Lesson00 - Loops (while loops)

This lesson is intended to be taught to high school students, grades 10-12. The materials are presented in Java, so it can be used for either AP CS A or an Introduction to Computer Science class (either in Java, or adapted to another language).

Computer Science and Digital Fluency Standards: 9-12.CT.7, 9-12.CT.8

Overview

Students start with a quick excercise reviewing if/else statements and conditions in the Do Now. This will lead into the introduction of loops through a lecture using a few slides. Students will get the formal definition/ explanation of the concept and then see a few examples in the slides.

I think it will be good to have code tracing problems for them to work on at this point, in order for them to really get into thinking about how computers act when they are executing a while loop. (It helps them understand what it means to be the “iterator”, which will be good for the next lesson about for loops). Being successful at code tracing will also give students confidence that they understand the concept, and is also a good way for the teacher to get a pulse of the room.

It will be important to show students an example code of a loop in action, which is why the lesson will then shift to a live-coding demo. I also have a few physical examples that could be useful in order to express it in a kinetic or visual way (that could be used during the introduction or during the powerpoint slides), but ultimately I think students seeing the code itself will be very important. By using a little bit of a more intermediate/lenghtier program compared to the code tracing problems, students are able to see how while loops can fit into a larger program and why it is so essential.

At the very end, there will be a short exit ticket where students write a code block utilizing a while statement, related to a question posed in the Do Now. This will be a good culmination of the lesson, and the teacher will be able to assess whether or not students have been able to grasp the concept of while loops.

Idea for a follow-up lesson/homework:
There might even be room for scaffolded code similar to what I thought up for 06_scaffold, where I had heavily scaffolded Game of Nim that was focused on having students fill in the blanks in the program – filling in the conditions for the conditional statements/loops in the code. This could be added at the end of the lesson if students speed through the code tracing problems, or if the lesson ends up taking class periods.

Lesson Structure

Intro (3 - 5 min)

Aim: What is a loop in computer programming? How can we use a “while” statement to make loops?

Do Now:

Review on if/else statements – with a focus on the condition. This will lead into the main lesson (conditions of a while loop), and will also be related to the exit ticket. (included in slides)

Short powerpoint slide/lecture to introduce loops (7 - 10 min)

Materials in folder. Can also use a physical example or two during the explanation. For example, act out the difference between:

```
while(lights are off){
   turn on lights;
}
```

and

```
while (true)
        {      //or while(it is xth period)  with x being the
        current period
    if(lights are off){
      turn on lights;
      }
}
```

Code tracing (10-15 min)

Materials in folder
Students work in their table groups to complete a few short code tracing problems centered on while loops. One problem will be done as an entire class as an example. Afterwards, review a couple of the problems.

Live coding activity (15 min)

Materials in folder
Using a live-code demonstration, students will be shown an example of a while loop in action. This section can also be prefaced by showing examples from the code tracing problems

Exit ticket (3-5 min/whatever time is left)

Follow up to the Do Now

Loops!

Intro to CS

Aim:

What is a loop in computer programming?
How can we use the "while" statement to make loops?

Do Now:

```
int x = 10
If (x = 10){
System.out.println("My name is Jeff");
}
```

What does the code above do? Write out the output.
Change the code so that the print statement is executed 5 times.

Consider: How would you make a computer execute that print statement 100 times?

What exactly is a loop?

- Computers are pretty good at following instructions. Loops are a way for us to tell the computer these instructions so that it can execute lines of code many times – as many times as we see fit.

- Running the same code multiple times like this is called **iteration**.

- Similar to the if-else statements we have seen in the past, computer loops use **conditions** in order to figure out how many times it should execute the code.

- ***While loops*** and ***For loops*** are two of the most common ways to do iteration in computer programming. Today, we will be focusing on while loops!

While Loops

While the nail isn't all the way in, hammer it!

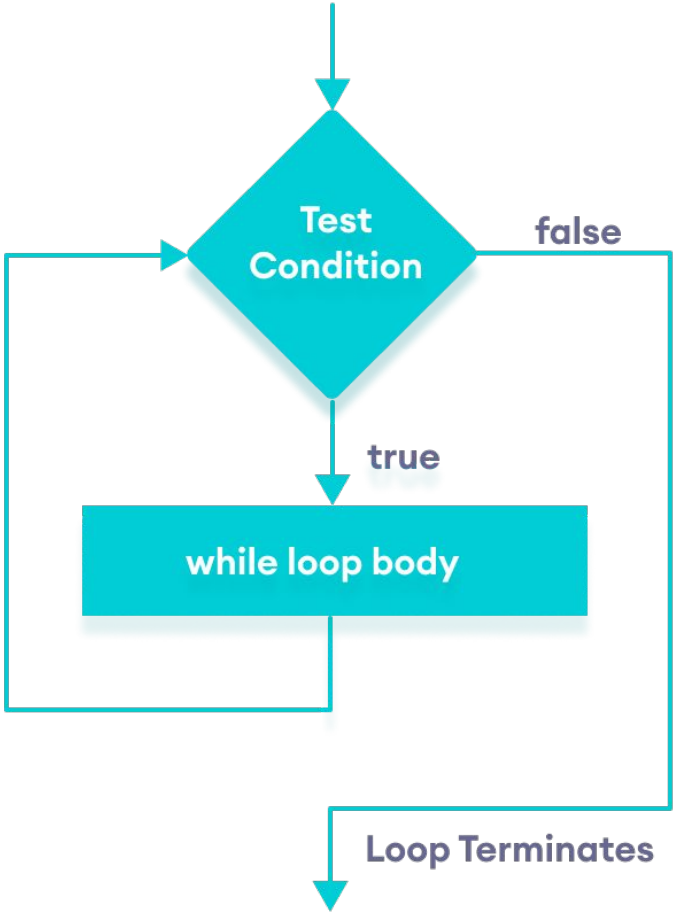
You can read while loops almost as if they were in English:

```
while  (this thing is true) {
        do this thing!
        }  // end loop
```

A few notes:

- As long as the **condition** is true, this part of the program will continue to run.
- Of course, if the **condition never becomes false,** the computer will run that piece of code forever! This is called an **infinite loop**.
- You will often see a **loop variable** in loops, specifically the letter "**i**" which stands for iterator.

We will look at a example in java next!


Test
Condition
false
true
while loop body
Loop Terminates

Using increment/decrements to iterate

```
int iterator = 0;
while (iterator < 5) {
        System.out.println("Hello world!");
        iterator = iterator +1;
}
```

We've established that the while loop will run forever until the condition is false. This means that in order to tell the computer how many times we want the code inside the while loop to run, **we need to figure out a way to turn the condition from true to false.**

One way to do this is by using a loop variable/iterator. We create an integer variable named "iterator" and assign it the value of 0. At the end of the loop body, we can add 1 to our iterator. This way, everytime we run through the loop body, iterator goes up by 1.

In the above example, the iterator starts at value 0, but after the computer prints out "Hello world!", the iterator goes up to 1, (and since 1 is still less than 5), it prints it out again, then the iterator goes up to 2, (and since 2 is still less than 5), it prints again, etc.

Once iterator reaches 5, it will no longer be less than 5, and the computer will no longer go through the while loop.

Increment/decrements contd.

```
int iterator = 5;
while (iterator > 0) {
        System.out.println(“Hello world!”);
        iterator = iterator - 1;
}
```

This does the same exact thing as the previous, but notice we have the iterator start at 5, then subtract 1 from the iterator at the end of the loop body. This is called **decrementing.**

Break; and Continue;

There are a few ways to break out of a loop. If the condition is false, the program will exit the loop and continue down the rest of the program.

Another way is to use **break;** – which literally “breaks” out of the loop at any given point.

```
 int iterator = 1;
while (iterator <= 5) {
        System.out.println(“Hello world!”);
         iterator++;
         break;
}
```

This piece of code will only print “Hello world!” one time.

We start the while loop and print it out once and then increment the iterator, but before we get a chance to execute the body of the loop again, we break out of the loop.

```
 int iterator = 1;
while (iterator <= 5) {
        if (iterator == 3){
        break;
        }
        else
            System.out.println(“Hello world!”);
             iterator++;
}
```

What happens when this runs?

continue; is the opposite – if you write in a program, it will just continue onto the next iteration

Vocabulary

Iteration:
Executing a sequence of statements repeatedly.

Loop:
A statement that executes a sequence of statements repeatedly.

Loop body:
The statements inside the loop.

Infinite loop:
A loop whose condition is always true.

Loop variable: (will usually be the iterator “i”)
A variable that is initialized, tested, and updated in order to control a loop.

Increment:
Increase the value of a variable.

Decrement:
Decrease the value of a variable.

Break:
Exits the loop.

Continue:
Moves onto the next iteration.

Practice in your groups – Code Tracing

Let’s do #1 together!

We’ll be reviewing 1, 2, 6, and 7 together.

Live code!

Follow along – you can code alongside me or you can just pay attention and take notes.

Exit Ticket

Here is the piece of code from your Do Now today.

```
int x = 10
If (x = 10){
        System.out.println("My name is Jeff");
        }
```

Write a while loop that prints "My name is Jeff" 100 times!

Trace through each piece of code and write out what it the computer will output!

Q1)

```
int i = 0;
while (i < 3){
  System.out.println("Hi!");
i = i + 1;
}
```

What does the code output?
answer:

Q2)

```
int i = 3;
while (i > 0){
  System.out.println("Hi!");
i = i - 1;
}
System.out.println("Bye!");
```

What does the code output?
answer:

Q3)

```
int i = 3;
while (i < 6) {
  System.out.println(i);
  i += 1;    // i+=1 is the same as i = i +1;
}
```

What does the code output?
answer:

Q4)

```
int i = 1;
while (i <= 5) {
    System.out.println(i);
    i++;          // i++ is the same as i +=1;
}
```

What does the code output?
answer:

Q5)

```java
int i = 10;
while (i >= 1) {
    System.out.println(i);
    i--;
}
```

What does the code output?
answer:

Q6)

```java
int i = 0;
while (i <= 10) {
    System.out.println(i*2);
}
```

What does the code output?
answer:

Q7)

```java
int x = 3;
int i = 0;
while (i < 3) {
  x += 1;
  i += 1;
}
System.out.println(x);
```

What does the code output?
answer:

table provided: |i||x|
0 3

Teacher version w/ solutions

Trace through each piece of code and write out what it the computer will output!

Q1)

```
int i = 0;
while (i < 3){
  System.out.println("Hi!");
i = i + 1;
}
```

What does the code output?
answer:
Hi!
Hi!
Hi!

Q2)

```
int i = 3;
while (i > 0){
  System.out.println("Hi!");
i = i - 1;
}
System.out.println("Bye!");
```

What does the code output?
answer:
Hi!
Hi!
Hi!
Bye!

Q3)

```
int i = 3;
while (i < 6) {
  System.out.println(i);
  i += 1;    // i+=1 is the same as i = i +1;
}
```

What does the code output?
answer:
3
4
5

Q4)

```java
int i = 1;
while (i <= 5) {
    System.out.println(i);
    i++;          // i++ is the same as i +=1;
}
```

What does the code output?
answer:
1
2
3
4
5

Q5)

```java
int i = 10;
while (i >= 1) {
    System.out.println(i);
    i--;
}
```

What does the code output?
answer:
10
9
8
7
6
5
4
3
2
1

Q6)

```java
int i = 0;
while (i <= 10) {
    System.out.println(i*2);
}
```

What does the code output?
answer:
the computer will keep on printing out the number 2 on a new line. Infinite loop.

Q7)

```java
int x = 3;
int i = 0;
while (i < 3) {
```

```
  x += 1;
  i += 1;
}
System.out.println(x);
```

What does the code output?
answer:
6

table provided: (students will receive just the first row)
|i||x|
0 3
1 4
2 5
3 6

Live-Code Blueprint

```java
import java.io.*;
import java.util.Scanner;
import java.util.Random;    //make sure to provide this!

public class Main {

    public static void main(String[] args) {      //provide
        Scanner scanner = new Scanner(System.in); //provide
        Random random = new Random();      //provide

        int secretNumber = random.nextInt(20) + 1; // Review with students how random
        works again. Student-prompt: How do we make the computer choose a random number
        between 1 and 20 (not including 0)?
        int attempts = 0;      //provide this and tell students that it will be used to
        keep track of the number of attempts the user makes.

        int guess = 0;    //this is part of the Big idea, either skip this line and then
        go back, or initialize guess without assigning a value to it and then continue
        (will be a deliberate-error).

        System.out.println("Welcome to the Guessing Game!");              //provide these
        2 print statements
        System.out.println("I have chosen a number between 1 and 20. Can you guess it?");

        while (guess != secretNumber)     //BIG IDEA/MUST-ANSWER-Q: Ask students what the
        condition of the while loop needs to be. "When should we tell the program to
        stop?"

        {
            System.out.println();
            System.out.print("Enter your guess: ");        //walk through the print
        statements with students with small student-prompts
            guess = scanner.nextInt();
            attempts++;      //student prompt: "Now that the user's attempt has gone up,
        what should we change here?" -> concept of iteration

            if (guess == secretNumber) {      //student-prompt: Ask students for what they
        would input for the if-else statements in the while loop. (include a "too high"
        and "too low" print statement for their respective cases)
                System.out.println("Congratulations! You guessed the correct number in " +
        attempts + " attempts.");
                break;          //this break is technically not needed in this program,
        but include it and explain it/how it can be used
            } else if (guess < secretNumber) {
                System.out.println("Too low! Try again.");
            } else {
                System.out.println("Too high! Try again.");
            }
        }

    }
}

//After the program is done, go through the whole program from start to finish one more
        time.
//If time permits, play around by changing the while loop to see what happens (change the
        while loop condition to true and take out the break to see the infinite loop,
        change the condition to be related to # of attempts similar to videogame
        "lives",etc)
```