Activity 05-1: Program Your Friends and Classmates

Big Idea

Computer programs are a series of commands. Of course, complex programs require more complex coding commands and can be quite lengthy with subroutines written into them. While there are many places you can go online to practice basic programming, you can create a series of bulleted directions, cards, or signs that replicate simple coding, often referred to as pseudocode.

Materials

Paper and pencil, dry-erase boards, or word-processing or note-taking apps

Vocabulary

Coding
Subroutines
Pseudocode
Compile
Unplugged coding
Blockcoding
Python
Variables
Conditional Statements

Background

In this activity, you will replicate basic computer programming by creating a list of directions or printing them out on cards or sheets of paper. Give the pseudocode to your friends to see if they can complete the required steps or identify whether the code has to be “debugged” or not. It’s okay to use pseudocode rather than actual code because you’re not going to run (or compile) the code on a device.

Depending on your coding experiences, you may have done something similar when you were younger using something called unplugged coding with Scratch or other blockcoding apps. Or you may have coded robots using a similar process. The idea is the same, you’re just using more sophisticated coding.

If you are new to coding you may want to include some basic commands (see below). If you have some experienced coders, you may want to identify terms used in a common coding language, such as Python. You can find lists of Python Keywords online with examples of how they work, like this one from W3³ Schools. You can also mix it up by creating pseudocode using basic Windows command lines.

If you create a series of common computing commands and print them out as cards or posters, they can be mixed and recombined into unique sequences on the fly.

Activity Directions

Make a list of common simple coding commands that might be included in a routine your friends can follow. The number of commands in your list can vary based on the coding experience you and your teammates have.

Some commands to consider starting with if you are using Python:

and
break
continue
del
else, if, elif
False, True
for
Is, not, or
None
return
with

Think of a task you can have someone complete. Try something simple in the classroom, perhaps getting someone to walk somewhere and get something or do something. You're going to combine actual code with other actions to result in pseudocode people can follow.

For example, let's say you want to write code that someone will follow to walk to the pencil sharpener and sharpen a pencil, not a pen. Based on your classroom layout, you can describe the steps with a bulleted list of directions. For example, the overarching steps for someone sitting next to you to sharpen a pencil may be:

1. If your pencil lead is broken
2. Stand up
3. Walk 5 steps to the pencil sharpener
4. Sharpen your pencil

But that example will only work for anyone 5 steps away from the pencil sharpener. What if someone is 12 steps away? Or only 3? That's where variables and conditional statements come in.

Variables in computer code are like variables in math. **Variables** stand for something else, like a number. In computer code, variables can also stand for text information or conditions, like whether a statement is True or False.

You could make the number of steps to the pencil sharpener a variable for people at different distances. Knowing whether your pencil has to be sharpened could also be a variable.

Variables Examples
- `**distanceToSharpener**` (integer/variable): Number of steps to the pencil sharpener.
- `**pencilCondition**` (boolean/flag): True if the pencil lead is broken, False otherwise.

Pseudocode Example using Variable:
- Check `**pencilCondition**`. If True, then proceed; otherwise, stay seated.

Computer code also uses **conditional statements,** to allow the code itself to process information and run as long as it needs to. The commands `if` and `else` are good examples. You can include an `if` statement that seeks whether a condition is `True`, like `if pencilCondition = True` then your pencil lead is broken and should be sharpened.

Conditional Statement Examples:
- Use an `**if-else**` statement for the condition of the pencil.

Pseudocode Example for Conditional Statement:
- If `**pencilCondition**` is True (pencil is broken):
- Stand up
- Else:
- Continue working

You can also loop actions or so that you can control for variable information, like how far someone is from the pencil sharpener.

Another Example using a Loop and More Conditional Statements:
- Add a loop to simulate walking towards the sharpener.

Pseudocode:
- While `**distanceToSharpener**` is greater than 0:
  - Take a step forward
  - Decrease `**distanceToSharpener**` by 1

Once these conditions are `True`, like when your friend has taken the number of steps they need to take to reach the pencil sharpener, you can tell the computer code to go on and do something else or it can just stop. Otherwise, the computer code can loop through steps until they are `True`. (Remember, we're using pseudocode for now, not strict coding.)

Final Action:
- When `**distanceToSharpener**` is 0 (at the sharpener), sharpen the pencil.

Pseudocode:
- If `**distanceToSharpener**` equals 0:
  - Sharpen pencil

If these are all familiar concepts to you, you may want to extend your code with some advanced concepts, like using a function. Check out the following:

Optional Advanced Concepts:
- Introduce a function, `**walkToSharpener(steps)**`, which takes steps as input and reduces `**distanceToSharpener**` accordingly.

Pseudocode:
- Function `**walkToSharpener(steps)**`:
  - While `**steps**` > 0:
    - Take a step forward
    - Decrease `**steps**` by 1
    - Decrease `**distanceToSharpener**` by 1

If you put all of these steps together, our finished pseudocode might look something like this:
- Check `**pencilCondition**`
- If True:
  - Stand up
  - `**walkToSharpener(distanceToSharpener)**`
  - Sharpen pencil
- Else:
  - Continue working

Here's an example of what our finished code might look like in Python:

Python-like Pseudocode for Pencil Sharpening Activity

python Copy code

```python
# Variables
distanceToSharpener = 10  # Replace with actual number of steps
pencilStatus = True       # True if the pencil is broken, False if not

# Function to simulate walking to the sharpener
def walkToSharpener(steps):
    while steps > 0:
        print("Take one step forward.")
        steps -= 1

# Main sequence
if pencilStatus:
    print("Stand up from the desk.")
    walkToSharpener(distanceToSharpener)

    # At the sharpener
    if distanceToSharpener == 0:
        print("Sharpen the pencil.")
else:
    print("Continue with the current task.")
```

Want some more ideas? Consider these extension activities:

- **Practical Exercise**: Students determine their `**distanceToSharpener**` based on their seating arrangement.
- **Group Collaboration:** Students can pair up to write and act out the pseudocode, fostering teamwork and practical application of concepts.
- **Debugging Challenge:** Introduce scenarios where `**distanceToSharpener**` is miscalculated and discuss the importance of accurate data in programming.