**LAB SESSION 5**

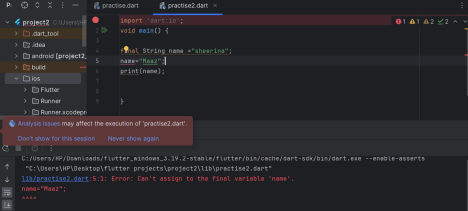
**Final, Constant Keywords, and Conditional Structures**

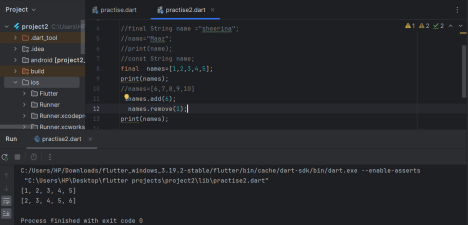
**Objective:**

To understand the usage of final and const keywords in Dart for creating immutable variables, and to explore various conditional structures like if, else if, and switch.

**Introduction:**

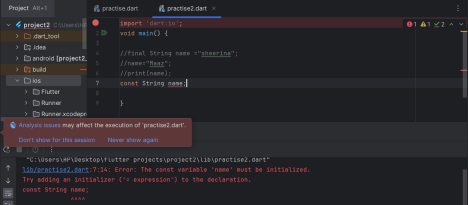
The **final keyword** in Dart is used to declare variables whose values cannot be changed once assigned. It essentially creates immutable variables. Once a final variable is assigned a value, it cannot be reassigned to a different value. Final variables must be initialized exactly once, either at the time of declaration or in a constructor. It’s important to note that final variables are not necessarily constants; their values can be determined at runtime.

Elements can be added and removed but cannot be updated

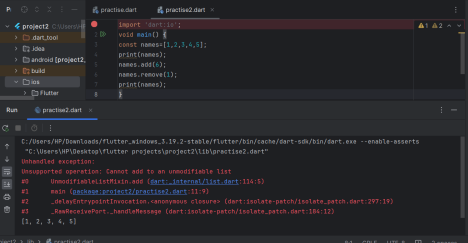


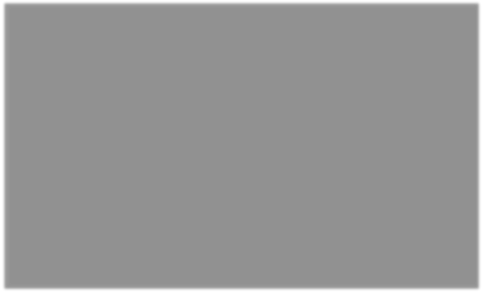
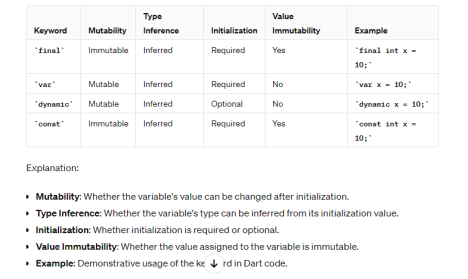
DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

The **const** keyword in Dart is used to declare compile-time constants. These are values that are fixed at compile-time and cannot be changed at runtime. const can be applied to variables, lists, sets, and maps. const variables must be initialized with a constant value at compile-time.Dart provides several built-in types that can be declared as const, such as numbers, strings, and booleans.



The const keyword makes the list non-modifiable



DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487) 

**Key Differences:**

**final** variables are initialized when they are accessed or at the time of object creation, while **const** variables are initialized at compile-time. final variables are not necessarily constants; their values can be determined at runtime, whereas **const** variables are true compile-time constants. **const** variables are implicitly final, but not all final variables are const.

**Use Cases:**

Use final when you want a variable whose value will not change after initialization, but the value might be computed at runtime. Use **const** when you want a variable whose value is known at compile-time and will not change throughout the program's execution.

In the context of programming or software development, "conditional structures" in Dart refer to constructs used for making decisions based on certain conditions. In Dart, the primary conditional structures are:

**if statement:** This structure allows you to execute a block of code if a specified condition is true. It can also be combined with else and else if clauses to handle alternative conditions.

if (condition) {

// Code to execute if condition is true

} else {

// Code to execute if condition is false

}

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

import 'dart:io';

void main()

{

var a =500;

var b=50;

if(a>200 && b >100)

{

print("block 1");

}

else if (a<50)

{

print("block 2");

}

else if(a>80)

{

print("block 3");

}

else

print("block 4");

}



**switch statement:** This structure allows you to select one of many blocks of code to execute, based on the value of an expression.

switch (expression) {

case value1:

// Code to execute if expression matches value1

break;

case value2:

// Code to execute if expression matches value2

break;

default:

// Code to execute if expression doesn't match any case }

These conditional structures are fundamental to controlling the flow of execution in Dart programs, allowing developers to create logic that responds dynamically to different scenarios.

In Dart, loops are used to execute a block of code repeatedly. Dart provides several types of loops:

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**for loop:** This loop executes a block of code a specified number of times.

for (int i = 0; i < 5; i++) {

print('Value of i: $i');

}

import 'dart:io';

void main() {

for (int i=0;i<=10;i++)

{

print("hello");

}

}

**while loop:** This loop executes a block of code as long as a specified condition is true.

int i = 0;

while (i < 5) {

print('Value of i: $i');

i++;

import 'dart:io';

void main() {

int i=100;

while (i<=200)

{

print("no=$i");

i=i+20;

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

}

}



**do-while loop:** Similar to a while loop, but the block of code is executed at least once before the condition is checked.

int i = 0;

do {

print('Value of i: $i');

i++;

} while (i < 5);

import 'dart:io';

void main() {

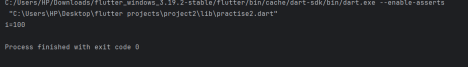
int i = 100;

do {

print("i=$i");

} while (i > 100);

}

**for-in loop:** This loop iterates over the elements of an iterable object (like lists, sets, or maps).

List<int> numbers = [1, 2, 3, 4, 5];

for (int number in numbers) {

print('Number: $number');

}

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**forEach loop:** Similar to a for-in loop, but specialized for iterating over the elements of an iterable object. It's commonly used with collections like lists.

List<int> numbers = [1, 2, 3, 4, 5];

numbers.forEach((number) {

print('Number: $number');

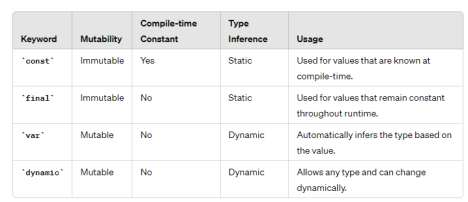
});

These loop structures in Dart provide developers with the flexibility to perform repetitive tasks efficiently, iterating over data structures or executing code based on specific conditions.

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**Lab Exercise:**

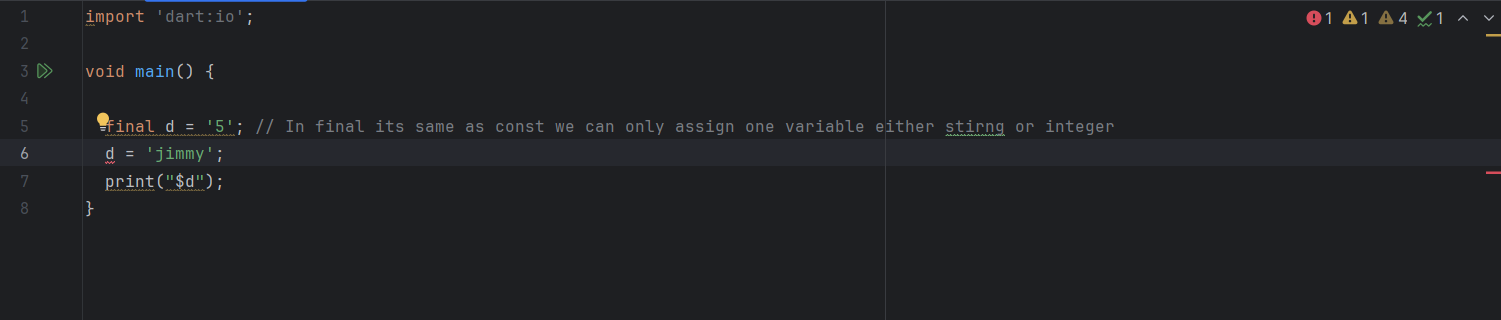
**Question 1**

****

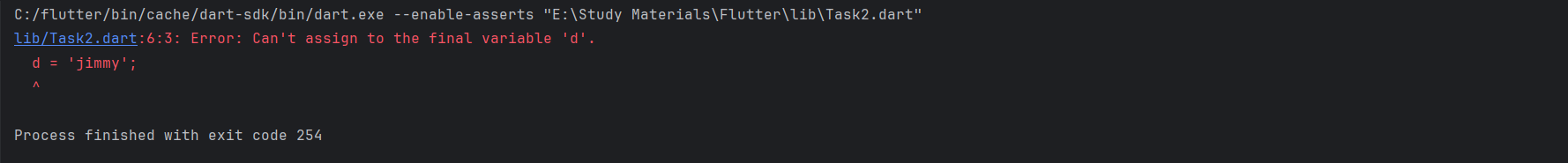
Consider you're designing a Dart application and need to declare variables for different purposes. Analyze the differences between `const`, `final`, `var`, and `dynamic` keywords and determine which one would be most suitable for each scenario:

1. **You have a value that will not change throughout the runtime and is known at compile-time.**

For this scenario, final is the most appropriate choice. When you declare a variable as final, you can only assign it a value once, and that value cannot be changed thereafter.

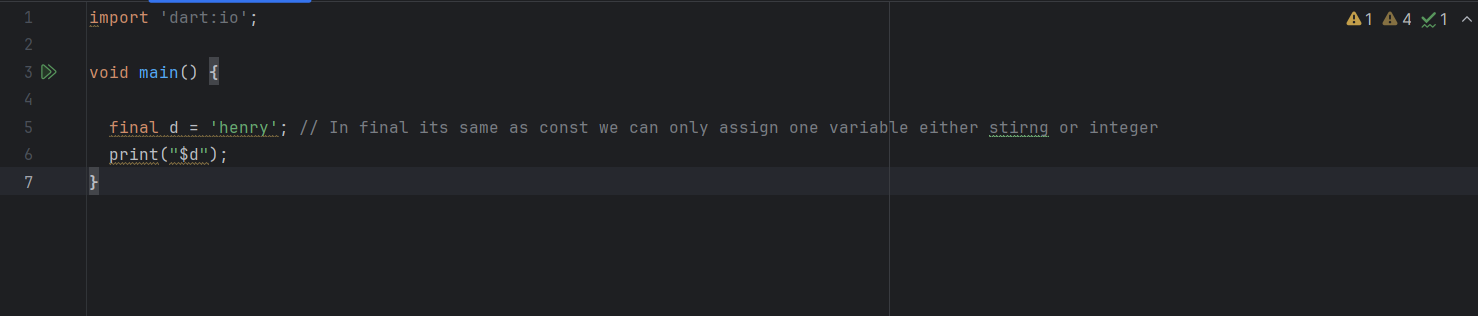


Output

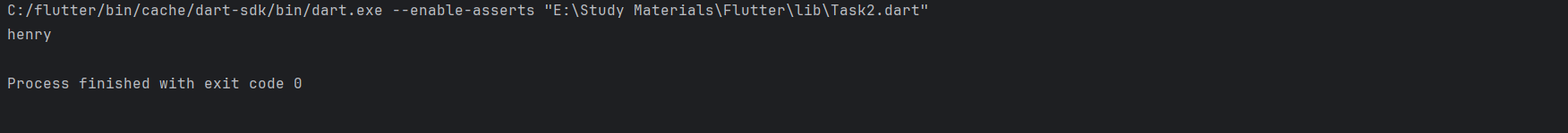


1. **You need a variable whose value won't change after initialization, but its value may be determined at runtime.**

final is the most suitable choice in Dart. It allows you to declare a variable whose value cannot be changed after initialization, regardless of whether that value is determined at compile-time or runtime.

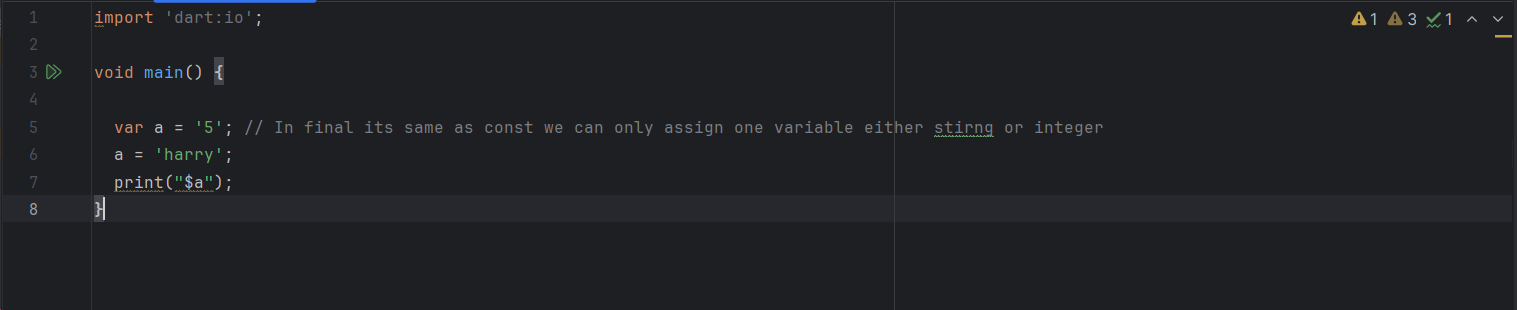


**Ouput**

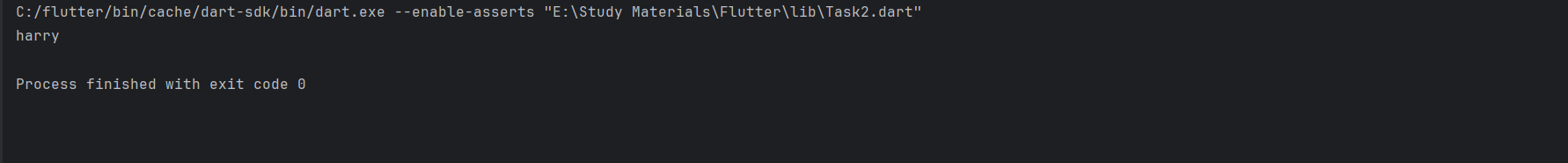


1. **You want the type of a variable to be inferred automatically based on its value.**

“Var” allows the type of the variable to be determined implicitly by the type of the value assigned to it at compile time**.**

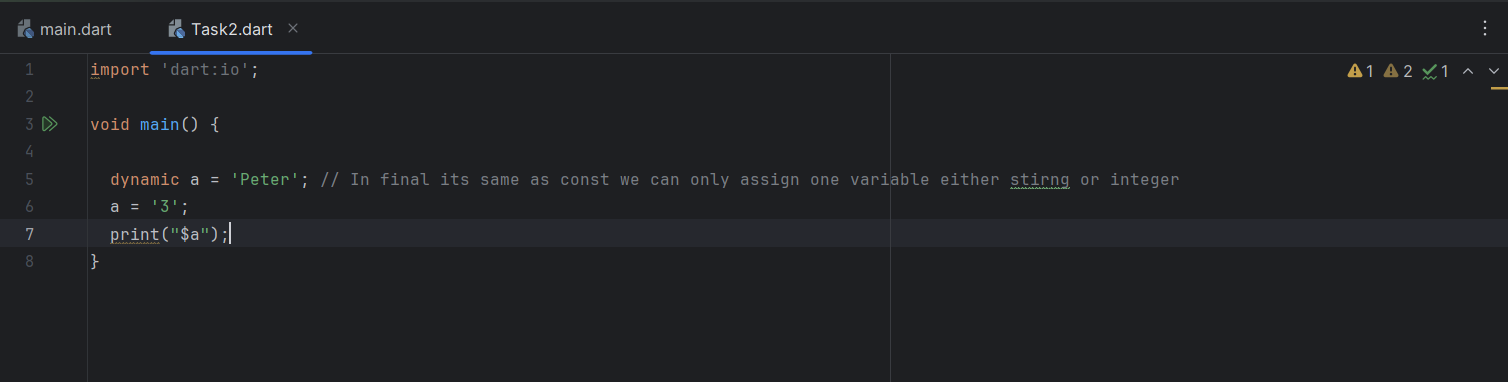
****

**Output**

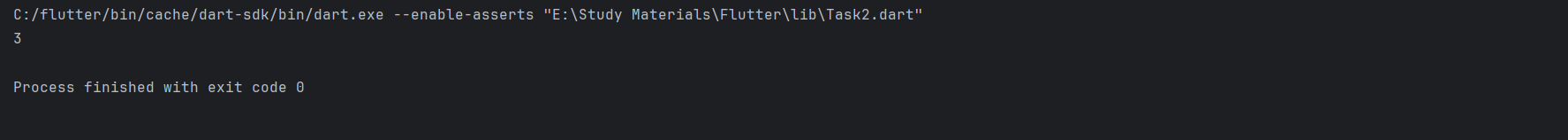
****

1. **You're working with a variable whose type may change dynamically during runtime. Choose the appropriate keyword for each scenario and justify your choice.**

The keyword “dynamic”. Think of it like having a container that can hold different types of things and can switch between them as needed, providing the flexibility you need for your variable.



Output



**Question 2**

**Number Guessing Game**

In this exercise, you'll create a simple number guessing game where the player has to guess a randomly generated number within a certain range. The game will provide feedback to the player based on their guesses, and it will use loops to allow multiple attempts.

**Instructions:**

∙ The program generates a random number between 1 and 100 (inclusive).

∙ It prompts the player to guess the number.

∙ After each guess, it provides feedback whether the guess is too high, too low, or correct. ∙ The player can continue guessing until they guess the correct number.

∙ The program keeps track of the number of attempts.

∙ Once the correct number is guessed, the program congratulates the player and displays the number of attempts.

DEPARTMENT OF SOFTWARE ENGINEERING MOBILE APPLICATION DEVELOPMENT (SE-487)

**Exercise Steps:**

∙ Implement the askForGuess() function to prompt the player for their guess and return the parsed integer.

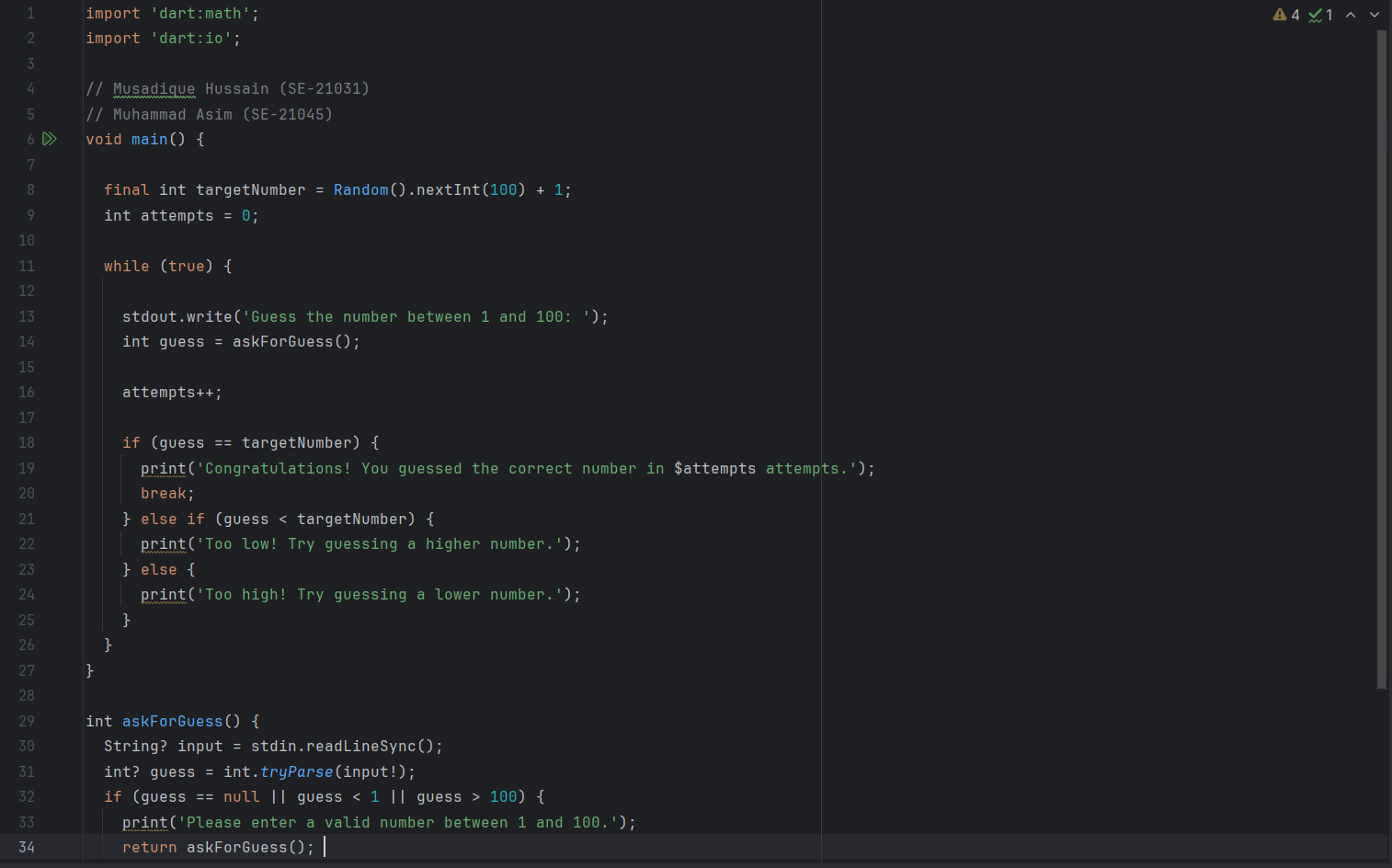
∙ Use a while loop to repeatedly prompt the player for guesses until they guess the correct number.

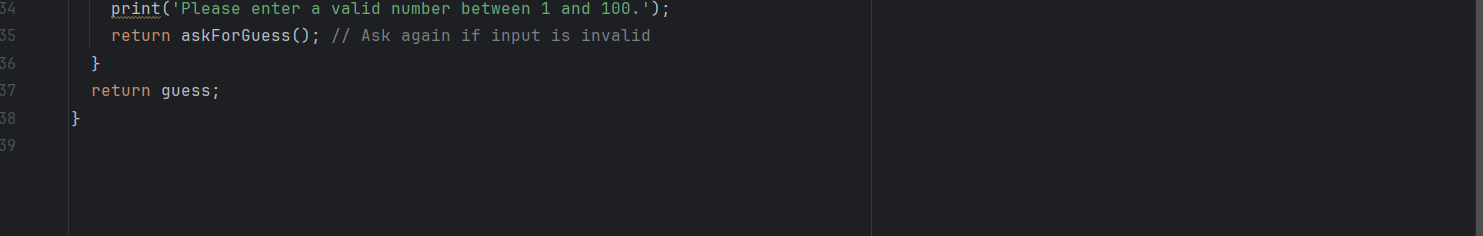
∙ Provide appropriate feedback to the player after each guess, indicating whether the guess is too high, too low, or correct.

∙ Keep track of the number of attempts.

∙ When the correct number is guessed, exit the loop and congratulate the player.

**Code**

****



**Output**

