

# CS131 - Week 9

---

UCLA Winter 2019

TA: Kimmo Karkkainen

# Announcements

- Project deadline extended - DL next Tuesday (03/12)
- No late submissions after Friday 03/15 for either project/HW6

# Today

- Homework 6
- Machine Learning & TensorFlow
- Kotlin basics & more interesting features

# Homework 6

---

# Homework 6

- Due **Friday 3/15**
  - No late submissions due to finals week
- A small coding warm-up + a report

# Homework 6

- Assume you've built a Python server herd like in your project
- This time, the servers do some machine learning task with TensorFlow
- With a large number of queries, the performance is too slow
  - I.e. Your Python code is the bottleneck
  - Can we improve this by using some other language?
- Compare three alternatives: **Java**, **Ocaml**, **Kotlin**
  - Compare these to each others and Python
- No need to write any prototype code
  - You can include code if you want to, but it is not expected

# Report

- 3-page executive summary
  - Write so that it can be understood by someone who doesn't know the specific languages
- What are the challenges when rewriting the server herd on the other languages? Are they any better than Python?
  - If there's e.g. a performance difference, discuss why that is
- Focus on “ease of use, flexibility, generality, performance, reliability”
- Citations!
  - E.g. if you found a nice comparison on performance differences between languages, add a citation to support your claims

# Kotlin Warm-up Problem

- Write a function *everyNth(L, N)*
  - Returns a new list that contains every Nth element of list L
  - E.g. List=[1,2,3,4,5,6], N=2 => [2,4,6]
- Write a Makefile
  - Compiles and tests your code with “*make check*”
  - Should work on SEASnet servers



# Machine Learning & TensorFlow

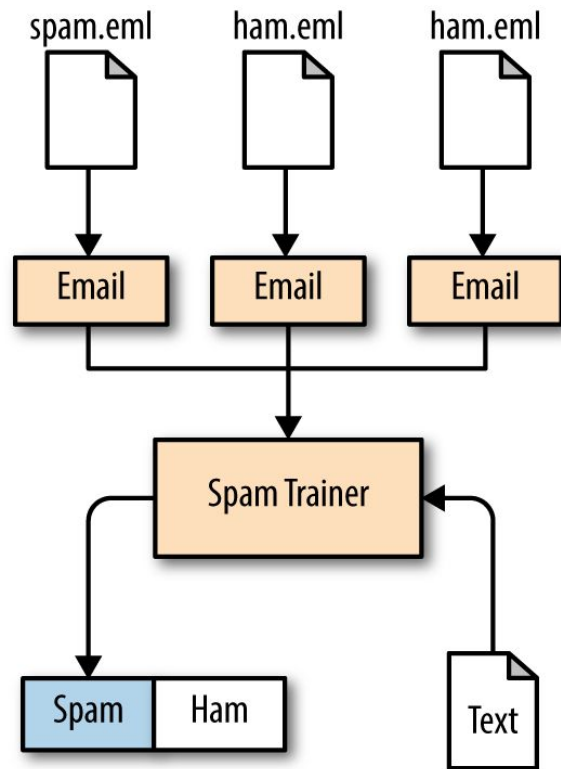
---

# Machine learning

- Use of algorithms and mathematical models to solve a problem
  - Goal is for the system to improve its performance as it “learns”
  - Compare with your typical programs, where you explicitly define what should happen in each situation
- Three main categories:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

# Machine Learning - Supervised Learning

- System tries to learn from examples given to it
- For example, a spam classifier can learn which words are more likely to occur in spam messages than in non-spam messages



# Machine Learning - Supervised Learning

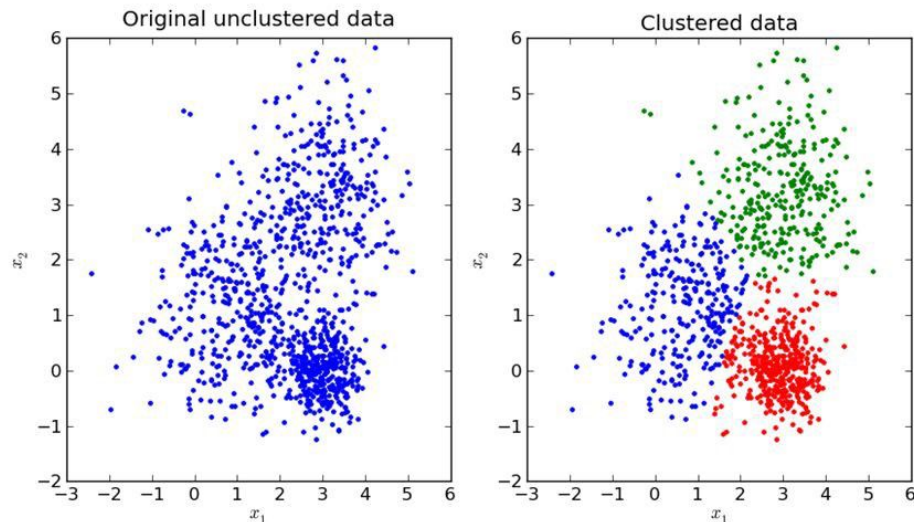


# Machine Learning - Supervised Learning



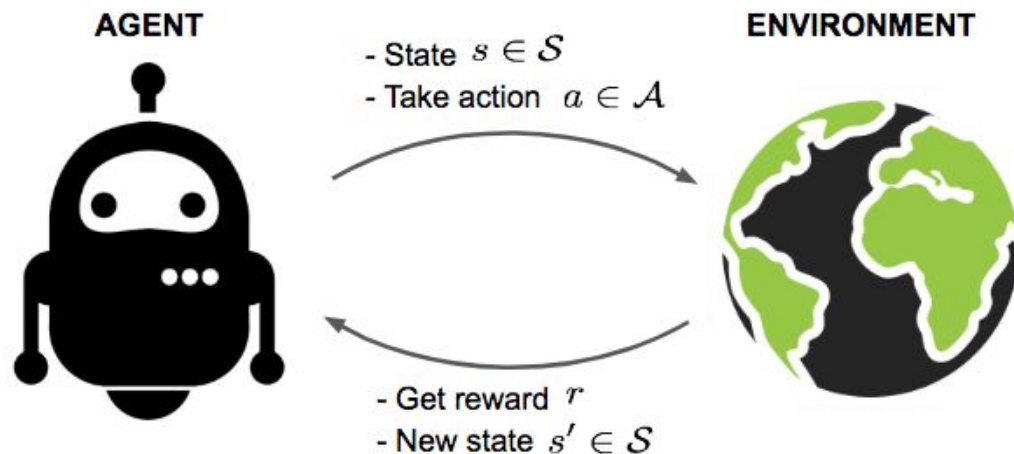
# Machine Learning - Unsupervised Learning

- System tries to find structure from data
- For example, which users are likely to enjoy a specific movie on Netflix?
  - Find groups of users and see which groups have lots of people enjoying that movie
- Is a credit card transaction typical for that user, or possibly a fraud? Detect outliers



# Machine Learning - Reinforcement Learning

- Try to perform different actions and see what the outcome is
  - Learn from the successes/mistakes
  - For example, a program that can play computer games without having rules explicitly coded into it



# Machine Learning - Phases

- Training phase
  - Preprocess data, extract features
  - Give features and labels to some learning algorithm
  - Algorithm produces a model (typically very slow)
- Testing / Prediction phase
  - Preprocess data, extract features (exactly the same way as in the training phase)
  - Give features to the model
  - Model gives predictions (typically quite fast)

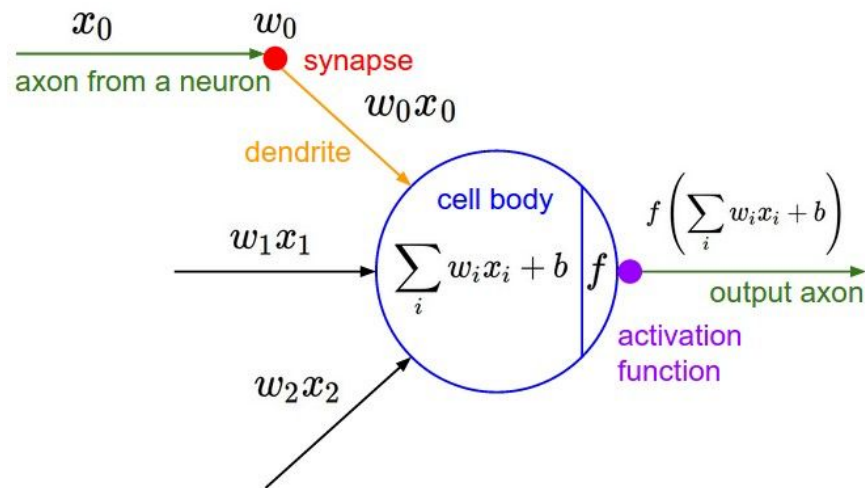
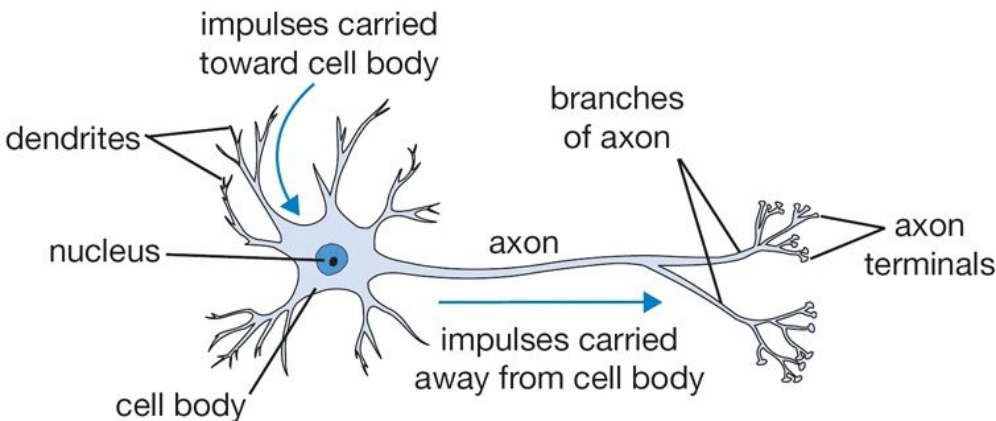


# TensorFlow

- Library for symbolic mathematics and neural networks
- Initially meant for Google's internal use, released for the public 2015
- Easy to define neural networks, TensorFlow takes care of running it efficiently on multiple CPUs or GPUs

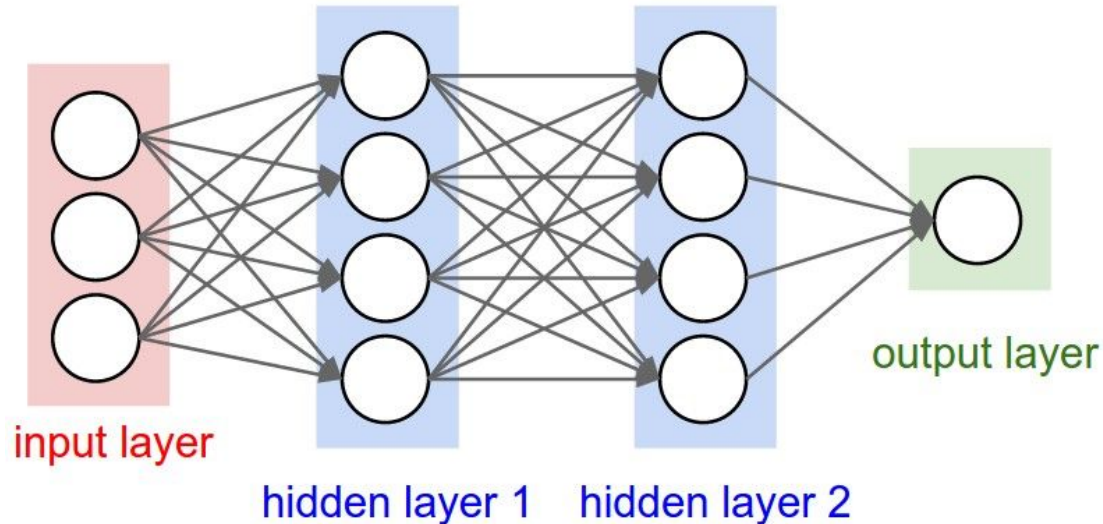
# Neural networks

- Powerful machine learning approach, has been shown to have great performance on tasks such as image recognition or signal processing
- Challenges: computationally heavy, difficult to reason about



# Neural networks

- Neural networks consist of a large number of neurons, each performing a simple mathematical operation on its input values



# TensorFlow Example

- Example

# TensorFlow - Minimal Example

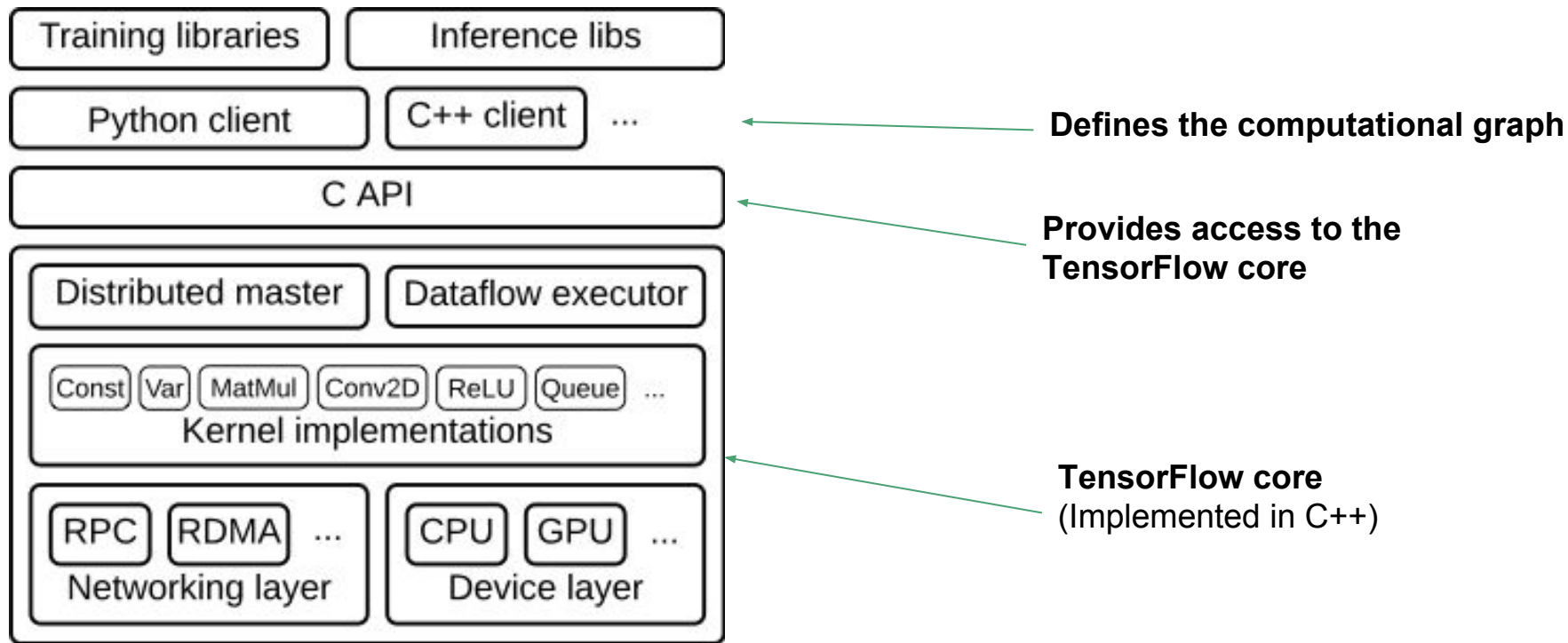
```
# Define network structure
model = Sequential()
model.add(Dense(4, input_dim=8, activation='relu'))
model.add(Dense(2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Define parameters that are needed to learn the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Learn the parameters
model.fit(X, Y, epochs=150, batch_size=10)

# Make predictions
predictions = model.predict(X)
```

# TensorFlow Architecture (see [documentation](#))



# TensorFlow Bindings

- Bindings allow you to call another language's code
  - Usually high-level languages (Python, Java, ...) call low-level libraries (C, C++, ...)
    - Plenty of libraries available
    - Better performance
  - Challenges
    - Different languages have different data types -> Might need to convert
    - Memory management
- For details on how bindings work, see e.g. [Python CFFI documentation](#)
- TensorFlow's focus has been on Python, but there are bindings for other languages as well
  - JavaScript, C++, Java, Go, Swift, C#, Haskell, Julia, Ruby, Rust, OCaml, Scala, ...
  - Not all features are available for all languages...
  - Some of these provided by TensorFlow team, some not

# TensorFlow & Python

- TensorFlow handles all the heavy mathematical operations etc
  - Can use multiple CPUs/GPUs to make it very fast
- Python needs to preprocess the data to a format that TensorFlow can handle
- Python is used to define the computational graphs etc
- In your project, assume that TensorFlow is not the bottleneck



# Kotlin

---

# History of Kotlin

- Released 2011 by JetBrains (developer of IntelliJ IDEA)
- Designed to be “better than Java” while still interoperable with existing Java code
  - Easy migration from old Java code
- Originally designed for Java Virtual Machine (JVM)
  - Like Java, Scala, Clojure, Groovy, ...
  - Language designers get all the benefits of JVM without having to put any effort into it!
    - Garbage collection
    - JIT compilation
    - ...

# Design principles

- Conciseness
  - Taking inspiration from functional programming
- Safety
  - Collections are immutable by default (e.g. *List* vs *MutableList*)
  - Immutable data types preferred in general (*val* vs *var*)
  - Variable can contain null only if it defined to be nullable (more on this later)
  - Static typing -> Many errors caught at compile-time
- Interoperability
  - Using existing libraries (e.g. Java libraries)

# Who uses Kotlin?

## Pinterest

Pinterest has successfully [introduced Kotlin](#) into their application, used by 150M people every month.

## Gradle

Gradle is [introducing Kotlin](#) as a language for writing build scripts.

## Evernote

Evernote recently [integrated Kotlin](#) into their Android client

## Uber

Uber team [uses Kotlin](#) for building internal tools

## Corda

is an open-source distributed ledger platform, supported by major banks, and [built entirely in Kotlin](#).

## Coursera

Coursera Android app is [partially written](#) in Kotlin

## Pivotal

Spring makes [use of Kotlin's language features](#) to offer more concise APIs

## Atlassian

All new code in the [Trello Android](#) app is in Kotlin.

# Kotlin Uses

- Android
- Server-side
- Native
- JavaScript

# Uses - Android Development

- Android used old version of Java for a long time -> Kotlin provided a nicer alternative for developers
  - Compiles like Java code, so could be used on Android even without Google's support
- Google made it one of the official Android languages in 2017
  - Android Studio has a full support for it, in addition to Java and C++

# Uses - Server-side Development

- Can be deployed anywhere where running Java is possible
  - AWS, Google Cloud Platform, Heroku, ...
- Can use existing Java libraries
- Many server-side libraries support Kotlin specifically
  - Easier development

# Uses - Native Development (Kotlin/Native)

- Sometimes running JVM is not possible -> compile native code instead
- Kotlin/Native can be compiled to following architectures:
  - iOS
  - MacOS
  - Android
  - Windows
  - Linux
  - WebAssembly
- Kotlin/Native can use C libraries and Swift / Objective-C frameworks



# Uses - JavaScript Development

- Transpiling = Compiling source code into another language's source code
- Your Kotlin code and Kotlin standard library can be transpiled into JavaScript
  - Excludes every other Java library!
- Client-side JavaScript
  - Interacting with DOM elements (your webpage's elements)
  - Creating graphical elements using WebGL
- Server-side JavaScript
  - Can interact with e.g. Node.js

# Installation

- Included in IntelliJ IDEA, Android Studio
- Available as a plugin for Eclipse
- Can install the compiler and use any text editor
  - See <http://kotlinlang.org> for instructions
  - Already on SEASnet Linux servers

# Hello, World!

```
fun main(args: Array<String>) {  
    val scope = "World"  
    println("Hello, $scope!")  
}
```

- Compare to Java:
  - No class needed
  - No semi-colons
  - Type after argument name
  - Variable inside string
  - val => immutable variable

# Compilation / Interactive Shell

```
$ kotlinc hello.kt -include-runtime -d hello.jar
```

```
$ java -jar hello.jar
```

**Hello, World!**

```
$ kotlinc-jvm
```

Welcome to Kotlin version 1.3.21 (JRE 1.8.0\_151-b12)

Type :help for help, :quit for quit

```
>>> println("Hello, World!")
```

**Hello, World!**

# Variables

- Type inference
- *val* defines immutable variables, *var* defines mutable variables

```
>>> val a = 5
```

```
>>> a
```

```
res2: kotlin.Int = 5
```

```
>>> a = 6
```

```
error: val cannot be reassigned
```

```
>>> var b = 5
```

```
>>> b = 6
```

```
>>> b
```

```
res5: kotlin.Int = 6
```

# Functions

- Return type comes **after** function name and arguments
  - Can be omitted if the type is *Unit* (similar to *void*)
- No need for curly braces if we only have one expression:

```
fun foo(): Int {  
    <Do multiple things here>  
    return 1  
}
```

```
fun len(x: String) = x.length
```

# Lambda functions

- Lambda functions defined with curly braces: *{ arg : argType -> expression }*

```
>>> { s: String -> println(s) } ("Hello")  
Hello
```

```
fun executeLambda(f: (s: String) -> Unit) {  
    f("Hello")  
}
```

```
>>> executeLambda({ s: String -> println(s) })  
Hello
```

# Lists & List Operations

```
>>> val myList = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
>>> myList.map { x -> x*2 }
res3: kotlin.collections.List<kotlin.Int> = [2, 4, 6, 8, 10]
```

```
>>> myList.map { it*2 }
res4: kotlin.collections.List<kotlin.Int> = [2, 4, 6, 8, 10]
```

```
>>> myList.filter { it < 5 }
res5: kotlin.collections.List<kotlin.Int> = [1, 2, 3, 4]
```

- By convention, function calls with a lambda argument do not use parentheses
- List iteration can directly use iterator (*it*) without declaring it as a parameter



# List Indices

```
>>> myList.withIndex().foreach { (index, value) -> println("$index: $value") }
```

**0: 1**

**1: 2**

**2: 3**

**3: 4**

**4: 5**

```
>>> myList.forEachIndexed { index, value -> println("$index: $value") }
```

**0: 1**

**1: 2**

**2: 3**

**3: 4**

**4: 5**

# Data Classes

- Meant for classes whose main purpose is to store data
  - Creates getters/setters automatically, provides equals()/copy()/toString() methods

```
data class Person(val name: String, val age: Int) {  
    fun printName() { println(name) }  
}
```

```
>>> val person = Person("John", 53)  
>>> person.toString()  
res50: kotlin.String = Person(name=John, age=53)  
>>> person.name  
res51: kotlin.String = John  
>>> person.printName()  
John
```

# Extension Methods

- Ability to add new functionality to existing classes without inheritance

```
package MyStringExtensions  
  
fun String.lastChar(): Char = get(length - 1)  
  
>>> println("Kotlin".lastChar())  
n
```

# Coroutines

- Similar to Python's *async/await*

```
import kotlinx.coroutines.*

fun main() = runBlocking {
    launch { doWorld() }
    println("Hello,")
}

suspend fun doWorld() {
    delay(1000L)
    println("World!")
}
```

# Nullable Data Types

- Nullable values must be defined with ?
- Safe navigation operator: ?.
  - Access a method/field only if the variable is not null, otherwise return null
- Null coalescing operator: ?:
  - Also known as Elvis Operator
  - Return the nullable value if it is not null, otherwise return an alternative value

```
>>> val name: String? = null
```

```
>>> name?.length
```

```
res5: kotlin.Int? = null
```

```
>>> val my_var = name ?: "Name was null"
```

```
>>> my_var
```

```
res9: kotlin.String = Name was null
```

# Safe casting

- Typical way of casting variables will throw an exception if the types are not compatible
- Safe casting will set the result variable null instead

```
>>> val y = 5
>>> val x: String = y as String
java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String
>>> val x: String? = y as? String
>>> x
res10: kotlin.String? = null
```

# Smart Casting

- If we check the type of a variable, it will cast itself to the desired type:

```
fun len(obj: Any): Int? {  
    if (obj is String) {  
        return obj.length  
    }  
  
    return null  
}
```

```
>>> len(5)  
res18: kotlin.Int? = null  
>>> len("Hello")  
res19: kotlin.Int? = 5
```

```
fun len(obj: Any): Int? {  
    if (obj !is String) return null  
  
    return obj.length  
}
```

```
>>> len(5)  
res25: kotlin.Int? = null  
>>> len("Hello")  
res26: kotlin.Int? = 5
```

# Resources

- [Kotlin Playground](#)
- [Kotlin Koans](#)
- [Kotlin Reference](#)



# Reminders

- Course feedback is open
  - Please fill it, shouldn't take more than a few minutes
  - Did we not cover something that you wish was covered? Should something be taught differently? Do you think some other language should be included in this course?
- All homeworks must be submitted by next **Friday 11:55pm**
  - **LATE SUBMISSIONS WILL NOT BE GRADED**

Questions?

---