	Swift	**Kotlin**
	Its enums are powerful	It Classes are final by default.
	It has no data class	It has no structs, no tuples, no guard statements,no typealias, and also no passing data by value.
	It does not have delegate class or delegate properties	
	It has no annotations	
	func	fun
	let	val
	var	var
	nil	null
	protocol	traits
	init	constructor
	print("!")	**println**("")
	let explicitTypesDouble: Double = 10	val explicitTypesDouble: Double = **10.0**
	let labelText = "Hello " let labelWidth = 5 let text = labelText + **String**(labelWidth)	let labelText = "Hello " let labelWidth = 5 let text = labelText + labelWidth
	let iPhone = 2 let iPad = 1 let nexus = 3 let totalDevices = "I have \(iPhone + iPad + nexus) " + "Number of devices."	let iPhone = 2 let iPad = 1 let nexus = 3 let totalDevices = "I have **${**iPhone + iPad + nexus**}** " + "Number of devices."

Range Operator	for i in 0..<count {
	print("\(i + 1) \(someArray[I])")
	}
		for (i in 0..count - 1) {
		println("${i + 1} ${someArray[i]}")
		}

Collections of Arrays	var countryList = ["UK", "US", "Japan"] countryList[1] = "India"	val countryList = arrayOf("UK", "US", "Japan") countryList[1] = "India"
Empty collections	let emptyArrayDeclaration = [String]() let emptyDictionaryDeclartaion = [String: Float]()	val emptyArrayDeclaration = arrayOf<String>() val emptyMapDeclaration = mapOf<String, Float>()

Functions	func congratulate(_ personName: String,_ reason: String) -> S
	return "Congratulation \(personName), for your \(reason)."
	}
	congratulate("Sam", "Marriage Anniversary")
		fun congratulate(personName: String, reason: String): String {
		return "Congratulation $personName, for your $reason."
		}
		congratulate("Sam", "Marriage Anniversary")

Tuples	func getSavingA/cNumberAndBalance() -> (Int, Double) {
	return (01234567, 31236979)
	}
		data class getSavingbalance(val a: Int, val b: Double)
		fun getSavingA/cNumberAndBalance() = getSavingbalance(01234567, 31236979)

Mapping	let prices = [1200, 1910, 710, 1210] prices.map { 3 * $0 }	val prices = listOf(1200, 1910, 710, 1210) prices.map { 3 * it }
Sort	var unsortedArray = [12, 25, 32, 212, 22] unsortedArray.sort()	listOf(12, 25, 32, 212, 22).sorted()

Classes	**class Square** {
	var numberOfFaces = 0
	func squareDescription() -> String {
	return "A square with \(numberOfFaces) sides."
	}
	}
	var square = Square()
	square. numberOfFaces = 4
	var squareDescription = square.simpleDescription()
		class Square {
		var numberOfFaces = 0
		fun squareDescription() =
		"A square with $numberOfFaces sides."
		}
		var square = Shape()
		square. numberOfFaces = 4
		var squareDescription = square. squareDescription()

Protocol	protocol ProtocolName {
	func name() -> String
	}
	func f<T: ProtocolName>(x: T) {
	print("Name is " + x.name())
	}
		interface ProtocolName {
		fun name(): String
		}
		fun f<T: ProtocolName>(x: T) {
		println("Name is " + x.name())
		}