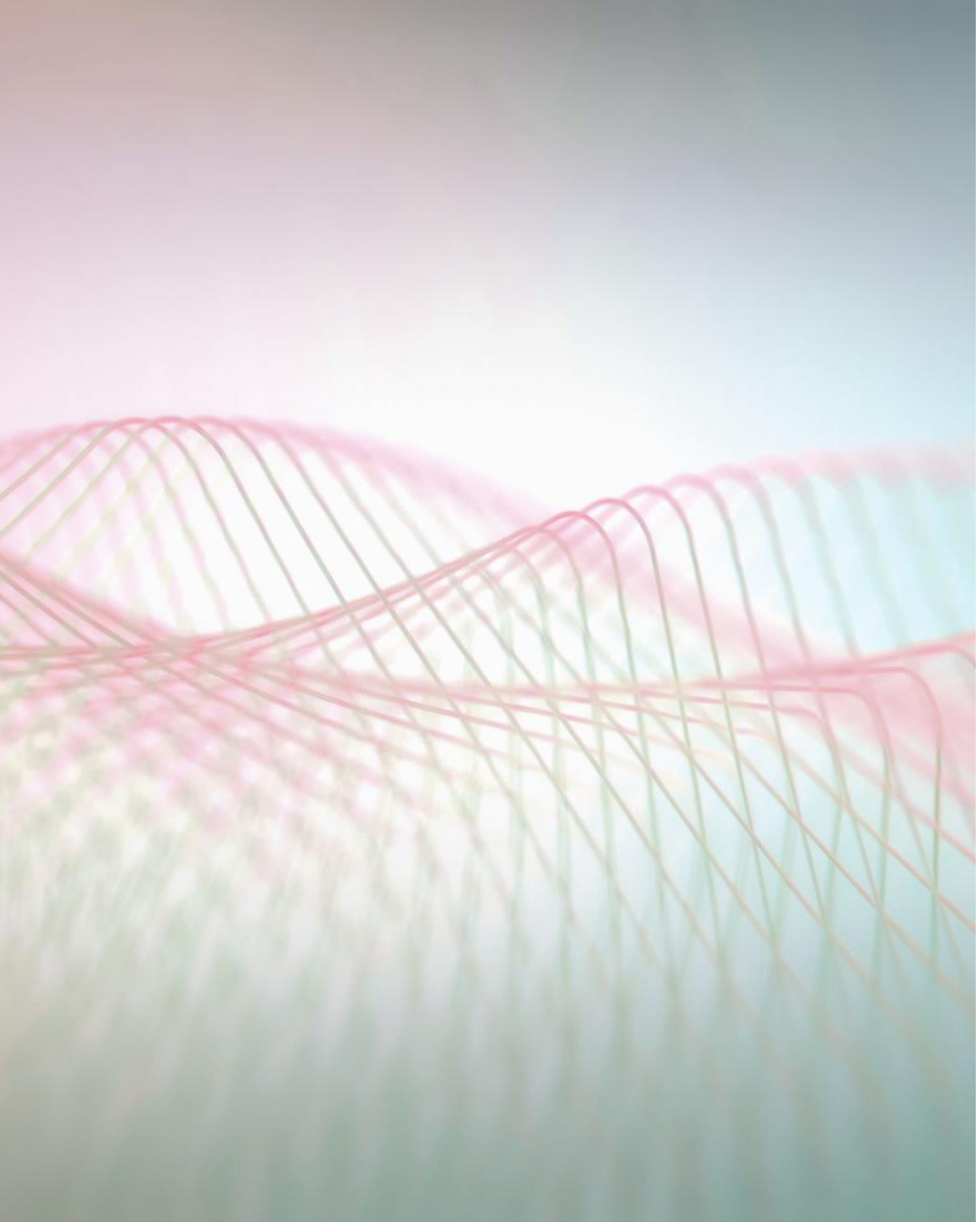


Kotlin

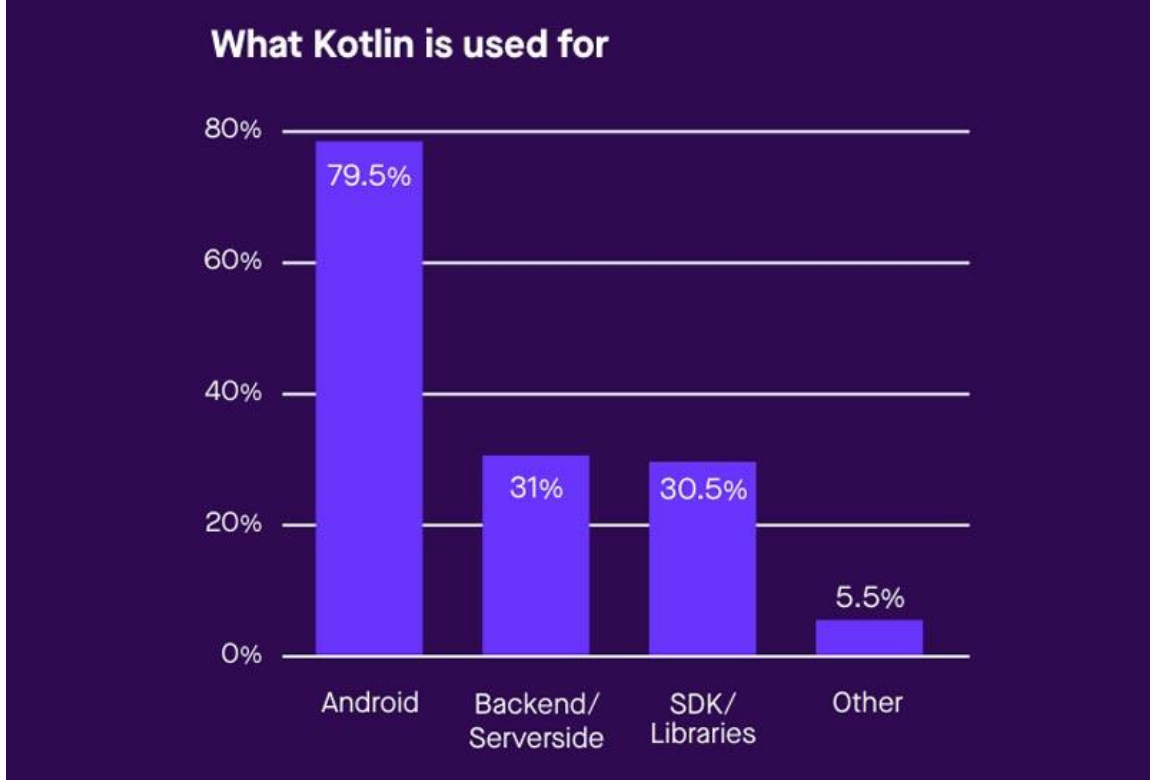
Introduzione al linguaggio Kotlin

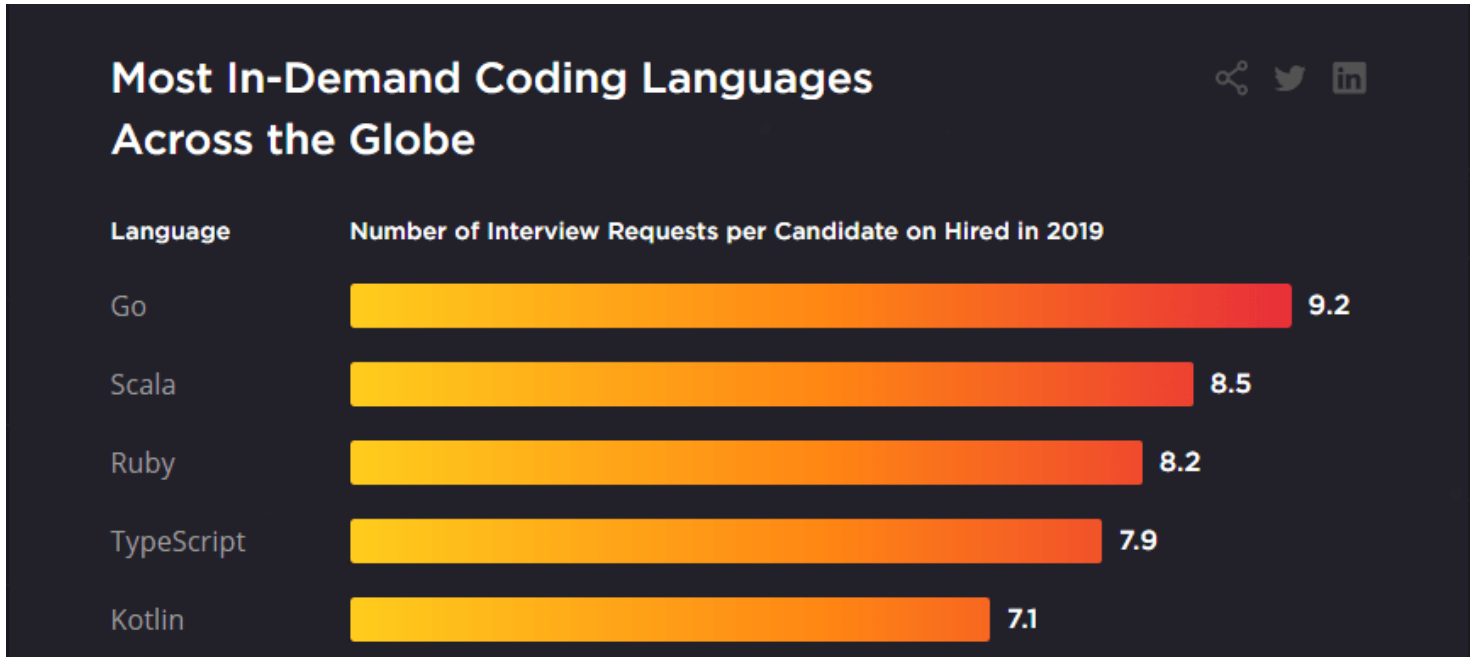
JAVA VS KOTLIN

WHICH IS BEST!?

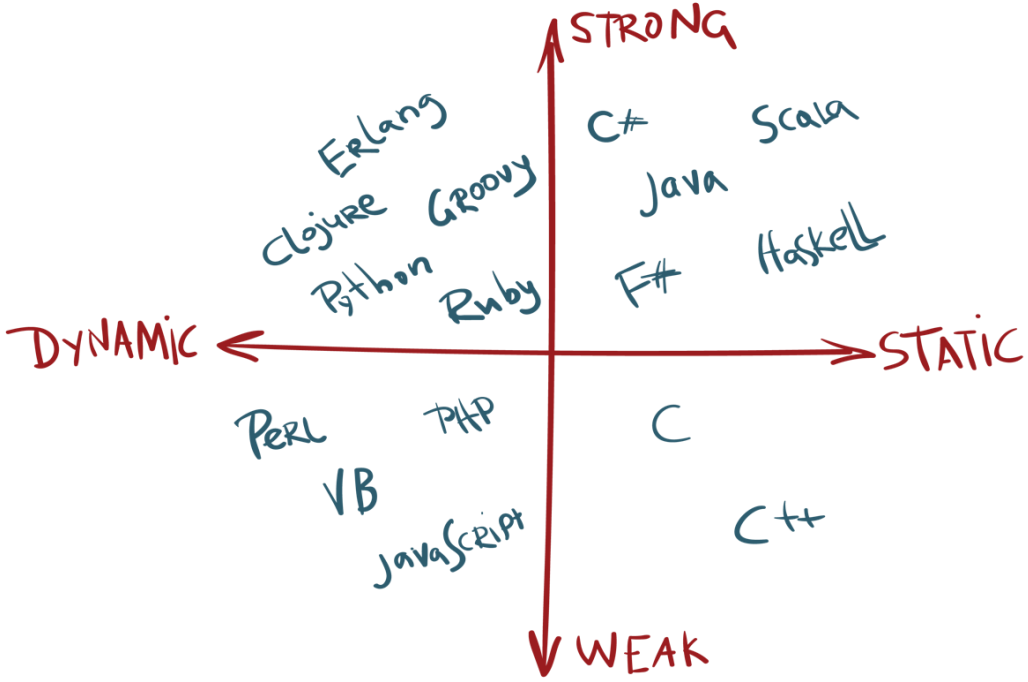
Some statistics

- According to the Hired rating, **Kotlin** is one of the five most popular programming languages in the world.
- It beat competitors such as Java and Python, giving in only to Go and Scala in its field.





What is Kotlin?



- **Kotlin** is a statically typed programming language that targets the JVM, Android, JavaScript, and Native language.
- It is developed by JetBrains.
- The project started in 2010 and was open source from its initial days (Apache 2 license).
- The first official 1.0 release was in February 2016.
- Kotlin does not aim to be unique but it draws inspiration from decades of language development.
- It exists in variants that target the JVM (Kotlin/JVM), JavaScript (Kotlin/JS), and Native code (Kotlin/Native LLVM).

Install kotlin

- How to install the Kotlin compiler https://kotlinlang.org/docs/tutorials/command-line.html

- Try running all the examples on the page

First example

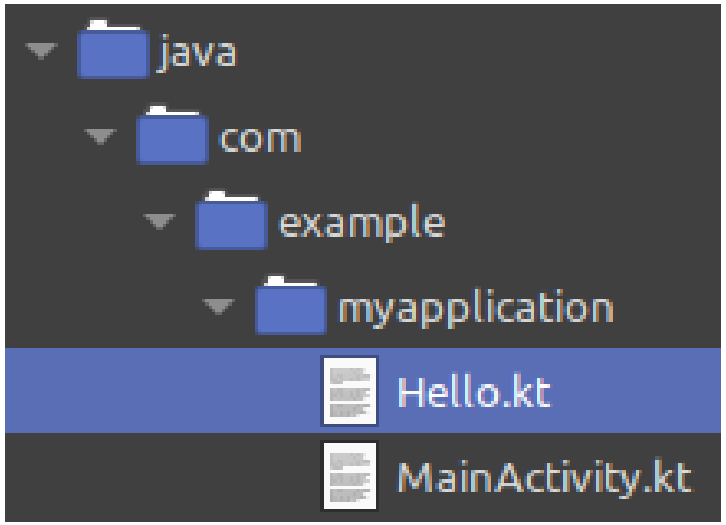
kotlinc com/example/myapplication/Hello.kt -d hello.jar

kotlin -cp hello.jar com.example.myapplication.HelloKt

- Hello-world:

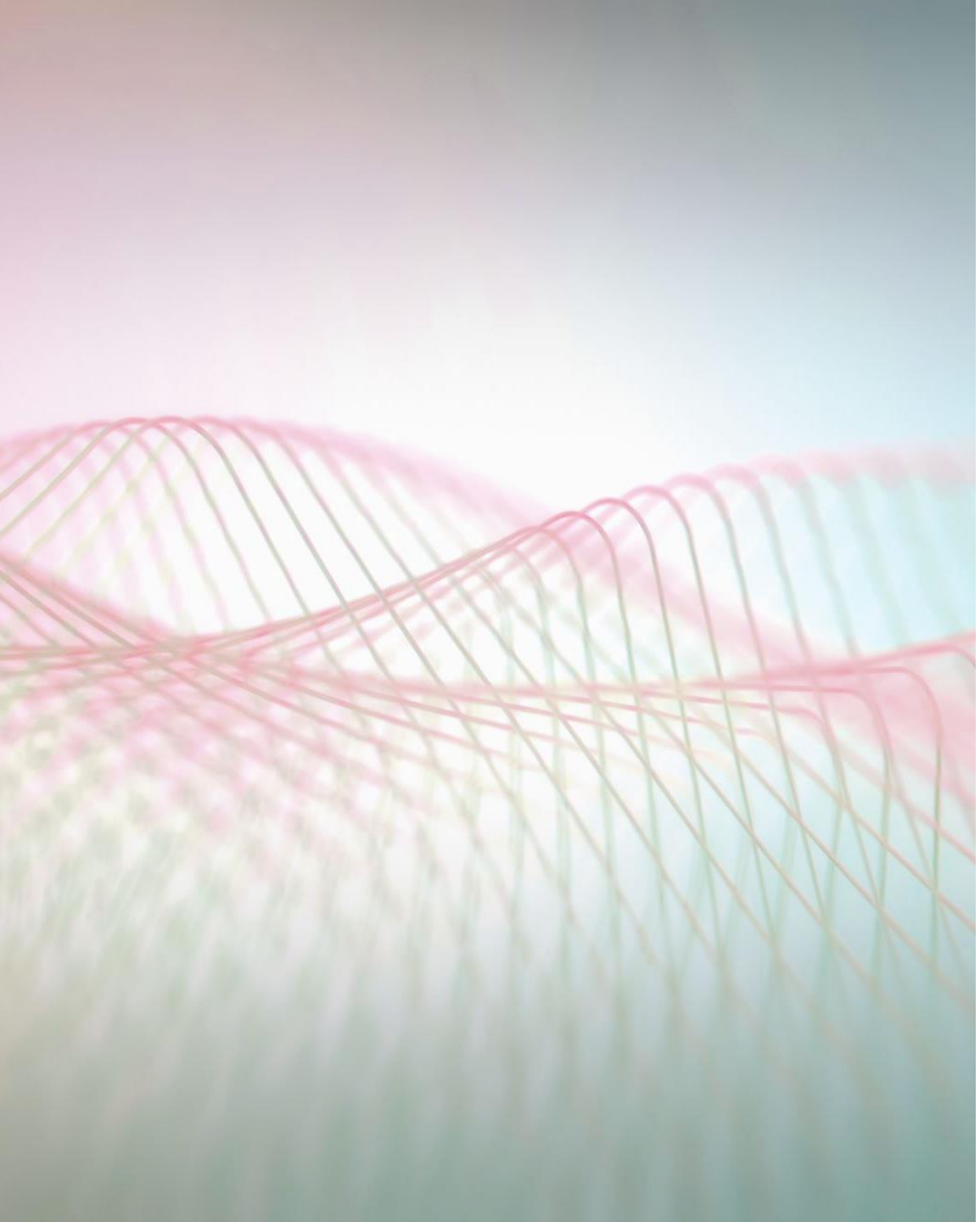
hello.kt

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



- Compilare: `> kotlinc hello.kt`

- Eseguire: `> kotlin HelloKt`

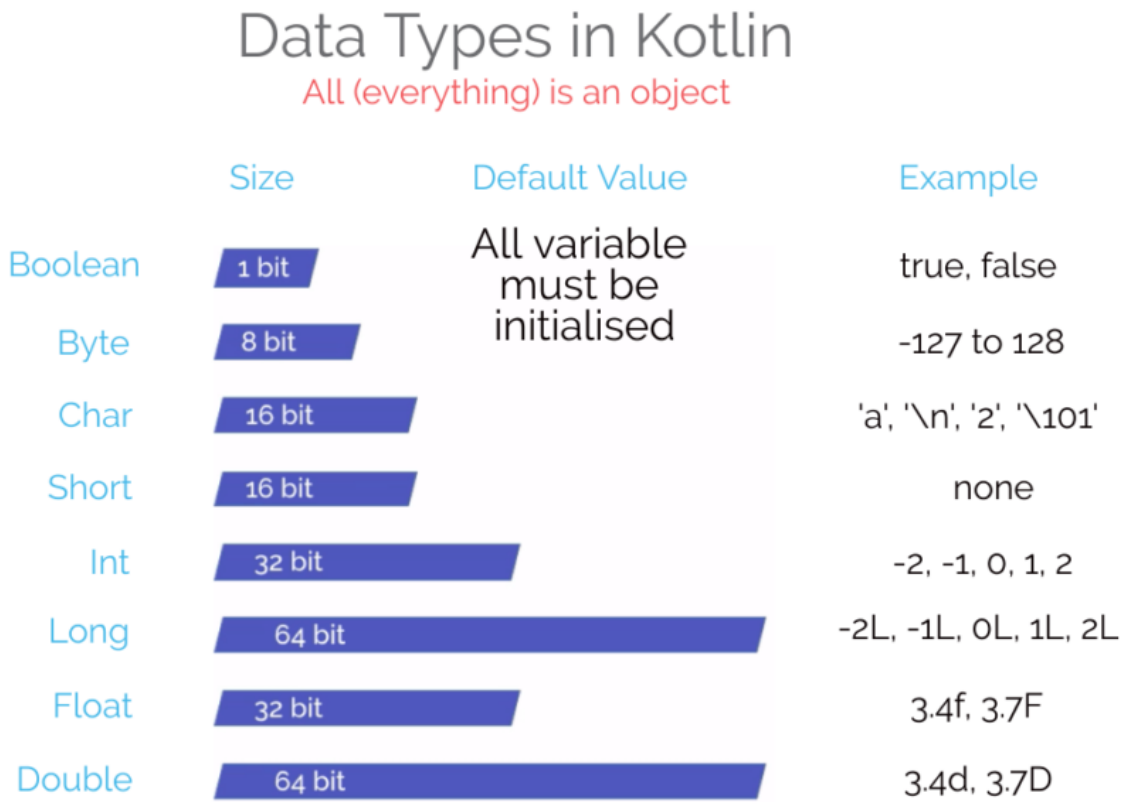


Kotlin Basics

Literals

- Kotlin provides literals for the basic types (numbers, character, Boolean, String).

```
var intLiteral = 5
var doubleLiteral = .02
var stringLiteral = "Hello"
var charLiteral = '1'
var boolLiteral = true
```



Variables

Variables in Kotlin can be easily defined as
- mutable (**var**)
- immutable (**val**)

The key concept: just use **val** as much as possible

- In Kotlin, everything is an object.
- We don't find primitive types as the ones we can use in Java.
- There are some differences compared to Java
  - There are no automatic conversions among numeric types.
  - Characters (Char) cannot directly be used as numbers.
  - A String can be accessed as an array and can be iterated

```
val i: Int = 7
val d: Double = i.toDouble()

val c: Char = 'c'
val ic: Int = c.toInt()
```

```
int i = 7;
double d = i;

char c = 'c';
int ic = c;
```

```
val s = "Example"
for(ch in s){
    println(ch)
}
```

```
String s = "Example";
for (char ch : s.toCharArray()) {
    System.out.print(ch);
}
```

Kotlin strings

- Much of what we've learned about Java Strings are still applicable in Kotlin;
- Two news we need to know about Kotlin strings:
  - They have iterators, so we can use a for loop:

```
val str = "The quick brown fox"
for (i in str) println(i)
```

  - Its elements can be accessed by the indexing operator (`str[elem]`)
  - The preferred way to do string composition in Kotlin is by using string templates, like

```
var name = "Ignazio Gallo"
var email = "ignazio.gallo@uninsubria.it"
var concat = "name: ${name} | email: ${email} "
println(concat)
```

Null Safety

- Per evitare le `NullPointerException`, tipiche di Java...
- Se hai bisogno di una variabile che può essere nulla, dichiarala nullable aggiungendo ? alla fine del suo tipo.

```
fun toUpper(str: String?): String? {
  if (str != null) {
    return str.toUpperCase()
  }
  return str
}

fun main(){
  var str: String? = null
  println("Hello ${toUpper(str)}")
}
```

Arrays

Array in Kotlin are fixed size (Immutable)

- Kotlin doesn't have an array object like the one created in Java using the square braces syntax.
- The Kotlin array is a generic class with a type parameter.
- Arrays in Kotlin can be created using
  - the arrayOf () and arrayOfNulls () functions,
  - the Array constructor. (docs)

```
var emptyArray = arrayOfNulls<String>(5)
println(emptyArray)
```

```
val arrayname = Array(4, { i -> i * i })
arrayname.forEach { item -> println(item) }
```

```
val num1 = arrayOf(1, 2, 3, 4) //implicit type declaration
val num2 = arrayOf<Int>(1, 2, 3) //explicit type declaration
```

The when Statement

- Kotlin doesn’t have a switch statement, but it has the when construct.
- we don’t need to put a break,

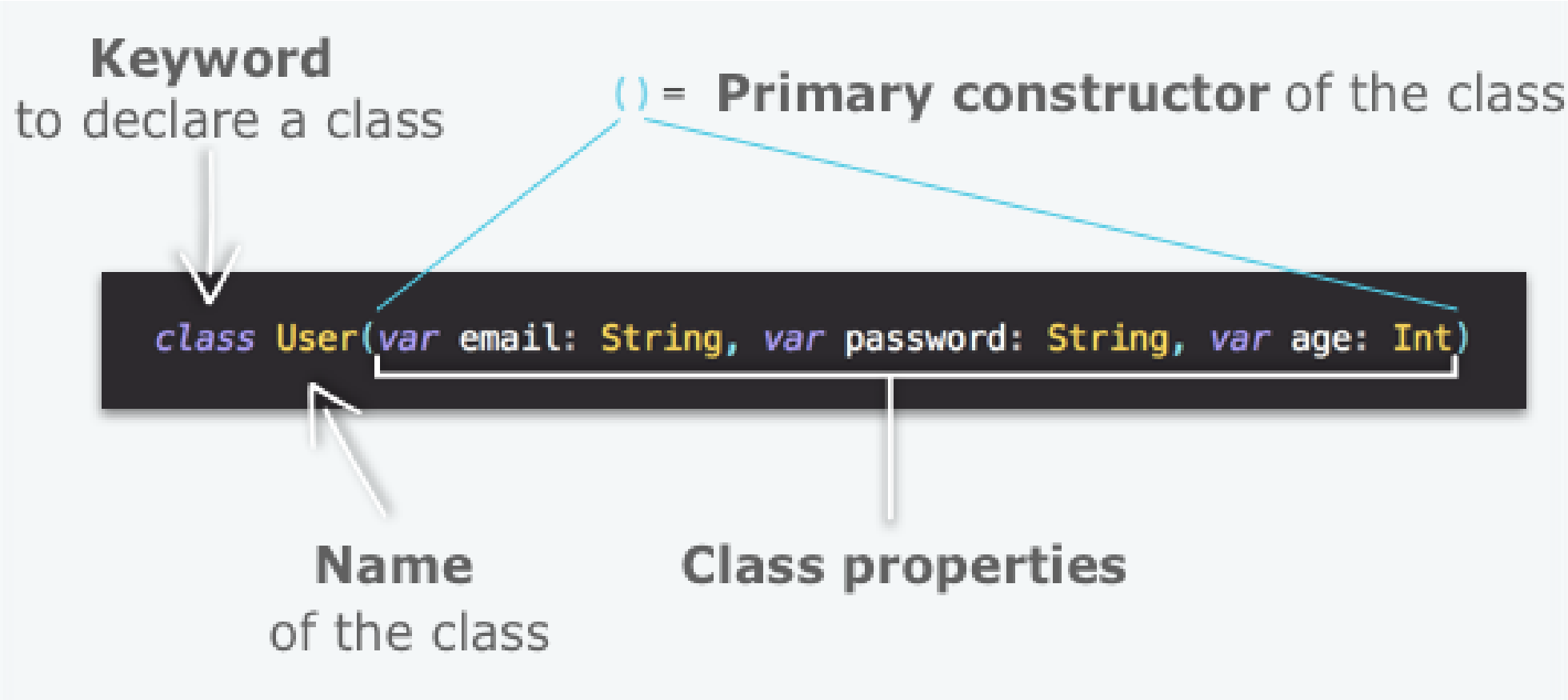
```
val d1 = Date()
val c1 = Calendar.getInstance()
val day = c1.get(Calendar.DAY_OF_WEEK)
when (day) {
  1 -> println("Sunday")
  2 -> println("Monday")
  3 -> println("Tuesday")
  4 -> println("Wednesday")
}
```

Ranges and Loops

- If you need to work with numbers on the for loop, you can use Ranges.
- A range is a type that represents an arithmetic progression of integers.
- Ranges are created with the rangeTo() function, but we usually use it in its operator form (. .)
- To create a range of integers from 1 to 10, we write like this:

```
var zeroToTen = 0..10
if (9 in zeroToTen) println("9 is in zeroToTen")
for (i in 1..10) {
    println(i)
}
var k = 0
while (k in zeroToTen){
   println(k++)
}
```

Classes

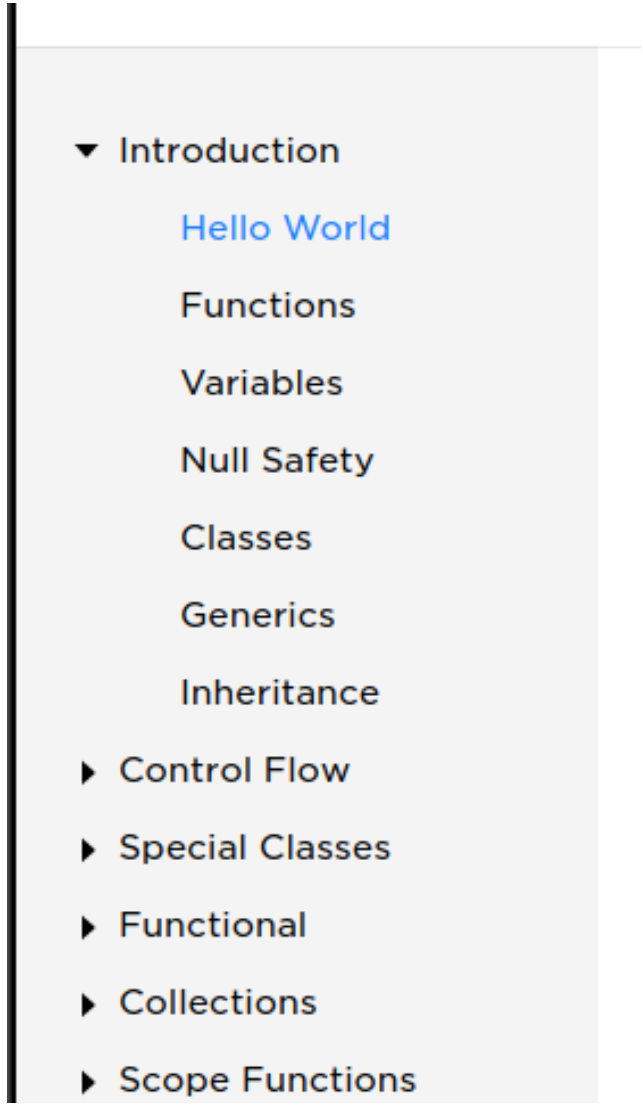



```
val p = Person("Pippo", "xxxxx",32)
```

Inheritance

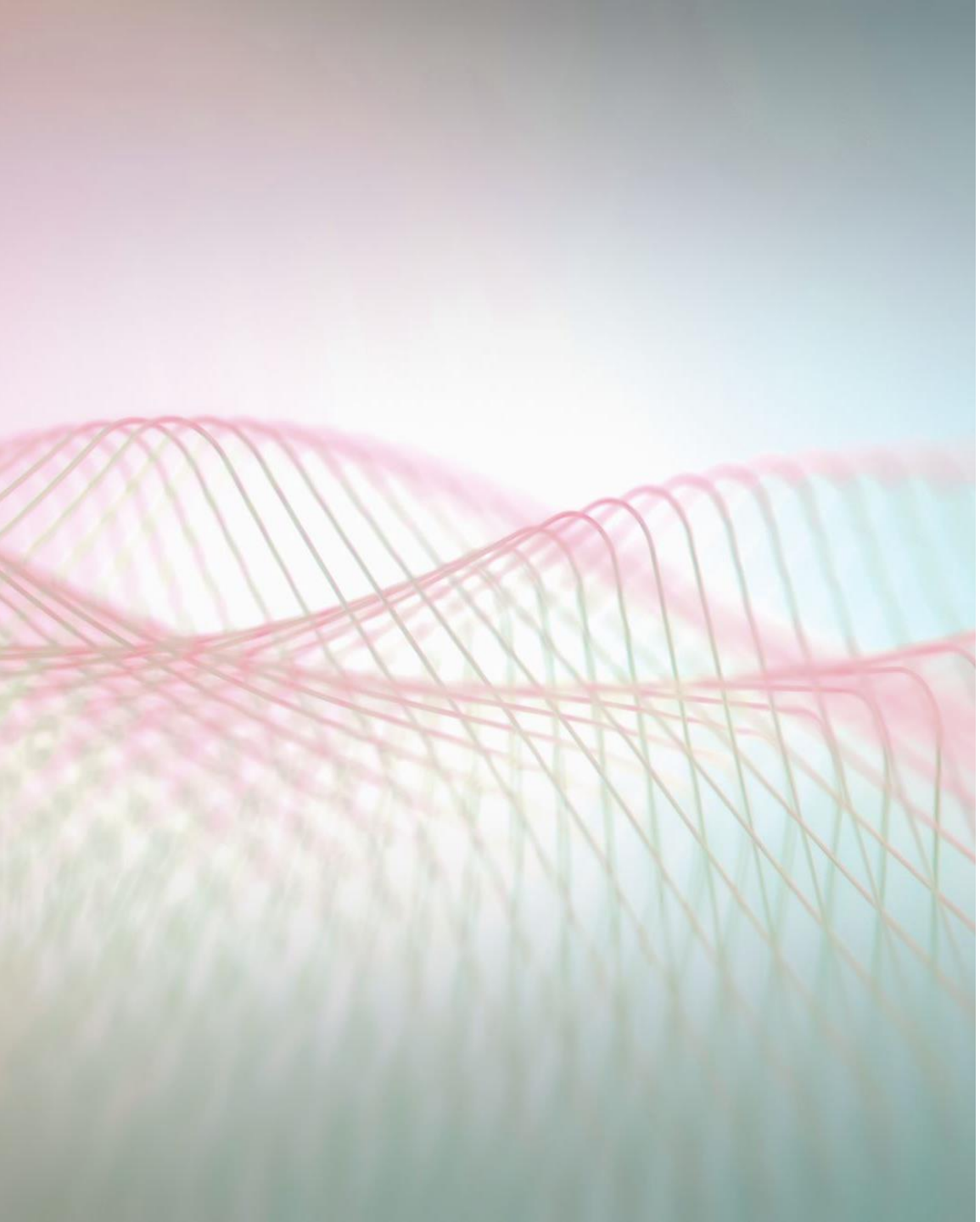
```
class Rectangle(a: Double, ...) ... {
  var a: Double
    get() = 0.0
    set(value) {a=value}
...
}
```

```
abstract class Polygon {
  abstract fun draw()
}

class Rectangle(var a: Double, var b: Double) : Polygon() {
  override fun draw() {
    println("Rectangle: $a x $b")
  }
}

fun main(){
  var rect: Polygon = Rectangle(2.0, 5.0)
  rect.draw()
}
```



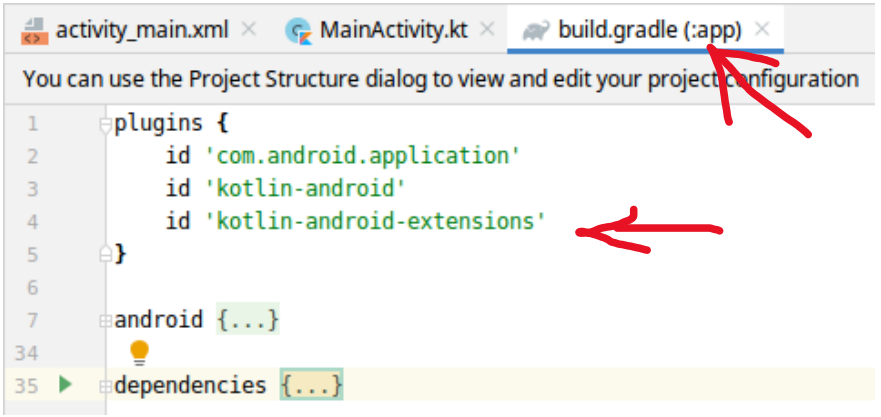
Kotlin by examples

https://play.kotlinlang.org/byExample/overview



Kotlin e Android Studio

Let's create a simple App using Kotlin



- Now that we know the Kotlin language a bit, we can finally try to make our App in Kotlin using Android Studio
- Create a new App -> Empty Activity -> Kotlin
- add *kotlin-android-extensions* plugin
- Let’s see how our basic MainActivity looks like in Kotlin...

Let's create a simple App using Kotlin

… pretty much the same as in Java!

kotlin

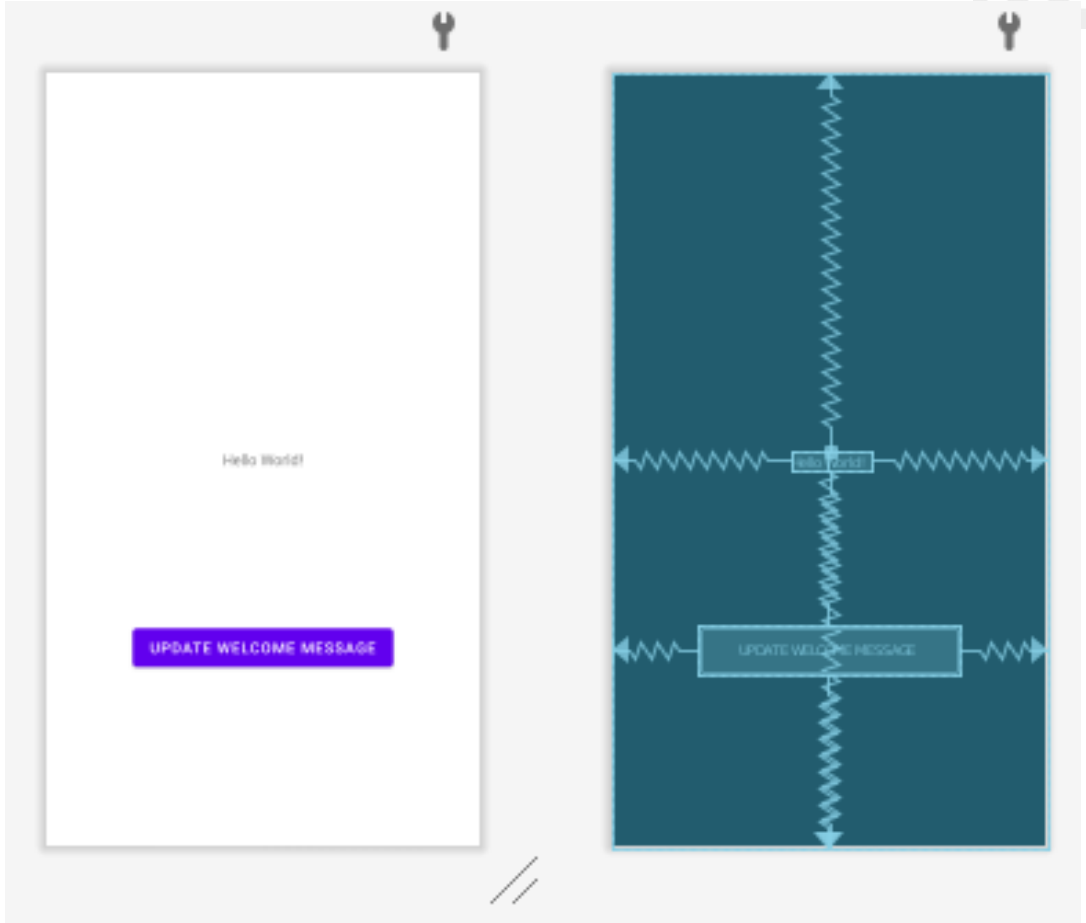
```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Java

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView
android:id="@+id/welcomeTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
android:id="@+id/updateTextButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Update welcome message"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="@id/welcomeTextView"
app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Let's add a button



Let's update the text when the button is clicked

kotlin

```kotlin
class MainActivity : AppCompatActivity(){
    override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      updateTextButton.setOnClickListener{ welcomeTextView.text = "Hello Kotlin World!" }
    }
}
```

Java

```java
public class MainActivity extends AppCompatActivity{
@Override
protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final Button button = findViewById(R.id.updateTextButton);
    final TextView welcomeTextView = findViewById(R.id.welcomeTextView);
    button.setOnClickListener(new View.OnClickListener(){
      @Override
      public void onClick(View v) {
        welcomeTextView.setText("Hello Kotlin World!");
       }
    });
}}
```

Functional support (Lambdas)

Esercizio 1

- Ad ogni click sul button incrementare il contatore n sulla stringa "Hello Kotlin World! (n)"
- Valore iniziale di n=0

Esercizio 2

- Ad ogni click sul button invertire la stringa "Hello Kotlin World!"