# Inferência automática de tipos;

O Kotlin ele sabe automaticamente qual o tipo da variável, seja String, int, float...

# Como definir uma variável em Kotlin

Var = (mutável) > posso mudar o valor

Val = (imutável) > variável que não posso mudar o valor (constantes)

package Variaveis  
  
fun main() {  
 *println*("Hello Word!!!")  
 var nome = "Alleph";  
 val cpf = 15679025701;  
}

Segue o exemplo, nome podemos mudar, CPF é único.

# Paradigmas

Kotlin trabalha com Orientação a Objetos e Funcional.

# Como iniciar o Kotlin

Assim como o Java vamos iniciar com uma função main

package Variaveis  
  
fun main() {  
 *println*("Hello Word!!!")  
 var nome = "Alleph"  
 val cpf = 15679025701  
}

# Concatenando no Kotlin

fun main() {  
 *println*("Hello Word!!!")  
 var nome = "Alleph"  
 val cpf = 15679025701  
  
 *println*("Seja bem-vindo $nome $cpf")  
}

Saída:

Hello Word!!!

Seja bem-vindo Alleph 15679025701

# Variáveis – Tipos – Constantes - Instância

fun main() {  
 val nome: String = "Alleph"  
 val cpf: Int = 12345678910  
}

## Tipos das variáveis

Byte, Short, Int, Long, Float, Double, Boolean, Char.

## Constantes

A constante ela deve ser utilizada de duas maneiras, dentro da uma função.

fun main() {  
 val nome: String = "Alleph"  
 val cpf: Long = 12345678910 // Tempo de execução  
}

Fora de uma função

package Variaveis  
  
const val *nomeUsuario*: String = "Alleph Nogueira" // Constante global

// Tempo de compilação  
fun main() {  
 val nome: String = "Alleph"  
 val cpf: Long = 12345678910  
}

Uma variável global, ela está disponível para todas as funções dentro do arquivo.

## Tempo de compilação x Tempo de execução

Tempo de compilação: O computador já tem que saber o valor da variável antes de tudo.

Tempo de execução: O valor pode ser adicionado a uma variável val na hora de compilar.

## Instância com Kotlin

package Variaveis  
  
class Usuario {}  
  
fun main() {  
 val usuario = Usuario()  
 // Lembrar que quando eu faço a instancia com val o usuario não pode ser mudado.  
  
}

# Null Safety

Imagina que você tem um objeto carro, e aí tudo certo, mas você também pode não conseguir instanciar o objeto e aí ficaria nulo, isso geraria um erro e fecharia a aplicação. Ou seja, você não pode deixar de hipótese nenhuma um item nulo, mas para isso o Kotlin tem uma solução; Null Safety (Nulo seguro) podemos colocar dentro de um objeto um valor nulo, ou seja não existe o carro ainda, mas em algum momento o carro vai existir.