JULIA PROGRAMMING

Variables, Data Types, Operations

dr. ilker arslan

Dynamically typed language

Optionally typed language

User defined types

Parameterized types

JULIA PROGRAMMING

Variables

dr. ilker arslan

variable : : varType

x : : Int64

y : : String

text::String = “Julia Language”

ERROR: LoadError: syntax: type declarations on global variables are not yet supported

area(height::Float64, width::Float64)

1. Multiple dispatch
2. Human readability
3. Catch errors

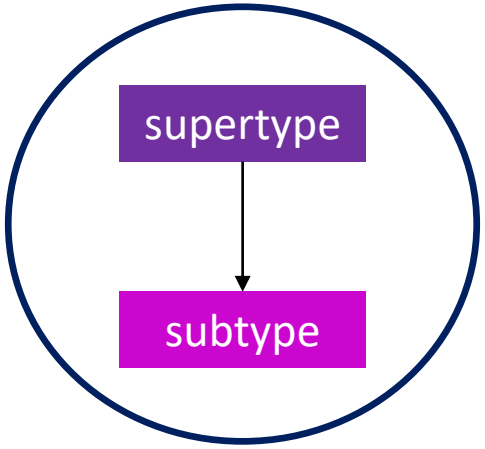
(expression)::DataType

JULIA PROGRAMMING

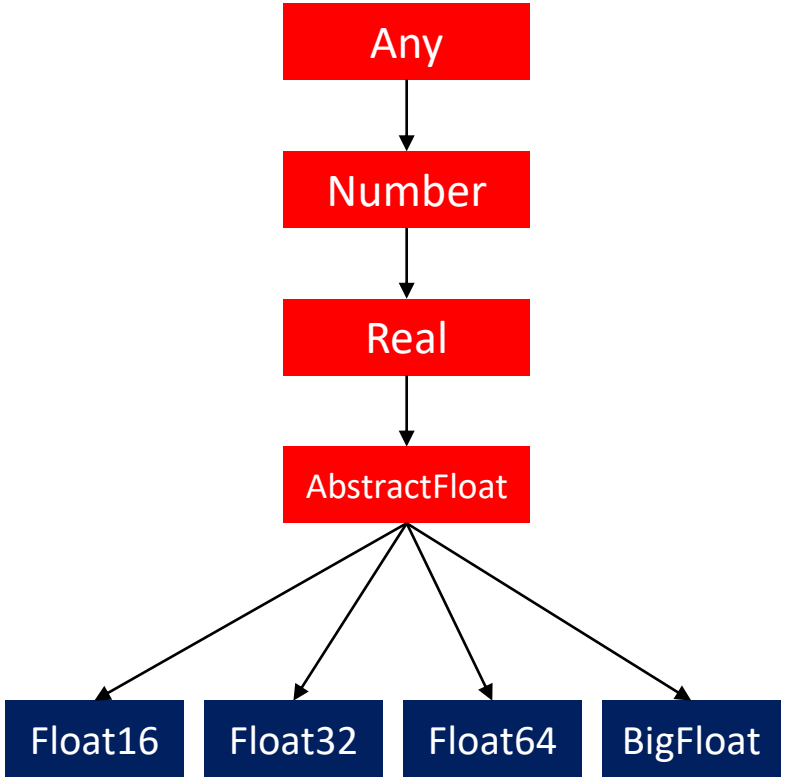
Type Hierarchy in Julia

dr. ilker arslan

Type Hierarchy



Abstract Types Concrete Types



Defining **Abstract Types**

abstract type «type-name» end

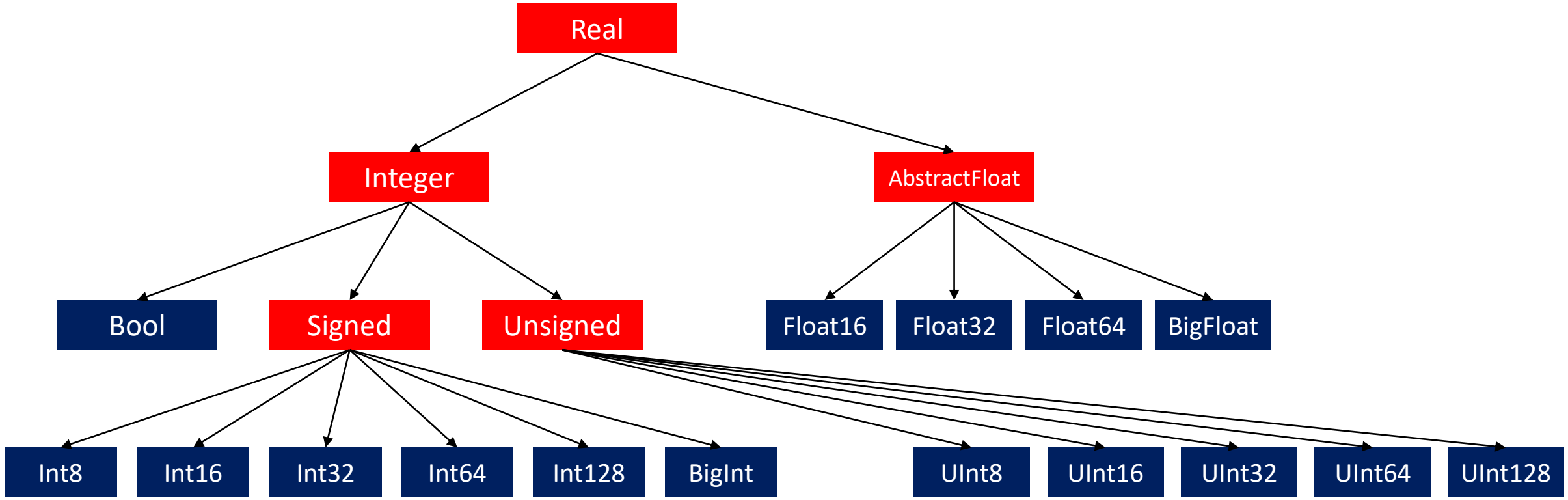
abstract type «type-name» <: «supertype» end

JULIA PROGRAMMING

Numerical data types: Integers and Floating-Point Numbers

dr. ilker arslan

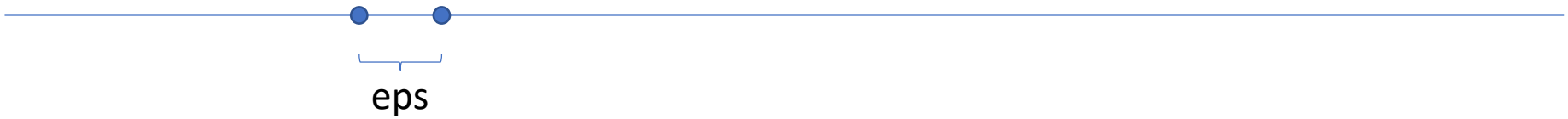
Numerical Data Types: Integers and Floating Point Numbers



Max & Min Integer Values

	Min Value	Max Value
IntN	$(-2)^{N-1}$	$2^{N-1}-1$
UIntN	0	$2^{N}-1$

Machine Epsilon



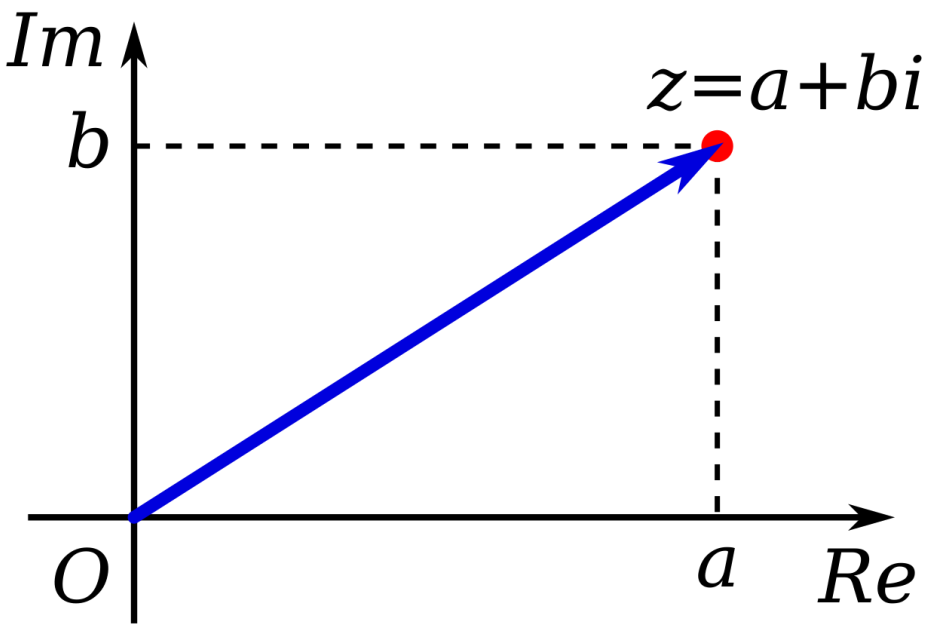
JULIA PROGRAMMING

Numerical data types: Complex and Rational Numbers

dr. ilker arslan

Complex Numbers

$$\sqrt{-1} = i$$

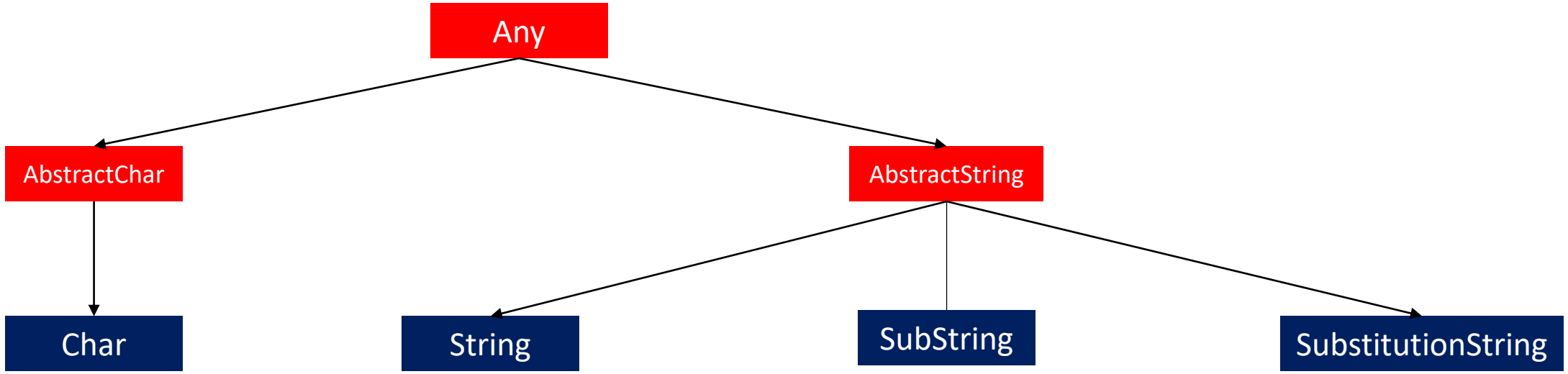


JULIA PROGRAMMING

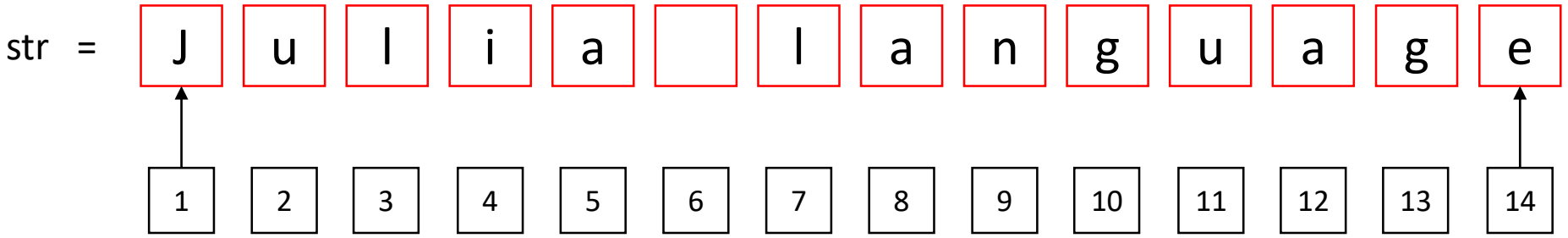
Character & Strings

dr. ilker arslan

Type Hierarchy



Indexing Strings



JULIA PROGRAMMING

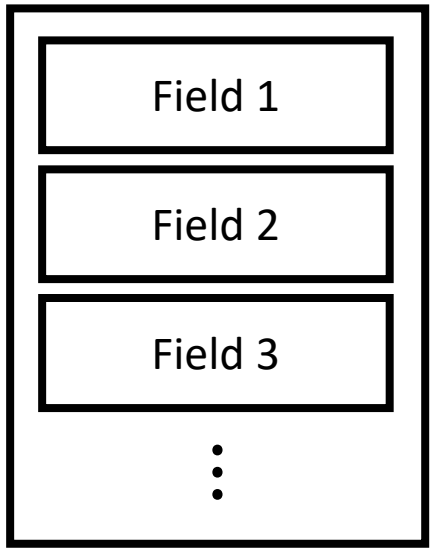
Primitive & Composite Types

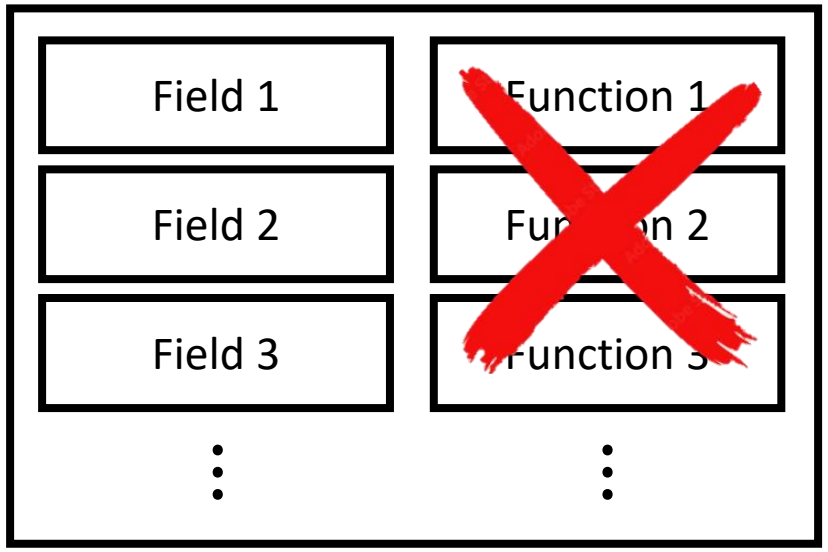
dr. ilker arslan

Primitive Types

```
primitive type «name» «bits» end

primitive type «name» <: «supertype» «bits» end

primitive type Float16 <: AbstractFloat 16 end
primitive type Float32 <: AbstractFloat 32 end
primitive type Float64 <: AbstractFloat 64 end

primitive type Bool <: Integer 8 end
primitive type Char <: AbstractChar 32 end

primitive type Int8 <: Signed 8 end
primitive type UInt8 <: Unsigned 8 end
primitive type Int16 <: Signed 16 end
primitive type UInt16 <: Unsigned 16 end
primitive type Int32 <: Signed 32 end
primitive type UInt32 <: Unsigned 32 end
primitive type Int64 <: Signed 64 end
primitive type UInt64 <: Unsigned 64 end
primitive type Int128 <: Signed 128 end
primitive type UInt128 <: Unsigned 128 end
```

Composite Types



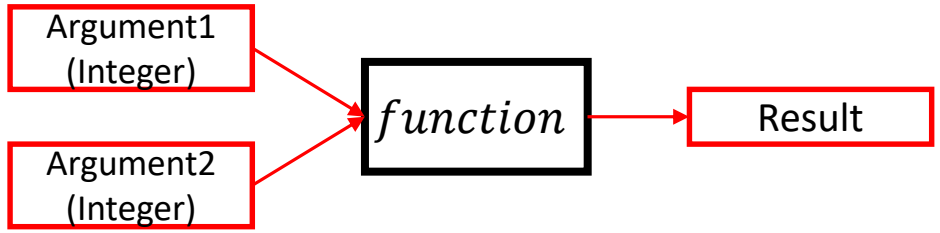


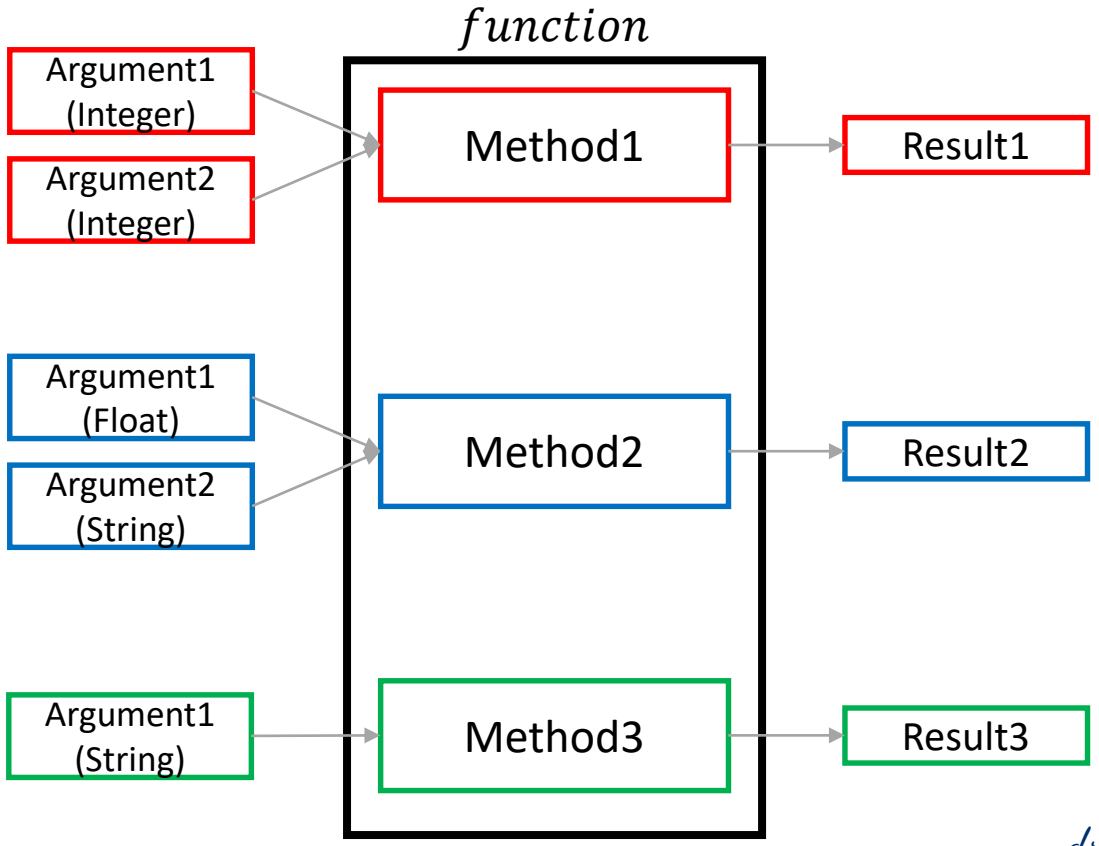
struct

record

object

Multiple Dispatch





$$f(x :: int, y :: int) = x^2 + y^2$$

$$f(x :: float, y :: string) = print(\text{"y ="}, x)$$

$$f(x :: string) = print(\text{"Hello, "}, x, \text{" how are you? "})$$

JULIA PROGRAMMING

Parametric Types

dr. ilker arslan

JULIA PROGRAMMING

Basic Operations

dr. ilker arslan

Arithmetic Operations

Operation	Sign	Explanation
Plus	x + y	Addition
Minus	x - y	Subtraction
Times	x * y	Multiplication
Division	x / y	Division
Integer Division	x ÷ y	Truncates x/y to an integer
Inverse Division	x \ y	Calculates y / x
Power	x^y	Calculates x to the power y
Remainder	x%y	Calculates mod x divided by y

Numeric Comparisons

Operator	Explanation
==	equality
=== ≡	equivalent (same object)
!= ≠	inequality
<	less than
<= ≤	less than or equal to
>	greater than
>= ≥	greater than or equal to

Boolean Operations

Operation	Expression
Negation	!x
AND	x && y
OR	x || y

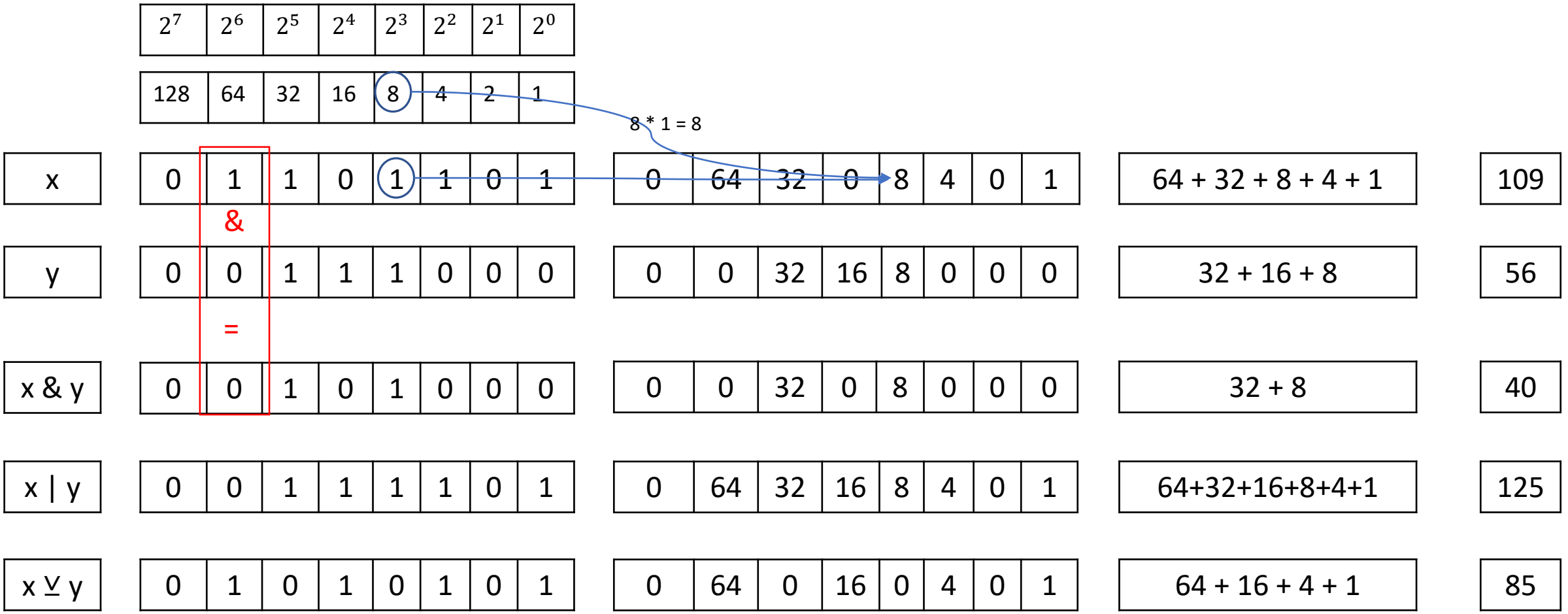
a	b	a && b
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

a	b	a || b
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

Bitwise Operators

Operator	Explanation
~x	bitwise not
x & y	bitwise and
x | y	bitwise or
x ⊻ y	bitwise xor (exclusive or) (\xor)
x ⊼ y	bitwise nand (not and) (\nand)
x ⊽ y	bitwise nor (not or) (\nor)
x >>> y	logical shift (right)
x >> y	arithmetic shift right
x << y	logical /arithmetic shift left

Bitwise Operators: AND OR XOR



Bitwise Operators: NOT

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

1	0	1	0	1	1	0	1	-128 + 32 + 8 + 4 + 1	-83

0	1	1	0	1	1	0	1	64 + 32 + 8 + 4 + 1	109

1	0	0	1	0	0	1	0	-128 + 16 + 2	-110