EVIDENCIA DE CODIGOS DE KOTLIN: CURSO

AUTOR: PATRICK ALEXANDER BAEZA ZAMUDIO

Primer programa en Kotlin

```
fun main() {
println("Hello, world!")
}
```

```
/**
* You can edit, run, and share this code. * play.kotlinlang.org */
fun main() {
println("Hello, Android!")
}
```

```
 /**
* You can edit, run, and share this code. * play.kotlinlang.org */
fun main() {
println("Hello, Android!")
println("Hello, Android!")
}
```

```
/**
* You can edit, run, and share this code. * play.kotlinlang.org */
fun main() {
println("Today is sunny!")
}
```

- 1. R=1 2 3

```
2. fun main() {
println("I'm")
println("learning")
println("Kotlin!")
}
3. fun main() {
```

println("Monday")

println("Tuesday")

println("Wednesday")

println("Thursday")

println("Friday")

}

4. fun main() {

println("Tomorrow is rainy")

}

5. fun main() {

println("There is a chance of snow")

}

6. fun main() {

println("Cloudy")

println("Partly Cloudy")

println("Windy")

}

7. fun main() {

println("How's the weather today?")

}

Crear y usar variables

```
fun main() {
val count: Int = 2
println(count)
}
```

```
 fun main() {
val count: Int = 2
println("You have $count unread messages.")
}
```

```
fun main() {
val numberOfPhotos = 100
val photosDeleted = 10
println("$numberOfPhotos photos")
println("$photosDeleted photos deleted")
println("${numberOfPhotos - photosDeleted} photos left")
}
```

```
 fun main() {
var cartTotal = 0
cartTotal = 20
println("Total: $cartTotal")
}
```

```
fun main() {
var cartTotal = 0
println("Total: $cartTotal")
cartTotal = 20
println("Total: $cartTotal")
}
fun main() {
var count = 10
println("You have $count unread messages.")
```

count-- println("You have $count unread messages.")

}

Doble

fun main() {

val trip1: Double = 3.20

val trip2: Double = 4.10

val trip3: Double = 1.72

val totalTripLength: Double = trip1 + trip2 + trip3

println("$totalTripLength miles left to destination")

}

String

fun main() {

val nextMeeting = "Next meeting is: " val date = "January 1" val reminder = nextMeeting + date + " at work" println(reminder)

}

Booleano

fun main() {

val notificationsEnabled: Boolean = false

println("Are notifications enabled? " + notificationsEnabled)

}

Código comentado

/**

* This program displays the number of messages * in the user's inbox. */

fun main() {

// Create a variable for the number of unread messages. var count = 10

println("You have $count unread messages.")

// Decrease the number of messages by 1. count-- println("You have $count unread messages.")

}

Resumen

Una variable es un contenedor de un único dato. Debes declarar una variable antes de usarla. Usa la palabra clave val para definir una variable que sea de solo lectura, en la que el valor no puede cambiar una vez que se asigne. Usa la palabra clave var a fin de definir una variable que sea mutable o

modificable. En Kotlin, se prefiere usar val en lugar de var cuando sea posible.

Para declarar una variable, comienza con la palabra clave val o var. Luego, especifica el nombre de la variable, el tipo de datos y el valor inicial. Por ejemplo:

val count: Int = 2. Con la inferencia de tipo, omite el tipo de datos en la declaración de variablesi se proporciona un valor inicial. Algunos tipos comunes de datos de Kotlin incluyen: Int, String, Boolean, Float y Double. Usa el operador de asignación (=) para asignar un valor a una variable

durante la declaración de la variable o cuando se actualiza la variable. Solo puedes actualizar una variable que se haya declarado como variablemutable (con var). Usa el operador de incremento (++) o el operador de disminución (--) paraaumentar o disminuir el valor de una variable de número entero en 1, respectivamente. Usa el símbolo + para concatenar strings. También puedes concatenar

variables de otros tipos de datos, como Int y Boolean, a Strings. Como crear y usar funciones

```
fun main() {

birthdayGreeting()

}

fun birthdayGreeting() {

println("Happy Birthday, Rover!")

println("You are now 5 years old!")

}
```

El tipo unit

```
fun main() {

birthdayGreeting()

}

fun birthdayGreeting(): Unit {

println("Happy Birthday, Rover!")
```

```
println("You are now 5 years old!")
}
```

Cómo mostrar String de birthdayGreeting()

```
fun birthdayGreeting(): String {
val nameGreeting = "Happy Birthday, Rover!" val ageGreeting = "You are now 5 years old!"
return "$nameGreeting\n$ageGreeting"
}
fun main() {
val greeting = birthdayGreeting()
println(greeting)
}
```

Cómo agregar un parámetro a la función birthdayGreeting()

```
fun birthdayGreeting(name: String): String {
val nameGreeting = "Happy Birthday, $name!"
val ageGreeting = "You are now 5 years old!"
return "$nameGreeting\n$ageGreeting"
}
fun main() {
println(birthdayGreeting("Rover"))
}
```

Funciones con varios parámetros

```
fun birthdayGreeting(name: String, age: Int): String {
val nameGreeting = "Happy Birthday, $name!" val ageGreeting = "You are now $age years old!"
return "$nameGreeting\n$ageGreeting"
}
fun main() {
println(birthdayGreeting("Rover", 5))
```

println(birthdayGreeting("Rex", 2))

}

Argumentos con nombre

println(birthdayGreeting(name = "Rex", age = 2))

println(birthdayGreeting(age = 2, name = "Rex"))

Argumentos predeterminados

fun birthdayGreeting(name: String = "Rover", age: Int): String {

return "Happy Birthday, $name! You are now $age years old!"

}

println(birthdayGreeting(age = 5))

println(birthdayGreeting(age = 2))

Resumen

Las funciones se definen con la palabra clave fun y contienen fragmentos decódigo reutilizables. Las funciones facilitan el mantenimiento de los programas más grandes y

evitan la repetición innecesaria de código. Las funciones pueden mostrar un valor que puedes almacenar en una

variable para usarlo más tarde. Las funciones pueden tomar parámetros, que son variables disponibles

dentro del cuerpo de una función. Los argumentos son los valores que pasas cuando llamas a una función. Puedes nombrar argumentos cuando llamas a una función. Cuando usas

argumentos con nombre, puedes reordenarlos sin afectar el resultado. Puedes especificar un argumento predeterminado que te permita omitirlocuando llames a una función. Impresión de mensajes

Ejercicio1.-¿Puedes escribir una función main() que imprima estos mensajes en cuatro líneas separadas?Use the val keyword when the value doesn't change. Use the var keyword when the value can change. When you define a function, you define the parameters that can be passed to it.

When you call a function, you pass arguments for the parameters. R= fun main() {

println("Use the val keyword when the value doesn't change.")

println("Use the var keyword when the value can change.")

println("When you define a function, you define the parameters that can be passedto it.")

println("When you call a function, you pass arguments for the parameters.")

}

Correccion de un error de correccion

Este programa imprime un mensaje que le notifica al usuario que recibió un mensaje de chat de un amigo. fun main() {

println("New chat message from a friend'}

}

1. ¿Puedes determinar la causa raíz de los errores de compilación de este programa y corregirlos?

2. ¿El código usa los símbolos apropiados para indicar la apertura y el cierre de la cadena y el argumento de la función?

Pista: Puedes usar Kotlin Playground a fin de ejecutar el código y ver los errores de compilación. Después de corregir los errores, el programa debe compilarse sin problemas y, luego, imprimir este resultado:

New chat message from a friend

R= fun main() {

println("New chat message from a friend")

}

Plantillas de Strings

Este programa informa a los usuarios sobre la próxima oferta promocional de un artículo en particular. Tiene una plantilla de cadenas, que se basa en la variable discountPercentage para el porcentaje de descuento y la variable item para el artículo en oferta. Sin embargo, existen errores de compilación en el código.

fun main() {

val discountPercentage: Int = 0

val offer: String = "" val item = "Google Chromecast" discountPercentage = 20

offer = "Sale - Up to $discountPercentage% discount on$item!

Hurry up!"

println(offer)

}

1. ¿Puedes determinar la causa raíz de esos errores y corregirlos?

2. ¿Puedes determinar el resultado de este programa antes de ejecutar el código en Kotlin Playground?

Pista: ¿Puedes reasignar un valor a una variable de solo lectura?

Después de corregir los errores, el programa debe compilarse sin problemas y, luego, imprimir este resultado:

Sale - Up to 20% discount on Google Chromecast! Hurry up!

R=fun main() {

var discountPercentage: Int = 0

var offer: String = "" val item = "Google Chromecast" discountPercentage = 20

offer = "Sale - Up to $discountPercentage% discount on $item! Hurry up!" println(offer)

}

Concatenación de strings

Este programa muestra una cantidad total de personas en una fiesta. Entre ellas, hay adultos y niños. La variable numberOfAdults contiene la cantidad de adultos en el grupo, y la

variable numberOfKids, la cantidad de niños. fun main() {

val numberOfAdults = "20" val numberOfKids = "30" val total = numberOfAdults + numberOfKids

println("The total party size is: $total")

}

Paso 1

¿Puedes determinar el resultado de este programa antes de ejecutar el código en Kotlin Playground?

Después de determinar el resultado, ejecuta el código en Kotlin Playground y, luego, verifica si el resultado coincide con el que se muestra. Pista: ¿Qué sucede cuando usas el operador + en dos cadenas?

Paso 2

El código funciona y, además, imprime algunos resultados, pero los resultados no muestran la cantidad total de personas que asistirán a la fiesta.

¿Puedes encontrar el problema del código y corregirlo de modo que imprima este resultado?

The total party size is: 50

R=fun main() {

val numberOfAdults = 20

val numberOfKids = 30

val total = numberOfAdults + numberOfKids

println("The total party size is: $total")

}

Formato de mensajes

Este programa muestra el salario total que recibe un empleado este mes. El salario total se divideen dos partes: la variable baseSalary, que es lo que el empleado recibe todos los meses, y la variable bonusAmount, que es una bonificación adicional otorgada al empleado. fun main() {

val baseSalary = 5000

val bonusAmount = 1000

val totalSalary = "$baseSalary + $bonusAmount" println("Congratulations for your bonus! You will receive a total of

$totalSalary (additional bonus).")

}

1. ¿Puedes determinar el resultado de este código antes de ejecutarlo en Kotlin Playground?R=solo se pondrán las dos variables ya que en ningún momento se suman

2. Cuando ejecutas el código en Kotlin Playground, ¿se imprime el resultado que esperabas?R=no, ya que se busca que se sumen las cantidades y el programa solo mostraría el mensaje y las dos cantidades sin sumar

Implementación de operaciones matemáticas básicas

En este ejercicio, escribirás un programa que realice operaciones matemáticas básicas y, luego, imprima el resultado. Paso 1

La función main() contiene un error de compilación: fun main() {

val firstNumber = 10

val secondNumber = 5

println("$firstNumber + $secondNumber = $result")

} □ ¿Puedes corregir el error de modo que el programa imprima este resultado?

10 + 5 = 15

R=fun main() {

val firstNumber = 10

val secondNumber = 5

val result = firstNumber + secondNumber

println("$firstNumber + $secondNumber = $result")

}

Paso 2

El código funciona, pero la lógica para sumar dos números se encuentra dentro de la variable de resultado, lo que hace que el código sea menos flexible a la hora de volver a usarlo. En su lugar, puedes extraer la operación de suma en una función add() para que el código se pueda volver a usar. Para ello, actualiza el código con el que se muestra a continuación. Observa que el código ahora presenta una nueva variable val, llamada thirdNumber, e imprime el resultado de esta variable nueva.

firstNumber. fun main() {

val firstNumber = 10

val secondNumber = 5

val thirdNumber = 8

val result = add(firstNumber, secondNumber)

val anotherResult = add(firstNumber, thirdNumber)

println("$firstNumber + $secondNumber = $result")

println("$firstNumber + $thirdNumber = $anotherResult")

}

// Define add() function below this line

□ ¿Puedes definir la función add() de modo que el programa imprima este resultado?

10 + 5 = 15

10 + 8 = 18

R=fun main() {

val firstNumber = 10

val secondNumber = 5

val thirdNumber = 8

val result = add(firstNumber, secondNumber)

val anotherResult = add(firstNumber, thirdNumber)

println("$firstNumber + $secondNumber = $result")

println("$firstNumber + $thirdNumber = $anotherResult")

}

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

return a + b

}

Paso 3

Ahora tienes una función reutilizable capaz de sumar dos números. ¿Puedes implementar la función subtract() de la misma manera en que implementaste la

función add()? Modifica la función main() también para usar la función subtract(), de modo que puedas verificar que funcione como se espera.

Pista: Piensa en la diferencia entre la suma, la resta y otras operaciones matemáticas. Comienza a trabajar en el código de solución a partir de allí. R=fun main() {

val firstNumber = 10

val secondNumber = 5

val thirdNumber = 8

val result = add(firstNumber, secondNumber)

val anotherResult = add(firstNumber, thirdNumber)

val subtractionResult = subtract(firstNumber, secondNumber)

println("$firstNumber + $secondNumber = $result")

println("$firstNumber + $thirdNumber = $anotherResult")

println("$firstNumber - $secondNumber = $subtractionResult")

}

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

return a + b

}

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

return a - b

}

Parametros Predeterminados

Paso1

□□ ¿Puedes implementar la función displayAlertMessage() en este programa de modo que imprima el resultado que se muestra?

□□ ¿El programa imprime este resultado?

There's a new sign-in request on Chrome OS for your Google Account sample@gmail.com. R=fun main() {

val operatingSystem = "Chrome OS" val emailId = "sample@gmail.com" println(displayAlertMessage(operatingSystem, emailId))

}

fun displayAlertMessage(operatingSystem: String, emailId: String): String {

return "There's a new sign-in request on $operatingSystem for your Google Account $emailId."

}

Paso 2

Bien hecho. Mostraste el mensaje. Sin embargo, en algunos escenarios, notas que no puedes determinar el sistema operativo del usuario. En esos casos, deberás especificar el nombre del sistema operativo como Unknown OS. Puedes optimizar aún más el código para que no necesites pasar el argumento Unknown OS cada vez que se llame a la función. 1¿Puedes encontrar una manera de optimizar el código con esta información de modo que imprima este resultado?

There's a new sign-in request on Unknown OS for your Google Account user_one@gmail.com. There's a new sign-in request on Windows for your Google Account user_two@gmail.com. There's a new sign-in request on Mac OS for your Google Account user_three@gmail.com. R=fun main() {

val operatingSystem = "Chrome OS" val emailId = "sample@gmail.com" println(displayAlertMessage(operatingSystem, emailId))

}

fun displayAlertMessage(operatingSystem: String, emailId: String): String {

return "There's a new sign-in request on $operatingSystem for your Google Account $emailId."

}

2 Para imprimir el mensaje anterior, reemplaza la implementación de la función main() por la siguiente: fun main() {

val firstUserEmailId = "user_one@gmail.com"

// The following line of code assumes that you named your parameter as emailId. // If you named it differently, feel free to update the name.
println(displayAlertMessage(emailId = firstUserEmailId))

println()

val secondUserOperatingSystem = "Windows" val secondUserEmailId = "user_two@gmail.com" println(displayAlertMessage(secondUserOperatingSystem, secondUserEmailId))

println()

val thirdUserOperatingSystem = "Mac OS" val thirdUserEmailId = "user_three@gmail.com" println(displayAlertMessage(thirdUserOperatingSystem, thirdUserEmailId))

println()

}

R=fun main() {

val firstUserEmailId = "user_one@gmail.com" println(displayAlertMessage(emailId = firstUserEmailId))

println()

val secondUserOperatingSystem = "Windows" val secondUserEmailId = "user_two@gmail.com" println(displayAlertMessage(secondUserOperatingSystem, secondUserEmailId))

println()

val thirdUserOperatingSystem = "Mac OS" val thirdUserEmailId = "user_three@gmail.com"

println(displayAlertMessage(thirdUserOperatingSystem, thirdUserEmailId))

println()

}

fun displayAlertMessage(operatingSystem: String = "Unknown OS", emailId: String): String {

return "There's a new sign-in request on $operatingSystem for your Google Account $emailId."

}

Podómetro

 ¿Puedes cambiar el nombre de las funciones, de los parámetros de las funciones y de las variables utilizados en este programa según las prácticas recomendadas?

R=fun main() {

val steps = 4000

val caloriesBurned = calculateCaloriesBurned(steps)

println("Walking $steps steps burns $caloriesBurned calories")

}

fun calculateCaloriesBurned(steps: Int): Double {

val caloriesPerStep = 0.04

val totalCaloriesBurned = steps * caloriesPerStep

return totalCaloriesBurned

}

Movimientos del código duplicado a una función

1. ¿Puedes crear una función que imprima los detalles del clima de una sola ciudad para reducir la repetición en la función main() y, luego, hacer lo mismo en las ciudades restantes?

2. ¿Puedes actualizar la función main() a fin de llamar a la función que creaste para cada ciudad y pasar los detalles apropiados del clima como argumentos?

R=fun main() {

printWeatherDetails("Ankara", 27, 31, 82)

println()

printWeatherDetails("Tokyo", 32, 36, 10)

```
println()
printWeatherDetails("Cape Town", 59, 64, 2)
println()
printWeatherDetails("Guatemala City", 50, 55, 7)
println()
}
fun printWeatherDetails(city: String, lowTemp: Int, highTemp: Int, chanceOfRain: Int) {
println("City: $city")
println("Low temperature: $lowTemp, High temperature: $highTemp")
println("Chance of rain: $chanceOfRain%")
}
```