Основи Kotlin.

Мова програмування Kotlin

- **Kotlin** (Ко́тлін) — статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту.
- Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE.
- Мова розробляється з 2010 року, публічно представлена в липні 2011. Початковий код було відкрито в лютому 2012.
- В грудні 2015 року з'явився реліз-кандидат версії 1.0, а 15 лютого 2016 року відбувся реліз версії 1.0.

Мова програмування Kotlin

- З 17 травня 2017 року входить в список офіційно підтримуваних мов для розробки застосунків для платформи Android.
- З 7 травня 2019 року є рекомендованою мовою програмування для розробки Android застосунків.

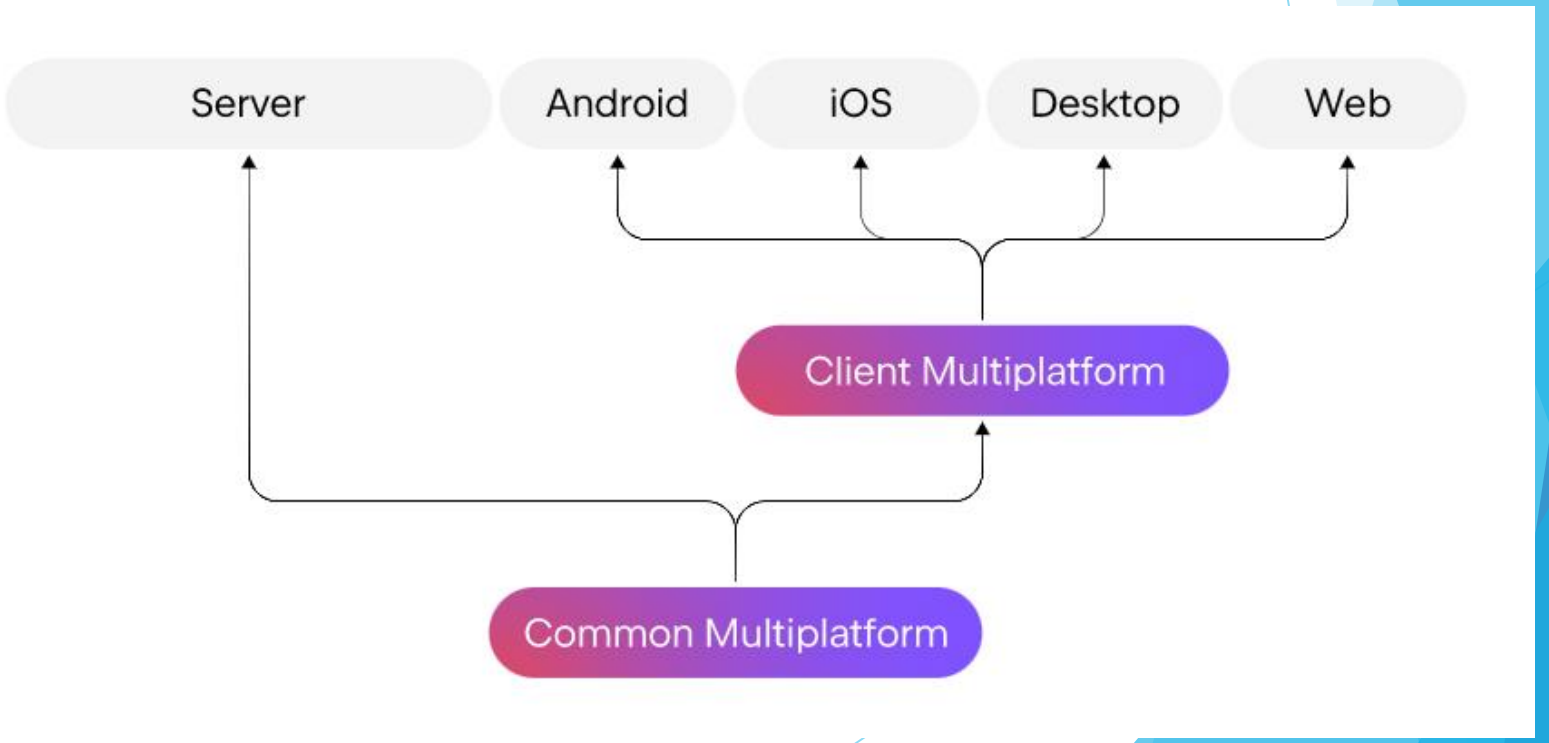
Офіційна документація з прикладами
https://kotlinlang.org/docs/home.html

Тюторіал https://www.w3schools.com/KOTLIN/index.php

Основи (вікіпідручник українською)
https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8_Kotlin

Kotlin Multiplatform

Технологія Kotlin Multiplatform спрощує розробку кросплатформних проєктів. Це скорочує час, витрачений на написання та підтримку того самого коду для різних платформ, зберігаючи при цьому гнучкість і переваги рідного програмування.



Java VS Kotlin

```
public class Program2 {
    public static void main(String[] args) {
        int t, tm;
        System.out.println("Введіть число секунд, " +
                "що пройшли від початку доби: ");
     java.util.Scanner scanner =
                new java.util.Scanner(System.in);
        t = scanner.nextInt();//інструкція вводу даних
        tm = t / 60;    int s = t % 60;
        int h = tm / 60;   int m = tm % 60;
        System.out.printf("Час %02d:%02d:%02d\n", h, m, s);
    }
}
```

Java VS Kotlin

```
object Program2b {
    @JvmStatic
    fun main(args: Array<String>) {
        val t: Int
        val tm: Int
        println( "Введіть число секунд, " +
                "що пройшли від початку доби: " )
        val scanner = java.util.Scanner(System.`in`)
        t = scanner.nextInt() //інструкція вводу даних
        tm = t / 60;          val s = t % 60
        val h = tm / 60;          val m = tm % 60
        System.out.printf("Час %02d:%02d:%02d\n", h, m, s)
    }
}
```

Java VS Kotlin

```
fun main(args: Array<String>) {
    println("Введіть число секунд, " +
        "що пройшли від початку доби: ")
    val t: Int = readln().toInt()
    val tm = t / 60;     val s = t % 60
    val h = tm / 60;     val m = tm % 60
    println("%02d".format(h) + ":" +
        "%02d:%02d".format(m,s))
}
```

Змінні

```
var x=5 //оголошення цілочисельної змінної
val y=5 //оголошення константи
var num: Int // цілочисельна змінна з вказанням типу
var hugeNumber = 6L //створення змінної типу Long
var hugeNumber2: Long =6 //теж саме
var isVar= true // створення змінної типу Boolean
var letter='D' //змінна типу Char
var name="Tom" // змінна типу String
var p=3.14 // змінна типу Double
```

Числові типи даних

Цілочисельні			
Type	Size (bits)	Min value	Max value
Byte	8	-128	127
Short	16	-32768	32767
Int	32	-2,147,483,648 (-2^{31})	2,147,483,647 ($2^{31} - 1$)
Long	64	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 ($2^{63} - 1$)

Дійсні числа				
Type	Size (bits)	Significant bits	Exponent bits	Decimal digits
Float	32	24	8	6-7
Double	64	53	11	15-16

Символи та текстові рядки

```
val myGrade: Char = 'B'
println(myGrade)
// Символи не Int
val myLetter: Char = 66
println(myLetter) // Error
// Рядок символів
val myText: String = "Hello World"
println(myText)
```

Приведення типів

```
val num1: Int = 101
var num2: Long = num1 // error
val num3: Long = num1.toLong()
```

toChar() – To convert a type to Char type.

toInt() – To convert a type to Int type.

toLong() – To convert a type to Long type.

toFloat() – To convert a type to Float type.

toDouble() – To convert a type to Double type.

toByte() – To convert a type to Byte type.

toShort() – To convert a type to Short type.

Ввід та вивід даних

```
import kotlin.math.sqrt
fun main(args: Array<String>) {
    println("Як вас звати?")
    val name: String = readln()
    println("Привіт, ${name}!")
    print("Введіть число: ")
    var x: Double = readln().toDouble()
    x = sqrt(x)
    println("Число впало під корінь")
    println("маєте: %11.7f".format(x))
}
```

Масиви

```
val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
cars[0] = "Opel"
println(cars[0]) // Outputs Opel instead of Volvo
println(cars.size)   // Outputs 4
if ("Volvo" in cars) {
    println("It exists!")
} else {
    println("It does not exist.")
}
for (x in cars) println(x)
```

Розгалуження

```
var x=5;
if (x==5) {
    println ("співпало")
}
if (x>0) {
    println ("додатнє число")
} else if (x<0) {
    println ("від'ємне число")
} else println ("ні додатнє, ні від'ємне (0)")
```

Вибір

```
val result = when (day) {
    1 -> "Monday"
    2 -> "Tuesday"
    3 -> "Wednesday"
    4 -> "Thursday"
    5 -> "Friday"
    6 -> "Saturday"
    7 -> "Sunday"
    else -> "Invalid day."
}
```

Цикли

```
var i = 0
while (i < 10) println(++i)

i = 0
do {
    println(++i)
}while (i < 10)

val m = arrayOf(1,3,5,7,9)
for (k in m){
println(k)
}

for (n in 1..10) println(n)
```

Функції

```
fun myFunction1(fname: String, age: Int) {
    println(fname + " is " + age)
}
fun main(args: Array<String>) {
   var result = myFunction3(3, 7)
    println(result) // 10
    result = myFunction2(result)
    println(result) // 15
    myFunction1("John", 35) // John is 35
    myFunction1("Jane", 32) // Jane is 32
    myFunction1("George", 15) // George is 15
}
fun myFunction2(x: Int): Int {
    return (x + 5)
}
fun myFunction3(x: Int, y: Int) = x + y
```

Null Safety

```
//Declares a non-null String variable.
// Assigns a nullable string to a variable
val a: String? = "Kotlin"
// Assigns null to a nullable variable
val b: String? = null

// Checks for nullability and returns length or null
println(a?.length)
// 6
println(b?.length)
// null
```

Null Safety

```
//Declares a non-null String variable.
    var neverNull: String = "This can't be null"
    // neverNull = null
// When trying to assign null to non-nullable variable,
// a compilation error is produced.

// Declares a nullable String variable.
    var nullable: String? = "You can keep a null here"
//Sets the null value to the nullable variable. This is OK.
    nullable = null

 nullable = "You can keep a null"

//Convert nullable to a non-null String variable.
 var n: Int = nullable!!.lenth
    // neverNull = null
```

Elvis Operator

```
//When we have a nullable reference r, we can say
//"if r is not null, use it, otherwise use some non-null value x":

val l: Int = if (b != null) b.length else -1
//Along with the complete if-expression,
//this can be expressed with the Elvis operator, written ?::
val l = b?.length ?: -1
/*If the expression to the left of ?: is not null, the elvis operator
returns it, otherwise it returns the expression to the right. Note
that the right-hand side expression is evaluated only if the left-
hand side is null.*/
/*Note that, since throw and return are expressions in Kotlin,
they can also be used on the right hand side of the elvis operator.
*/
```