# Day-1, Assignment-1

Task 1: Install Kotlin and configure IntelliJ IDEA. Verify the setup by running a "Hello, World!" program.

fun main()

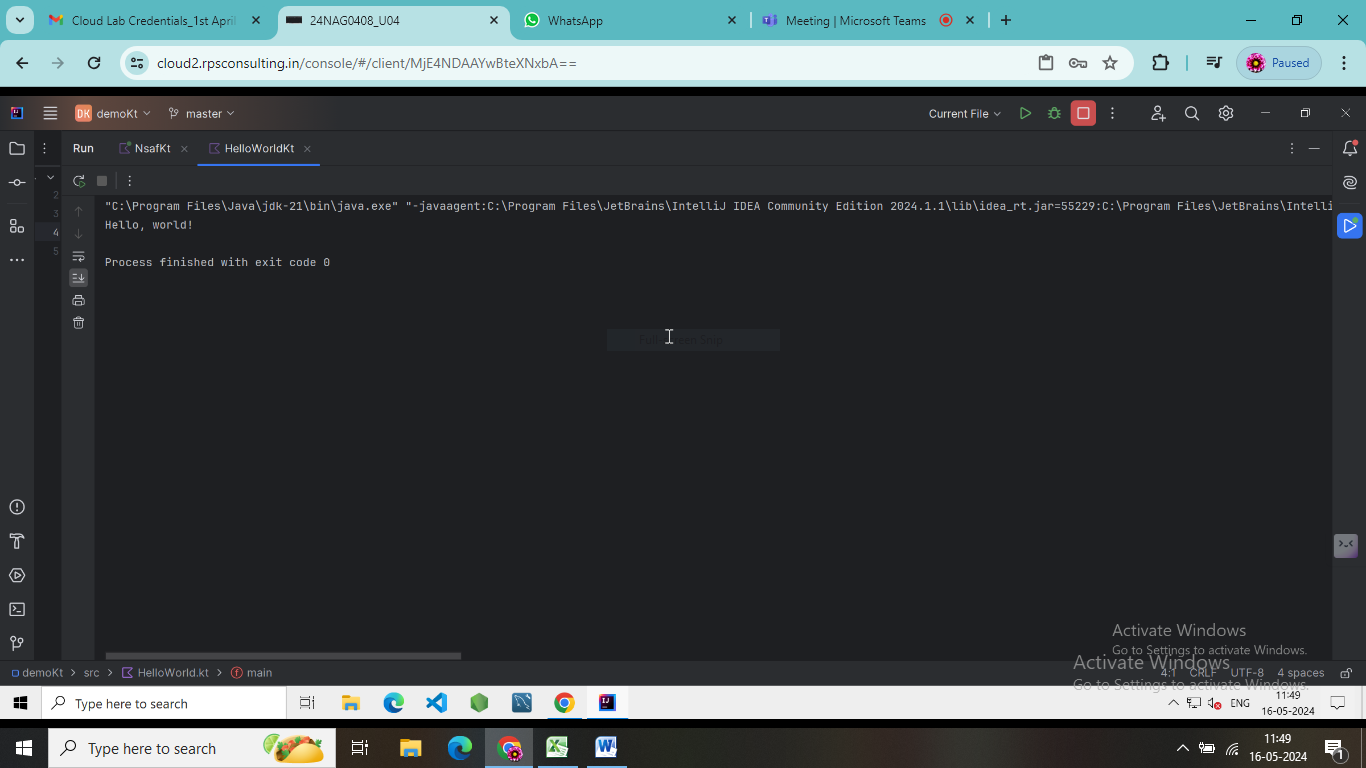
{

Val name=”World”

println(“Hello, “ + name + “!”)

}

### Output:



Task 2: Explore Kotlin REPL (Read-Eval-Print Loop) to familiarize with Kotlin syntax and basic operations.

fun main() {

//Addition

val sum = 1+ 6

println(sum)

val difference = 6 - 1

println(difference)

val product = 6 \* 7

println(product)

val quotient = 25 / 5

println(quotient)

val name: String = "Yesh"

println("Hello, $name!")

var age: Int = 20

age = 24

println("Age: $age")

// String Interpolation

val firstName = "sravani"

val lastName = "motupalli"

val fullName = "$firstName $lastName"

println("Full Name: $fullName")

// Conditional Statements

val number = 10

if (number > 0) {

println("Number is positive")

} else if (number < 0) {

println("Number is negative")

} else {

println("Number is zero")

}

// Loops

// For loop

for (i in 1..5) {

println("for loop is:$i")

}

// While loop

var count = 0

while (count < 3) {

println("Count while loop: $count")

count++

}

// Functions

// Function to add two numbers

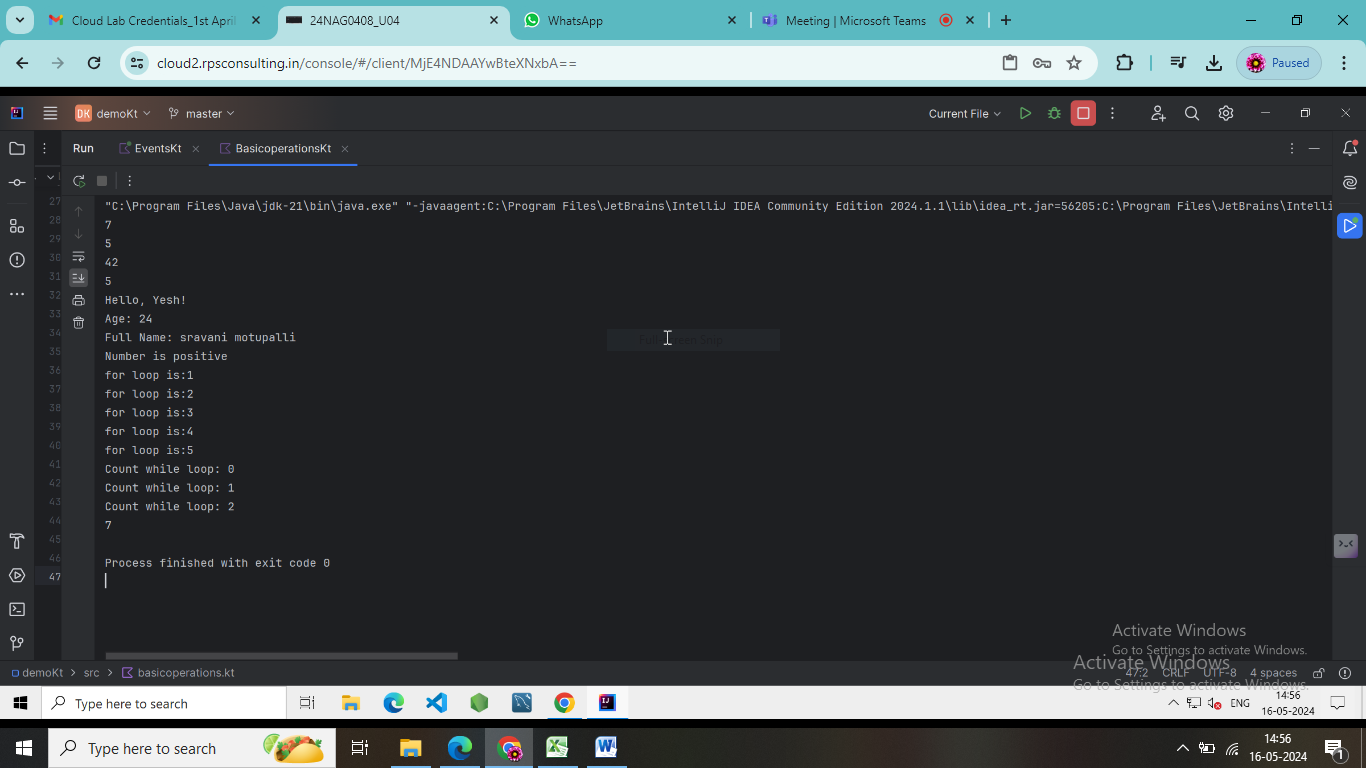
fun add(a: Int, b: Int): Int {

return a + b

}

println(add(3, 4))}

## Output:



Task 3: Create a Transaction class with properties such as amount, date, and category.

import java.time.LocalDate

class Transaction1(

val amount: Double,

val date: LocalDate,

val category: String

)

fun main() {

// Creating transactions

val transaction1 = Transaction1(100.0, LocalDate.of(2024, 5, 15), "Groceries")

val transaction2 = Transaction1(50.0, LocalDate.of(2024, 5, 16), "Dining")

val transaction3 = Transaction1(200.0, LocalDate.of(2024, 5, 17), "Shopping")

// Accessing transaction properties

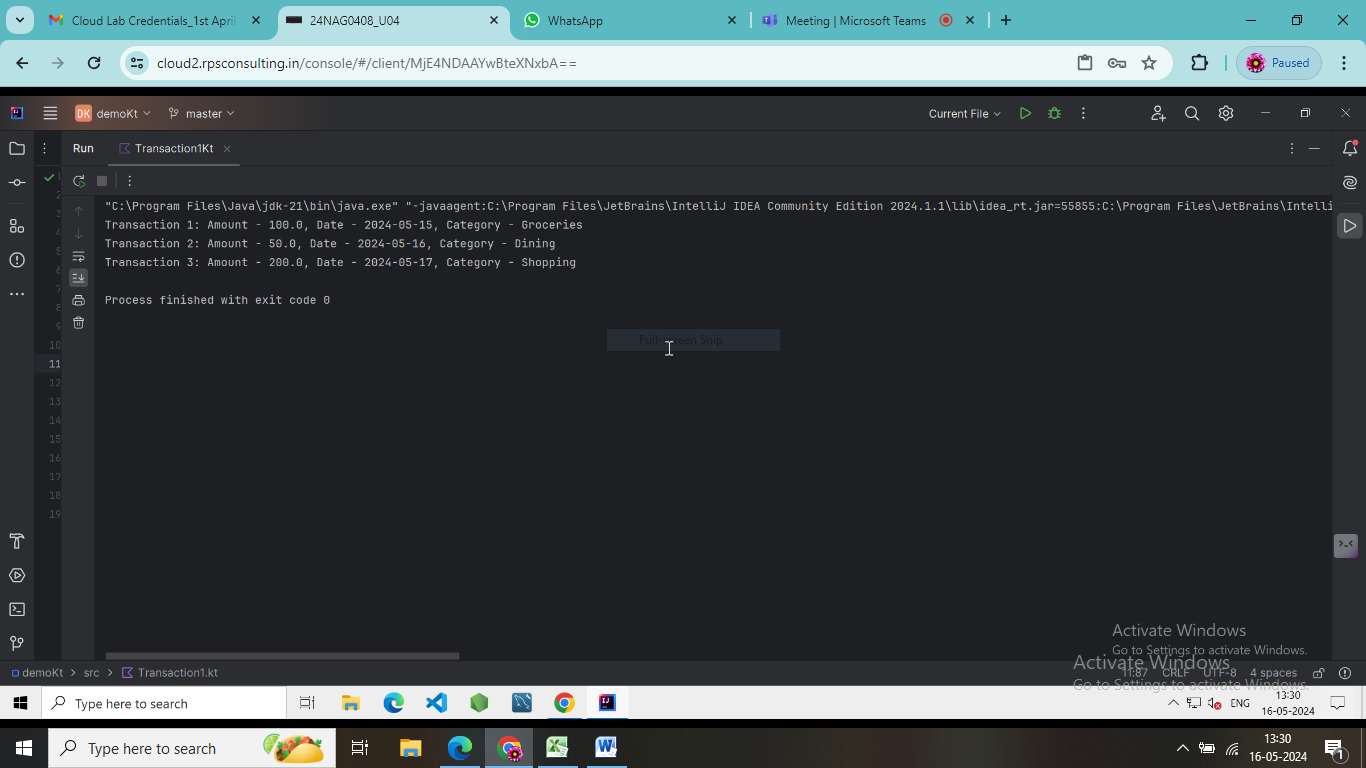
println("Transaction 1: Amount - ${transaction1.amount}, Date - ${transaction1.date}, Category - ${transaction1.category}")

println("Transaction 2: Amount - ${transaction2.amount}, Date - ${transaction2.date}, Category - ${transaction2.category}")

println("Transaction 3: Amount - ${transaction3.amount}, Date - ${transaction3.date}, Category - ${transaction3.category}")

}

### Output:



Task 4: Implement control structures to categorize transactions (e.g., Food, Utilities, Entertainment) using when statements.

import java.time.LocalDate

class Transaction(

val amount: Double,

val date: LocalDate

)

fun categorizeTransaction(transaction: Transaction): String {

return when {

transaction.amount <= 0 -> "Invalid Transaction"

transaction.amount <= 50 -> "Food"

transaction.amount <= 100 -> "Utilities"

else -> "Entertainment"

}

}

fun main() {

// Sample transactions

val transaction1 = Transaction(30.0, LocalDate.of(2024, 5, 15))

val transaction2 = Transaction(80.0, LocalDate.of(2024, 5, 16))

val transaction3 = Transaction(150.0, LocalDate.of(2024, 5, 17))

// Categorize transactions

val category1 = categorizeTransaction(transaction1)

val category2 = categorizeTransaction(transaction2)

val category3 = categorizeTransaction(transaction3)

// Print categories

println("Transaction 1: ${transaction1.amount} - Category: $category1")

println("Transaction 2: ${transaction2.amount} - Category: $category2")

println("Transaction 3: ${transaction3.amount} - Category: $category3")

}

### Output:

