Elixir Basics

Date Types

Elixir has the usual data types. Coming from Python, you might find them very easy to digest.

```
5
x = 34
x

x = 10.0   # Can rebind variables
"String -- has double quotes, not single quotes."

true
false

# Atoms are important
# An atom is a constant whose name is its own value
:foo
```

Built-in Data Structures

```
[1, 2, 4]   # lists
{1, 2, 3}   # tuples
```

Important difference from Python is that *both* lists and tuples are immutable in elixir – difference is only in performance in different scenarios.

We also have Maps which are essentially Python's dictionaries.

```
%{"x" => 34, :y => 25}
%{"x" => 34, y: 25}     # equivalent to the above

m = %{:a => 1, 2 => :b}
Map.get(m, :a)   # verbose
m[:a]            # slightly shorter
m.a              # simple since a is an atom
```

Basic Arithmetic

```
1 + 2
5 * 5
9/2   # Python 3 style
```

```elixir
div(9, 2)     # Integer division
rem 9, 2      # Notice the missing parens
# div (9, 2)  # Can't have a space before the paren
```

Booleans and Stuff

```elixir
true == true
3 == 3
3 == 3.0
3 === 3.0    # Value and data type

is_integer("something")
is_integer(2)
is_boolean(1)

"hello" == 'hello'  # These are completely different animals
```

String Interpolation

```elixir
"héllo #{:world}"   # UTF-8 strings by default

"hello
world"

IO.puts "Something"  # Notice the :ok
IO.inspect "Something"

String.upcase("hello")
```

List Operations

```elixir
[1, 2, 3] ++ [4, 5, 6]     # Concat
[1, 2, 3, 4, 5] -- [1, 2]

list = [1, 2, 3]
hd(list)
hd

tl(list)
tl
```

Studying Types

```
i 35

x = 56
i x
```

Tuple Operators

```
tuple = {:ok, "hello"}
elem(tuple, 1)      # 0-based index
```

Let's see an Error

```
missing_function(24)

# Gives:
# ** (CompileError) iex:26: undefined function missing_function/1
```

It's important to understand the error. We see fun/arity here.

"Elixir and Phoenix: Real World Functional Programming"

Video Course by Dr. Nauman

`http://recluze.net/learn`