cards
_build
config
lib
cards.ex
test
.gitignore
mix.exs
README.md

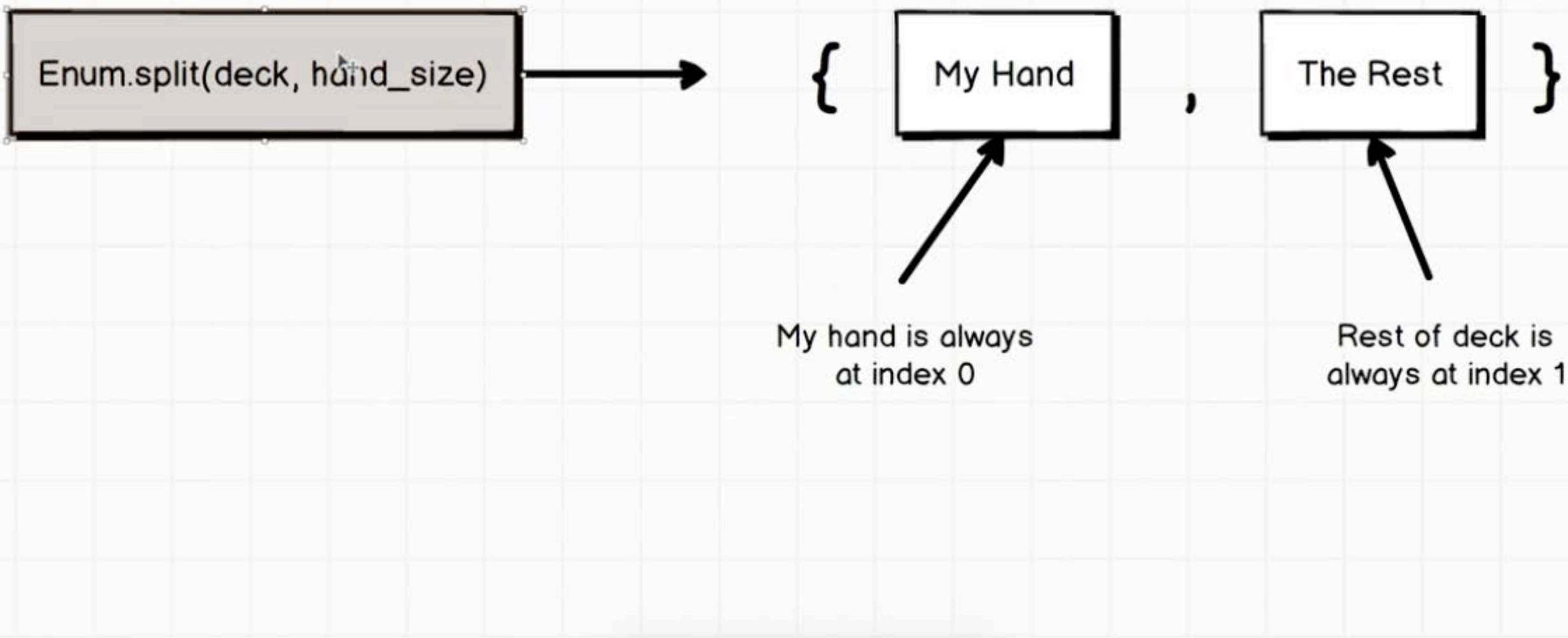
cards.ex

```
15    def contains?(deck, card) do
16      Enum.member?(deck, card)
17    end
18
19    def deal(deck, hand_size) do
20      Enum.split(deck, hand_size)
21    end
22  end
23
```

```
iex(23)> deck = Cards.create_deck
```

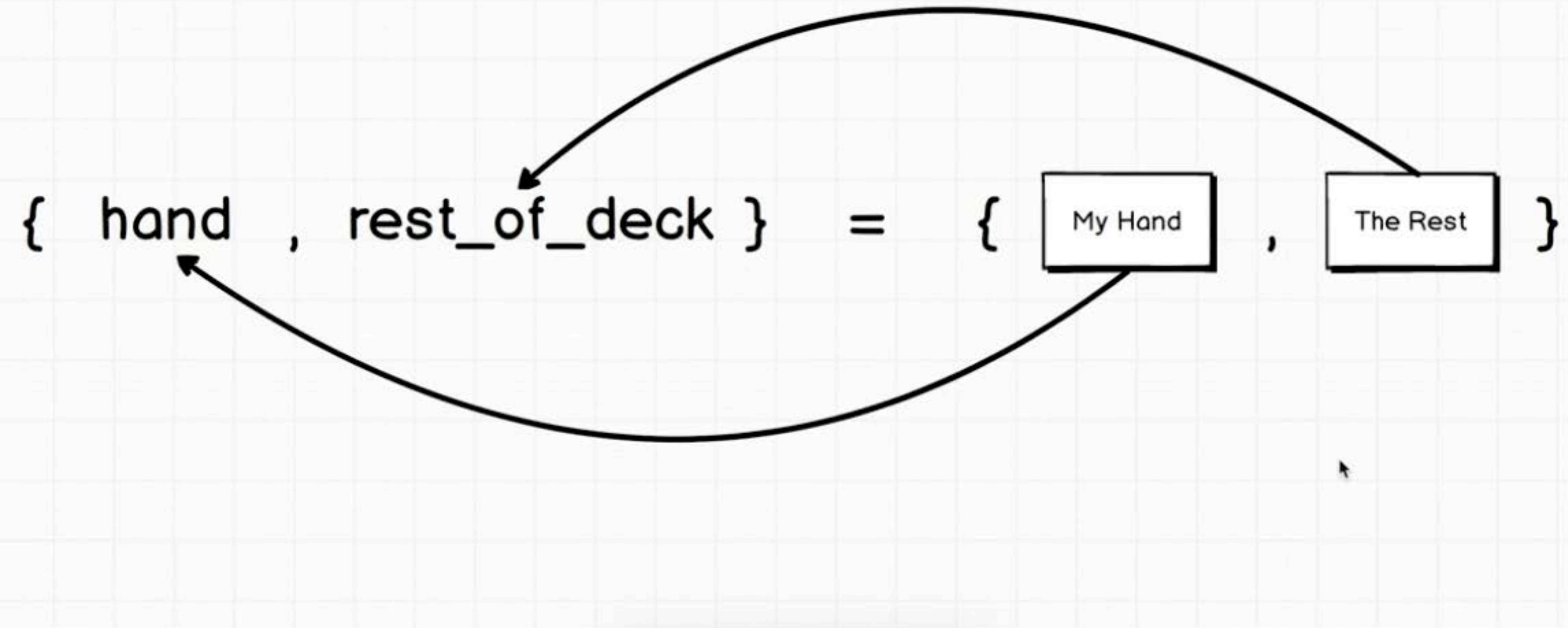
```
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds",
 "Three of Diamonds", "Four of Diamonds",
 "Five of Diamonds"]
iex(24)> Cards.deal(deck, 4)
{["Ace of Spades", "Two of Spades", "Three of Spades",
  "Four of Spades"],
 ["Five of Spades", "Ace of Clubs", "Two of Clubs",
  "Three of Clubs", "Four of Clubs", "Five of Clubs",
  "Ace of Hearts", "Two of Hearts", "Three of Hearts",
  "Four of Hearts", "Five of Hearts", "Ace of Diamonds",
  "Two of Diamonds", "Three of Diamonds",
  "Four of Diamonds", "Five of Diamonds"]}
iex(25)>
```

```
  "Four of Diamonds", "Five of Diamonds"]}
iex(25)> Cards.deal(deck, 4)[0]
** (FunctionClauseError) no function clause matching in A
ccess.fetch/2
    (elixir) lib/access.ex:147: Access.fetch({["Ace of Sp
ades", "Two of Spades", "Three of Spades", "Four of Spade
s"], ["Five of Spades", "Ace of Clubs", "Two of Clubs", "
Three of Clubs", "Four of Clubs", "Five of Clubs", "Ace o
f Hearts", "Two of Hearts", "Three of Hearts", "Four of H
earts", "Five of Hearts", "Ace of Diamonds", "Two of Diam
onds", "Three of Diamonds", "Four of Diamonds", "Five of
Diamonds"]}, 0)
    (elixir) lib/access.ex:179: Access.get/3
iex(25)>
```



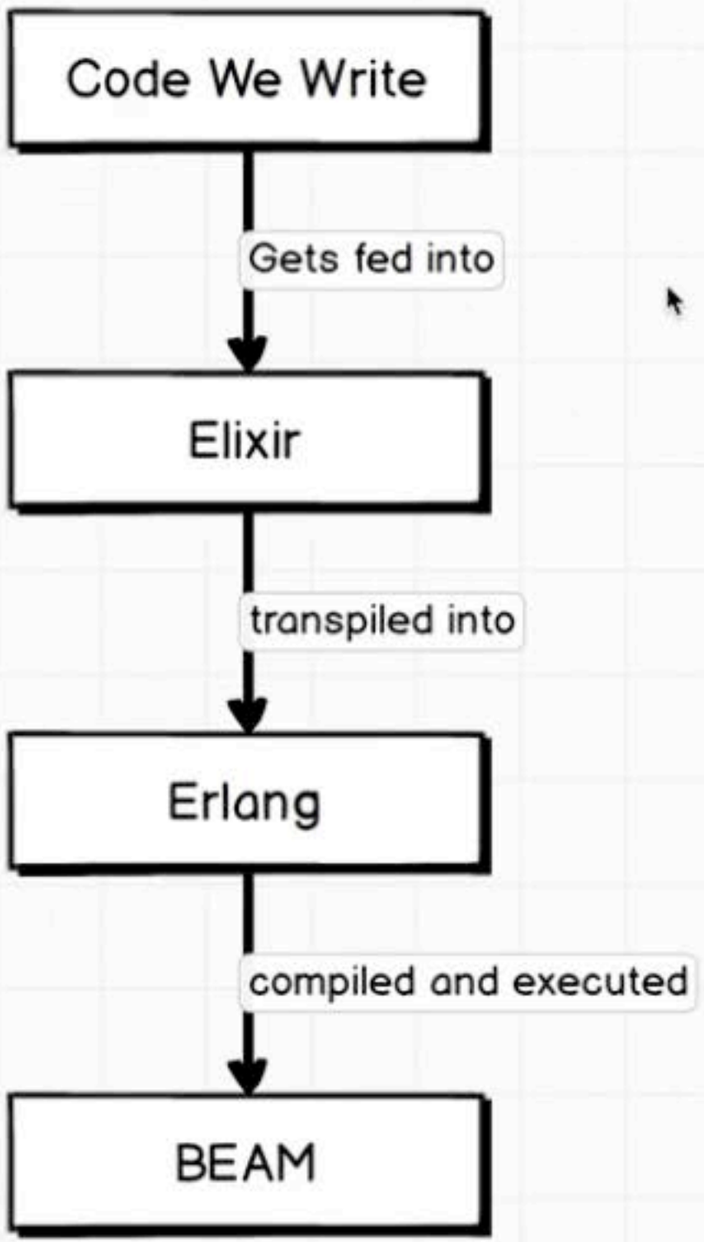
```
 "Ace of Diamonds", "Two of Diamonds",
 "Three of Diamonds", "Four of Diamonds",
 "Five of Diamonds"]
iex(27)> { hand, rest_of_deck } = Cards.deal(deck, 5)
{["Ace of Spades", "Two of Spades", "Three of Spades",
  "Four of Spades", "Five of Spades"],
 ["Ace of Clubs", "Two of Clubs", "Three of Clubs",
  "Four of Clubs", "Five of Clubs", "Ace of Hearts",
  "Two of Hearts", "Three of Hearts", "Four of Hearts",
  "Five of Hearts", "Ace of Diamonds", "Two of Diamonds",

  "Three of Diamonds", "Four of Diamonds",
  "Five of Diamonds"]}
iex(28)>
```

```
  "Three of Diamonds", "Four of Diamonds",
  "Five of Diamonds"]}
iex(28)> hand
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades"]
iex(29)> rest_of_deck
["Ace of Clubs", "Two of Clubs", "Three of Clubs",
 "Four of Clubs", "Five of Clubs", "Ace of Hearts",
 "Two of Hearts", "Three of Hearts", "Four of Hearts",
 "Five of Hearts", "Ace of Diamonds", "Two of Diamonds",
 "Three of Diamonds", "Four of Diamonds",
 "Five of Diamonds"]
iex(30)>
```



```
iex(30)> color1 = ["red"]
["red"]
iex(31)> color1
["red"]
iex(32)> [ color1 ] = ["red"]
["red"]
iex(33)> color1
"red"
iex(34)>
```

```
["red"]
iex(31)> color1
["red"]
iex(32)> [ color1 ] = ["red"]
["red"]
iex(33)> color1
"red"
iex(34)> [color1, color2] = ["red", "blue"]
["red", "blue"]
iex(35)> color1
"red"
iex(36)> color2
"blue"
iex(37)>
```



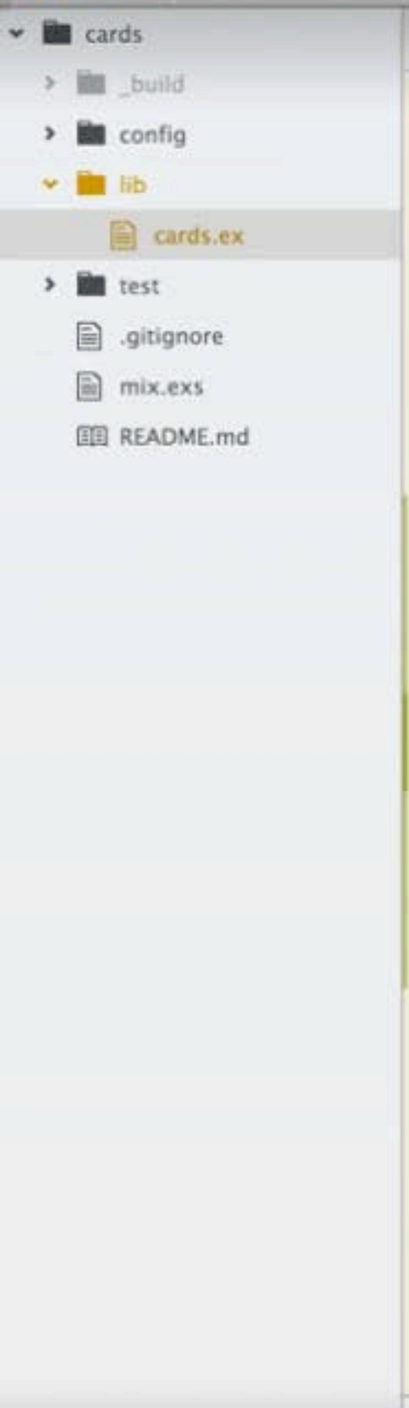
Atom File Edit View Selection Find Packages Window Help

Thu 2:29 PM





cards.ex — /Users/stephengrider/workspace/ElixirWorkspace/prod/cards




cards.ex

```
18
19    def deal(deck, hand_size) do
20      Enum.split(deck, hand_size)
21    end
22
23    def save(deck, filename) do
24      binary = :erlang.term_to_binary(deck)
25      File.write(filename, binary)
26    end
27  end
28
```

```
iex(37)> recompile
Compiling 1 file (.ex)
:ok
iex(38)> deck = Cards.create_deck
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades", "Ace of Clubs",
 "Two of Clubs", "Three of Clubs", "Four of Clubs",
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds",
 "Three of Diamonds", "Four of Diamonds",
 "Five of Diamonds"]
iex(39)>
```

```
Compiling 1 file (.ex)
:ok
iex(38)> deck = Cards.create_deck
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades", "Ace of Clubs",
 "Two of Clubs", "Three of Clubs", "Four of Clubs",
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds",
 "Three of Diamonds", "Four of Diamonds",
 "Five of Diamonds"]
iex(39)> Cards.save(deck, 'my_deck')
:ok
iex(40)>
```

```
➜  cards git:(026-saving-a-deck) ✗ iex -S mix
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [smp:4:4] [async-thr
eads:10] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.3.1) - press Ctrl+C to exit (type h() ENT
ER for help)
iex(1)> File.read("my_deck")
{:ok,
 <<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
   111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
   84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
   109, 0, 0, 0, 15, 84, 104, ...>>}
iex(2)>
```

```
Interactive Elixir (1.3.1) - press Ctrl+C to exit (type h() ENT
ER for help)
iex(1)> File.read("my_deck")
{:ok,
 <<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
   111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
   84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
   109, 0, 0, 0, 15, 84, 104, ...>>}
iex(2)> {status, binary} = File.read("my_deck")
{:ok,
 <<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
   111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
   84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
   109, 0, 0, 0, 15, 84, 104, ...>>}
iex(3)> st
```

```
   109, 0, 0, 0, 15, 84, 104, ...>>}
iex(2)> {status, binary} = File.read("my_deck")
{:ok,
 <<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
   111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
   84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
   109, 0, 0, 0, 15, 84, 104, ...>>}
iex(3)> status
:ok
iex(4)> binary
<<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
  111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
  84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
  109, 0, 0, 0, 15, 84, 104, 114, ...>>
iex(5)>
```

```
:ok
iex(4)> binary
<<131, 108, 0, 0, 0, 20, 109, 0, 0, 0, 13, 65, 99, 101, 32,
  111, 102, 32, 83, 112, 97, 100, 101, 115, 109, 0, 0, 0, 13,
  84, 119, 111, 32, 111, 102, 32, 83, 112, 97, 100, 101, 115,
  109, 0, 0, 0, 15, 84, 104, 114, ...>>
iex(5)> :erlang.binary_to_term(binary)
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades", "Ace of Clubs",
 "Two of Clubs", "Three of Clubs", "Four of Clubs",
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds", "Three of Diamonds",
 "Four of Diamonds", "Five of Diamonds"]
iex(6)>
```

cards
_build
config
lib
cards.ex
test
.gitignore
mix.exs
my_deck
README.md

cards.ex

```
18
19    def deal(deck, hand_size) do
20      Enum.split(deck, hand_size)
21    end
22
23    def save(deck, filename) do
24      binary = :erlang.term_to_binary(deck)
25      File.write(filename, binary)
26    end
27
28    def load(filename) do
29      {status, binary} = File.read(filename)
30      :erlang.binary_to_term binary
31    end
32  end
33
```

```
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds", "Three of Diamonds",
 "Four of Diamonds", "Five of Diamonds"]
iex(6)> recompile
:noop
iex(7)> Cards.load("my_deck")
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades", "Ace of Clubs",
 "Two of Clubs", "Three of Clubs", "Four of Clubs",
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds", "Three of Diamonds",
 "Four of Diamonds", "Five of Diamonds"]
iex(8)>
```

```
iex(6)> recompile
:noop
iex(7)> Cards.load("my_deck")
["Ace of Spades", "Two of Spades", "Three of Spades",
 "Four of Spades", "Five of Spades", "Ace of Clubs",
 "Two of Clubs", "Three of Clubs", "Four of Clubs",
 "Five of Clubs", "Ace of Hearts", "Two of Hearts",
 "Three of Hearts", "Four of Hearts", "Five of Hearts",
 "Ace of Diamonds", "Two of Diamonds", "Three of Diamonds",
 "Four of Diamonds", "Five of Diamonds"]
iex(8)> Cards.load("my_deckasdfghjkqwertyuiosdfghjk")
** (ArgumentError) argument error
             :erlang.binary_to_term(:enoent)
    (cards) lib/cards.ex:30: Cards.load/1
iex(8)>
```

cards.ex — /Users/stephengrider/workspace/ElixirWorkspace/prod/cards

cards
_build
config
lib
cards.ex
test
.gitignore
mix.exs
my_deck
README.md

cards.ex

```
27
28    def load(filename) do
29      {status, binary} = File.read(filename)
30      :erlang.binary_to_term binary
31    end
32
33    def load() do
34      {status, binary} = File.read(filename)
35
36      if(status === :error) {
37        return "Something went wrong"
38      }
39    end
40  end
41
```

cards
_build
config
lib
cards.ex
test
.gitignore
mix.exs
my_deck
README.md

cards.ex

```
23  def save(deck, filename) do
24    binary = :erlang.term_to_binary(deck)
25    File.write(filename, binary)
26  end
27
28  def load(filename) do
29    {status, binary} = File.read(filename)
30
31    case status do
32      :ok -> :erlang.binary_to_term binary
33      :error -> "That file does not exist"
34    end
35  end
36 end
37
```

```
iex(11)>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^C%
➜  cards git:(026-saving-a-deck) ✗
➜  cards git:(026-saving-a-deck) ✗
➜  cards git:(026-saving-a-deck) ✗ iex -S mix
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [smp:4:4] [async-thr
eads:10] [hipe] [kernel-poll:false] [dtrace]

Interactive Elixir (1.3.1) - press Ctrl+C to exit (type h() ENT
ER for help)
iex(1)> Cards.load("asldkfjalksdfj")
"That file does not exist"
iex(2)>
```