Expressions & Conditionals

CSE 120, Winter 2020

Instructor Sam Wolfson

Teaching Assistants Yae Kubota, Eunia Lee, Erika Wolfe

Cashless businesses are now banned in NYC

"New York City's restaurants and other retail establishments will no longer be allowed to reject cash payments under legislation passed by the City Council on Thursday.

Supporters of the bill say cashless businesses requiring credit cards and electronic payments like Apple Pay discriminate against poor people who may not have bank accounts or credit cards — as well as minors.

"The City of New York cannot allow the digital economy to leave behind the 25 percent of New Yorkers who are chronically unbanked and underbanked," said Councilman Ritchie Torres (D-Bronx), the bill's sponsor."

- https://nypost.com/2020/01/24/cashless-businesses-are-now-banned-in-nyc/

Administrivia

- Assignments:
  - Animal Functions due tomorrow (1/28)
  - Reading Check 4 due Thursday @ 3:30 (1/30)
  - Jumping Monster due Friday (1/31) 
- "Big Ideas" this week: Digital Distribution
- Quiz 2 this Friday
  - Topics posted on course website
  - **New:** memorization of *short* code snippets

Outline

- **Expressions & Operators**
- Conditionals

Expressions

- "An expression is a combination of one or more *values*, *constants*, *variables*, *operators*, and *functions* that the programming language interprets and computes to produce another value."
  - https://en.wikipedia.org/wiki/Expression_(computer_science)

- Expressions are *evaluated* and resulting value is used
  - Assignment: `x = x + 1;`
  - Assignment: `xPos = min(xPos + 3, 460);`
  - Argument: `ellipse(50+x, 50+y, 50, 50);`
  - Argument: `drawMouse(rowX+4*50,rowY,rowC);`

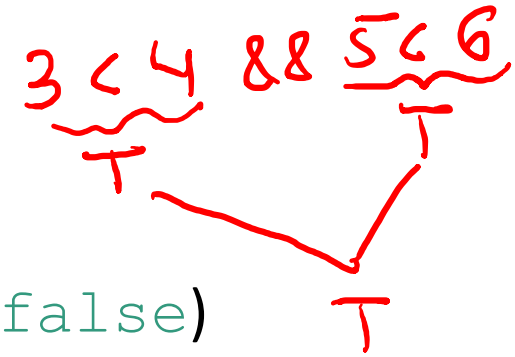
Operators

- Built-in "functions" in Processing that use special symbols:
  - Multiplicative: * mult / div % modulous
  - Additive: + add - sub
  - Relational: < > <= >= ≤ ≥
  - Equality: == !=
  - Logical: && || !

- Operators can only be used with certain data types and return certain data types
  - Multiplicative/Additive: give numbers, get number 3+4 ⇒ 7
  - Relational: give numbers, get Boolean 3<4 ⇒ true
  - Logical: give Boolean, get Boolean
  - Equality: give same type, get Boolean

Operators

- Built-in "functions" in Processing that use special symbols:
  - Multiplicative: `*` `/` `%`
  - Additive: `+` `-`
  - Relational: `<` `>` `<=` `>=`
  - Equality: `==` `!=`
  - Logical: `&&` `||` `!`



- Logical operators use Boolean values (`true`, `false`)

AND (`&&`) truth table

x	y	x && y
false	false	F
false	true	F
true	false	F
true	true	T

OR (`||`)

x	y	x || y
false	false	F
false	true	T
true	false	T
true	true	T

NOT (`!`)

x	!x
false	T
true	F

Operators

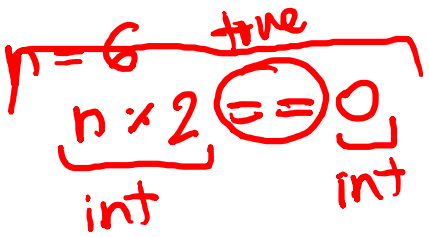
- Built-in "functions" in Processing that use special symbols:
  - Multiplicative: `*` `/` `%`
  - Additive: `+` `-`
  - Relational: `<` `>` `<=` `>=`
  - Equality: `==` `!=`
  - Logical: `&&` `||` `!`

- In expressions, use parentheses for evaluation ordering and readability
  - e.g. `x + (y * z)` is the same as `x + y * z`, but easier to read

Modulus Operator: %

- `x % y` is read as "`x` mod `y`" and returns the remainder after `y` divides `x`
  - For short, we say "mod" instead of modulus

17 % 4 = 1



17 / 4 = 4 R1

- Example Uses:
  - Parity: Number `n` is even if `n%2 == 0`
  - Leap Year: Year `year` is a leap year if `year%4 == 0`
  - Chinese Zodiac: `year1` and `year2` are the same animal if `year1%12 == year2%12`

Conditionals Worksheet

- Work just on **Page 1 (Questions 1-6)**

- Operators:
  - Arithmetic: `+` `-` `*` `/` `%`
  - Relational: `<` `>` `<=` `>=`
  - Equality: `==` `!=`
  - Logical: `&&` `||` `!`
- Data Types:
  - Arithmetic: give numbers, get number
  - Relational: give numbers, get boolean
  - Logical: give Boolean, get boolean
  - Equality: give same type, get boolean

Modulus Example in Processing

- Use mod to "wrap around"
  - Replace `min`/`max` function to "connect" edges of drawing canvas

xPos = 459;

462

- `xPos = min(xPos + 3, 460);` ⇒ 460
- `xPos = (xPos + 3) % 460;` ⇒ 2

462

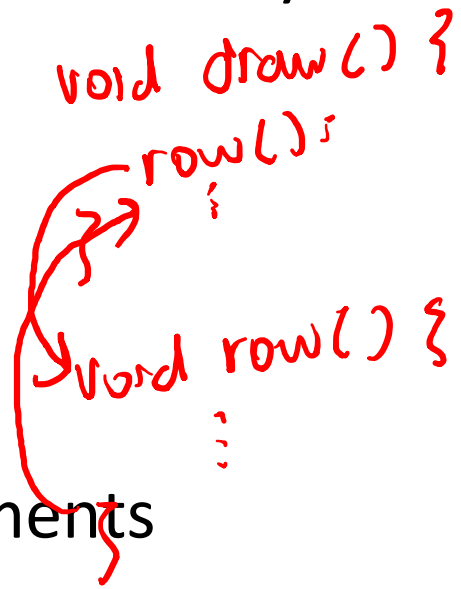
Control Flow

- The order in which instructions are executed

- We typically say that a program is executed in sequence from top to bottom, but that's not always the case:
  - Function calls and `return` calls
  - Conditional/branching statements
  - Loops

today!

next week



- Curly braces { } are used to group statements
  - Help parse control flow
  - Remember to use indentation!

Outline

- Expressions & Operators
- **Conditionals**

If-Statements

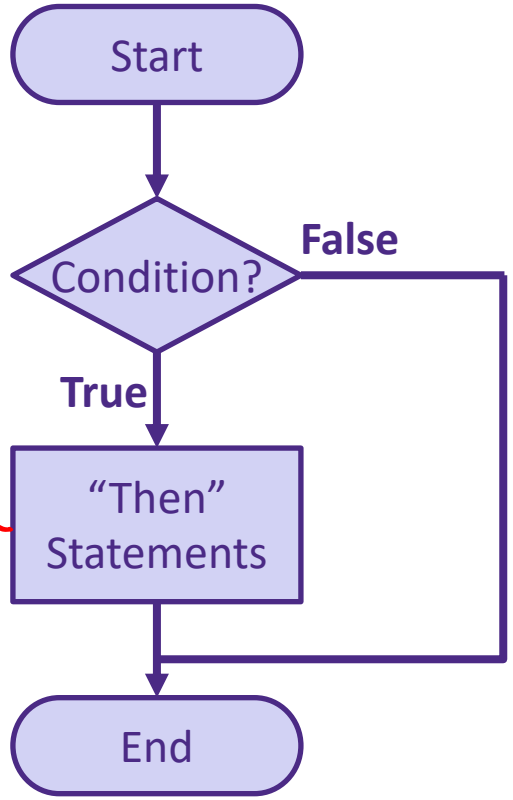
- Sometimes you don't want to execute *every* instruction
  - Situationally-dependent

- Conditionals give the programmer the ability to make decisions
  - The next instruction executed depends on a specified *condition*
    - The condition must evaluate to a `boolean` (*i.e.* `true` or `false`)
    - Sometimes referred to as "branching"
  - This generally lines up well with natural language intuition

If-Statements

```
if (condition) {
    // "then"
    // statements
}
```

Code continues below



- Example conditions:
  - Expression: `if( done == true )`
  - Variable: `if( done )`
  - Expression: `if( xPos > 460 )`
  - Expression: `if( xPos > 100 && yPos > 100 )`

boolean boolean boolean

Practice Question

- Which value of `x` will get the following code to print out `"Maybe"`?

A. 1
B. 3
C. 5
D. 7

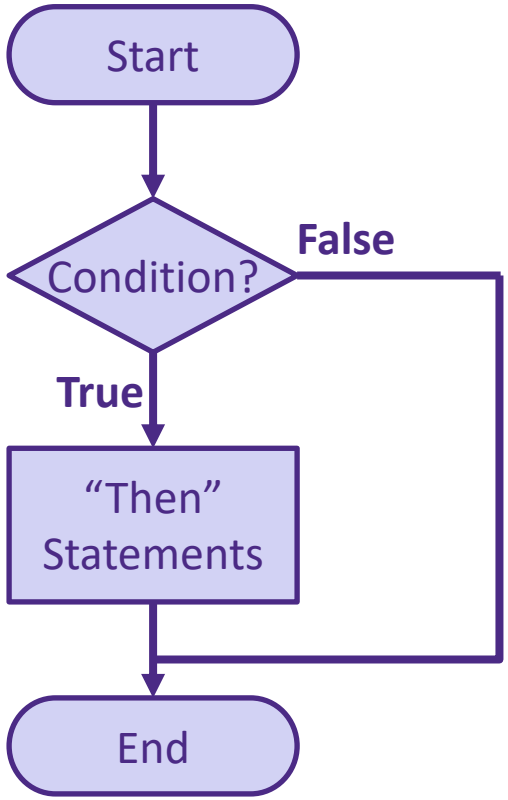
```
if (x == 5) {
    print("Yes");
}
if ((x >= 6) || (x < 2)) {
    print("No");
}
if ((x != 5) && (x < 6) && (x >= 2)) {
    print("Maybe");
}
```

- Think for a minute, then discuss with your neighbor!

Conditionals Worksheet

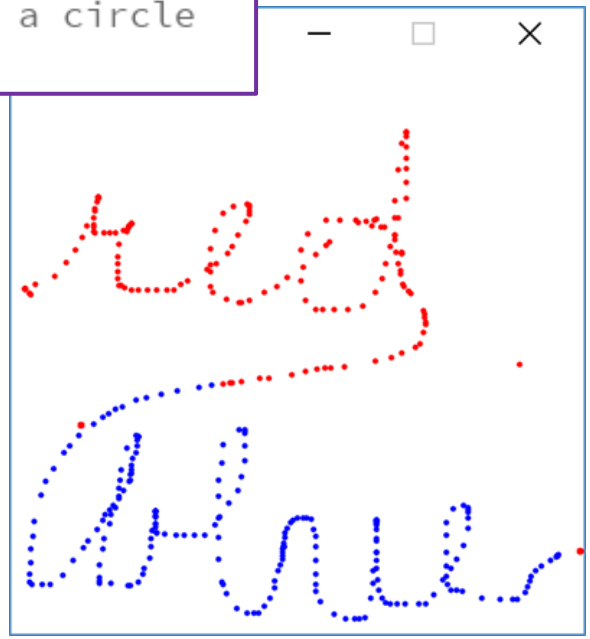
- Work on **Page 2 (Questions 7-9)**

```
if (condition) {
    // "then"
    // statements
}
```



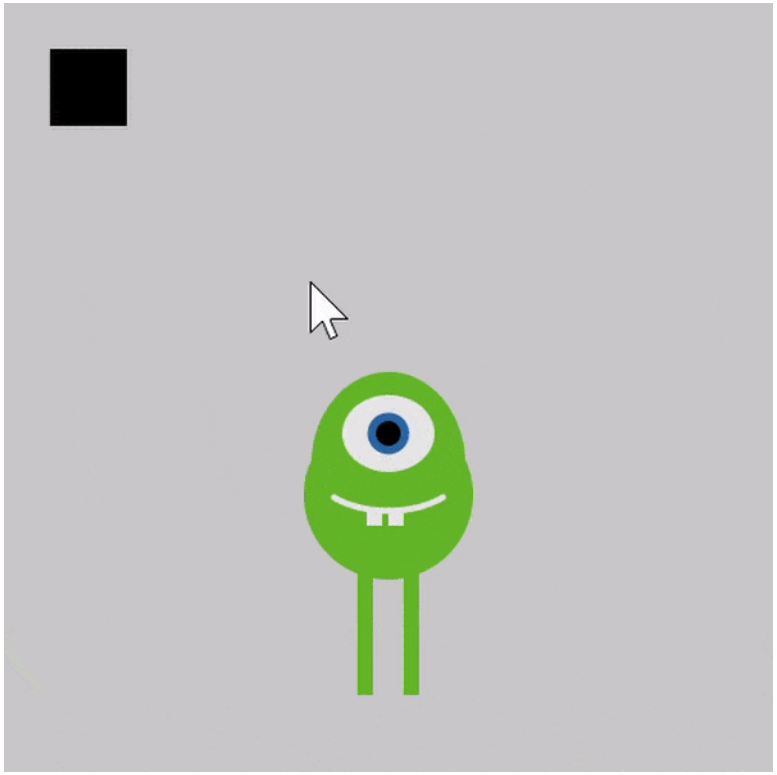
Processing Demo: Drawing Dots

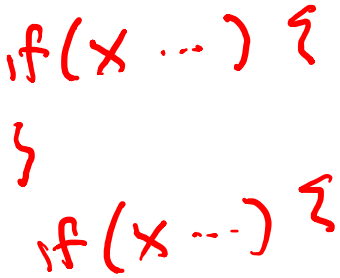
```
7  void draw() {
8    if (mousePressed) {
9      fill(0, 0, 255);  // blue if mouse is pressed
10   }
11   if (!mousePressed) {
12     fill(255, 0, 0);  // red if mouse is not pressed
13   }
14   ellipse(mouseX, mouseY, 5, 5);  // draw a circle
15 }
```



Jumping Monster

- Using *expressions* and *conditionals* in conjunction with *variables* and *user input* (Wed) to control what is drawn as well as motion:





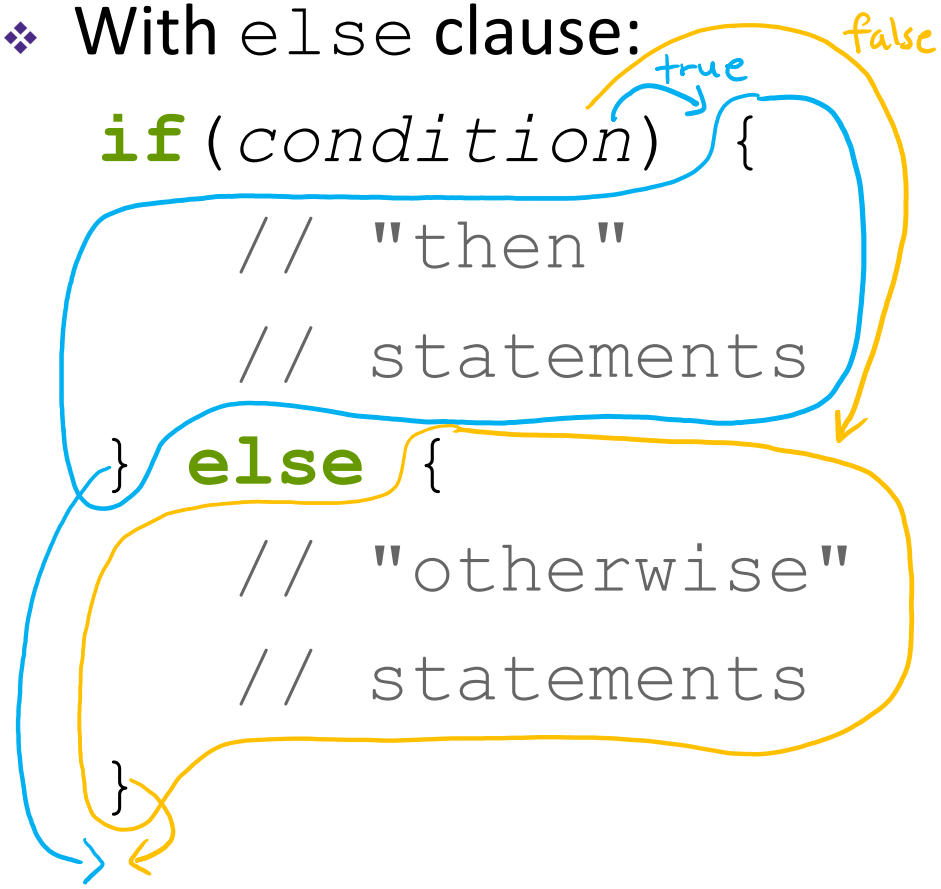
Additional Material For if-statements

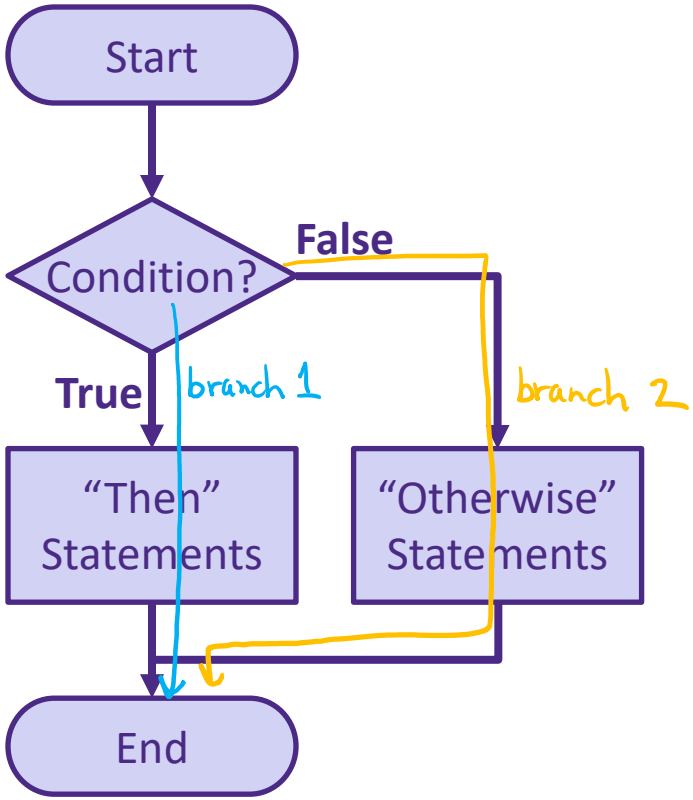
This material is optional, and you won't be tested on it, but it may help you to write more concise code.

If-Statements

- With `else` clause:

```
if(condition) {
    // "then"
    // statements
} else {
    // "otherwise"
    // statements
}
```

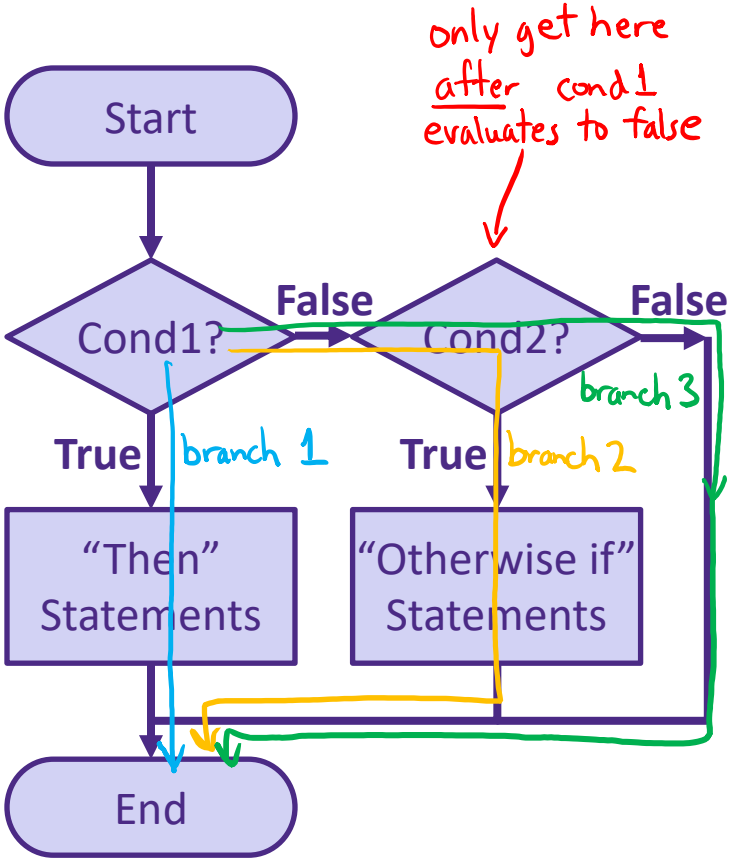




If-Statements

- With `else if` clause:

```
if (cond1) {
    // "then"
    // statements
} else if (cond2) {
    // "otherwise if"
    // statements
}
```



If-Statements

- Notice that conditionals *always* go from Start to End
  - Choose one of many *branches*
  - A conditional must have a single **`if`**, as many **`else if`** as desired, and at most one **`else`**

"catch all" / default

- Can nest and combine in interesting ways: