Suppose you want to print

1 2 3 4 5 6 ... *till the number n*

* Need a way to execute statements over and over again
* Loops are used to execute statement repeatedly

### **While**

* See example in various languages [here](https://en.wikipedia.org/wiki/While_loop#Demonstrating_while_loops)
* [(Links to an external site.)](https://en.wikipedia.org/wiki/While_loop#Demonstrating_while_loops)
* .
* Of the form

while (<conditional>) {

s1

s2

s3

}

* Executes statements while the condition is true
* Can be followed by block or by a single statement
* Example:

int n = 25

int i = 1;

while (i <= n) {

System.out.printf("%d\t", i++);

}

* Factorial example:

3! = 3 \* 2 \* 1

5! = 5 \* 4 \* 3 \* 2 \* 1

int n = 5;

int factorial = 1;

while (n > 0) {

factorial = n \* factorial;

n = n - 1;

}

System.out.println(factorial);

* Can you simplify the above further?
* If your condition is incorrect, you may get an off-by-one condition (eg. n >= 0)
* If you forget the increment (n = n-1) you will get an **infinite loop**
* An **iteration** is a single execution of the loop statement
* **NOTE**: Do not use floating point numbers in the condition

### **Do-While**

* See examples in various languages [here](https://en.wikipedia.org/wiki/Do_while_loop#Demonstrating_do_while_loops)
* [(Links to an external site.)](https://en.wikipedia.org/wiki/Do_while_loop#Demonstrating_do_while_loops)
* .
* Same as while except executes body and then performs the check
* Of the form:

do {

s1

s2

s3

} while (condition);

* Must be followed by semicolon
* Can contain a block of statements or a single statement
* Sometimes this is more convenient than a while loop
* For example:

do {

System.out.print("Enter Number: ");

num = input.nextInt();

} while (num > 0);

### **For**

* Often we loop on a particular number value, as in:

int i = 1; // initialize

while (i <= n) { // check

...

i++; // increment

}

* This can be more naturally implemented with a for loop as follows:

for (i=0; i < N; i++) {

...

}

* For loops are of the form:

for (initialize control variable(s); check condition; update control variable(s)) {

s1

s2

s3

}

* Can contain a block of statements or a single statement
* Must use parentheses after for
* Best practice is declare control variable along with the initialization; the variable is then not usable outside the loop. For example:

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

...

}

* The factorial example using for loop:

int n = 5;

int factorial = 1;

for (int i = n; i > 0; i--)

factorial \*= i;

System.out.println(factorial);

* Can have multiple control variables and update statements separated by commas.

for (i = 0, j = 0 ; i+j > 0; i++, j++)

...

* Of while, do-while, and for, use the loop construct most natural to the problem.

### **Nested Loops**

* Can have loops within loops, as

// print multiplication table

for (int i = 1; i <= 9; i++) {

System.out.print(i + " | ");

for (int j = 1; j <= 9; j++)

System.out.print(i\*j);

System.out.println();

}

* Beware of the performance implications of having nested loops. For example:

for (i = 0; i < 10000; i++)

for (j = 0; j < 10000; j++)

for (k = 0; k < 10000; k++)

s1;

If statement s1 takes a microsec (1 millionth of a second), the above loop will take 227 hours.

### **Break and Continue**

* You can use a break (last seen in a switch statement) and continue in the body of loops.
* The break statement will exit out of the whole loop.
* The continue statement will exit only the current iteration of the loop.
* Use of neither is recommend and it is always possible to write alternate code with the same effect.

### **Pitfalls**

* Note that "for", "do", "while" are reserved words
* Do not use floating point numbers in the conditions for loops
* Can have infinite loops if not updating control variables in loop
* Can have Off-by-one error if your conditional expression is not correct