Kotlin Part 1

Upon completion of this module, a student will be able to

- write Kotlin functions
- create Kotlin variables
- use Kotlin String interpolation
- use conditionals in Kotlin
- create Kotlin classes
- use Kotlin constants

https://www.udacity.com/course/kotlin-bootcamp-for-programmers--ud9011

Assignment

- Task
  - Build classes for a SnackBar Command line app
- Repo
  - https://github.com/LambdaSchool/Android_KotlinSnackBar
- Submission
  - Fork on github and submit pull request

A Student Can

create Kotlin functions

Functions

- Simple Method
- Return Value
- Parameters
- Default Parameters
  - Named Values

```kotlin
// Simple Method
fun main() {
    // This method also happens to be the equivalent of the Java main method
    ...
}

// Method with return value
fun randomDay(): String {
    ...
    return "Today"
}

// Can return an expression when a function is a single line
fun randomDay() = "Today"

// Method which accepts a value and returns a method
fun fishFood(day: String): String {
    println(day)
    return day
}

// Method which accepts a value or uses a default value and returns a method
fun getFortuneCookie(value: Int = 3): String {
    return whatShouldIDoToday("happy", temp = 3) // named value
}

// Method with multiple default values that can be passed as named values
fun whatShouldIDoToday(mood: String, weather: String = "sunny", temp: Int = 24) : String {
    ...
    return todaysTask
}
```

https://kotlinlang.org/docs/reference/basic-syntax.html

Challenge

- Write a function which accepts 2 Int values, multiplies them together, and returns the value
- Give the second int a default value
- Test with passing 1 and 2 values

3 Minutes

Solution

```
fun functionPractice() {
    println(multiply(3, 7))
    println(multiply(3))
}

fun multiply(x: Int, y: Int = 1): Int {
    return x.times(y)
}
```

A Student Can

Kotlin string interpolations

String Injections

- $ - injects variable into string
- ${} - injects expression into string
  - Method call
  - Statement
  - Expression

```
override fun toString(): String {
    return "Name: $name Year: $year ${move()} ${breath()} ${reproduce()}"
}
```

https://kotlinlang.org/docs/reference/basic-syntax.html

Challenge

- Write a function which accepts an Int value and returns the string “My Int value is “ followed by the value

2 Minutes

Solution

```
println(writeIntValue(113))



fun writeIntValue(num: Int): String {
    return "My Int value is $num"
}
```

A Student Can

work with Kotlin variables

Variables

- val – final variable
- var – mutable variable
- Type Inference

```
// types can be declared when variable is initialized
val variableWithType: String
variableWithType = "Hello World"

// types do not need to be declared due to type inference
val finalVariable = 5
// variables declared with val can't be reassigned, they are final
finalVariable = 7

var variable = 8
// variables declared with var can be reassigned
variable = 10
// type is inferred upon declaration and can't be replaced
variable = "15"
```

https://kotlinlang.org/docs/tutorials/kotlin-for-py/declaring-variables.html

Primitives as Objects

- Can call methods on primitives

```
val number = 5.times(10)

val floatPlus = 1.23.plus(3)

val div = 30.div(50)
```

https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-number/index.html

Challenge

- Write a method to declare a variable and give it a value
- Divide that value by a number, store and print the result
- Convert the value to a String and store it

3 Minutes

Solution

```
fun variablePractice() {
    val var1 = 5

    val var2 = var1 / 5
    println(var2)

    val var3 = var2.toString()
}
```

A Student Can

conditionals

If Else Expression

- Single Line
- As Expression

```
// Traditional usage
var max = a
if (a < b) max = b

// With else
var max2 = if (a > b) {
    a
} else {
    b
}

var max3: Int
if (a > b) {
    max3 = a
} else {
    max3 = b
}

// As expression
val max4 = if (a > b) a else b
```

https://kotlinlang.org/docs/reference/control-flow.html

When Expression

- Switch statement
- Value
- Expressions (if, else if)

```
sealed class Spice(
    val name: String,
    private val spiciness: String = "mild",
    color: SpiceColor = YellowSpiceColor) :
    SpiceColor by color {
    val heat: Int
        get() {
            return when (spiciness) {
                "mild" -> 1
                "medium" -> 2
                "spicy" -> 3
                "very spicy" -> 4
                "extremely spicy" -> 5
                else -> 0
            }
        }

    abstract fun prepareSpice()
}

fun dayOfWeek() {
    println("What day is it today?")
    when (Calendar.getInstance().get(Calendar.DAY_OF_WEEK)) {
        1 -> println("Sunday")
        2 -> println("Monday")
        3 -> println("Tuesday")
        4 -> println("Wednesday")
        5 -> println("Thursday")
    }
}

when {
    x == 0 -> print("x is odd")
    x > 1 -> print("x is even")
    else -> print("x is funny")
}
```

https://kotlinlang.org/docs/reference/control-flow.html

Challenge

- Write a when statement that will accept an int rating 1-5 and convert it to a string representation of your rating (ie. hated it, loved it)

- STRETCH: Adapt the statement to accept a float and adapt it to a 1-5 rating

3 Minutes

Solution

```
convertRating(5f)

fun convertRating(rating: Float): String {
    return when(rating) {
        1 -> "Hated It"
        2 -> "Didn't Like it"
        3 -> "Liked It"
        4 -> "Really Liked It"
        5 -> "Loved It"
        else -> "unknown"
    }

    /*** OR ***/

    return when {
        rating <= 1                -> "Hated It"
        rating > 1 && rating <= 2 -> "Didn't Like it"
        rating > 2 && rating <= 3 -> "Liked It"
        rating > 3 && rating <= 4 -> "Really Liked It"
        rating > 4 && rating <= 5 -> "Loved It"
        else -> "unknown"
    }
}
```

A Student Can

Classes

Basic Class

- Constructor
- Data Members
- New

```
class Food(val name: String = "Curry", val spiciness: String = "mild")

fun getFood() {
    val steak = Food(spiciness = "None", name = "Steak")
    val food = Food()

    println(steak.name)
    println(steak.spiciness)
    println(food)
}
```

https://kotlinlang.org/docs/reference/classes.html

Getters and Setters

- Included with properties by default
- Can provide different implementations

```
class Food(val name: String = "Curry", val spiciness: String = "mild") {

    var numServings: Int = 1

    var servingSize: Int = 1
        set(value) {
            field = value
        }

    val quantity: Int
        get() = numServings * servingSize

}

fun getFood() {
    val steak = Food(spiciness = "None", name = "Steak")
    val food = Food()

    println(steak.name)
    println(steak.spiciness)
    println(food.quantity)
    println(food.servingSize)
}
```

https://kotlinlang.org/docs/reference/classes.html

Member Methods

- Regular functions
- Called as other functions

```
class Food(val name: String = "Curry", var spiciness: String = "mild") {
    var cooked = false

    fun cook() {
        cooked = true
    }

    fun addSpice(spice: String) {
        spiciness = "mild"
    }
}
```

https://kotlinlang.org/docs/reference/classes.html

Init and Secondary Constructor

- Init Block**s**
- Secondary Constructor

```
class Meal {
    private val plate = mutableListOf<Food>()

    fun addFood(food: Food) {
        plate.add(food)
    }
}

class Food(val name: String = "Curry", var spiciness: String = "mild") {
    var cooked = false

    init {
        cooked = true
    }

    constructor(meal: Meal, name: String, spiciness: String) :
            this(name, spiciness) {
        meal.addFood(this)
    }

    fun addSpice(spice: String) {
        spiciness = "mild"
    }
}

fun getFood() {
    val dinner = Meal()

    val steak = Food(spiciness = "None", name = "Steak")
    val food = Food(dinner, name = "Hot Wings", spiciness = "Hot")
}
```

https://kotlinlang.org/docs/reference/classes.html

Challenge

- Create a Vehicle class
  - Parameters: name, number of wheels, number of doors, number of seats
  - Methods:
    - Drive method, no parameters
    - Open door method, accept parameter for door number default to 1

5 Minutes

Solution

```
fun classCaller() {
    val vehicle = Vehicle("Coupe", 4, 2, 2)
    println(vehicle.name)
    vehicle.drive()
    vehicle.openDoor()
    vehicle.openDoor(2)
}

class Vehicle(val name: String, var wheels: Int, val doors: Int, var seats: Int) {
    fun drive() {
        println("Driving on $wheels wheels.")
    }

    fun openDoor(doorNum: Int = 1) {
        println("Opening door number $doorNum")
    }
}
```

Advanced Solution

```
class Vehicle(val name: String, var wheels: Int, val doors: Int, seats: Int) {

    var color: String = "Red"
        set(value) {
            println("setting seats to $value")
            field = value
        }

    val seats: Int = seats

    /*var seats: Int = seats
        get() = this.seats
        private set*/


    fun drive() {
        println("Driving on $wheels wheels.")
    }

    fun openDoor(doorNum: Int = 1) {
        println("Opening door number $doorNum")
    }
}
```

```
fun classCaller() {
    val vehicle = Vehicle("Coupe", 4, 2, 2)
    println(vehicle.name)
    vehicle.drive()
    vehicle.openDoor()
    vehicle.openDoor(2)

    println(vehicle.color)
    vehicle.color = "Blue"

    println(vehicle.seats)

    vehicle.seats = 3
}
```

A Student Can

Constants

Constants

- Const
- Class members can’t be constant
  - Companion object

```
const val MAX_BORROWED = 3 // top level constant

open class Book(val title: String, val author: String, val year: Int = 2019) {

    companion object { // class level constant
        const val MAX_BORROW = 5
        const val BASE_URL = "google.com"
    }

    private var currentPage = 1
    private var numBorrowed = 1

    ...

    fun canBorrow(): Boolean {
        return numBorrowed < MAX_BORROW
    }
}
```

https://kotlinlang.org/docs/reference/properties.html#compile-time-constants