U

UNPLUGGED

Computational Thinking

Lesson time: 25 Minutes Basic lesson time includes activity only. Introductory and Wrap-Up suggestions can be used to delve deeper when time allows.

LESSON OVERVIEW

For this activity, no instructions are provided. Instead, students will use examples of what imaginary players have done to figure out how to play the game. This lesson gives students the opportunity to practice the four arts of computational thinking (decomposition, pattern matching, abstraction, and algorithms) in one cohesive activity.

TEACHING SUMMARY

Getting Started - 15 minutes

1) Vocabulary
2) Figuring it Out

Activity: Computational Thinking - 25 minutes

3) Computational Thinking

Wrap-up - 10 minutes

4) Flash Chat - What did we learn? 5) Vocab Shmocab

Assessment - 5 minutes

6) Computational Thinking Assessment

LESSON OBJECTIVES

Students will:

- Analyze information to draw conclusions
- Match identical portions of similar phrases to match patterns
- Identify differences in similar phrases and abstract them out

TEACHING GUIDE

MATERIALS, RESOURCES AND PREP

For the Student

- One die per group
- One Computational Thinking Kit per group
- Pens, Pencils, & Scissors
- Computational Thinking Assessment for each student

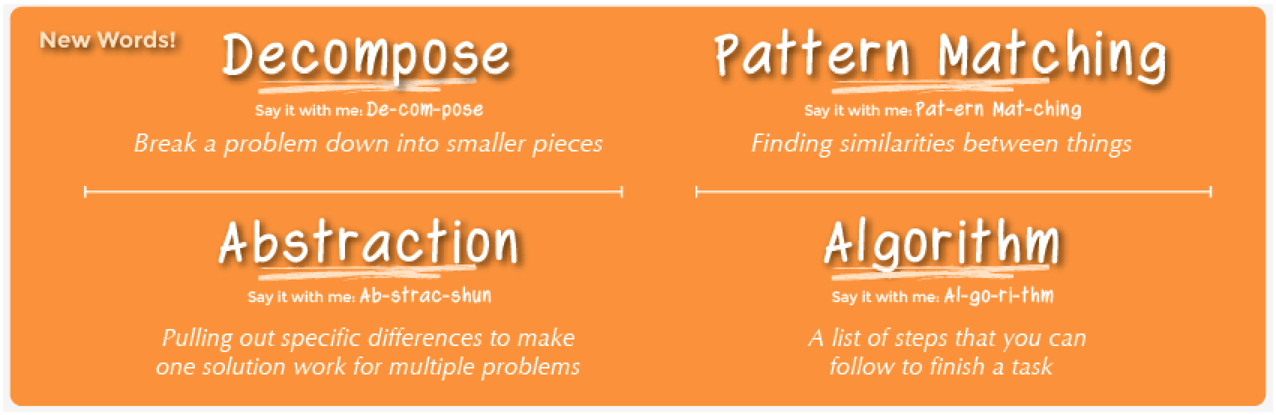
For the Teacher

- Lesson Video
- This Teacher Lesson Guide
- Print one Computational Thinking Kit per group
- Print one Computational Thinking Assessment for each student

GETTING STARTED (15 MIN)

1) Vocabulary

This lesson has four new and important words:



New Words!

Decompose

Say it with me: De-com-pose

Break a problem down into smaller pieces

Pattern Matching

Say it with me: Pat-ern Mat-ching

Finding similarities between things

Abstraction

Say it with me: Ab-strac-shun

Pulling out specific differences to make one solution work for multiple problems

Algorithm

Say it with me: Al-go-ri-thm

A list of steps that you can follow to finish a task

Algorithm - Say it with me: Al-go-ri-thm
A list of steps that you can follow to finish a task

Decompose - Say it with me: De-com-pose
Break a problem down into smaller pieces

Abstraction - Say it with me: Ab-strac-shun
Pulling out specific differences to make one solution work for multiple problems

Pattern Matching - Say it with me: Pat-ern Matching
Finding similarities between things

2) Figuring it Out

- Tell your students that you want them to sum up all of the numbers between 1 & 200.
    - Use your body language to indicate that this is not a "serious" or graded exercise.
        - Now, let them know that they must do it all in their heads.
        - Add the time constraint of thirty seconds.
        - They may feel overwhelmed. This is intentional. You can indicate with your tone and demeanor that you might be crazy asking this of them, but begin timing with a resounding: "Starting NOW".
- Watch the class as you keep time. How many are lost in thought?
- When time is up, ask if anyone was able to get the total.
- Ask if there is anyone who thought the problem was so hard that they didn't even attempt it.
- Did anyone attempt it and just not finish?
    - What did they try?
- Guide students toward thinking a little smaller.
    - If we break the problem up into smaller pieces, it becomes easier to manage.
    - Let's start at the two ends. What is 200 + 1?
    - What is 199 + 2?
    - What is 198 + 3?
    - See a pattern?
    - How many of these pairs will we have?
        - What is the last pair we will find? 100 + 101
        - That means that we have 100 total pairs.
        - If we have 100 total pairs of sums of 201, how do we find the final total?
        - What is 100 * 201?
            - Now, what if we wanted to find the trick to do this with other numbers?
    - Can we do it easily with 2,000?
    - How about 20,000?
    - What stays the same? What is different?
    - If we use abstractions to make our end goal something that can change (say we name it "blank") then we can make an algorithm that will work for any number
- Work through the problem until you ultimately get ? = ("blank"/2) * ("blank"+1)
- Do a few simple examples to show that the algorithm is correct for blanks= 2, 3, 4, & 5.

"This is all to show that if you use the tools of Computational Thinking (decomposition, pattern matching, abstraction, and algorithms), then you can figure out how to solve problems that no one has already taught you how to solve...just like we did here! This will be an extremely powerful skill for the rest of your life!"

ACTIVITIES: (25 MIN)

3) Computational Thinking

This lesson is all about a "Game with No Instructions." Students will be charged with figuring out how to play the game as a small group. The small details of their final algorithm are unimportant. What *is* important is that they were able to take a huge task like "figuring out how to play a game on their own" and take small steps toward achieving the goal.

Students will be guided toward discovering the rules using the steps of computational thinking. Resist the temptation to point the students toward "doing it right" and allow them just to do it on their own. If they feel stumped or confused, encourage the students to look at the information that has been given to them, or if they must, ask a classmate.

Directions:

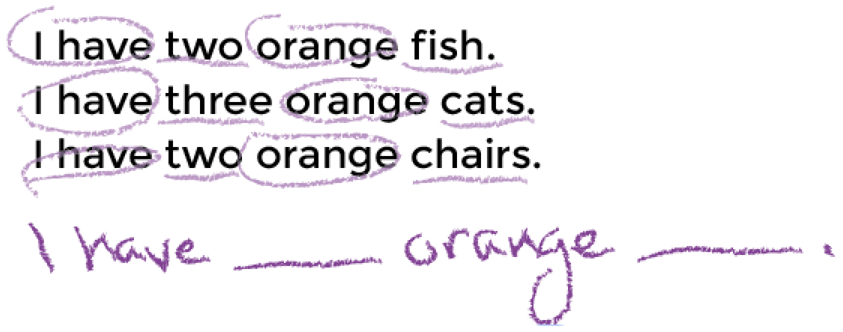
1) Divide students into groups of 2-4.

2) Have the groups read over user experiences to get an idea of how other students have played the “Game with No Instructions.”

3) Encourage them to pattern match between each experience by circling the sections of words that are identical from player to player.

4) Next, have them abstract away differences from each experience by underlining words that change from player to player.

5) Using pattern matching and abstraction, have them make a script template for game play by writing up the circled parts of the other students’ experiences, and leaving the underlined sections as blanks. For example:



6) Give students a blank sheet of paper to write a list of instructions for how they think this game should be played based on the user experiences that they just read. This will be their algorithm.

7) Have students play the game using the algorithm that they just made. Each player should get at least two turns.

WRAP-UP (5 MIN)

4) Flash Chat: What did we learn?

- What should you try to do when you're asked to do something and you don't know how?
- If a problem is too hard, what should you try to do?
- If you find similarities in lots of solutions to different problems, what does that probably tell you?
- If you have a problem that is just a little different from a problem that you have a solution for, what would you do?

LESSON TIP

Flash Chat questions are intended to spark big-picture thinking about how the lesson relates to the greater world and the students' greater future. Use your knowledge of your classroom to decide if you want to discuss these as a class, in groups, or with an elbow partner.

5) Vocab Shmocab

- Which one of these definitions did we learn a word for today?

"Bringing two pieces together"
"Breaking a problem down into smaller pieces"
"An educated guess"

...and what is the word that we learned?

ASSESSMENT (5 MIN)

6) Computational Thinking Assessment

- Hand out the assessment worksheet and allow students to complete the activity independently after the instructions have been well explained.
- This should feel familiar, thanks to the previous activities.