Inf1-OP
Conditionals and Loops

Timothy Hospedales, adapting earlier version by Perdita Stevens and Ewan Klein

School of Informatics

December 23, 2016

\(^1\)Thanks to Sedgewick&Wayne for much of this content
Learning Outcomes for this week

- Use if and if-else statements to execute a sequence of statements based on the truth value of Boolean expressions.
- Use nested if-else statements to compute results based on a number of mutually exclusive alternatives.
- Use while-loops to repeatedly execute a sequence of statements based on the truth value of Boolean expressions.
- Use for-loops to repeatedly execute a sequence of statements based on an initialization statement, a Boolean test, and an increment statement.
- Use for-loops to compute finite sums and finite products.
A Foundation for Programming

primitive data types
assignment statements
Math
text I/O
conditionals and loops
arrays
functions and modules
objects
graphics, sound and image I/O
arrays
conditionals and loops
Math
text I/O
primitive data types
assignment statements
A Foundation for Programming

- primitive data types
- assignment statements
- Math
- text I/O
- conditionals and loops
- arrays
- graphics, sound and image I/O
- functions and modules
- objects

primitive data types | assignment statements
A Foundation for Programming

- Primitive data types
- Assignment statements
- Math
- Text I/O
- Arrays
- Graphics, sound, and image I/O
- Functions and modules
- Objects
- Conditionals and loops
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 / 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.

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

```java
if (boolean expression) {
    statement T;
} else {
    statement F;
}
can be any sequence of statements
```

Example:

```java
if (x > y) {
    int t = x;
    x = y;
    y = t;
}
```
If Statement

If / conditional statement — sometimes called branching structures:

```
If boolean expression Then
    statement T
Else
    statement F
End If
```
If Statement

If / conditional statement:

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

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

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

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

true
false
```

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

true
false
# If Statement: Examples

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

Diagram:
- Boolean expression
- True flow to statement 1
- False flow to 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.

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

Output:

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
</tr>
</tbody>
</table>

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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
</tr>
</tbody>
</table>
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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ( \leq n )</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ( \leq ) n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

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

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i $\leq$ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>0 1 2</td>
</tr>
</tbody>
</table>
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.

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

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i $\leq$ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ( \leq n )</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

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;
}
```
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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>( i \leq n )</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
</tbody>
</table>
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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i $\leq$ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td></td>
<td>4 16</td>
</tr>
</tbody>
</table>

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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ( \leq n )</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i \leq n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
</tbody>
</table>
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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$val$</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td>6 64</td>
</tr>
</tbody>
</table>

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 \( \text{val} \) each time.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i ≤ n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td>6 64</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>$i \leq n$</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td>6 64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

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;
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>val</th>
<th>i \leq n</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
<td>1 2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
<td>2 4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
<td>3 8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
<td>4 16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
<td>5 32</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
<td>6 64</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
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
Q: Is anything wrong with the following version of PowersOfTwo?

```java
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`?

```java
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

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

```java
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

```java
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++;`

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

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

```java
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```
Subdivisions of a Ruler

Output

% java Ruler
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
Subdivisions of a Ruler — the hard way

```java
public class Ruler {
    public static void main(String[] args) {
        String ruler1 = "1";
        String ruler2 = ruler1 + " 2 " + ruler1;
        String ruler3 = ruler2 + " 3 " + ruler2;
        String ruler4 = ruler3 + " 4 " + ruler3;
        System.out.println(ruler1);
        System.out.println(ruler2);
        System.out.println(ruler3);
        System.out.println(ruler4);
    }
}
```

Output

```
% java Ruler
1
1 2 1
1 2 1 3 1 2 1
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
```
Subdivisions of a Ruler — with for loop

- Initialize ruler to " " (empty string).
- For each value `i` from 1 to `n`:
  - sandwich two copies of ruler on either side of `i`.

```java
public class RulerN {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        String ruler = " ";
        for (int i = 1; i <= n; i++) {
            ruler = ruler + i + ruler;
        }
        System.out.println(ruler);
    }
}
```

<table>
<thead>
<tr>
<th>i</th>
<th>ruler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot; 1 &quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot; 1 2 1 &quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot; 1 2 1 3 1 2 1&quot;</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td></td>
</tr>
</tbody>
</table>
| % java Ruler 1
| 1 |
| % java Ruler 2
| 1 2 1 |
| % java Ruler 3
| 1 2 1 3 1 2 1 |
| % java Ruler 4
| 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 |
| % java Ruler 100
Exception in thread "main" java.lang.OutOfMemoryError |
Loop Examples 1

Print largest power of two that is \( \leq n \)
Loop Examples 1

Print largest power of two that is \( \leq n \)

```java
int val = 1;
while (val <= n / 2) {
    val = 2 * val;
}
System.out.println(val);
```
Print the result of computing the finite sum 
\((1 + 2 + \ldots + n)\)
Print the result of computing the finite sum 
$(1 + 2 + \ldots + n)$

```java
int sum = 0;
for (int i = 1; i <= n; i++) {
    sum += i;
}
```
Print the result of computing the finite product 
\( n! = 1 \times 2 \times \ldots \times n \)
Print the result of computing the finite product 
\((n! = 1 \times 2 \times \ldots \times n)\)

```java
int product = 1;
for (int i = 1; i <= n; i++) {
    product *= i;
}
```
Nested If Statements

How to classify Scottish weather:

<table>
<thead>
<tr>
<th>degrees C</th>
<th>verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -5</td>
<td>wear a sweater</td>
</tr>
<tr>
<td>-5 to 0</td>
<td>nippy</td>
</tr>
<tr>
<td>1 to 10</td>
<td>normal</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>roastin'</td>
</tr>
</tbody>
</table>

4 mutually exclusive alternatives
Nested If Statements

How to classify Scottish weather:

<table>
<thead>
<tr>
<th>degrees C</th>
<th>verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -5</td>
<td>wear a sweater</td>
</tr>
<tr>
<td>-5 to 0</td>
<td>nippy</td>
</tr>
<tr>
<td>1 to 10</td>
<td>normal</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>roastin'</td>
</tr>
</tbody>
</table>

4 mutually exclusive alternatives

```java
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.

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

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'";

<table>
<thead>
<tr>
<th>degrees C</th>
<th>verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -5</td>
<td>wear a sweater</td>
</tr>
<tr>
<td>-5 to 0</td>
<td>nippy</td>
</tr>
<tr>
<td>1 to 10</td>
<td>normal</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>roastin'</td>
</tr>
</tbody>
</table>

4 mutually exclusive alternatives
Summary

Control flow:

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

<table>
<thead>
<tr>
<th>Control Flow</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>straight-line programs</td>
<td>all statements are executed in the order given</td>
<td></td>
</tr>
<tr>
<td>conditionals</td>
<td>certain statements are executed depending on the values of certain variables</td>
<td>if, if-else</td>
</tr>
<tr>
<td>loops</td>
<td>certain statements are executed repeatedly until certain conditions are met</td>
<td>while, for</td>
</tr>
</tbody>
</table>
Tutorials

Start this week — please let the ITO know if you need to switch tutorial groups.
Labs continue this week and every week (except ILW).
This Week’s Reading

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