Lesson 00 - Conditionals

Intended Audience and Course

This lesson should be appropriate for any high school student's first computer science class centered around written code. I imagine that means that this lesson would mostly be used in intro computer science classes, but I am aware that some intro CS classes start with block coding like Scratch or Snap, meaning this might be useful for classes that come after that. The materials for this lesson are centered around Java as programming language and does get into some specifics of Java, which means it could potentially fit in an AP CS A course. At the same time, the main activity for this lesson centers more around pseudocode and this lesson could potentially be adapted for other written programming languages.

This lesson in particular might also be much easier if students have gone over logic and/or truth tables in their math classes prior to this lesson, but it is by no means a necessity.

Explanation of Methods Used

1. CS Unplugged - Although it could be argued that this is not a fully unplugged activity as it involves writing and interpreting pseudocode, I believe that at its core it still counts as it can definitely be done without using a computer. Anyways, part of the reason I wanted to use an Unplugged activity for this lesson was because I wanted to make sure that students really understood what conditionals were capable of outside of the bounds of coding. The idea is to encourage computational thinking both inside and outside of CS class, and I feel like this activity would help students apply a programmatic understanding of conditionals to other areas. For example, maybe a student could apply similar logic here to science classes when it comes to identifying animal species or different types of rocks. I also just thought that it would be a fun idea, and that when we need to talk about conditionals in the future we could go, “remember that time we played Guess Who?”
2. Group Work - Although maybe not group work in the traditional sense, I think that the multiplayer aspect of the activity enhances it. Although not high stakes, I think that the competitive nature of part 1 of the activity will really have students thinking. Even if its not exactly what’s going through their mind, the drive to win before their opponent might get them thinking “What are the best questions to ask? What leads to the most information gain?”. Playing with other students should also expose the students to other ways of thinking they might not have thought of on their own.
3. Open-ended Questions - This is not one of the methods listed, might not be the best way to describe what I’m thinking of, and might not even really be a thing, but a big thing I kept in mind while designing this lesson was to ask questions with many different possible answers. The idea here was that by doing so, students would be able to link conditionals to their own experiences and share those experiences with others. Also by having questions with so many possible answers and interpretations, it might encourage students who don’t normally speak up or are afraid of getting things wrong to participate.

Lesson Goals

This lesson is meant to be an introduction to conditionals. While it does go over the syntax for how to make conditional statements in Java, I don’t consider students being able to create their own conditional statements in Java a primary goal of the lesson, as that could be saved for the next lesson or a follow-up of some kind. Instead, my two main goals for this lesson are for students to make connections between conditionals in their own lives and conditionals in computer science, as well as to introduce to students how conditionals can be used to make decisions.

Lesson Structure

Aim: What are conditionals and how are conditionals used?

Warm-Up (3-5 minutes)

Do Now: Think of a game, any kind of game, whether it be a video game, card game, board game, physical game, etc. How do you win and/or lose that game?

Students will be given 1-2 minutes to answer this question. If they have their answer early, they will be encouraged to share with the students sitting next to them.

After the 1-2 minutes are up, we'll discuss the Do Now question as a class. Students will be asked who wants to share the game they thought of to the class. Potentially a way for students and myself to learn about new games.

Slide Presentation (13 - 20 minutes)

Present the slides provided in the materials

A few of the slides in the materials have questions to ask the students in bold. For these questions (except the one on the slide titled “Relational Operators (cont.)”, give students time to think about the question (no more than a minute). Then, ask students to share their thoughts. This might be a good opportunity to call on students who haven’t spoken yet today. Furthermore, while there are many possible answers and interpretations to these questions, if a student says an answer that doesn’t obviously fit ask them to explain their reasoning.

For the “Relational Operators (cont.)” question, give students 1-2 minutes to write down their answer for each variable in their notebook (or some equivalent). Afterwards, ask the class to answer all together whether each variable is true or false. You can explain the the reasoning (or get a student to) for each variable, especially if there seems to be an audible disagreement in the class

Guess Who Activity (20 - 25 minutes)

The rules for this activity are also on the slides but I will repost them here as well to add some extra teacher notes and directions

Below are the instructions for part 1 of the activity, which should take 10-12 minutes, enough time for students to really think about their questions

- Get students into groups of 4
    - Not all group members will be interacting at the same time
    - If doing this physically groups can be decided on the spot, but if doing this through the computer (like by using Repl.it multiplayer, it's a good idea to set up the groups before hand)
- Students choose one of their group members to face in Guess Who?
- Before starting the game, students secretly select one of the characters on the board. The goal of the game is to guess who your opponent's secretly selected character is.
- The first player to go starts by asking the other player a question. This question has to be answerable with either "yes" or "no" and can't be open-ended.
    - If your opponent doesn't or can't know about the answer to your question, they may request you ask if different one. For example, if you ask "is your character over 6ft tall?" but they don't know the heights of every character, they can't answer. It's best to stick with questions based on the visual information available
    - Since you will most likely be walking around during this, you can also act as an intermediary for such questions
- Based on the opposing player's answer, eliminate all characters the answer rules out. For example, if the question was "Is your character a Pokemon?" and the answer is no, you would eliminate Pikachu, Pichu, Mewtwo, and Jigglypuff
    - It might be a good idea in this case to demonstrate what this looks like in front of the class. Make sure to emphasize that students are trying to cross out characters that don't fit the condition to narrow their possible choices
- Unlike regular Guess Who?, this version has an extra step. Students should try to convert their question and the answer to pseudocode.
    - It is also a good idea to demonstrate what this pseudocode might look like with an example. You can emphasize that since they are writing pseudocode, it doesn't have to be perfect
- Next, the opposing player takes their turn to ask a question and write down their pseudocode and other player answers their question.
- Students keep alternating turns until they feel confident enough to guess who the opponent's secret character is. The first person to guess correctly wins, but keep playing until both secret characters are discovered.
    - It might be useful to keep track of who the winners are and what questions they asked, but not necessary
- Students should keep note of who their opponents secret character was separate from the pseudocode, as this will be important for part 2 of the activity.

Below are the rules for part 2 of the activity, which should also take 10-12 minutes

- Students choose one of their group members to play with that they didn't play with already
- They'll be trying to guess the other player's secret character but this time they aren't allowed to ask any questions of the other player.
    - Clarifying questions should be fine, but not the same type of yes or no questions they were asking earlier
- Students should swap the pseudocode they wrote in part 1 with the pseudocode their current partner wrote in part 1.
- Students should try to follow along with the pseudocode instructions left for them by their fellow player to see if they can guess their secret character
    - It might help to emphasize that the secret character they're trying to guess will be whatever their current partner's opponent's secret character was in the previous part
- Once they feel confident about an answer, they can ask their fellow player if their guess is correct.
- If their guess is correct, they win! However, they should let their fellow player keep playing too if they didn't guess correctly already.
- If their guess is incorrect, try to go over the pseudocode you were provided with again to see if they missed anything.
    - Note that they shouldn't be afraid to ask for help if they feel really unsure or confused about something. You should be going an available resource for the students
    - Alternatively, if their partner already answered correctly they can help the student with their confusion and see if it stems from a misconception of conditionals or a mismatch in the perception of such conditionals (i.e. what one student considers red another considers orange)
- If both students are done playing and there's still time left, they can try playing part 1 again. Or, if everyone in their group is done, they can swap their original pseudocode with the last person in your group they haven't played with.

It would also be very helpful to have the relational operators and boolean operators somewhere visible for all students to use as a reference (that could potentially mean on the repl.it itself if done that way).

This activity can lead into if-else if-else by saying something like "When we played Guess Who, we wanted do every conditional that was true. But what if we only wanted to do one of a few choices instead."

Conditions in Games and Other Contexts

- As discussed while reviewing the Do Now, most games have some kind of victory condition -- conditions that need met in order to win the game.
- Another way of saying this is that IF the victory conditions are met THEN you win the game.
- Using Tic-Tac-Toe as an example, we could say IF you get three of the same symbol in a row THEN you win the game.
- This IF condition THEN action structure is not only limited to games though. **What else can you think of that has a similar structure, where you only do something if certain conditions are met?**

Translating Conditionals to Code

- If… Then statements exist in programming languages as well and can be used to make decisions
- Like most other things in programming languages, while the code may be somewhat similar to English, we need to make sure we're following our programming language's syntax
- Before we go over how to translate conditional statements from English to Java, there's a few things we need to go over first

Booleans

- The name “boolean” comes from George Boole (1815-1864), who is considered to be one of the founders of Computer Science
- Booleans are unique as they only have two possible values: true or false
- Other concepts like on/off or yes/no can also be expressed as booleans in programming

Booleans (cont.)

- In Java, booleans are a primitive data type like int, double, or char. That means that we can have boolean variables or functions that return booleans
- In Java, true and false are their own recognized values. We don’t need to put quotes around them.
- Most light switches for example, are either on or off. We could express this as a boolean in Java by writing something like this
- boolean is_light_on = true
- **What are some other examples of things that can be expressed as booleans?**

Boolean Expressions

- Although we can use true or false directly like the example in the previous slide, it is more common to get boolean values from boolean expressions
- Think about how arithmetic expressions evaluate to a numeric value, like how 9 + 10 evaluates to 19
- Boolean expressions (also called logical expressions), evaluate to boolean values, true or false
- Similarly, the same way there are arithmetic operators like addition, subtraction, multiplication or division, boolean expressions have their own operators

Relational Operators (cont.)

a == b : a is equal to b (notice that two equal signs are used for comparing values as opposed to one equal sign for assignment)

a < b : a is less than b

a <= b : a is less than or equal to b

a > b : a is greater than b

a >= b : a is greater than or equal to b

a != b : a is NOT equal to b

Relational Operators (cont.)

What are the values of the following boolean variables?

```
boolean boo1 = (5 == 5)

boolean boo2 = ((9 * 1) > 9)

boolean boo3 = ((9 * 1) >= 9)

boolean boo4 = ((1 + 1) != 2)
```

If Statements

- If… Then statements can be translated into Java by using the following syntax:

```
if (condition) {
        //code to be run if condition is true
}
```

- if is a Java keyword
- Inside the parenthesis we put a condition. The condition should be something that evaluates to a boolean value
- The code inside the curly brackets will only get run if the condition is true

If Statements (cont.)

- Let’s use another game as an example while also practicing with relational operators
- For many (but not all) sports, the winner is determined by who has the most points at the end. We could translate that into Java using something like this

```
if (your_score > opponent_score) {

        winner = “you”

}
```

- **What are some other If… Then statements you can think of that use relational operators?**

Boolean Operators

- Boolean expressions can be combined by using boolean operators (also called logical operators) to create more complex conditional statements.
- There are 3 boolean operators: and, or, and not. These boolean operators operate similarly in programming as they do in math or English.
- And: a && b evaluates to true if both a and b are true, otherwise it evaluates to false.
- Or: a || b evaluates to true if a or b or both are true. It is only false if both a and b are false
- Not: !a evaluates to true if a is false and false if a is true, inverting a’s value

Boolean Operator Examples

```
if (age > 12 && age < 20) {
        System.out.println(“You are a teenager”);
}
if (num == 2 || num == -2) {
        System.out.println(“num is a square root of 4”);
}
if (!(false)) {
        System.out.println(“This line will always print”);
}
```

Guess Who?

You all will be playing a modified game of Guess Who? Here are the rules for part 1 of today's activity.

- Get into groups of 4 (although not all group members will be interacting at the same time)
- Choose one of your group members to face in Guess Who?
- Before starting the game, secretly select one of the characters on the board. The goal of the game is to guess who your opponent's secretly selected character is.
- The first player to go starts by asking the other player a question. This question has to be answerable with either "yes" or "no" and can't be open-ended.
    - If your opponent doesn't or can't know about the answer to your question, they may request you ask if different one. For example, if you ask "is your character over 6ft tall?" but they don't know the heights of every character, they can't answer. It's best to stick with questions based on the visual information available
- Based on the opposing player's answer, eliminate all characters the answer rules out. For example, if the question was "Is your character a Pokemon?" and the answer is no, you would eliminate Pikachu, Pichu, Mewtwo, and Jigglypuff
- Unlike regular Guess Who?, this version has an extra step. Try to convert your question the answer to pseudocode. For example, the previous question might be written like this

```
if (creature_type == "Pokemon") {

eliminateCharacters(); //assume eliminateCharacters() is what always goes in the curly brackets.

}
```

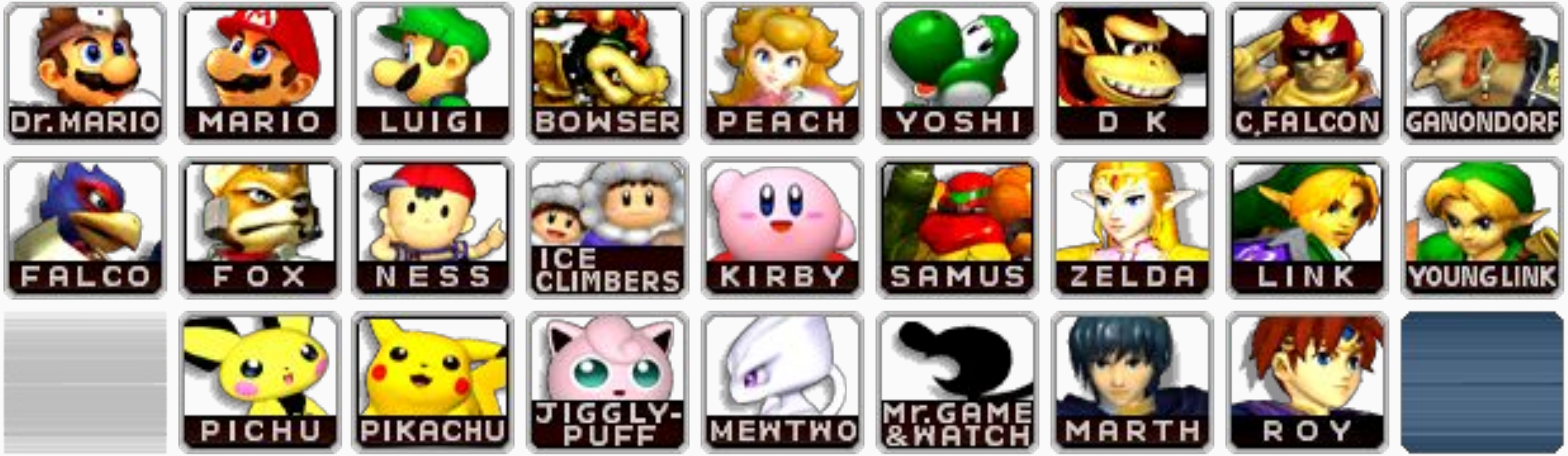
- Next, the opposing player takes their turn to ask a question and write down their pseudocode and you answer their question.
- Keep alternating turns until you feel confident enough to guess who the opponent's secret character is. The first person to guess correctly wins, but keep playing until both secret characters are discovered.
- Keep note of who your opponents secret character was separate from the pseudocode, as this will be important for part 2 of the activity.

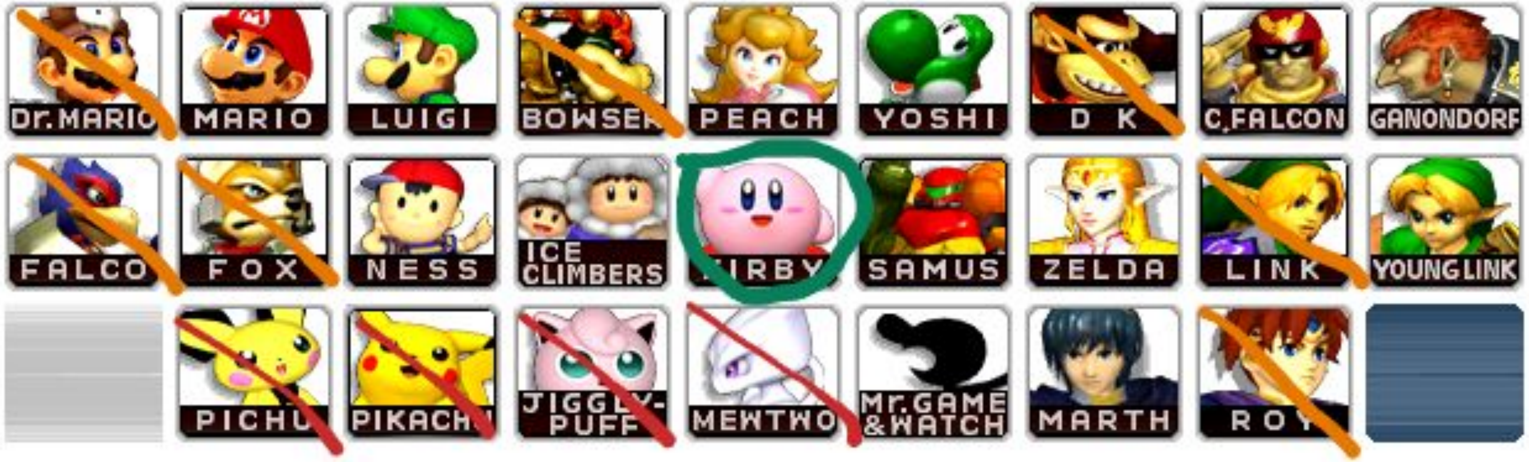
Guess Who? Part Two

Here are the rules for Part 2 of today's activity, Guess Who with an even bigger twist:

- Choose one of your group members to play with that you didn't play with already
- You'll be trying to guess the other player's secret character but this time you aren't allowed to ask any questions of the other player.
- Swap the pseudocode you wrote in part 1 with the pseudocode your partner wrote in part 1.
- Try to follow along with the pseudocode instructions left for you by your fellow player to see if you can guess their secret character
- Once you feel confident about an answer, ask your fellow player if your guess is correct.
- If your guess is correct, you win! However, you should let your fellow player keep playing too if they didn't guess correctly already.
- If your guess is incorrect, try to go over the pseudocode you were provided with again to see if you missed anything. Don't be afraid to ask for help if you feel really unsure or confused about something.
- If you and your partner are done playing and there's still time left, try playing part 1 again with your new partner. Or, if everyone in your group is done, you can swap your original pseudocode with the last person in your group you haven't played with..

Materials


Dr.MARIO
MARIO
LUIGI
BOWSER
PEACH
YOSHI
D K
C.FALCON
GANONDORF
FALCO
FOX
NESS
ICE CLIMBERS
KIRBY
SAMUS
ZELDA
LINK
YOUNGLINK
PICHU
PIKACHU
JIGGLY-PUFF
MEWTWO
Mr.GAME &WATCH
MARTH
ROY



●
```
if (creature_type == "Pokemon") {
    eliminateCharacters();
}
```
●
```
if (facing == "right") {
    eliminateCharacters();
}
```