# Inf1B

## Conditionals and Loops[1]

Fiona Mcneill
adapting earlier versions by Perdita Stevens, Ewan Klein, Volker Seeker, et al.

School of Informatics

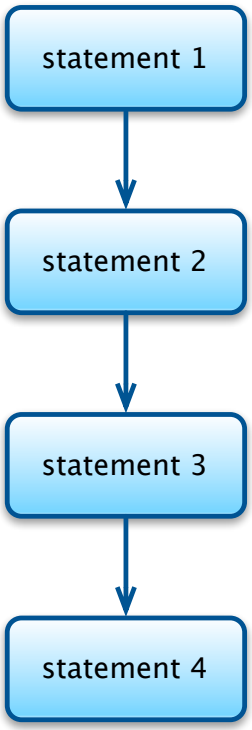
[1] Thanks to Sedgewick&Wayne for much of this content

# Conditional Statements

# Control Flow

Control flow:

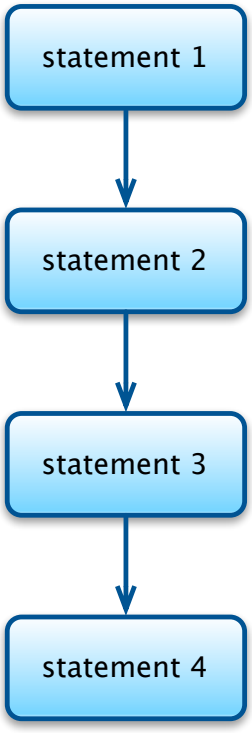
- A sequence of statements that are actually executed in a program

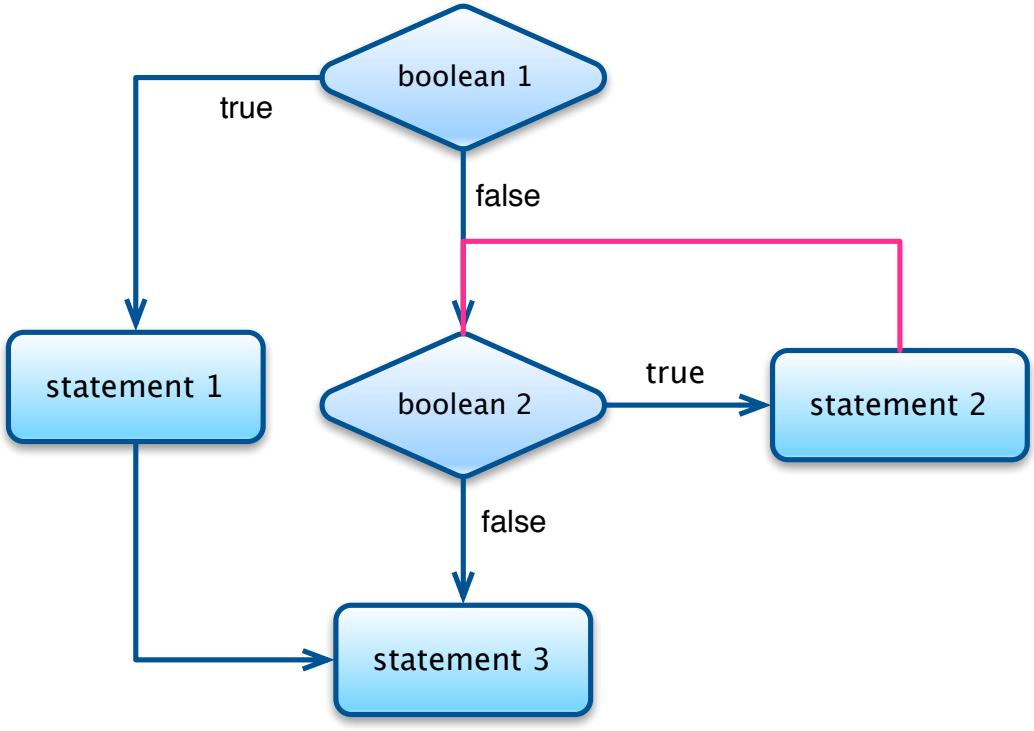


# Control Flow

Control flow:

- A sequence of statements that are actually executed in a program
- Conditionals and loops enable us to choreograph control flow



# If Statement

If / conditional statement:

- Evaluate a `boolean` expression $E$.
- If value of $E$ is `true`, execute some statements.
- If value of $E$ is `false`, execute some other statements — this is the *else* part of a conditional statement.

# If Statement

If / conditional statement:

- Evaluate a `boolean` expression $E$.
- If value of $E$ is `true`, execute some statements.
- If value of $E$ is `false`, execute some other statements — this is the *else* part of a conditional statement.

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

can be any sequence of statements

# If Statement

If / conditional statement:

- Evaluate a `boolean` expression $E$.
- If value of $E$ is `true`, execute some statements.
- If value of $E$ is `false`, execute some other statements — this is the *else* part of a conditional statement.

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

can be any sequence of statements

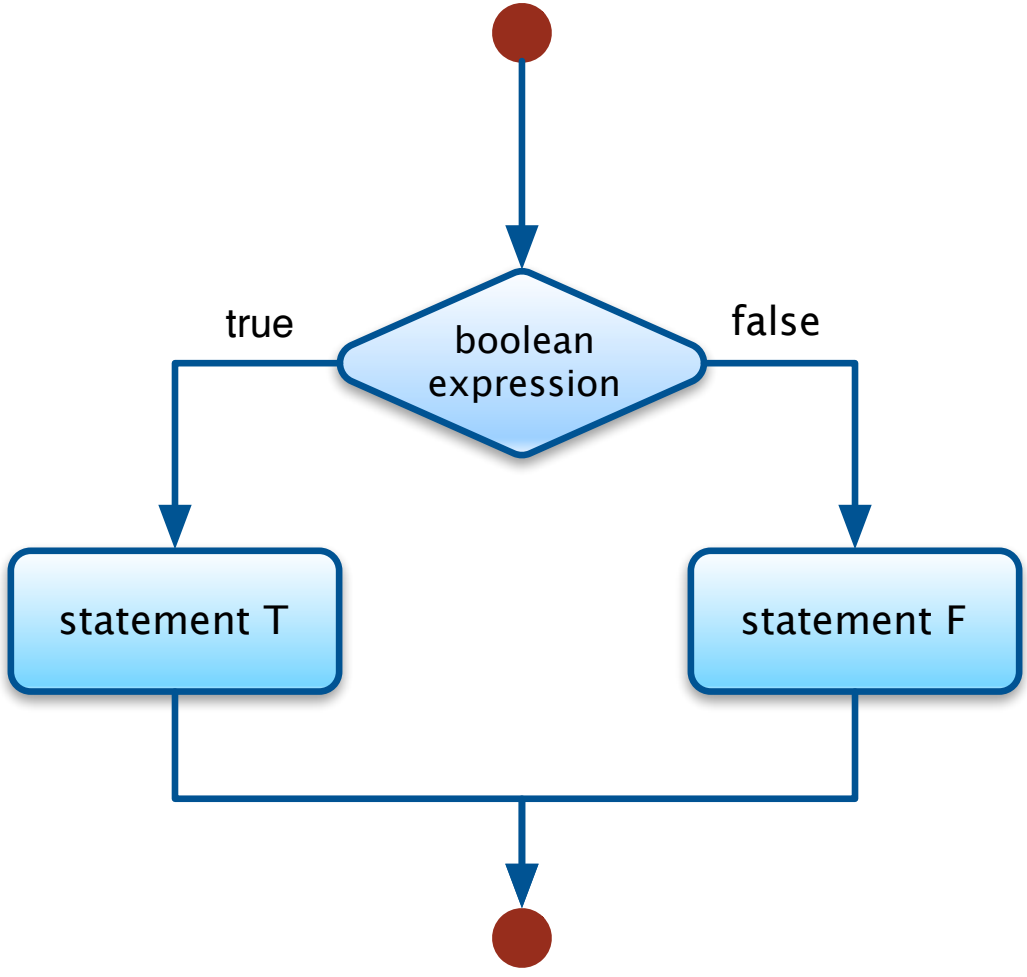
*boolean expression*

```
if (x > y) {

    int t = x;
    x = y;
    y = t;

}
```

*sequence of statements*

# If Statement

If / conditional statement — sometimes called branching structures:

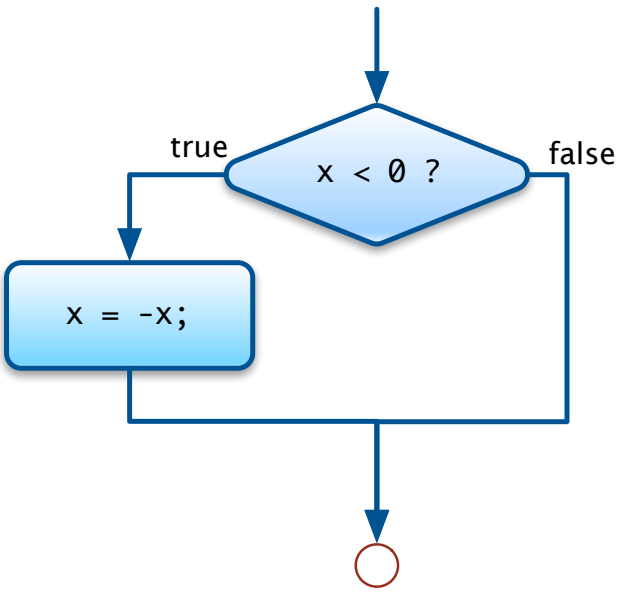


# If Statement

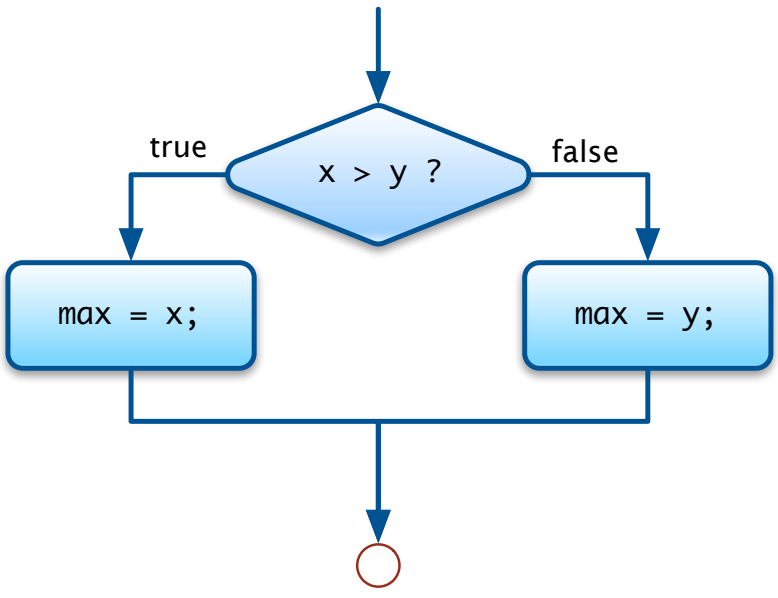
If / conditional statement:

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- If `false`, execute some other statements.

```
if (x < 0) x = -x;
```




```
if (x > y) max = x;
else       max = y;
```



# If Statement: Examples

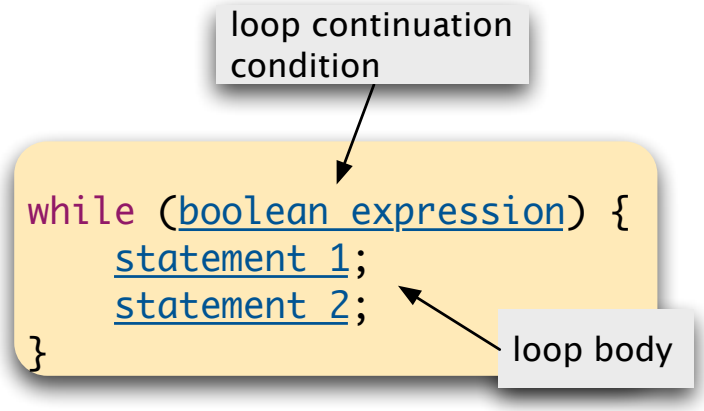
| | |
|---|---|
| *absolute value* | `if (x < 0) x = -x;` |
| *put* `x` *and* `y` *into ascending order (swap)* | `if (x > y) {`<br>`  int temp = x;`<br>`  x = y;`<br>`  y = temp;`<br>`}` |
| *maximum of* `x` *and* `y` | `if (x > y) max = x;`<br>`else  max = y;` |
| *error check for division operation* | `if (den == 0) {`<br>`  System.out.println("Division by zero");`<br>`} else {`<br>`  System.out.println("Quotient = " + num / den);`<br>`}` |

# Loops (While)

# While Loop

The `while` loop is a structure for expressing repetition.

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Repeat.



# While Loop

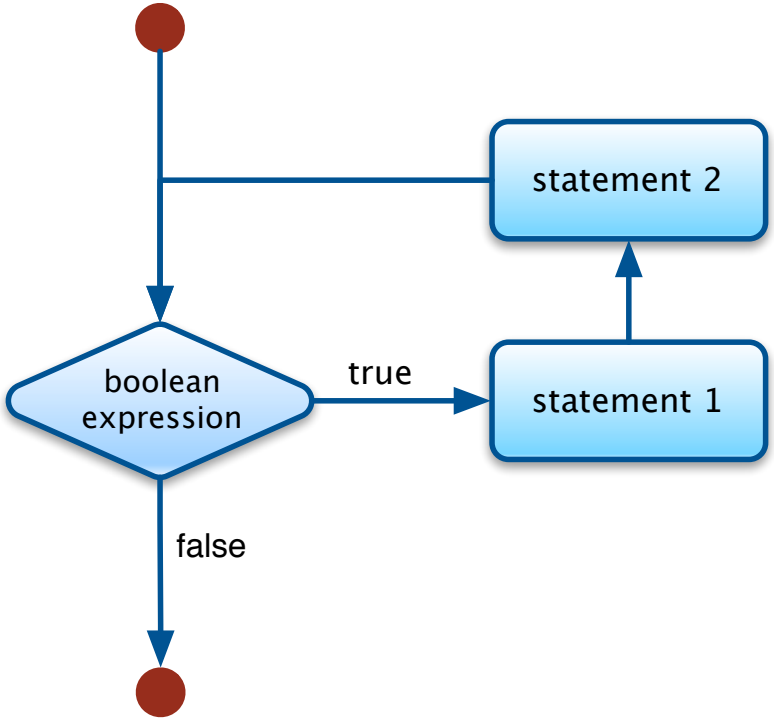
The `while` loop is a structure for expressing repetition.

- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Repeat.

```
while (boolean expression) {
    statement 1;
    statement 2;
}
```

loop continuation condition

loop body



# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

i
0

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val |
|---|---|
| 0 | 1 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n |
|---|---|---|
| 0 | 1 | true |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output |
|---|---|---|---|
| 0 | 1 | true | 0 1 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output |
|---|---|---|---|
| 0 | 1 | true | 0 1 |
| 1 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | | | | |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | true | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | true | 6 | 64 |

Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | true | 6 | 64 |
| 7 | | | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | true | 6 | 64 |
| 7 | 128 | | | |

▸ Start Again

# While Loop: Powers of Two

Print powers of 2 that are $\leq 2^n$ for some $n$. Set `n = 6`.

- Increment loop counter `i` by `1`, from `0` to `n`.
- Double `val` each time.

```
int i = 0;
int val = 1;
while (i <= n) {
  System.out.println(i + " " + val);
  i = i + 1;
  val = 2 * val;
}
```

| i | val | i ≤ n | Output | |
|---|---|---|---|---|
| 0 | 1 | true | 0 | 1 |
| 1 | 2 | true | 1 | 2 |
| 2 | 4 | true | 2 | 4 |
| 3 | 8 | true | 3 | 8 |
| 4 | 16 | true | 4 | 16 |
| 5 | 32 | true | 5 | 32 |
| 6 | 64 | true | 6 | 64 |
| 7 | 128 | false | | |

▸ Start Again

# Powers of Two

```
public class PowersOfTwo {
   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      int i = 0;
      int val = 1;
      while (i <= n) {
         System.out.println(i + " " + val);
         i = i + 1;
         val = 2 * val;
      }
   }
}
```

```
% java PowersOfTwo 3
0 1
1 2
2 4
3 8
```

# While Loop Challenge

Q: Is anything wrong with the following version of `PowersOfTwo`?

```
int i = 0;
int val = 1;
while (i <= n)
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
```

# While Loop Challenge

Q: Is anything wrong with the following version of `PowersOfTwo`?

```
int i = 0;
int val = 1;
while (i <= n)
   System.out.println(i + " " + val);
i = i + 1;
val = 2 * val;
```

A: Need curly braces around statements in `while` loop. Otherwise, only the first of the statements is executed before returning to `while` condition; enters an infinite loop, printing `0 1` for ever.

(How to stop an infinite loop? At the Linux command-line, hit Control-c.)

# The Increment Operator

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
}
```

# The Increment Operator

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
}
```

- standard assignment: `i = i + 1;`
- semantically equivalent shorthand: `i++;`

# The Increment Operator

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i = i + 1;
   val = 2 * val;
}
```

- standard assignment: `i = i + 1;`
- semantically equivalent shorthand: `i++;`

```
int i = 0;
int val = 1;
while (i <= n) {
   System.out.println(i + " " + val);
   i++;
   val = 2 * val;
}
```

# Loops (For)

# For Loop

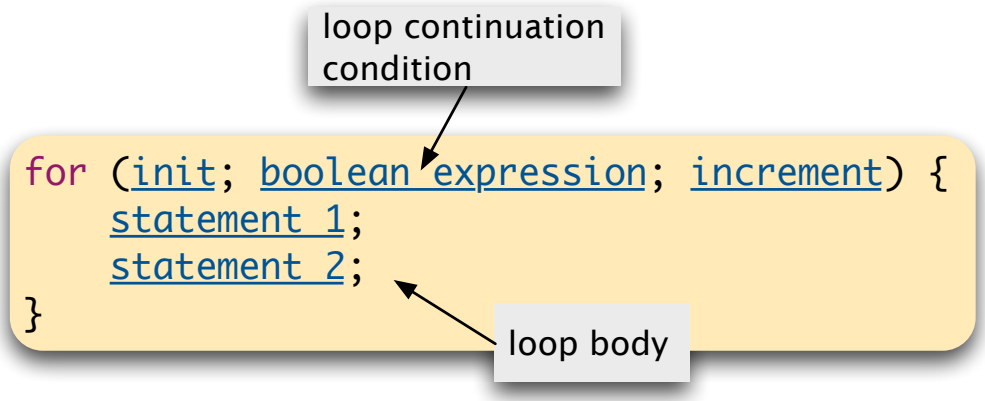
The `for` loop is another common structure for repeating things.

- Execute initialization statement.
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Then execute the increment statement.
- Repeat.

# For Loop

The `for` loop is another common structure for repeating things.

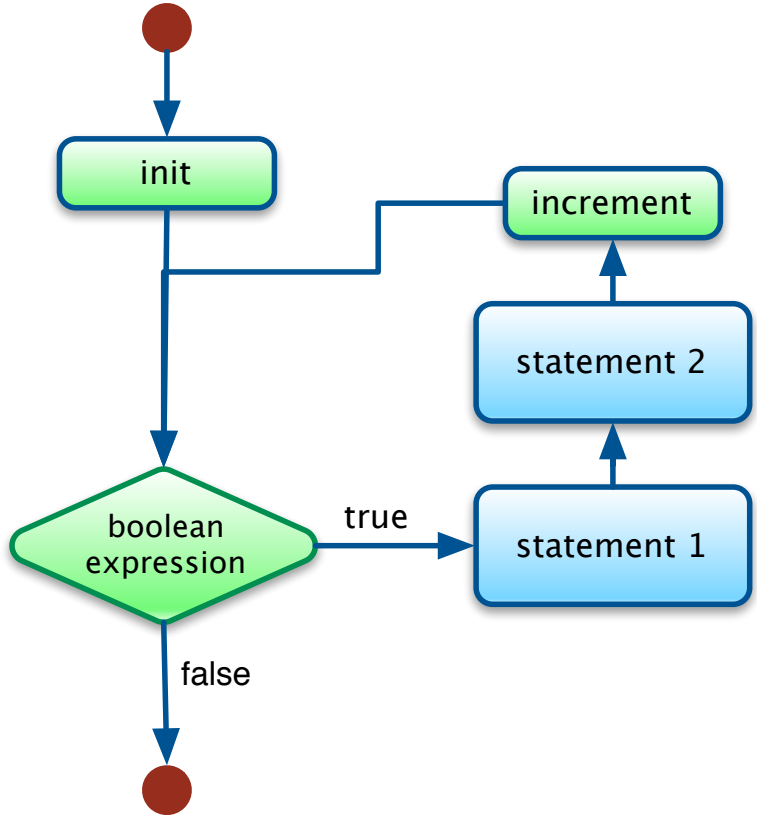
- Execute initialization statement.
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Then execute the increment statement.
- Repeat.



# For Loop

The `for` loop is another common structure for repeating things.

- Execute initialization statement.
- Evaluate a `boolean` expression.
- If `true`, execute some statements.
- Then execute the increment statement.
- Repeat.

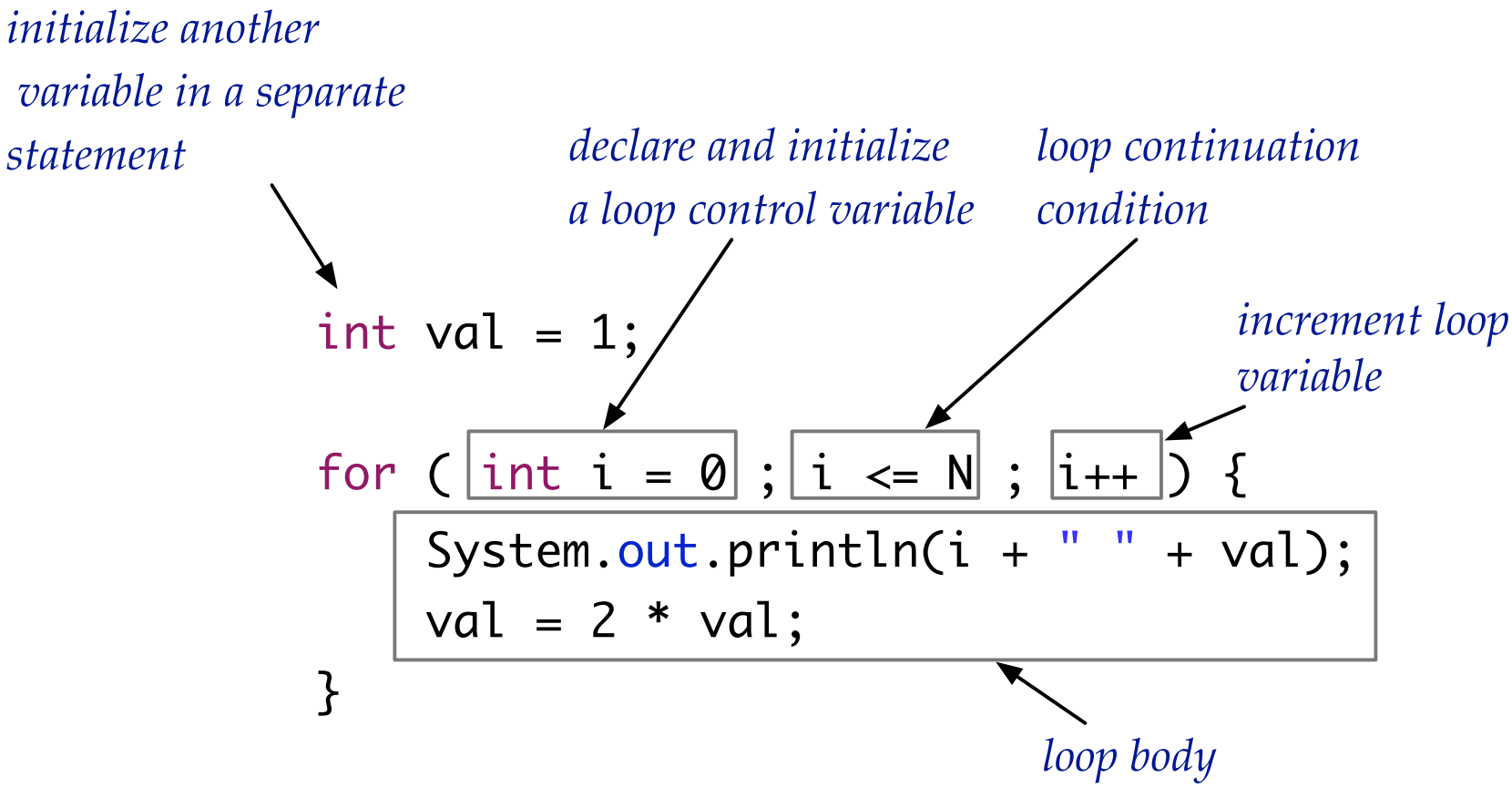



loop continuation condition

```
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```

loop body

# Anatomy of a For Loop



# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

val
1

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i |
|---|---|
| 1 | 0 |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n |
|---|---|---|
| 1 | 0 | true |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | | | | |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | | | | |

▸ Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | | | | |

▸ Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | | |

▸ Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | | | | |

▸ Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | | | |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | true | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | true | 6 | 64 |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | true | 6 | 64 |
| 128 | | | | |

Start Again

# For Loop: Powers of Two

Print the first $n$ powers of 2. Set `n = 6`.

- Double `val` each time.

```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i $\leq$ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | true | 6 | 64 |
| 128 | 7 | | | |

Start Again

# For Loop: Powers of Two

Print the first *n* powers of 2. Set `n = 6`.

- Double `val` each time.

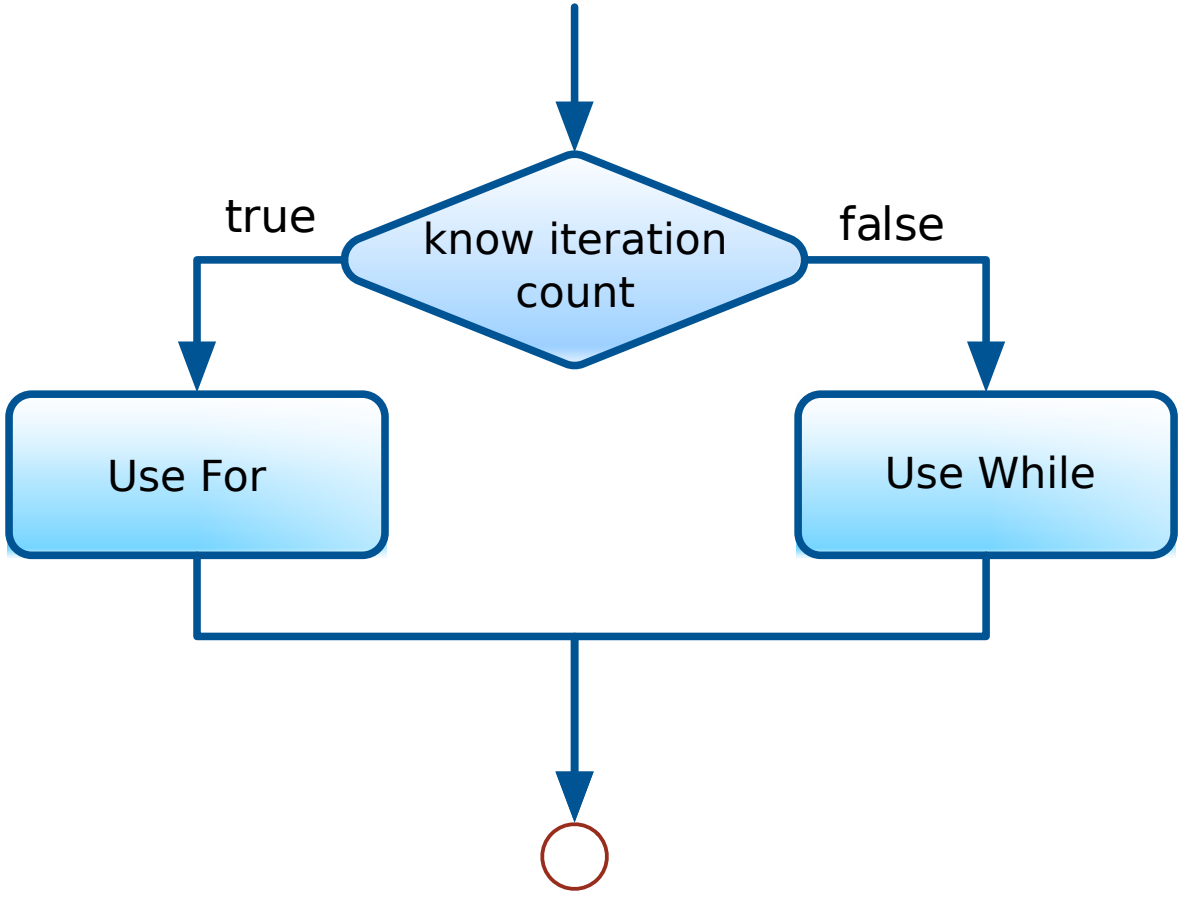
```
int val = 1;
for (int i = 0; i <= n; i++) {
  System.out.println(i + " " + val);
  val = 2 * val;
}
```

| val | i | i ≤ n | Output | |
|---|---|---|---|---|
| 1 | 0 | true | 0 | 1 |
| 2 | 1 | true | 1 | 2 |
| 4 | 2 | true | 2 | 4 |
| 8 | 3 | true | 3 | 8 |
| 16 | 4 | true | 4 | 16 |
| 32 | 5 | true | 5 | 32 |
| 64 | 6 | true | 6 | 64 |
| 128 | 7 | false | | |

▸ Start Again

# When to use While and when to use For?

## Rule of thumb



# Bailing Out Early

# What if you need to leave a loop early?

Sometimes you don't want to end the execution of a loop but instead break out early, e.g. for search algorithms.

`break` allows you to break out of a loop immediately

# What if you need to leave a loop early?

## Breaking out early

```
// find first number divisible by n
int start = 50;
int end = 5000;
int n = 344;
for(int i = start; i < end; i++) {
  if (i % n == 0) {
    System.out.println("Number found: " + i);
    break;
  }
  // some complex calculations
}
```

# What if you need to leave a loop early?

At other times, you might want to skip a loop iteration for certain input and continue with the next one, e.g. when processing data and skipping invalid entries.

`continue` allows you to skip the remainder of the loop body and continue with the next iteration

# What if you need to leave a loop early?

## Skipping iterations

```
// skip numbers divisible by n
int start = 0;
int end = 100;
int n = 5;
for (int i = start; i < end; i++) {
  if (i % n == 0) {
    continue;
  }

  // run some complex calculations
}
```

# A word of caution

It is easy to end up writing complex, hard-to-read, error-prone code using `break` and `continue`.

Keep them for cases like the examples, where the whole loop body is irrelevant for certain values, or after a certain value.

Usually nested conditionals are better style, because they make it easier to reason about the circumstances under which a given line is reached...

# Nested Conditionals

# Nested If Statements

How to classify Scottish weather:

| degrees C | verdict |
|---|---|
| < -5 | wear a sweater |
| -5 to 0 | nippy |
| 1 to 10 | normal |
| > 10 | roastin' |

4 mutually exclusive alternatives

# Nested If Statements

How to classify Scottish weather:

| degrees C | verdict |
|---|---|
| < -5 | wear a sweater |
| -5 to 0 | nippy |
| 1 to 10 | normal |
| > 10 | roastin' |

4 mutually exclusive alternatives

```
String verdict;
if (temp < -5) verdict = "wear a sweater";
else {
    if (temp < 1) verdict = "nippy";
    else {
        if (temp < 11) verdict = "normal";
        else verdict = "roastin'";
    }
}
```

# Nested If Statements

We don't necessarily need all those braces.

```
public class ScottishWeather {
    public static void main(String[] args) {
        String verdict;
        int temp = Integer.parseInt(args[0]);
        if      (temp < -5) verdict = "wear a sweater";
        else if (temp < 1)  verdict = "nippy";
        else if (temp < 11) verdict = "normal";
        else                verdict = "roastin'";
        System.out.println("Verdict: " + verdict);
    }
}
```

## Output

```
% java ScottishWeather -1
Verdict: nippy

% java ScottishWeather 1
Verdict: normal
```

# Nested If Statements

Is there anything wrong with the logic of the following code?

| degrees C | verdict |
|---|---|
| < -5 | wear a sweater |
| -5 to 0 | nippy |
| 1 to 10 | normal |
| > 10 | roastin' |

4 mutually exclusive alternatives

```
String verdict;
int temp = Integer.parseInt(args[0]);
if (temp < -5) verdict = "wear a sweater";
if (temp < 1)  verdict = "nippy";
if (temp < 11) verdict = "normal";
if (temp >= 11) verdict = "roastin'";
```

# Summary

Control flow:

- Sequence of statements that are actually executed in a program run.
- Conditionals and loops: enable us to choreograph the control flow.

| **Control Flow** | **Description** | **Examples** |
|---|---|---|
| straight-line programs | all statements are executed in the order given | |
| conditionals | certain statements are executed depending on the values of certain variables | `if`, `if-else` |
| loops | certain statements are executed repeatedly until certain conditions are met | `while`, `for` |

# Reading

## Java Tutorial

pp68-86, i.e. Chapter 3 *Language Basics* from *Expressions, Statements and Blocks* to the end of the chapter.

## Objects First

Appendix *C.2* - *C.3*, Appendix *D.1* - *D.3*