



# WIT COMP1000

**for** Loops



# while Loops

- **while** loops are often used to repeat a task a fixed number of times, which leads to a similar structure based on a counter variable

```
int count;  
count = 1;  
while (count <= 8) {  
    System.out.println(count + " squared is " + count*count);  
    count++;  
}
```

counter variable initialization

counter variable boolean expression

counter variable update



# for loops

- **for** loops are specialized loops based on that counter structure

counter variable  
initialization

counter variable  
boolean expression


counter variable  
update

```
int count;  
for (count = 1; count <= 8; count++) {  
    System.out.println(count + " squared is " + count * count);  
}
```



# Generic Form

```
for (INITIALIZATION; BOOLEAN_EXPRESSION; UPDATE) {  
    STATEMENT1;  
    STATEMENT2;  
    ...  
}
```



- INITIALIZATON is done one time, before the first loop iteration
- UPDATE is done every loop iteration after the last loop body statement
- BOOLEAN\_EXPRESSION is checked every loop iteration, after UPDATE (and once after INITIALIZATION)



# Another Example

```
import java.util.Scanner;
import java.util.Random;

public class ForExample {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Random generator = new Random();

        int i;
        for (i = 1; i <= 20; i++) {
            int random_value = generator.nextInt(10);
            System.out.println(random_value);
        }
    }
}
```

Create a new random number generator

Generate a random number between 0 and 9



# Gotchas

---

- There are only two semicolons
  - » Between the initialization step and the boolean expression
  - » Between the boolean expression and the update step
- No semicolon after the update step
- No semicolon after the parentheses
- If you are doing an increment, be sure you do something like `i++`, not just `i+1`
  - » That is, either `i++` or `i=i+1`
  - » Just `i+1` doesn't do anything!



## Exercise

---

- Write a **for** loop that prints all the numbers between 100 and 200 (inclusive, in increasing order)



# Answer

---

```
int i;  
for (i = 100; i <= 200; i++) {  
    System.out.println(i);  
}
```





## for and while

---

- Both kinds of loops work basically the same way
- The only difference is that the initialization and update pieces are part of the **for** syntax directly
- There is no particular benefit to using either loop, so you should use the one that makes the most sense to you in each situation



## Complex Update Steps

---

- The update step can be more complex than a simple increment
- It can be *any* assignment operation
  - » Usually updates the loop variable
  - » Watch out for infinite loops!

```
int x;  
for (x = 10; x >= 0; x--) {  
    System.out.println(x);  
}
```



# Another Example

```
double val;  
double total = 0;  
  
for (val = 1; val <= 1000; val = val * 10) {  
    total = total + val;  
}  
System.out.println(total);
```

	total	val
before <b>for</b> loop	0.0	undefined
after loop initialization	0.0	1.0
after first iteration	1.0	10.0
after second iteration	11.0	100.0
after third iteration	111.0	1000.0
after fourth iteration	1111.0	10000.0
after <b>for</b> loop	1111.0	10000.0



## Exercise

---

- Write a **for** loop that prints all the powers of two between 1 and 1 billion
- Do not use the `Math.pow()` function!
- Think about how to get from one power of two to the next
  - » 1, 2, 4, 8, 16, 32, 64, ...



# Answer

---

```
int x;  
  
for (x = 1; x <= 1000000000; x = x * 2) {  
    System.out.println(x);  
}
```



## Take Home Points

---

- Use **for** loops when you need to repeat a task a certain number of times
- The counter/iteration variable is initialized, checked, and updated as part of the **for** loop syntax
- Always check your semicolons to be sure they are in the correct place, with no extras