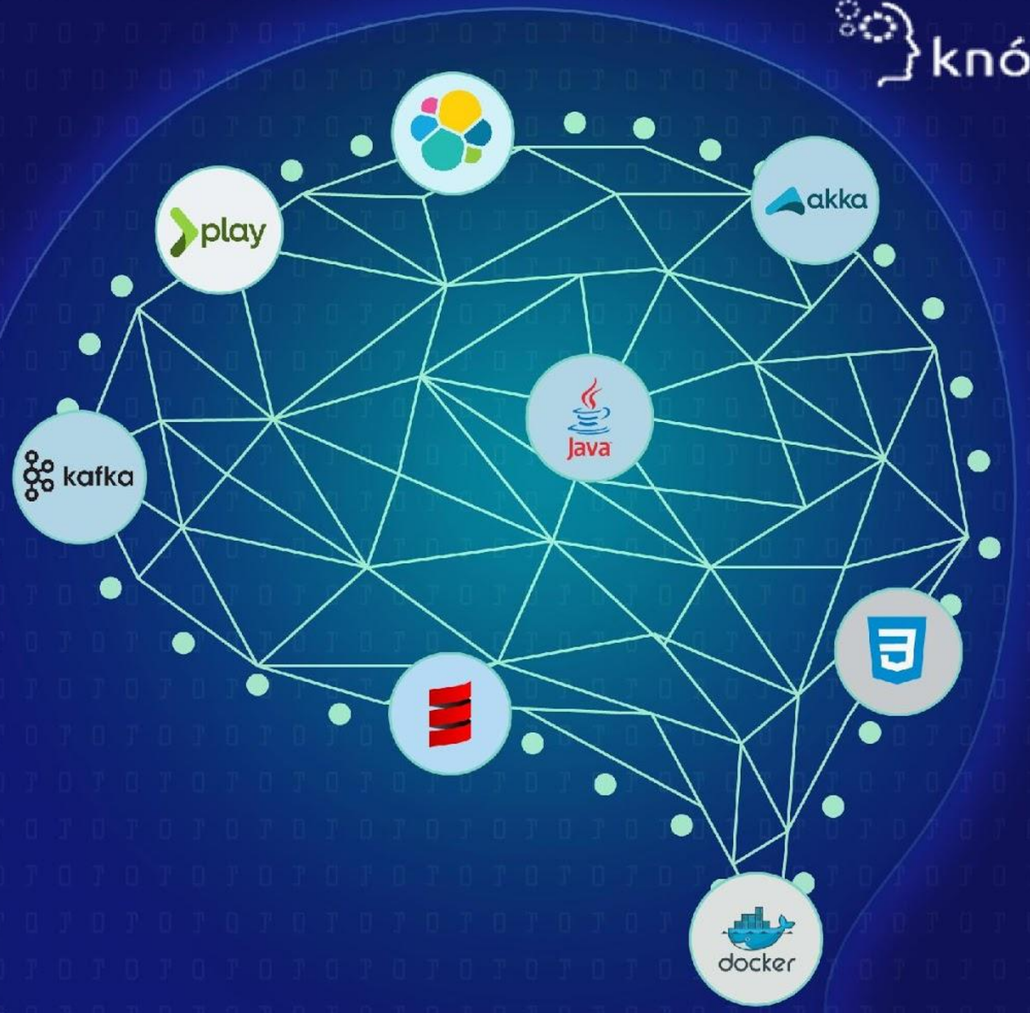
# Numbers




**Presented By**:
Himanshu Gupta

# Agenda

- Converting Between Numeric Types (Casting)
- Overriding the Default Numeric Type
- Replacements for ++ and −−
- Comparing Floating-Point Numbers
- Formatting Currency

# Converting Between Numeric Types (Casting)

## Problem

We want to convert from one numeric type to another, such as from an Int to a Double.

# Converting Between Numeric Types (Casting)

Solution

Instead of using the “cast” approach in Java, use the to* methods that are available on all numeric types.

# Converting Between Numeric Types (Casting)

Example

## Demo

# Overriding the Default Numeric Type

Problem

Scala automatically assigns types to numeric values when we assign them, and we need to override the default type it assigns as we create a numeric field.

# Overriding the Default Numeric Type

Solution

The following examples show one way to override simple numeric types:

- val a = 1d
- val a = 0:Byte
- val a: Byte = 0

# Overriding the Default Numeric Type

Example

## Demo

# Replacements for ++ and −−

Problem

We want to increment or decrement numbers using operators like ++ and −− that are available in other languages, but Scala doesn’t have these operators.

# Replacements for ++ and −−

Solution

Because val fields are immutable, they can’t be incremented or decremented, but var Int fields can be mutated with the += and −= methods.

# Replacements for ++ and −−

Example

## Demo

# Comparing Floating-Point Numbers

## Problem

We need to compare two floating-point numbers, but as in some other programming languages, two floating-point numbers that should be equivalent may not be.

# Comparing Floating-Point Numbers

## Solution

As in Java and many other languages, we solve this problem by creating a method that lets we specify the precision for our comparison.

# Comparing Floating-Point Numbers

Example

## Demo

# Formatting Currency

Problem

We want to format currency to control decimal places and commas, typically for printed output.

# Formatting Currency

Solution

Use the *java.text.NumberFormat.getCurrencyInstance* formatter or *java.util.{Currency, Locale}*

# Formatting Currency

Example

## Demo

# Q/A


Thanks