Program-1 Write simple program

fun main()  
{  
 *print*("Hello World")  
}

Program-2 Write a Program for arithmetic operations

fun main(args: Array<String>) {  
 // Ask for user input for number1 and number2  
 *println*("Enter the first number:")  
 val number1 = *readLine*()!!.*toDouble*() // Reads and converts the input to Double  
  
 *println*("Enter the second number:")  
 val number2 = *readLine*()!!.*toDouble*() // Reads and converts the input to Double  
  
 var result: Double  
  
 // Perform addition  
 result = number1 + number2  
 *println*("number1 + number2 = $result")  
  
 // Perform subtraction  
 result = number1 - number2  
 *println*("number1 - number2 = $result")  
  
 // Perform multiplication  
 result = number1 \* number2  
 *println*("number1 \* number2 = $result")  
  
 // Perform division  
 result = number1 / number2  
 *println*("number1 / number2 = $result")  
  
 // Perform modulus  
 result = number1 % number2  
 *println*("number1 % number2 = $result")  
}

Program-3 Write a program to print a string up to particular character.

fun main(args : Array<String>)  
{  
 var name : String = "Learn Kotlin"  
 *println*("Remove last character from name :" + name.*dropLast*(2))  
}

Program-4 Write a program to calculate LCM using While loop and if statement

fun main()  
{  
 val input1 = 24  
 val input2 = 28  
 var answer : Int  
 *println*("The numbers are $input1 and $input2")  
 answer = if(input1 > input2)  
 input1  
 else  
 input2  
 while(true)  
 {  
 if (answer % input1 == 0 && answer % input2 ==0)  
 {  
 *println*("The answer of LCM is : $answer")  
 break  
 }  
 answer++  
 }

Program-5 Write a program to calculate sum of natural numbers using recursion

fun main(args:Array<String>)  
{  
 val number = 20  
 val sum = *AddNum*(number)  
 *println*("Sum is : $sum")  
  
}  
fun AddNum(num:Int):Int  
{  
 if (num!=0)  
 return num + *AddNum*(num - 1)  
 else  
 return num  
  
}

Program-6 Write a program to calculate to find GCD of given numbers

/\* \*/

fun bef(n1: Int, n2: Int): Int {  
 if (n2 != 0)  
 return *bef*(n2, n1 % n2)  
 else  
 return n1  
}  
  
fun main(args: Array<String>) {  
 val n1 = 366  
 val n2 = 60  
 val bef = *bef*(n1, n2)   
 *println*("G.C.D of $n1 and $n2: $bef")  
}

Program-7 Sum of two matrices

fun main() {  
 val rows = 2  
 val cols = 3  
  
 // Defining the matrices properly  
 val firstMatrix = *arrayOf*(  
 *intArrayOf*(2, 3, 4),  
 *intArrayOf*(5, 2, 3)  
 )  
  
 val secondMatrix = *arrayOf*(  
 *intArrayOf*(2, 3, 4),  
 *intArrayOf*(5, 2, 3)  
 )  
  
 // Initialize the sum matrix with the same size  
 val sum = Array(rows) **{** IntArray(cols) **}** // Adding the matrices  
 for (i in 0 *until* rows) {  
 for (j in 0 *until* cols) {  
 sum[i][j] = firstMatrix[i][j] + secondMatrix[i][j]  
 }  
 }  
  
 // Printing the result  
 *println*("Sum of two matrices is:")  
 for (i in 0 *until* rows) {  
 for (j in 0 *until* cols) {  
 *print*("${sum[i][j]} ")  
 }  
 *println*()  
 }  
}

Program-8 Write a program to create a class of student with roll no, name, stream and semester.

// Define the class with proper capitalized name  
class Student {  
 var rollno: Int = 0  
 var name: String = ""  
 var city: String = ""  
 var stream: String = ""  
 var sem: Int = 0  
  
 // Setter functions for each property  
 fun setStudentRollno(rollno: Int) {  
 this.rollno = rollno  
 }  
  
 fun setStudentName(name: String) {  
 this.name = name  
 }  
  
 fun setStudentStream(stream: String) {  
 this.stream = stream  
 }  
  
 fun setStudentSem(sem: Int) {  
 this.sem = sem  
 }  
  
 // Getter function to get student details  
 fun getStudentDetails(): String {  
 return "Roll No = $rollno, Name = $name, Stream = $stream, Semester = $sem"  
 }  
}  
  
// Main function  
fun main() {  
 val student1 = Student()  
 student1.setStudentRollno(30)  
 student1.setStudentName("Raj")  
 student1.setStudentStream("BCA")  
 student1.setStudentSem(6)  
  
 // Printing the details of the student  
 *println*("Student: ${student1.getStudentDetails()}")  
}

Program-9 Mutable and Immutable List

fun main(){  
 val a = *listOf*('1', '2', '3') //Immutable list  
 *println*(a.size)  
 *println*(a.indexOf('2'))  
 *println*(a[2])  
  
 //Mutable list  
 val names = *arrayListOf*("Ram", "Raj", "Veer")  
 names.add("Krishna")  
 *println*(names)  
}

Program-10 Hashmap

fun main(args: Array<String>){  
  
 val hashMap:HashMap<Int,String> = HashMap<Int,String>() //define empty hashmap  
 hashMap.put(1,"Ajay")  
 hashMap.put(3,"Vijay")  
 hashMap.put(4,"Praveen")  
 hashMap.put(2,"Ajay")  
 *println*(".....traversing hashmap.......")  
 for(key in hashMap.keys){  
 *println*("Element at key $key = ${hashMap[key]}")  
 }  
}

Program-11 Mutable Map

fun main(args: Array<String>) {  
  
 val mutableMap: MutableMap<String, String> = *mutableMapOf*<String, String>()  
 mutableMap.put("name", "Ashu")  
 mutableMap.put("city", "Delhi")  
 val hashMap: HashMap<String, String> = *hashMapOf*<String, String>()  
 hashMap.put("department", "Development")  
 hashMap.put("hobby", "Playing")  
 *println*("......traverse mutableMap.......")  
 for (key in mutableMap.keys) {  
 *println*("Key = " + key + ", " + "Value = " + mutableMap[key])  
 }  
 mutableMap.putAll(hashMap)  
 *println*("......traverse mutableMap after mutableMap.putAll(hashMap).......")  
 for (key in mutableMap.keys) {  
 *println*("Key = " + key + ", " + "Value = " + mutableMap[key])  
 }  
}

Program-12 Set

fun main(args: Array<String>){  
 val intSet = *setOf*(2,6,4,29,4,5)  
 val mySet: Set<Any> = *setOf*(2,6,4,29,4,5,"Ashu","Ajay")  
 *println*(".......print Int set.........")  
 for(element in intSet){  
 *println*(element)  
 }  
 *println*(".......print Any set.........")  
 for(element in mySet){  
 *println*(element)  
 }  
  
 }

Program-13 Mutable Set

fun main(args: Array<String>) {  
 val intmutableSet = *mutableSetOf*<Int>(2, 6, 4, 29, 4, 5)  
 val anymutableSet: Set<Any> = *setOf*(2, 6, 4, 29, 4, 5, "Ajay", "Ashu", "Ajay")  
 *println*("....intmutableSet....")  
 for(element in intmutableSet){  
 *println*(element)  
 }  
 *println*("....anymutableSet......")  
 for(element in anymutableSet){  
 *println*(element)  
 }  
 }

Program-14 Write a program for method overriding.

// Base class  
open class Animal {  
 open fun run() {  
 *println*("Animals can run")  
 }  
}  
  
// Derived class  
class Tiger : Animal() {  
 override fun run() {  
 *println*("Tiger can run fast")  
 }  
}  
  
fun main() {  
 val t = Tiger()  
 t.run()  
}

Program-15 Write a program for interface.

// Interface definition  
interface Vehicle {  
 fun start()  
 fun stop()  
}  
  
// Class implementing the interface  
class Car : Vehicle {  
 override fun start() {  
 *println*("Car started")  
 }  
  
 override fun stop() {  
 *println*("Car stopped")  
 }  
}  
  
fun main() {  
 val obj = Car()  
 obj.start()  
 obj.stop()  
}

Program-16 Write a program for abstract class.

abstract class motorCar {  
 abstract fun run()  
}  
  
// Derived class that implements the abstract method  
class Honda1 : motorCar() {  
 override fun run() {  
 *println*("Honda is running safely...")  
 }  
}  
  
fun main(){  
 val obj = Honda1()  
 obj.run()  
}

Program-17 Write a program for inheritance.

open class Base {  
 // Mark 'x' as 'open' to allow access in derived classes  
 open val x = 10  
}  
  
class Derived : Base() {  
 fun f1() {  
 *println*("x is equal to $x") // Kotlin string templates can be used here  
 }  
}  
  
fun main(args: Array<String>) {  
 val derived = Derived()  
 derived.f1() // Call the foo method  
}

Program-18 Write a program using this keyword.

class person(val name: String) {  
 // Constructor parameter `name` and property `name` are different  
 fun greet(name: String) {  
 *println*("Hello, $name! My name is ${this.name}.") // 'this.name' refers to the class property  
 }  
}  
  
fun main() {  
 val person = person("Raj")  
 person.greet("Dhvanit")// Output: Hello, Alice! My name is John.  
}

Program-20 Write a program using super keyword

open class Animal {  
 open fun makeSound() {  
 *println*("Animal sound")  
 }  
}  
  
class Dog : Animal() {  
 override fun makeSound() {  
 super.makeSound() // Calls the parent class method  
 *println*("Bark!")  
 }  
}  
  
fun main() {  
 val dog = Dog()  
 dog.makeSound() // Output: Animal sound  
}

Program-21 Write a program for second constructor.

class Person(val name: String, val age: Int) {  
 // Primary constructor (used when creating a person with both name and age)  
  
 // Secondary constructor: Only name is provided  
 constructor(name: String) : this(name, 0) { // Calls the primary constructor with default age  
 *println*("Person created with name: $name and age: 0")  
 }  
  
 // Secondary constructor: Only age is provided  
 constructor(age: Int) : this("Unknown", age) { // Calls the primary constructor with default name  
 *println*("Person created with name: Unknown and age: $age")  
 }  
}  
  
fun main() {  
 val person1 = Person("Raj", 25) // Calls the primary constructor  
 val person2 = Person("Shasvat") // Calls the secondary constructor  
 val person3 = Person(30) // Calls the secondary constructor  
}

**How start a Kotlin file in Android Studio, follow these steps:**

1. Create or Open a Project

- Open Android Studio and either

- Create a new project by clicking on "New Project" and selecting a template like Empty Activity, ensuring Kotlin is selected as the programming language.

- Open an existing project.

2. Add a Kotlin File

- In the Project View (left-hand side of Android Studio):

1. Right-click on the desired folder (e.g., app/src/main/java/your\_package\_name).

2. Navigate to New > Kotlin File/Class.

3. Enter a name for your file.

4. Choose the file type (e.g., File for a standard .kt file or Class for a Kotlin class).

5. Click OK.

3. Write Kotlin Code

- Open the newly created file and start coding in Kotlin.

4. Run or Test the Code

- If the file is part of an Activity, update your project’s code (e.g., MainActivity.kt) and run the app on an emulator or a physical device.

- For non-Activity Kotlin files, write functions and call them from your main Activity or elsewhere in the project.

Additional Tips:

- Ensure the Kotlin plugin is installed (by default in newer Android Studio versions). Go to File > Settings > Plugins to check.

- Update your Gradle files if Kotlin is not configured:

1. Add the Kotlin plugin to the top-level build.gradle:

kotlin

plugins {

id 'org.jetbrains.kotlin.android' version '1.x.x' apply false

}

2. In the app-level build.gradle:

kotlin

plugins {

id 'com.android.application'

id 'org.jetbrains.kotlin.android'

}

You're ready to start using Kotlin in Android Studio