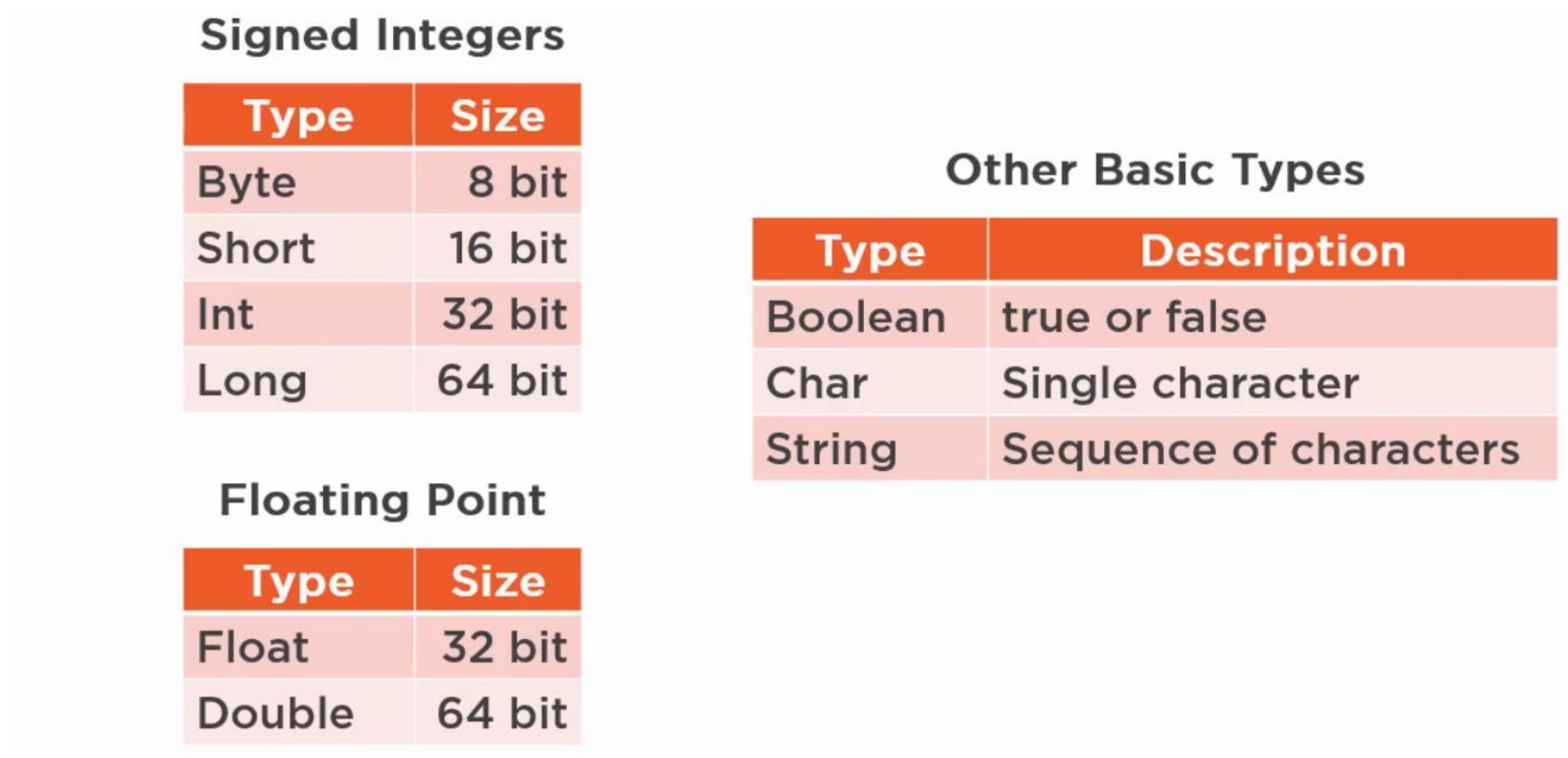
Week 2 : Session Recap

Agenda

- Object-oriented Programming (OOP)
- Kotlin Class
  - Structure of a class.
  - Data members and function/method
    - Getter and setter methods
    - Method/function Overloading
  - Constructors
    - Primary Constructor
    - Parametrized Constructor
  - Initializers
- Exercise 6

Basic Types



Signed Integers

Type	Size
Byte	8 bit
Short	16 bit
Int	32 bit
Long	64 bit

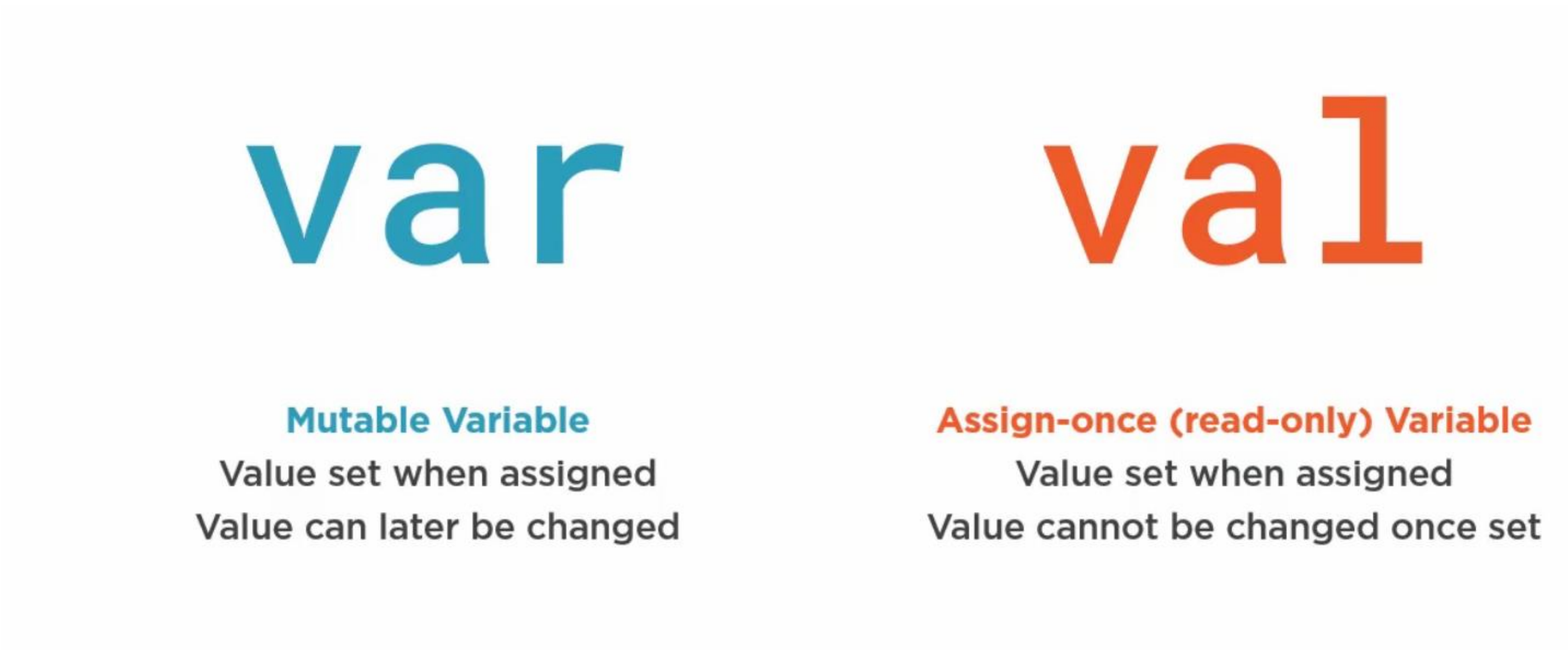
Floating Point

Type	Size
Float	32 bit
Double	64 bit

Other Basic Types

Type	Description
Boolean	true or false
Char	Single character
String	Sequence of characters

Declaring Variables



var

Mutable Variable

Value set when assigned

Value can later be changed

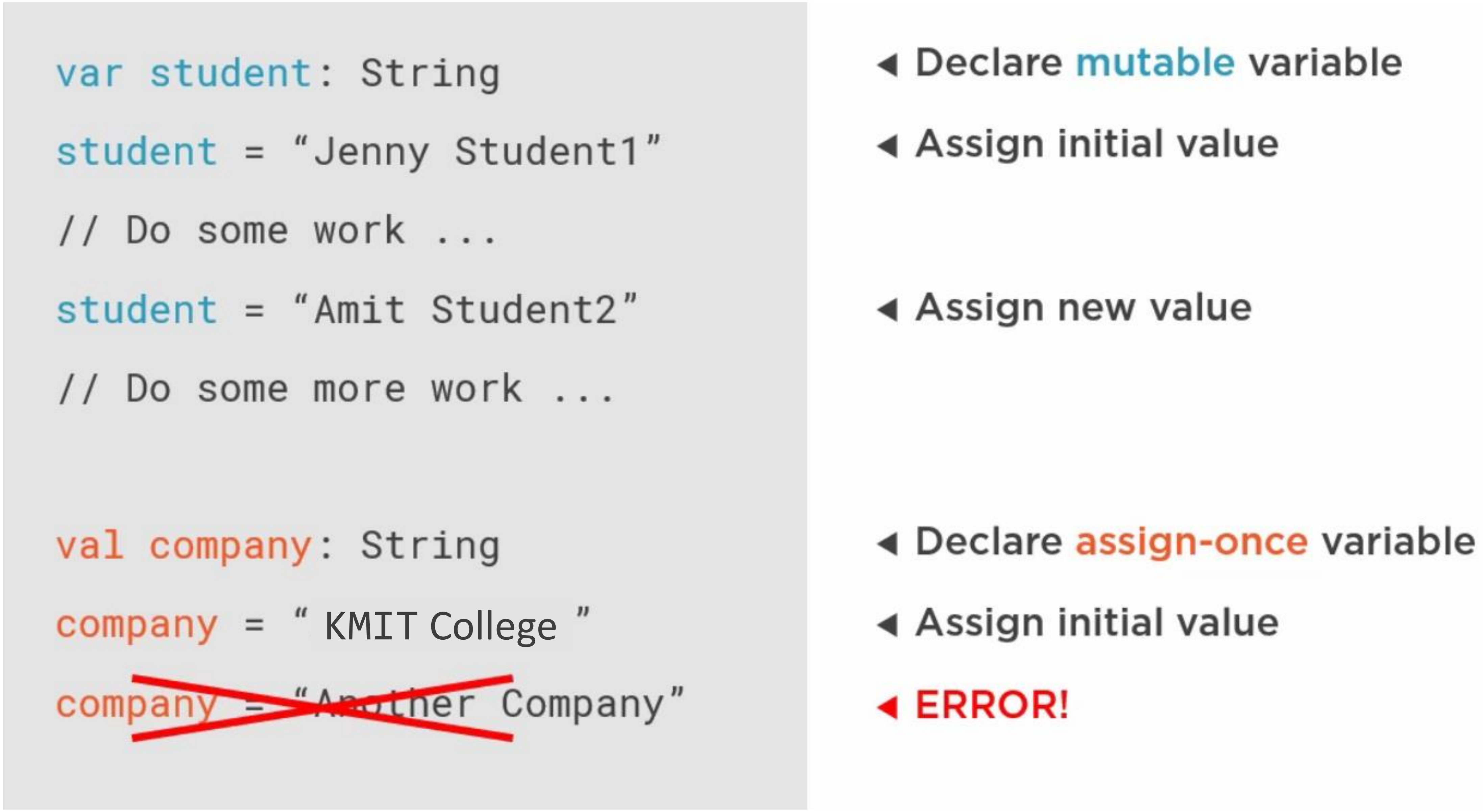
val

Assign-once (read-only) Variable

Value set when assigned

Value cannot be changed once set

Declaring Variables



```
var student: String
student = “Jenny Student1”
// Do some work ...
student = “Amit Student2”
// Do some more work ...

val company: String
company = “ KMIT College ”
~~company = “Another~~ Company”
```

◀ Declare mutable variable

◀ Assign initial value

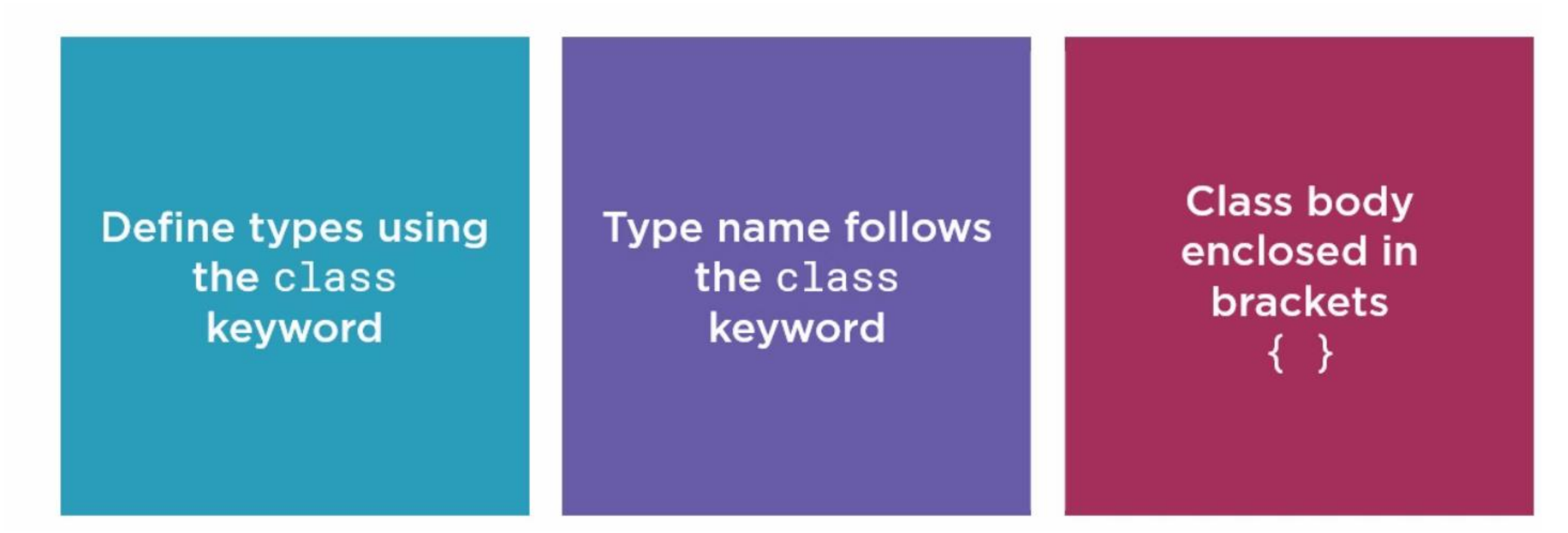
◀ Assign new value

◀ Declare assign-once variable

◀ Assign initial value

◀ ERROR!

Defining Types

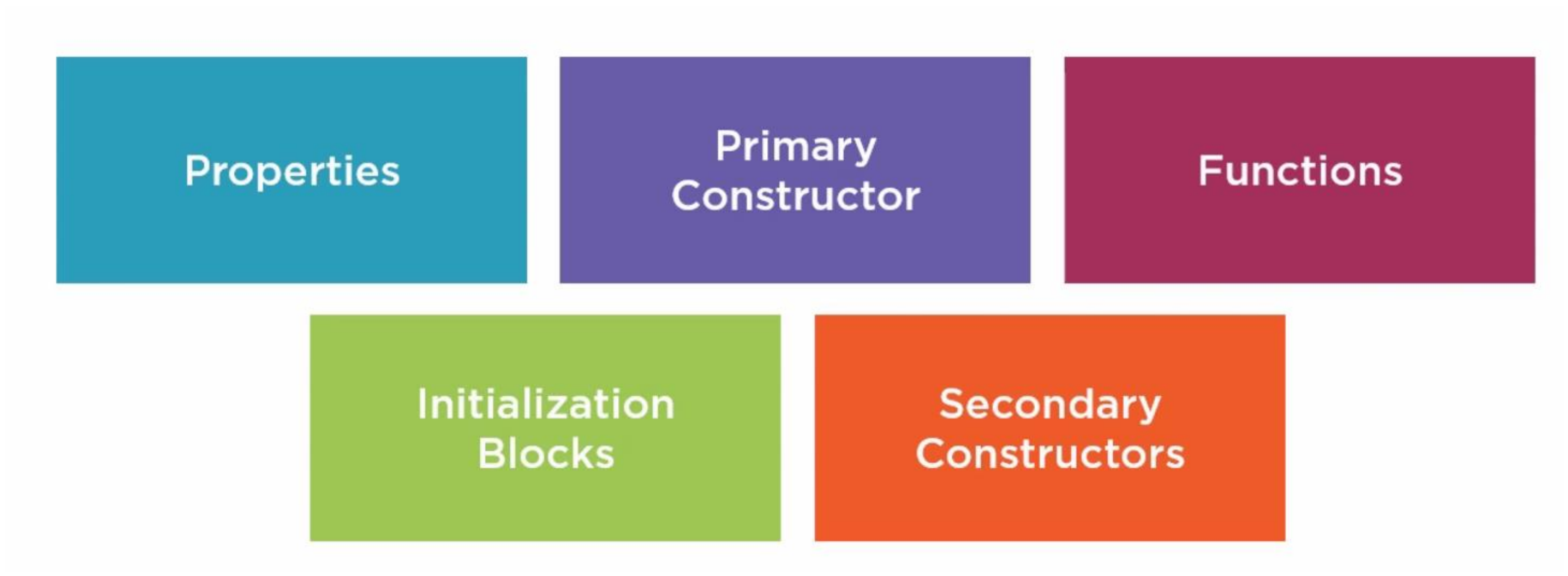


Define types using the `class` keyword

Type name follows the `class` keyword

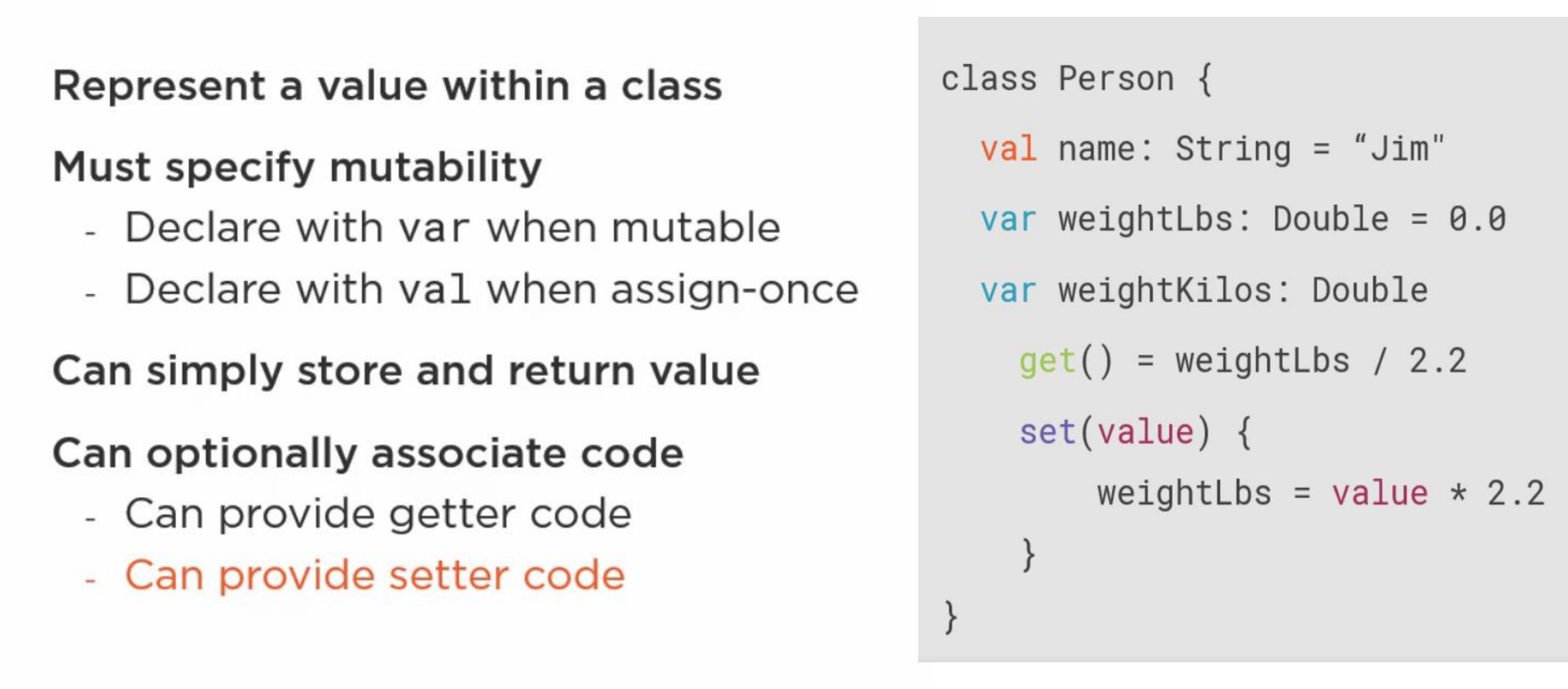
Class body enclosed in brackets
{ }

Class consist



- Properties
- Primary Constructor
- Functions
- Initialization Blocks
- Secondary Constructors

Properties



Represent a value within a class

Must specify mutability

- Declare with var when mutable
- Declare with val when assign-once

Can simply store and return value

Can optionally associate code

- Can provide getter code
- Can provide setter code

```
class Person {
  val name: String = “Jim"
  var weightLbs: Double = 0.0
  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

Properties

```
class Person {
  val name: String = “Jim"
  var weightLbs: Double = 0.0
  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

```
val p = Person()
val name = p.name


p.weightLbs = 220.0
val kilos = p.weightKilos


p.weightKilos = 50.0
val lbs = p.weightLbs
```

◀ Creates new instance of Person

◀ Returns “Jim”

◀ Stores 220.0 in weightLbs

◀ Runs weightKilos getter
Returns 100.0

◀ Runs weightKilos setter

◀ Returns 110.0

Primary Constructor

Accepts list of construction parameters

- Appears after the class name
- Optionally use the constructor keyword
- Parameters used to initialize class
- Contains no code

```
class Person(name: String, weightLbs: Double) {
  val name: String = name
  var weightLbs: Double = weightLbs
  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

Primary Constructor

```
class Person(name: String, weightLbs: Double) {
  val name: String = name
  var weightLbs: Double = weightLbs
  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

```
val p = Person("Bob", 176.0)


val name = p.name
val lbs = p.weightLbs
val kilos = p.weightKilos
```

◀ Creates new instance of Person

◀ Returns "Bob"

◀ Returns 176.0

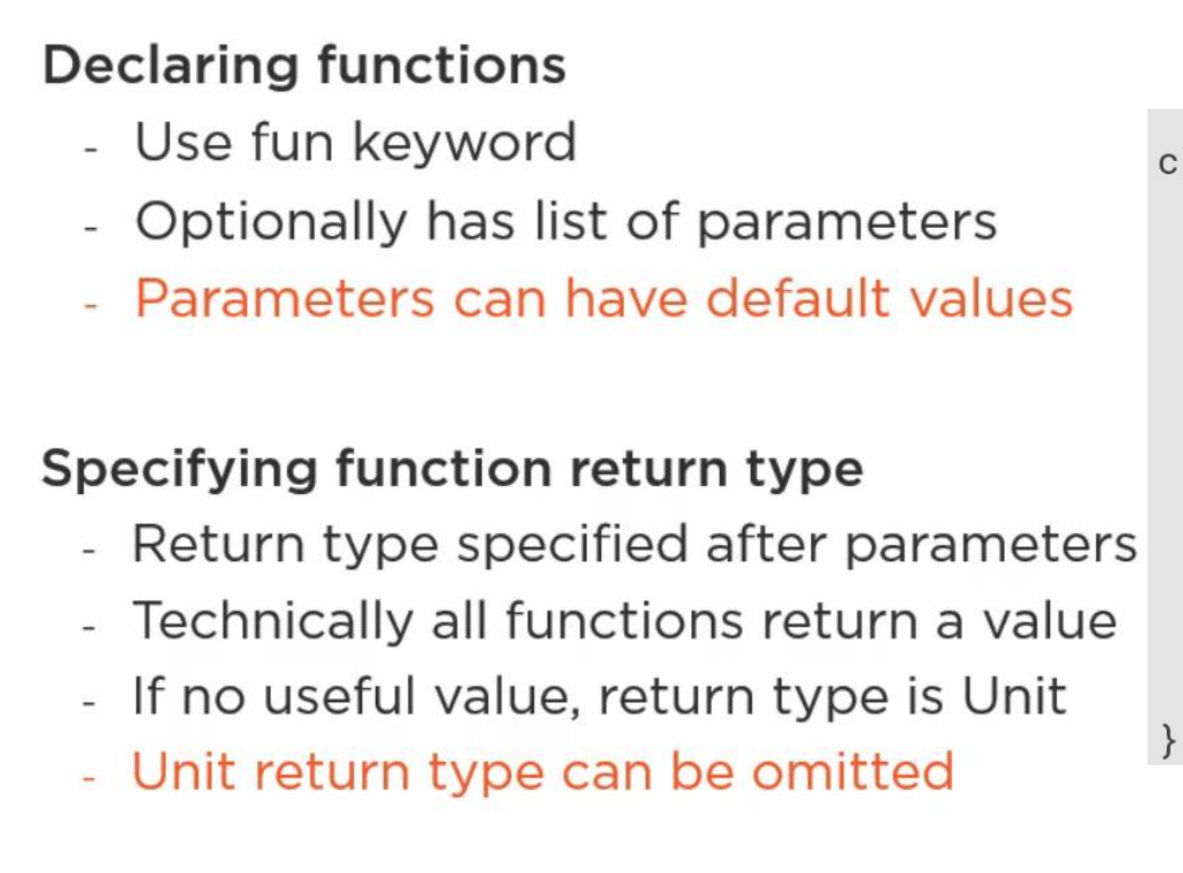
◀ Runs weightKilos getter
Returns 80.0

Primary Constructor declaring properties

```
class Person(name: String, weightLbs: Double) {
  val name: String = name
  var weightLbs: Double = weightLbs
  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

```
class Person(val name: String, var weightLbs: Double) {


  var weightKilos: Double
    get() = weightLbs / 2.2
    set(value) {
        weightLbs = value * 2.2
    }
}
```

Declaring Functions (Methods)



Declaring functions

- Use fun keyword
- Optionally has list of parameters
- Parameters can have default values

Specifying function return type

- Return type specified after parameters
- Technically all functions return a value
- If no useful value, return type is Unit
- Unit return type can be omitted

```
class Person(val name: String, var weightLbs: Double) {
  // weightKilos declaration elided for clarity
  fun eatDessert(addedIceCream: Boolean = true) {
    weightLbs += if (addedIceCream) 4.0 else 2.0
  }
  fun calcGoalWeightLbs(lbsToLose: Double = 10.0): Double {
    return weightLbs - lbsToLose
  }
}
```

Passing Parameter values to function

```
class Person(val name: String, var weightLbs: Double) {
  // weightKilos declaration elided for clarity
  fun eatDessert(addedIceCream: Boolean = true) {
    weightLbs += if (addedIceCream) 4.0 else 2.0
  }
  fun calcGoalWeightLbs(lbsToLose: Double = 10.0): Double {
    return weightLbs - lbsToLose
  }
}
```

```
val p = Person("Bob", 176.0)             ◄ Creates new instance of Person

p.eatDessert(false)                      ◄ addedIceCream passed as false
val lbs = p.weightLbs                    ◄ Returns 178.0

p.eatDessert()                           ◄ addedIceCream passed as
                                                      default (true)
lbs = p.weightLbs                        ◄ Returns 182.0

val gw = p.calcGoalWeightLbs()           ◄ lbsToLose passed as default (10)
                                            Returns 172
```

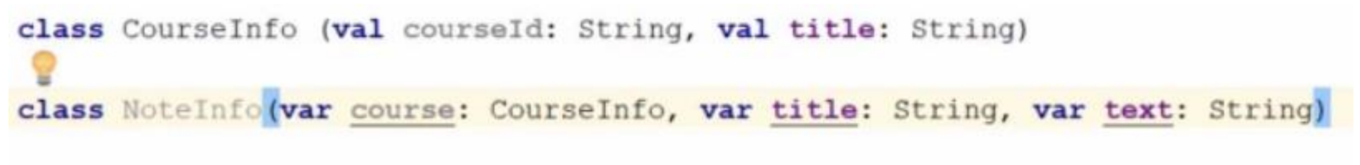
```
val p1 = Person("Jim", 185.0)

val p2 = Person(weightLbs = 185.0, name = "Jim")
```

Initialization Block and Secondary Constructor

Initializer block	Secondary constructor
Always runs during construction	Runs only when used
Can have multiple if desired	Can have multiple if desired
All will run every time	Runs when used to construct instance
	Code runs after all initializer blocks
	Must delegate to primary constructor if type includes one

Exercise – 7_1

Example : Create a program in kotlin to help Bob, what food should he feed to fishes in the aquarium on particular day and does he need to change the water



```
class CourseInfo (val courseId: String, val title: String)

class NoteInfo(var course: CourseInfo, var title: String, var text: String)
```