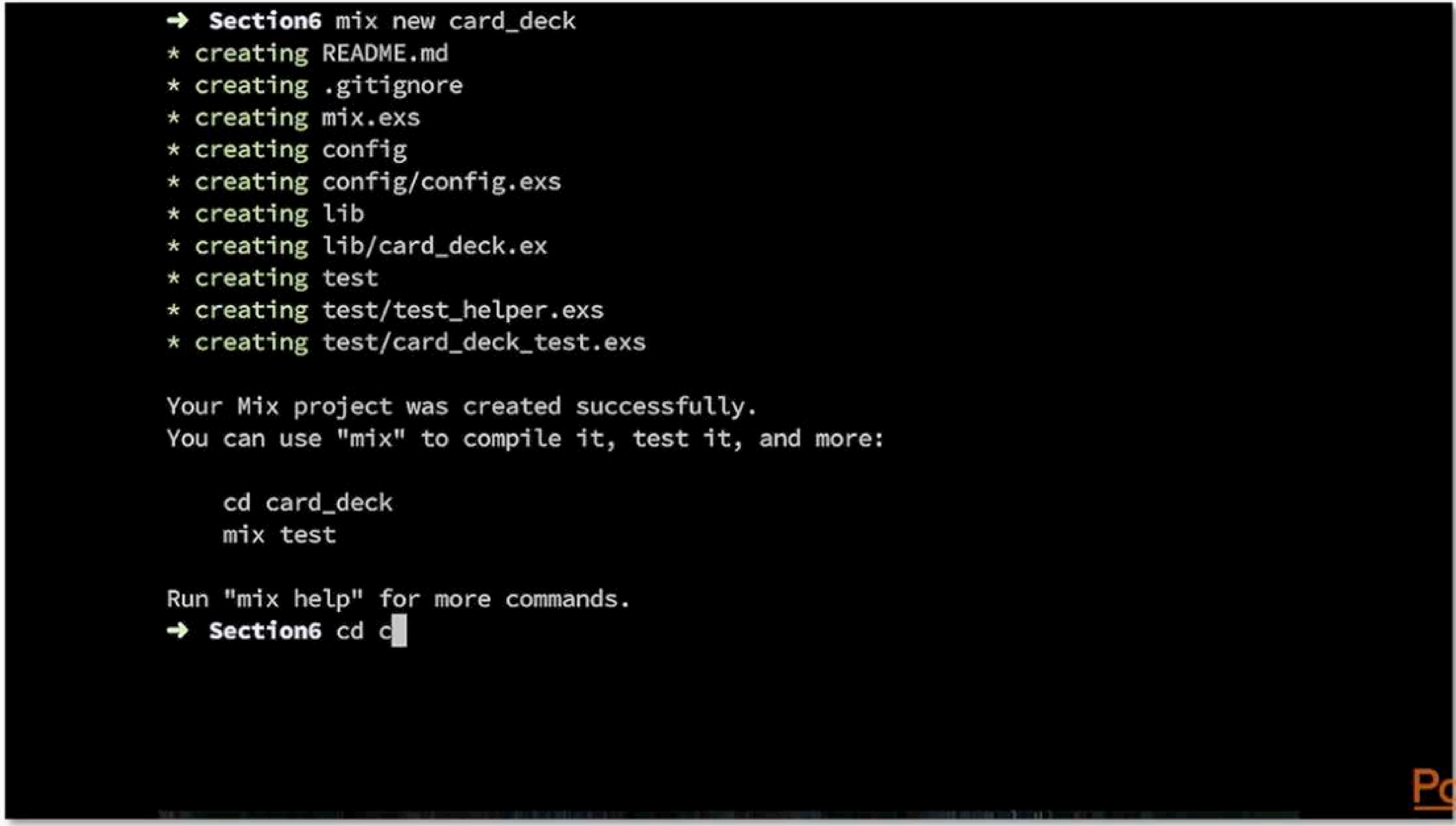


Deck of Cards

52 unique cards

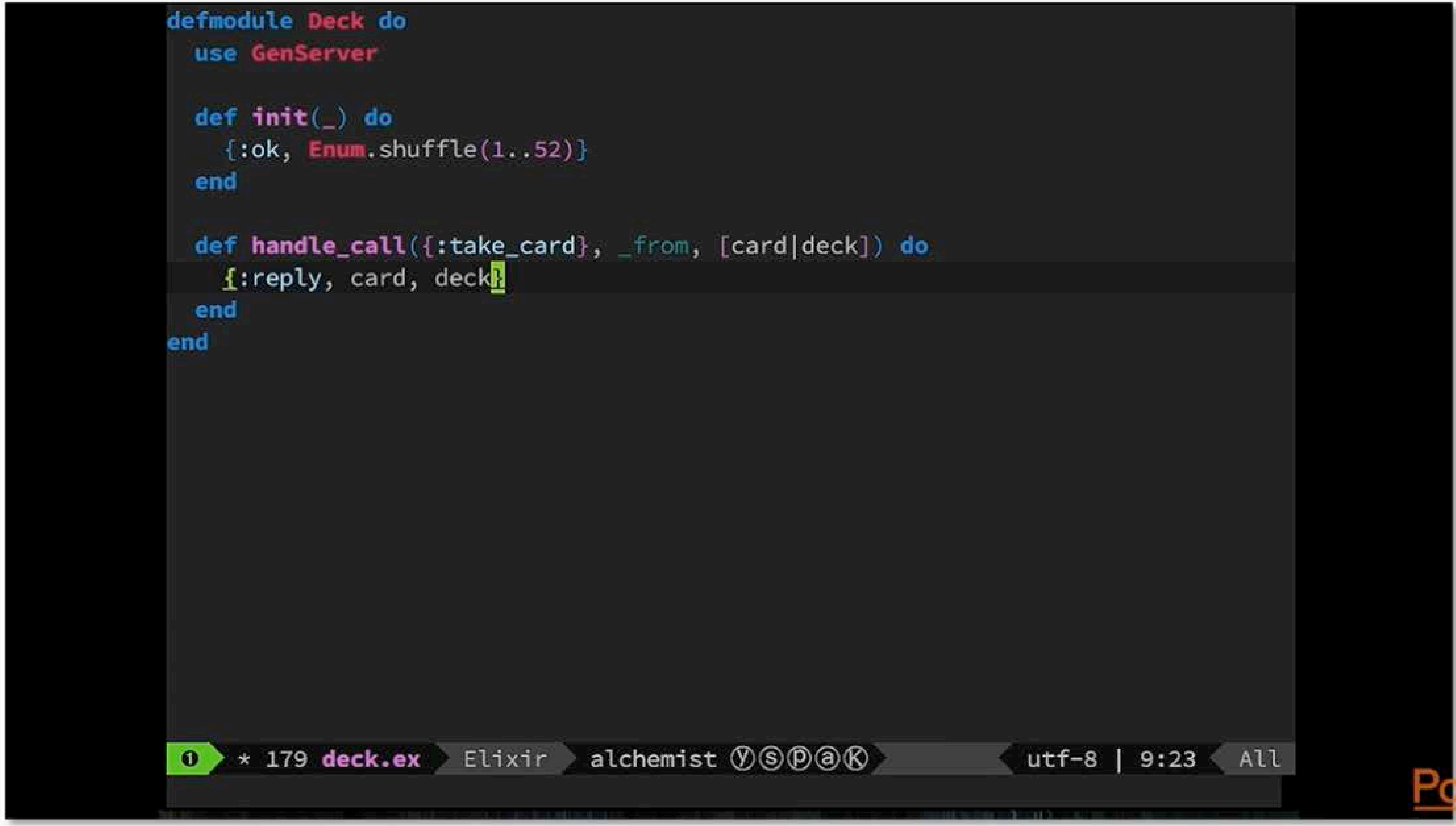
Take one card at a time

Deck of Cards



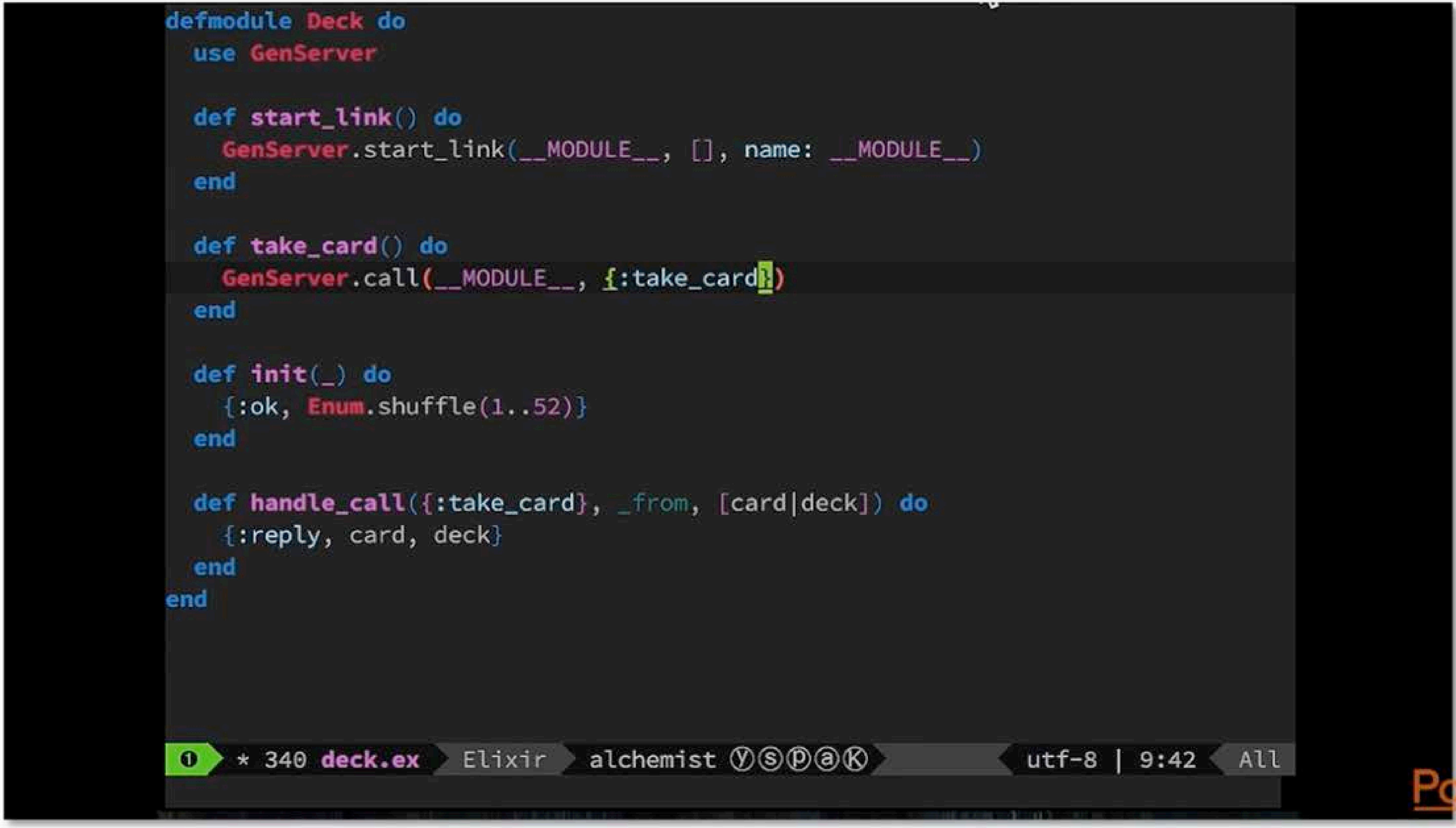
```
➜  Section6 mix new card_deck
* creating README.md
* creating .gitignore
* creating mix.exs
* creating config
* creating config/config.exs
* creating lib
* creating lib/card_deck.ex
* creating test
* creating test/test_helper.exs
* creating test/card_deck_test.exs

Your Mix project was created successfully.
You can use "mix" to compile it, test it, and more:

    cd card_deck
    mix test

Run "mix help" for more commands.
➜  Section6 cd c
```

Deck of Cards



```
defmodule Deck do
  use GenServer

  def init(_) do
    {:ok, Enum.shuffle(1..52)}
  end

  def handle_call({:take_card}, _from, [card|deck]) do
    {:reply, card, deck}
  end
end
```

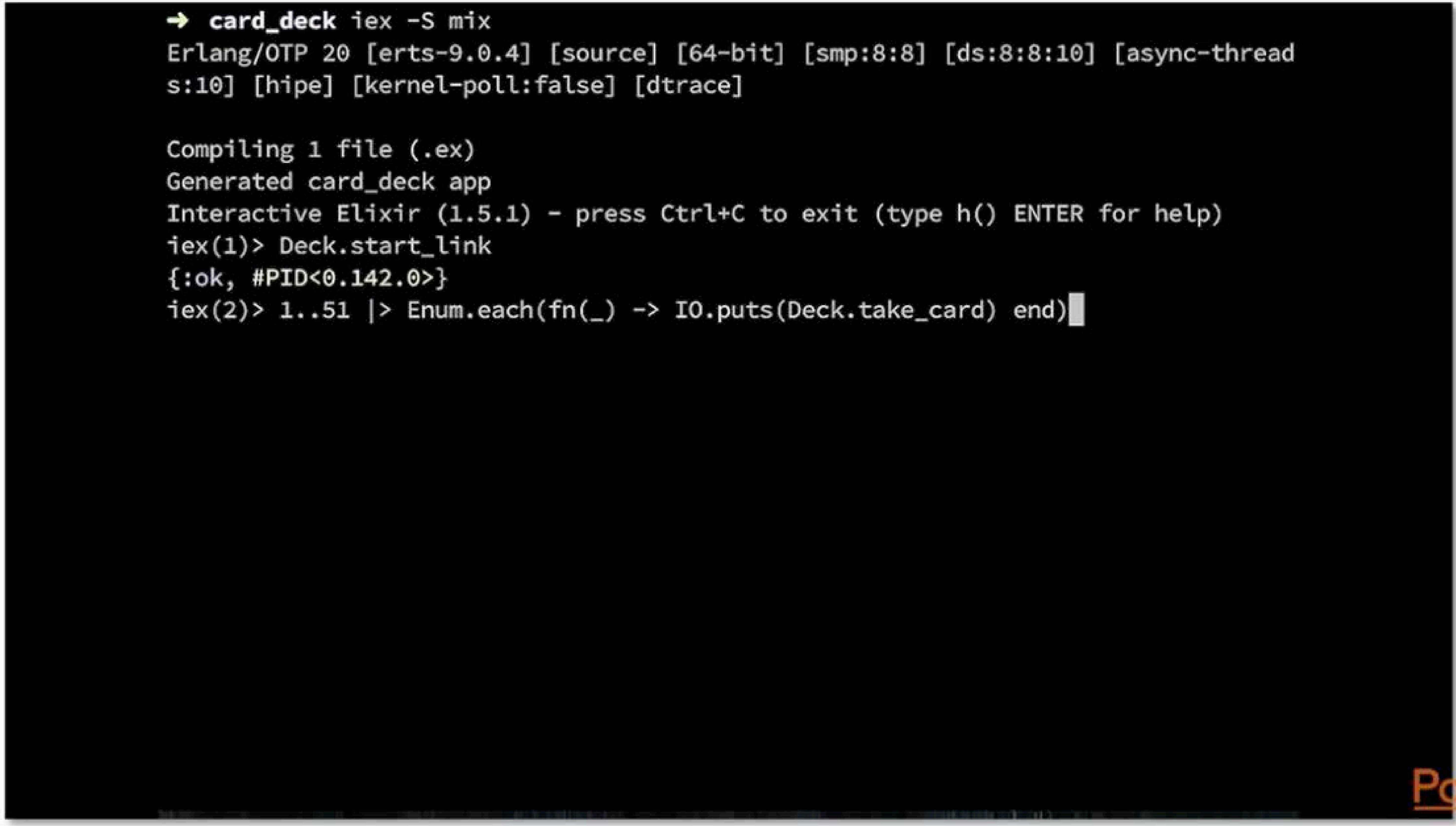
Deck of Cards



```elixir
defmodule Deck do
  use GenServer

  def start_link() do
    GenServer.start_link(__MODULE__, [], name: __MODULE__)
  end

  def take_card() do
    GenServer.call(__MODULE__, {:take_card})
  end

  def init(_) do
    {:ok, Enum.shuffle(1..52)}
  end

  def handle_call({:take_card}, _from, [card|deck]) do
    {:reply, card, deck}
  end
end
```

```
❶  * 340 deck.ex   Elixir   alchemist ⓨⓢⓟⓐⓚ        utf-8 | 9:42   All
```

Deck of Cards

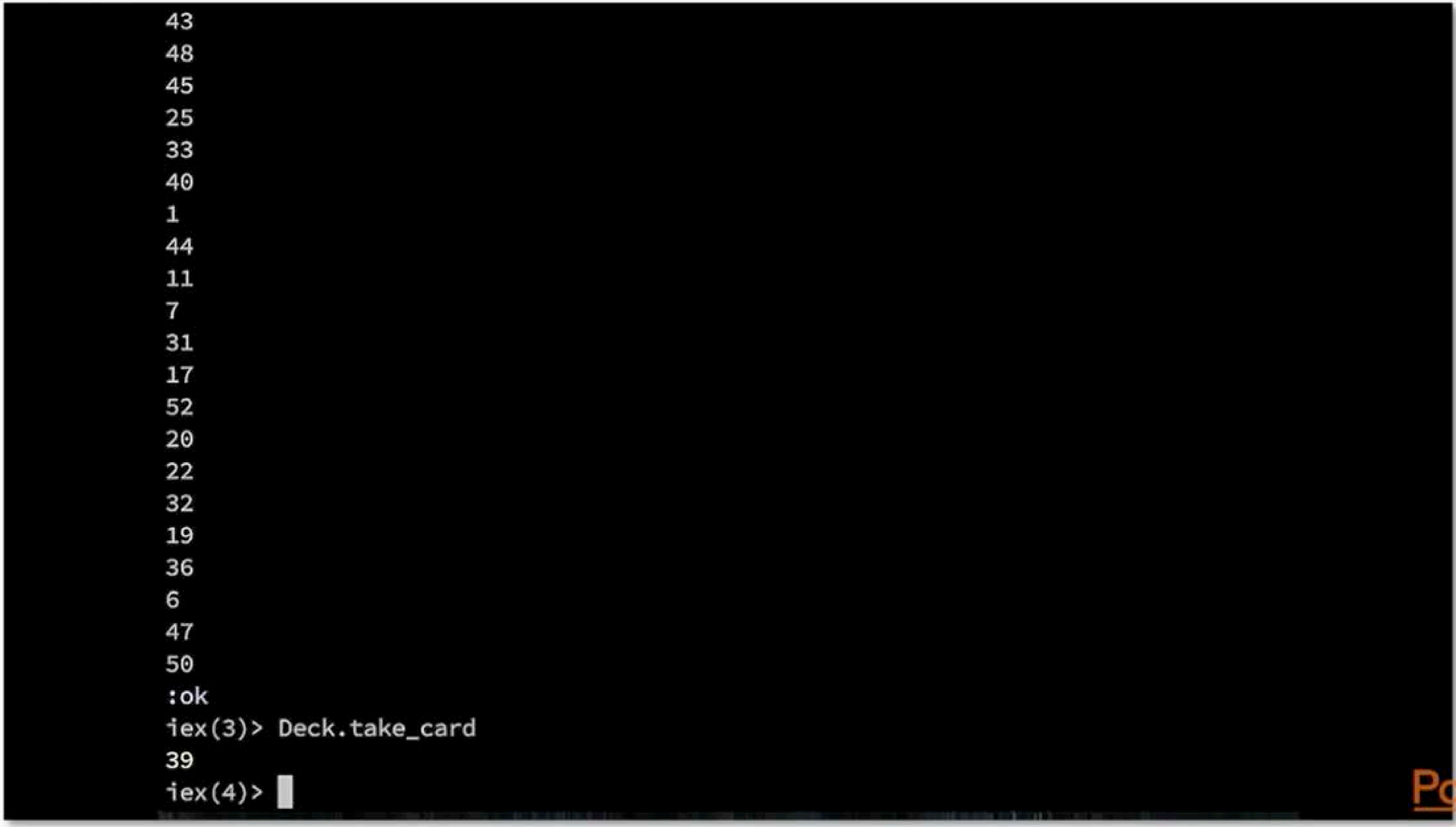


```
➜ card_deck iex -S mix
Erlang/OTP 20 [erts-9.0.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-thread
s:10] [hipe] [kernel-poll:false] [dtrace]

Compiling 1 file (.ex)
Generated card_deck app
Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Deck.start_link
{:ok, #PID<0.142.0>}
iex(2)> 1..51 |> Enum.each(fn(_) -> IO.puts(Deck.take_card) end)
```

Deck of Cards

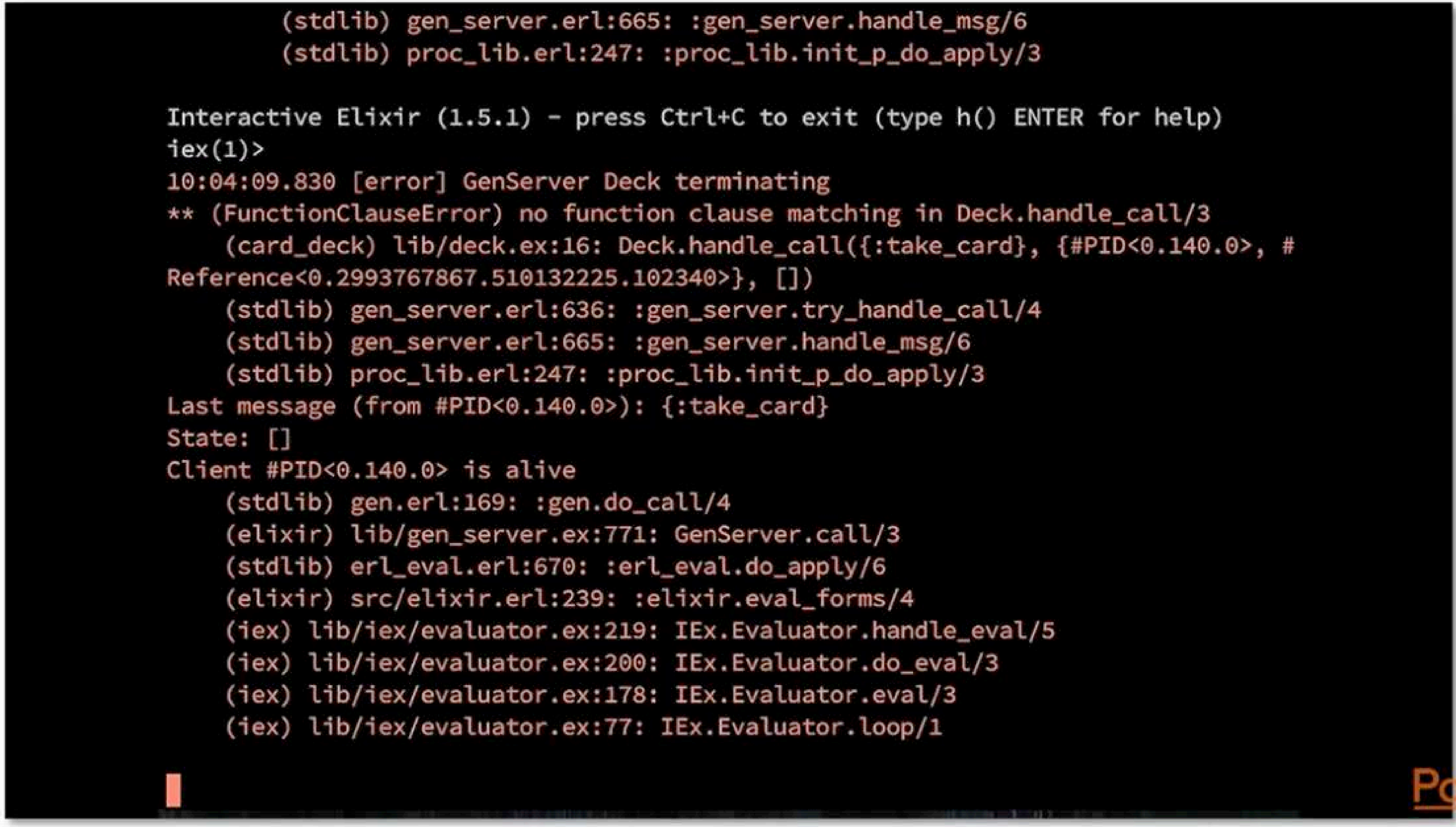
```
27
41
43
48
45
25
33
40
1
44
11
7
31
17
52
20
22
32
19
36
6
47
50
:ok
iex(3)> 
```

Deck of Cards



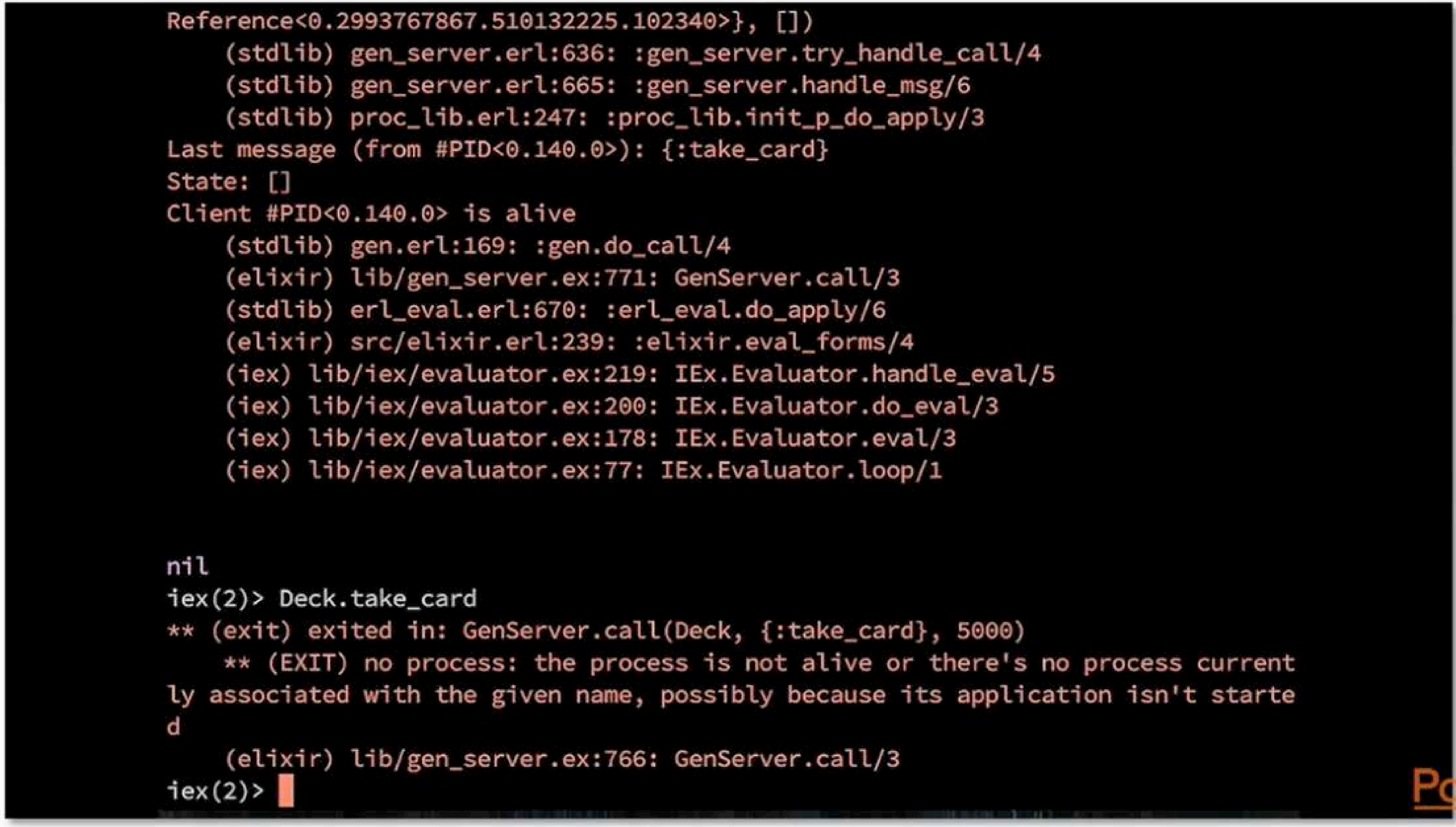
```
43
48
45
25
33
40
1
44
11
7
31
17
52
20
22
32
19
36
6
47
50
:ok
iex(3)> Deck.take_card
39
iex(4)> 
```

Deck of Cards



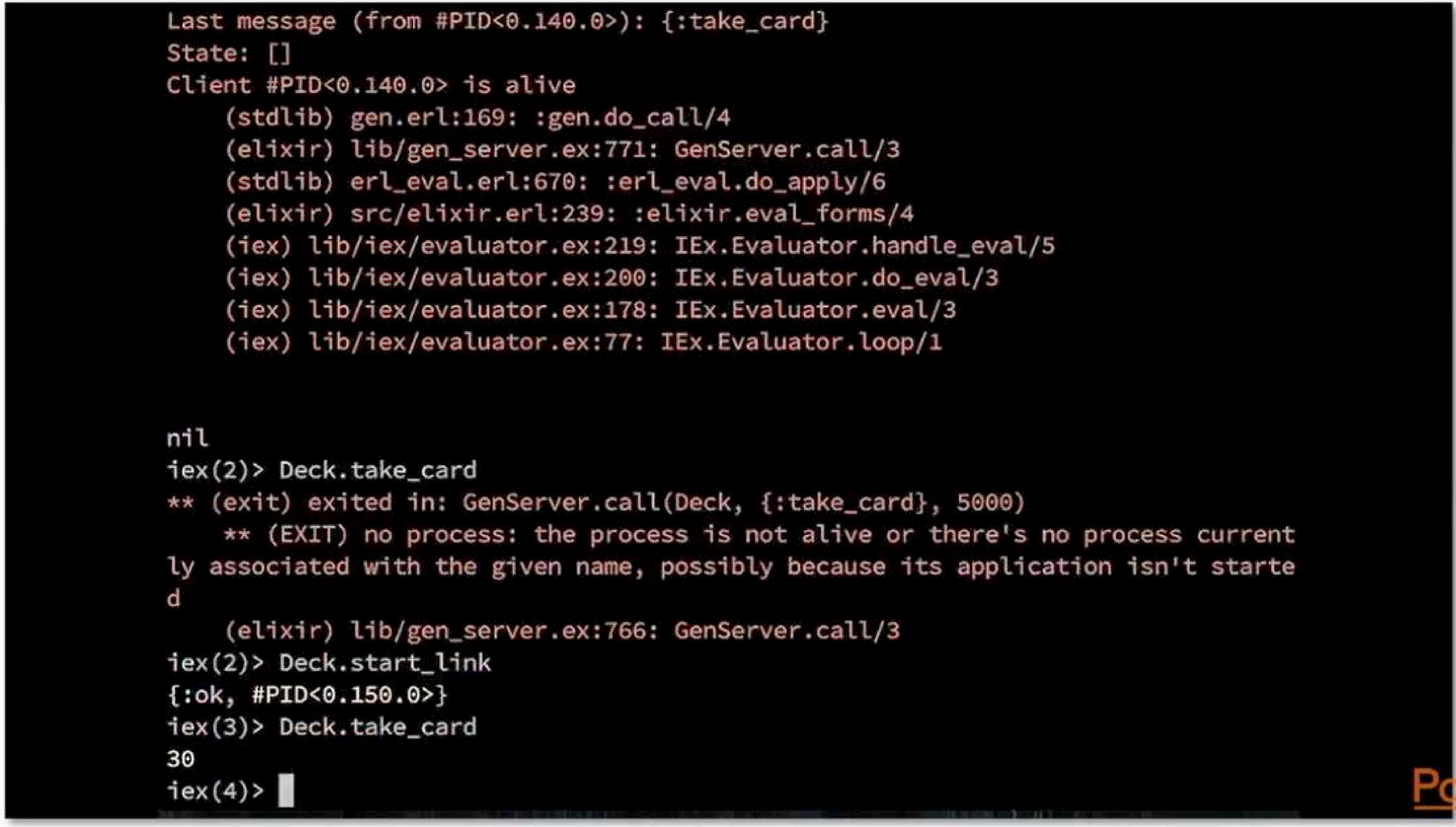
```
        (stdlib) gen_server.erl:665: :gen_server.handle_msg/6
        (stdlib) proc_lib.erl:247: :proc_lib.init_p_do_apply/3

Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)>
10:04:09.830 [error] GenServer Deck terminating
** (FunctionClauseError) no function clause matching in Deck.handle_call/3
    (card_deck) lib/deck.ex:16: Deck.handle_call({:take_card}, {#PID<0.140.0>, #
Reference<0.2993767867.510132225.102340>}, [])
    (stdlib) gen_server.erl:636: :gen_server.try_handle_call/4
    (stdlib) gen_server.erl:665: :gen_server.handle_msg/6
    (stdlib) proc_lib.erl:247: :proc_lib.init_p_do_apply/3
Last message (from #PID<0.140.0>): {:take_card}
State: []
Client #PID<0.140.0> is alive
    (stdlib) gen.erl:169: :gen.do_call/4
    (elixir) lib/gen_server.ex:771: GenServer.call/3
    (stdlib) erl_eval.erl:670: :erl_eval.do_apply/6
    (elixir) src/elixir.erl:239: :elixir.eval_forms/4
    (iex) lib/iex/evaluator.ex:219: IEx.Evaluator.handle_eval/5
    (iex) lib/iex/evaluator.ex:200: IEx.Evaluator.do_eval/3
    (iex) lib/iex/evaluator.ex:178: IEx.Evaluator.eval/3
    (iex) lib/iex/evaluator.ex:77: IEx.Evaluator.loop/1
```

Deck of Cards



```
Reference<0.2993767867.510132225.102340>}, [])
    (stdlib) gen_server.erl:636: :gen_server.try_handle_call/4
    (stdlib) gen_server.erl:665: :gen_server.handle_msg/6
    (stdlib) proc_lib.erl:247: :proc_lib.init_p_do_apply/3
Last message (from #PID<0.140.0>): {:take_card}
State: []
Client #PID<0.140.0> is alive
    (stdlib) gen.erl:169: :gen.do_call/4
    (elixir) lib/gen_server.ex:771: GenServer.call/3
    (stdlib) erl_eval.erl:670: :erl_eval.do_apply/6
    (elixir) src/elixir.erl:239: :elixir.eval_forms/4
    (iex) lib/iex/evaluator.ex:219: IEx.Evaluator.handle_eval/5
    (iex) lib/iex/evaluator.ex:200: IEx.Evaluator.do_eval/3
    (iex) lib/iex/evaluator.ex:178: IEx.Evaluator.eval/3
    (iex) lib/iex/evaluator.ex:77: IEx.Evaluator.loop/1



nil
iex(2)> Deck.take_card
** (exit) exited in: GenServer.call(Deck, {:take_card}, 5000)
    ** (EXIT) no process: the process is not alive or there's no process current
ly associated with the given name, possibly because its application isn't starte
d
    (elixir) lib/gen_server.ex:766: GenServer.call/3
iex(2)>
```

Deck of Cards



```
Last message (from #PID<0.140.0>): {:take_card}
State: []
Client #PID<0.140.0> is alive
    (stdlib) gen.erl:169: :gen.do_call/4
    (elixir) lib/gen_server.ex:771: GenServer.call/3
    (stdlib) erl_eval.erl:670: :erl_eval.do_apply/6
    (elixir) src/elixir.erl:239: :elixir.eval_forms/4
    (iex) lib/iex/evaluator.ex:219: IEx.Evaluator.handle_eval/5
    (iex) lib/iex/evaluator.ex:200: IEx.Evaluator.do_eval/3
    (iex) lib/iex/evaluator.ex:178: IEx.Evaluator.eval/3
    (iex) lib/iex/evaluator.ex:77: IEx.Evaluator.loop/1



nil
iex(2)> Deck.take_card
** (exit) exited in: GenServer.call(Deck, {:take_card}, 5000)
    ** (EXIT) no process: the process is not alive or there's no process current
ly associated with the given name, possibly because its application isn't starte
d
    (elixir) lib/gen_server.ex:766: GenServer.call/3
iex(2)> Deck.start_link
{:ok, #PID<0.150.0>}
iex(3)> Deck.take_card
30
iex(4)> 
```