Conditionals

Boolean Logic
Relational Expressions

Chapter 5, Examples 5-10, 5-11, 5-12, 5-13, 5-15, 5-16, 5-17, 5-18, 5-19, Robot 3 (p. 84-86).
Recap: What is Algorithm? (cf. Lecture 02 Algorithms)

Algorithms *typically* ...

... generate same **output**.

... make some **assumptions**.

... have **multiple** solutions.

... include **decisions**.

... are expressed in **modular** pieces.
Recap: What is Algorithm? (cf. Lecture 02 Algorithms)

Algorithms typically ... 

... generate same output.

... make some assumptions.

... have multiple solutions.

... include decisions.

... are expressed in modular pieces.
Decisions made in our everyday life

- “What should I eat for lunch?”
- “Should I do the lab/assignment today or later?”
- “Can I cross the road?”
- ... and more
How do we decide?

- “What should I eat for lunch?”
- “Should I do the lab/assignment today or later?”
- “Can I cross the road?”
- ... and more
How do we decide? (Decision-making process)

1. Check Condition

   “Is today taco special day?”

   “Is the deadline today?”

   “Is it green light?”
How do we decide? (Decision-making process)

1. Check Condition

- "Is today taco special day?"  "Yes"
- "Is the deadline today?"  "No"
- "Is it green light?"  "Yes"
How do we decide? (Decision-making process)

1. **Check Condition**
   - “Is today taco special day?”
   - “Is the deadline today?”
   - “Is it green light?”

2. **Take Action**
   - “Yes”
   - “No”
   - “Yes”
   - “Let’s eat tacos!”
   - “Let’s do it tomorrow”
   - “Let’s cross the road”
More examples (Decision-making process)

How Babies Make Decisions
More examples (Decision-making process)

How Babies Make Decisions

Is it food?

No

Yes
More examples (Decision-making process)

How Babies Make Decisions

Is it food?

Yes

Stare at it, poke it, drop it on the floor, smash it, throw it, or rub it in your hair.

No

Try to eat it immediately.
More examples (Decision-making process)

1. Check Condition

- Is it food?
  - Yes: Stare at it, poke it, drop it on the floor, smash it, throw it, or rub it in your hair.
  - No: Try to eat it immediately.
More examples (Decision-making process)

1. Check Condition

2. Take Action

- **Is it food?**
  - Yes: Stare at it, poke it, drop it on the floor, smash it, throw it, or rub it in your hair.
  - No: Try to eat it immediately.
Note: general decision-making process

1. Check Condition

2. Take Action
But *why* do we need programs to make decisions?
But *why* do we need programs to make decisions?

- We began by studying...
  - how to draw **static image** by learning about
    - method, canvas, coordinates
But why do we need programs to make decisions?

- then started learning how to make it **dynamic & interactive using**...
  - `draw()` function
  - variables (e.g., `mouseX`, `circleSize`, etc.)
  - `keyPressed()` and `mousePressed()` functions
But *why* do we need programs to make decisions?

- It allows us to do *even more* with our program  
  (in terms of making it more *dynamic* & *interactive*)
Example
Actually, we already saw our program make decisions

- Event functions follow the same decision-making process
  - e.g., `mousePressed()`, `keyPressed()`

```
function mousePressed() {
  // do the following
  // do the following
}
```
But we need user-defined events

- They are limited to keyPressed() and mousePressed() events
- We need to represent user-defined events, such as...
But we need user-defined events

- They are limited to keyPressed() and mousePressed() events
- We need to represent user-defined events, such as...

1. **Check Condition**
   - “Did pac-man lose all lives?”
     - “No”  “Continue the game”
   - “Did any ghost touch pac-man?”
     - “Yes”  “Remove one life & restart”
   - “Did pac-man complete the stage?”
     - “Yes”  “Move to next stage”
Boolean Logic

- It can help us define user-defined events
Boolean Logic

The Genius of George Boole (1:43)
What is boolean logic?

- Based on an idea that some statement $P$ is either true or false. (It can’t be anything in between.)
- Boole suggested that this forms the basis of our thought process
So how do we write boolean logic?
Let’s first look at boolean expression, and then logic

1. boolean expression

P

2. boolean logic

Yes (true)

No (false)
Boolean Expression (= some statement P)

- An expression that evaluates to true or false.
Pick a number
Boolean Expressions (i.e., some statement P)

- *Relational* boolean expressions:
  - is your number greater than 50?
  - is your number less than 40?
  - is your number equal to 73?
  - is your number greater than or equal to 75?
  - is your number less than or equal to 37?
  - is your number not equal to 99?
# Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&gt;</code></td>
<td>greater than 50?</td>
<td><code>number &gt; 50</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than 40?</td>
<td><code>number &lt; 40</code></td>
</tr>
<tr>
<td><code>===</code></td>
<td>equal to 73?</td>
<td><code>number === 73</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to 75?</td>
<td><code>number &gt;= 75</code></td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to 37?</td>
<td><code>number &lt;= 37</code></td>
</tr>
<tr>
<td><code>!==</code></td>
<td>not equal to 99?</td>
<td><code>number !== 99</code></td>
</tr>
</tbody>
</table>
// my number to guess
let number = 73;

// a relational expression used to guess my number
print(number > 50);

https://editor.p5js.org/sanghosuh/sketches/kgv_Tre1
True or False?

let a = 8;
let b = 5;
6 < a
a / 2 > 6
a * 5 === 8 * b
b !== 5

A. true
B. false
C. neither
true or false
Now, let’s look at how to write boolean logic
Conditional Statement

- *if* a boolean expression (\( P \)) is **true**, *then* execute a block of code

```java
if (P) {
    // code to execute if true
}
```
Conditional Statement

- *if* a number is greater than 50, *then* execute a block of code

```plaintext
if (number > 50) {
    // code to execute if true
}
```
Conditional Statement (Flowchart representation)

- *if* a number is greater than 50, *then* execute a block of code

```plaintext
if number > 50
{
    // code to execute if true
}
```
animation

using a conditional statement to "loop" an animation

Starter: https://editor.p5js.org/cs105/sketches/Mcl4bsUNJ

https://editor.p5js.org/cs105/sketches/xkfOIdfyI
conditional_dot1 (if)

...  

// the conditional statement  
if (mouseX > width / 2) {  
    fill("#FF0000"); // red  
    ellipse(0.75 * width, 50, 30, 30);  
}  

...  

https://editor.p5js.org/cs105/sketches/DorsbS_Hk
function draw() {
    background(220);

    // print debug info
    print(mouseX, mouseX > width / 2);

    // the conditional statement
    if (mouseX > width / 2) {
        fill("#FF0000"); // red
        ellipse(0.75 * width, 50, 30, 30);
    }

    // separating line
    line(width / 2, 0, width / 2, height);
}
see “08 Conditionals (trace)”
function draw() {
  background(200);
  fill(0); // black
  if (mouseX < 50) {
    fill(255); // white
  }
  ellipse(mouseX, mouseY, 20);
}

A, B, and C are all possible
Gas Station Analogy (if statement)

- A **single if statement** is like deciding what to do at gas station
  - the **if code block** is something you do if a condition is true
    (if tank is low, then get gas)
  - after checking this condition, you continue down the road ...
If statement (Flowchart representation)

```
if (tank fuel is low) {
    // code to execute if true
}
```

`{ Continue down the road }`
conditional_dot2 (if else)

... 

// the conditional statement
if (mouseX > width / 2) {
  fill("#FF0000"); // red
  ellipse(0.75 * width, 50, 30, 30);
} else {
  fill("#0000FF"); // blue
  ellipse(0.25 * width, 50, 30, 30);
}

...

https://editor.p5js.org/cs105/sketches/nuHTR21Ci
Gas Station Analogy (if else statement)

- A **if else statement** is like deciding what to do at gas station
  - the **if code block** is something you do if a condition is true (if tank is low, then get gas)
  - the **else code block** is something you do if a condition is false (if tank has enough, then just get a snack)
  - after doing one of the two, you continue down the road ...
If - else statement (Flowchart representation)

if (tank fuel is low)

code block to execute if true

{ // code to execute if true

}  

code block to execute if false

{ // code to execute if true

}  

Continue down the road
What can this code draw?

```javascript
function draw() {
  background(200);
  if (mouseY > 50) {
    fill(0); // black
  } else {
    fill(255); // white
  }
  ellipse(mouseX, mouseY, 20, 20);
}
```

A, B, and C are all possible
Gas Station Analogy (if - else if - else statement)

- A **if else if else statement** is deciding what to do at gas station
  - the **if code block** is something you do if a condition is true
    (if tank is low, then get gas)
  - the **else if code block** is something you do if a second condition is true
    (if tank is not low but you’re hungry, grab a pizza)
  - the **else code block** is something you do if none of the above conditions is true
    (if tank is not low and you’re not hungry, grab a coffee)
  - after doing one of the three, you will continue down the road
If - else if - else statement (Flowchart representation)

if
   tank fuel is low
      { // code to execute if true
      }
else if
   I am hungry
else
   { // code to execute if true
   }
conditional_dot3 (if else if)

... 

// the conditional statement
if (mouseX > width / 2) {
    fill("#FF0000"); // red
    ellipse(0.75 * width, 50, 30, 30);
} else if (mouseX < width / 2) {
    fill("#0000FF"); // blue
    ellipse(0.25 * width, 50, 30, 30);
} else {
    background(random(255));
}
...
Two Common Logic Errors with if statements

Adding semicolon after the boolean expression means no code block. A code block with no "if" means always execute.

```javascript
if (mouseX > 50) {
    fill("#FF0000"); // red ellipse(0.75 * width, 50, 30, 30);
}
```

Missing "else" makes two different (and independent) if statements.

```javascript
// the conditional statement
if (mouseX > 50) {
    fill("#FF0000"); // red ellipse(0.75 * width, 50, 30, 30);
}
if (mouseX < 50) {
    fill("#0000FF"); // blue ellipse(0.25 * width, 50, 30, 30);
} else {
    background(random(255));
}
```
checking what key was pressed:

```javascript
if (key === 'c') {
    shape = 1;
} else if (key === 'r') {
    shape = 2;
} else if (key === 't') {
    ...
}
```

using a variable to save 'state':

```javascript
if (shape === 1) {
    ellipse( ... );
} else if (shape === 2) {
    rect( ... );
} else if (shape === 3) {
    triangle( ... );
}
```

https://editor.p5js.org/cs105/sketches/d69-jjtUT
selecting a random shape:

```javascript
let shape = 0;

if (shape === 1) {
    ellipse( ... );
} else if (shape === 2) {
    rect( ... );
} else if (shape === 3) {
    triangle( ... );
}

shape = int(random(1, 4));
```

https://editor.p5js.org/sanghosuh/sketches/ftWomWsw
In JavaScript, there are two equality operators: `==` and `===`
- You should use `===` because it’s a strict comparison
- Same goes for `!=` and `!==` operators, use `!==`
  (In other languages, there is only `==` and `!=`, and these are strict)

https://editor.p5js.org/sanghosuh/sketches/_rv8pQhb
https://codeahoy.com/javascript/2019/10/12/==vs===in-javascript/
Equality vs. Assignment Operators

Equality Operator

`===` - is the left value equal to the right value?
- e.g. “state === 1” means "is state equal to 1?"
- left and right can be anything that reduces to a single value (a variable, a function that returns a value, a number, an expression)

Assignment Operator

`=` - assign the right side value to the variable on left
- e.g. “state = 1” means “assign 1 to state”
- left must be a variable, right can be anything that reduces to a single value (a variable, a function that returns a value, a number, an expression)
Office Hours

- My office hours are on **Fridays 10:00 - 12:00** at **DC2102**
- If you want to make an appointment, you can sign up using the following link:
  https://calendly.com/sanghosuh/cs105-office-hours
- You do **NOT** have to make an appointment, but then it will be on a **first come, first served** basis.