

## Section 2

# *Basic Types and Operators*



In this Section, we are going to take a look at...

- Literal types
- Basic operators
- Collection types
- Functions to manipulate collections

## Video 2.1

# *Literals and Operators*



In this video, we are going to take a look at...

- Observing the built-in literal types
- Exploring the basic operators

## Before We Start

### Terminal

```
>iex
```

```
Erlang/OTP 19 [erts-8.2]
```

```
Interactive Elixir (1.3.4) - press Ctrl+C to exit (type  
h() ENTER for help)
```

```
iex(1)>
```

# Numbers and Arithmetic Operators

- Integer
  - 1
  - -30
- Floating point
  - 1.54932
  - 1.5e-15

# Numbers and Arithmetic Operators

- Integer Bases

Base	Example	Decimal
Hexadecimal (16)	0xFE	254
Octal (8)	0o773	507
Binary (2)	0b1110	14

# Numbers and Arithmetic Operators

- Integer – Separator
  - 43312209
  - 43\_312\_209



# Numbers and Arithmetic Operators

- Precision

```
55438573489534785439534857345764234854236
78452367423546723456237452367452367453267
45237645237645237645326745236745762345631
245762345263
```

# Numbers and Arithmetic Operators

- Operators

Operator	Example	Result
Addition (+)	$1.5 + 9$	10.5
Subtraction (-)	$1 - 2$	-1
Multiplication (*)	$9 * 8$	72
Division (/)	$10 / 4$	2.5

# Numbers and Arithmetic Operators

- Integer Division Functions

Operator	Example	Result
Division (div)	<code>div(9, 7)</code>	1
Modulo/Remainder (rem)	<code>rem(9, 7)</code>	2

# Booleans

## Terminal

```
iex(1)> (1 + 2 + (3 * 12)) / div(6,3)  
13.0
```

# Booleans

- True
  - Everything is truthy
- False
  - nil is falsey

# Booleans

and, or, not → strict

&&, ||, ! → non-strict

# Booleans

## Terminal

```
iex(1)> true and false  
false
```

```
iex(2)> true && false  
false
```

```
iex(3)> 3 and true
```

```
** (ArgumentError) argument error: 3
```

```
iex(4)> 3 && true  
true
```

## Booleans – Comparison Operators

>, <, >=, <=, !=, ==, ===, ===



true  
false



## Booleans – Comparison Operators

>, <, >=, <=, !=, ==, !==, ===



The diagram shows a horizontal line with arrows pointing down to the labels 'Non-Strict' and 'Strict'. The line is divided into two segments: an orange segment on the left and a blue segment on the right. The orange segment contains the operators '!=', '==', and '!='. The blue segment contains the operators '===', '!==', and '==='.

Non-Strict

Strict

## Booleans – Comparison Operators

### Terminal

```
iex(1)> 15.0 == 15  
true  
iex(2)> 15.0 === 15  
false
```

# Booleans – Comparison Operators

- Not just for numbers
  - `number < atom < reference < function < port < pid < tuple < map < list < bitstring`
- Multiple types can be compared with one another by this order

# Strings

`"Hello, World!"`

`"γειά σου κόσμ"`

# Strings

“Hello, World!”

“γειά σου κόσμ”



UTF-8

# Strings

`"Hello, \nWorld!"`  `Hello,  
World!`

# Strings

“I am `#{2 * 15}` years old.”



I am 30 years old.

## Strings – Some fun(ctions)

### Terminal

```
iex(1)> String.reverse("Hello")
```

```
"olleH"
```

```
iex(2)> String.replace("Hello, World!", "World", "You")
```

```
"Hello, You!"
```



# Atoms

- A constant whose name is its value
  - `:hello`
  - `:Hey_you`
  - `:”>_<“`

## Atoms

:4\_seasons

Is invalid!



: "4\_seasons"

Valid atom