**R for loop:**

A for loop is a control flow statement that allows code to be executed repeatedly as long as a specified condition is true. The syntax of a for loop is as follows:

Code snippet

for (i in sequence) {

# code to be executed repeatedly

}

The sequence is an object of elements and i takes in each of those elements. In each iteration, the code inside the loop body is executed.

For example, the following code prints the numbers from 1 to 10:

Code snippet

# initialize a variable

i <\- 1

# for loop to print numbers from 1 to 10

for (i in 1:10) {

print(i)

}

The break statement can be used to terminate a for loop early. The syntax of the break statement is as follows:

Code snippet

break

The break statement can be used anywhere inside a for loop body. When the break statement is executed, the loop terminates immediately.

The next statement can be used to skip the current iteration of a for loop. The syntax of the next statement is as follows:

Code snippet

next

The next statement can be used anywhere inside a for loop body. When the next statement is executed, the current iteration of the loop is skipped and the next iteration is started.

Here is an example of a for loop with a break statement:

Code snippet

# for loop to print numbers from 1 to 10, but break when i is 5

for (i in 1:10) {

if (i == 5) {

break

}

print(i)

}

Here is an example of a for loop with a next statement:

Code snippet

# for loop to print odd numbers from 1 to 10

for (i in 1:10) {

if (i %% 2 == 0) {

next

}

print(i)

}

Nested for loops

You can include a for loop inside another for loop to create a nested loop.

For example, the following code prints all the combinations of two numbers from 1 to 3:

Code snippet

# for loop to print all combinations of two numbers from 1 to 3

for (i in 1:3) {

for (j in 1:3) {

print(paste(i, j))

}

}

This code will print the following output:

Code snippet

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

Xxxxxxx—-------------xxxxxxxx

Sure, here is a lecture note on R for loop:

What is a for loop?

A for loop is a control flow statement that allows code to be executed repeatedly for a specific number of times. The syntax of a for loop is as follows:

Code snippet

for (i in 1:10) {

# code to be executed 10 times

}

The variable i is initialized to 1 and is incremented by 1 after each iteration. The code inside the loop body is executed 10 times.

Example of a for loop

The following code prints the numbers from 1 to 10:

Code snippet

for (i in 1:10) {

print(i)

}

Break statement

The break statement can be used to terminate a for loop early. The syntax of the break statement is as follows:

Code snippet

break

The break statement can be used anywhere inside a for loop body. When the break statement is executed, the loop terminates immediately.

Next statement

The next statement can be used to skip the current iteration of a for loop. The syntax of the next statement is as follows:

Code snippet

next

The next statement can be used anywhere inside a for loop body. When the next statement is executed, the current iteration of the loop is skipped and the next iteration is started.

Exercises

1. Write a for loop that prints the even numbers from 2 to 10.
2. Write a for loop that prints the prime numbers from 2 to 100.
3. Write a for loop that asks the user for a number and prints "Too low" if the number is less than 10, "Too high" if the number is greater than 10, and "Just right" if the number is equal to 10.

Xxxxxx—--------xxxxxxx

Here's a lecture note on R for loop:

A for loop is a control flow statement that allows code to be executed repeatedly as long as a specified condition is true. The syntax of a for loop is as follows:

Code snippet

for (value in sequence) {

# code to be executed repeatedly

}

The sequence is an object of elements and value takes in each of those elements. In each iteration, the block of code is executed. For example,

Code snippet

numbers = c(1, 2, 3, 4, 5)

# for loop to print all elements in numbers

for (x in numbers) {

print(x)

}

Output

Code snippet

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

In this program, we have used a for loop to iterate through a sequence of numbers called numbers. In each iteration, the variable x stores the element from the sequence and the block of code is executed.

Example 1: Count the Number of Even Numbers Let's use a for loop to count the number of even numbers stored inside a vector of numbers.

Code snippet

# vector of numbers

num = c(2, 3, 12, 14, 5, 19, 23, 64)

# variable to store the count of even numbers

count = 0

# for loop to count even numbers

for (i in num) {

# check if i is even

if (i %% 2 == 0) {

count = count + 1

}

}

print(count)

Output

Code snippet

[1] 4

In this program, we have used a for loop to count the number of even numbers in the num vector. Here is how this program works:

We first initialized the count variable to 0. We use this variable to store the count of even numbers in the num vector. We then use a for loop to iterate through the num vector using the variable i.

Code snippet

for (i in num) {

# code block

}

Inside the for loop, we check if each element is divisible by 2 or not. If yes, then we increment count by 1.

Code snippet

if (i %% 2 == 0) {

count = count + 1

}

Example 2: for Loop With break Statement You can use the break statement to exit from the for loop in any iteration. For example,

Code snippet

# vector of numbers

numbers = c(2, 3, 12, 14, 5, 19, 23, 64)

# for loop with break

for (i in numbers) {

# break the loop if number is 5

if( i == 5) {

break

}

print(i)

}

Output

Code snippet

[1] 2

[1] 3

[1] 12

[1] 14

Here, we have used an if statement inside the for loop. If the current element is equal to 5, we break the loop using the break statement. After this, no iteration will be executed.

Example 3: for Loop With next Statement Instead of terminating the loop, you can skip an iteration using the next statement. For example,

Code snippet

# vector of numbers

numbers = c(2, 3, 12, 14, 5, 19, 23, 64)

# for loop with next

for (i in numbers) {

# use next to skip odd numbers

if( i %% 2 != 0) {

next

}

print(i)

}

Output

Code snippet

[1] 2

[1] 12

[1] 14

[1] 64

Here, we have used an if statement inside the for loop to check for odd numbers. If the number is odd, we skip the iteration using the next statement and print only even numbers.

Nested for Loops You can include a for loop inside another for loop to create a nested loop.

Consider the example below. Suppose we have two sequences of numbers. We want to print all the combinations where the sum of numbers in both the sequences is even.

Code snippet

# vector