Elixir Lesson 2018/7/10

アジェンダ

- これまでの復習
  - 環境構築とmix,iexのインストール
  - モジュールと関数
- Elixirの型を学ぶ
  - 文字列型
  - 数値型(TBD)
  - 真偽値(TBD)
  - Atom型
  - List型
  - Tuples型
  - Anonymous Function型(無名関数)
- 練習問題(関数に値をいろいろ入れてくみあわせてみる)

このレッスンの所要時間

- 1時間

Elixirの型を学ぶ

文字列型

- 特徴
  - Elixirでは、ダブルクォート(")で囲った値が 文字列型(String)の値として扱われます
  - 文字列は UTF-8にエンコードされます
- サンプルコード

```
iex> str0 = "A"
"A"

iex> is_bitstring(str0)
true

iex> bit_size(str0)
```

```
8

iex> str1 = "hello"
"hello"

iex> is_bitstring(str1)
true

iex> bit_size(str1)
40

iex> str2 = "こんにちは"
"こんにちは"

iex> is_bitstring(str2)
true

iex> bit_size(str2)
120
```

数値型

- 特徴
    - 数値型はElixirの基本型の１つ
    - 整数値と浮動小数値を表すことが可能
    - 数値型は、１０進数、１６進数がありそれぞれ表現可能
- サンプルコード

```
iex 1> 1
1
iex 2> is_integer(1)
true
iex 3> 0x1F
31
iex 4> is_integer(0x1F)
true
iex 5> 1.0
1.0
iex 6> is_float(1.0)
true
```

真偽値型

- 特徴

    - 真偽値型はElixirの基本型の１つ
    - 真(true)と偽(false)の２つの値を取り、Elixir内で定義されている

- サンプルコード

```
iex> true
true
iex> true == false
false

iex> is_boolean(true)
true

iex> is_boolean(false)
true

iex> age = 20
true

iex> is_smoking_allowed_jp = (age >= 20)
true

iex> is_boolean(is_smoking_allowed_jp)
true
```

Atom

- 特徴
    - 数値に名前をつけられる
    - true/falseもatom, クラス名はatomとして管理されている

- サンプルコード(TBD)

```
iex 7> is_atom(Tuple)
true
iex 8> is_atom(File)
true
```

List型

- 特徴
    - TBD
- サンプルコード

```
* リストを作成する
  * listTest = [1, 2, 3]
* リストの要素を取得する
  * Enum.at(listTest, 1)
* リスト同士の足し算
  * [1,2,3] ++ [4,5,6]
```

Tuples

- 特徴
    - TBD
- サンプルコード

```
iex(1)> tuple = {:v1, :v2, :v3}
{:v1,:v2,:v3}
iex(2)> tuple
{:v1,:v2,:v3}
iex(3)> tuple_size(tuple)
3
iex(4)> elem(tuple,1)
:v2
iex(5)> elem(tuple,2)
:v3
```

Anonymous Function(無名関数)

- 関数とは
    - 入力を与えると、出力が得られる装置
    - 入力とは
    - 出力とは
    - 関数に何らかの入力xを与えて実行すると、何らかの出力yが得られる
    - 参照: 入力xを与えたらx に100を足した値を出力する関数を考える
    - 関数を実行するには
    - まず関数を定義する
    - 定義した呼び出す(ここで実行される)
- 特徴
    - Elixirでは関数も 1 つの型として規定されている

    - 文字列や数値など他の型を扱う際と同じレベルで、 `関数` 値を表現する方法を学ぶ

- サンプルコード (入力xを与えたらx に100を足した値を出力する関数を考える)

```
iex(1)> x = 42
iex(2)> f = (fn (value) -> value + 100 end) # 変数fに `関数` 値を格納する
iex(2)> x = f.(x) # 関数を実行する(`関数` 値を格納した変数f を実行する)
iex(3)> x # 何が表示されるか
```

練習問題(関数に値をいろいろ入れてくみあわせてみる)

- TBD