[Dart Programming Language - Lecture [4]

[Basics of Loops]

1. Introduction

In any programming language one of the first topics to be covered is loops, this is a sign for their importance, allowing you to execute a block of code repeatedly based on certain conditions. In Dart, like in most programming languages, loops help you iterate over a collection of data, perform repetitive tasks, or execute code until a specific condition is met.

Learning Objectives

1. Understand the basics of loops.
2. Understand the differences between the types of loops
3. Be able to decide when to use loops and what type of loop to use

2. Context

Technical Background

Loops are powerful tools in Dart, providing flexibility and control over how and when code is executed repeatedly. Understanding the different types of loops and how to use them effectively is crucial for writing efficient and clean code. Whether you're iterating over a collection, performing repetitive tasks, or executing code based on a condition, loops are indispensable in programming. Relevance to Course

Key Terms and Definitions

1. Loop: execute a block of code repeatedly based on certain conditions.
2. Condition: a statement that is either true or false
3. Index: a variable used to track the current iteration of the loop

3. Main Content

1. For loop

The for loop is the most famous type of loops, and it is also the one that is usually used in almost all cases, it is used when you know beforehand how many times you want to iterate over a block of code.

```
for (initialization; condition; increment/decrement)
{
 // Code to execute
 }
```

What is initialization; it is the process of defining an index to determine from where the loop will start.

What is condition: condition is the statement that will make the loop stops if this condition is false. In another word, the loop will continue as long as the condition is true.

What is increment/decrement: it determines whether the loop will count up or count down.

```
void main()
{
for (int i = 0; i <= 5; i++)
        {
         print('Iteration $i');
        }
}
```

In this for loop, we defined an index that starts from 0, and the condition is that this variable is less than or equal 5, and it will count up from 0 to 5. And in each iteration it will print (Iteration $i) this means it will print the text “iteration” followed by the value of the variable i.

So the output will be as follows:
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5

There is another form of for, called for – in, and it will be studied in later lectures.

2. While:

While loops are the second most used form of loops. The while loop continues to execute as long as the specified condition is true.

```
while (condition)
 {
 // Code to execute
}
```

It is useful when the number of iterations is not known beforehand and depends on some condition that may change during the loop's execution.

```
void main()
{
 int i = 0;
 while (i < 5) {
 print('Iteration $i');
 i++;
} }
```

This code will produce the same result as the for loop.

There is another form of while called do-while and it not used, so we will not learn about it

3. Break and Continue:

Sometimes when we are iterating through a loop, we do not want the loop to finish until the end, or sometimes we don't want to execute a specific iteration, break and continue are used for these cases.

Let's take a look at this example.

```
void main()
{
for (int i = 0; i <= 5; i++)
        {
          if (i == 3)
            {
              print("this loop will stop here");
              break;
            }
          print('Iteration $i');
        }
}
```

If we follow this code the output will be
Iteration 0
Iteration 1
Iteration 2
this loop will stop here

In the iteration where i = 3 the loop will break.

```
void main()
{
for (int i = 0; i <= 5; i++)
        {
          if (i == 3)
            {
              print(“this loop will skip this iteration”);
              continue;
            }
          print('Iteration $i');
        }
}
```

In this case, the output will be as follows.

Iteration 0
Iteration 1
Iteration 2
this loop will skip this iteration
Iteration 4
Iteration 5

Break and continue are very important in controlling loops and making them more efficient

4. Examples:

One of the most important challenges a beginner programmer should overcome is having fear on looking at code that he does not understand. To solve this problem, the programmer should learn how to analyze code slowly and thoroughly.

Here, we look at an example code, and the challenge is analyzing it and understand how does it work and what it achieves.

```
void main()
 {
 String str = "Dart programming";
 String vowels = "aeiouAEIOU";
 String? firstVowel;
 for (int i = 0; i < str.length; i++)
           {
           if (vowels.contains(str[i]))
                  {
                  firstVowel = str[i];
                  break;
                  }
            }
 if (firstVowel != null)
          {
          print('First vowel in the string is $firstVowel');
          }
else
         {
 print('No vowels found in the string');
          }
 }
```

Another example,

```
void main()
{
int a = 48;
int b = 18;
while (a != b)
 {
          if (a > b)
         {
         a = a - b;
          }
        else
         {
         b = b-  a;
        }
 }
 print('GCD is $a');
}
```

One more example:

```
void main()
 {
 String str = "racecar";
 int left = 0;
 int right = str.length - 1;
 bool isPalindrome = true;
   while (left < right)
       {
          if (str[left] != str[right])
           {
               isPalindrome = false;
               break;
           }
           left++;
          right--;
       }
    if (isPalindrome)
      {
         print('$str is a palindrome');
      }
    else
       {
          print('$str is not a palindrome');
       }
}
```

4. Practical Examples

Assignment 1: [Rewrite the code]

Rewrite the previous codes using different type of loop, it the example uses for, solve it using while. And the opposite

5. Homework/Assignments

Due date 29/8/2024