User Input
Grades 6-12
Designed by Robolink

Summary:
For the most part, the programs that students have written do not have a lot of user interaction, but that will change with this lesson! Students will learn how to write programs that the user can interact with. They will then write their own remote control program and test it out in an obstacle course.

Guiding Question(s):
- How can you write a program to create more user interaction?
- How can a keyboard be programmed to act as a remote control?

Learning Objectives:
Students will be able to:
- Create and run programs that include user interaction
- Create and run a program that will enable their keyboard to act as a remote control

Step to Success:
1. Input
2. Getting the Info
3. Dizzy Drone Activity
4. Creating a Menu
5. Make Your Own Menu
6. Resetting and User Input
7. Making Decisions
8. Final Code
9. Challenge: Custom Polygon

Materials needed:
- CoDrone EDU, remote, and USB cable for each student or student group
- Laptop with Internet access and PyCharm EDU installed for each student or student group
- Charged CoDrone EDU batteries and extra chargers
- Chairs and/or desks to use for obstacles

Lesson Title: User Input **Time:** 1 hour 30 minutes

Engagement: (Introduction)

- In the past lessons, students have been flying their drone autonomously --- with a program. In this lesson, students will be flying their drone with a remote control. Ask them to come up with a list of situations where it would be better to use a drone autonomously and then with a remote control. Write answers on the board!
- Show students the videos below. Would these situations be better for autonomous or remote control flying? Why? Ask for thoughts, observations, and questions.
  Video: Watch the drone that was used in a search and rescue operation
  Video: DJI - M200 Series - Search and Rescue in Extreme Environments

Exploration: (Activity)

- Have students complete User Input on Robolink Basecamp, including the Super Program challenge. If students are having problems, they need to talk to classmates before asking the teacher!

Explanation: (Recap)

- Ask students to use pseudocode to explain their Super Program challenge program to a partner using both their own words and the appropriate academic language.

Elaboration: (Extension)

- Set up an obstacle course in the classroom (or the space that you are using). Desks and chairs are great for this, along with anything that is already in there, like doorways. PVC pipe gates and hoops also work really well for this activity.
- Assign point values to each obstacle, like 10 points for an easy obstacle, 20 points for an intermediate obstacle, and 30 points for a hard obstacle. You can also assign 50 points for tricks, like if a student does a flip over a gate.
- Have students test their Super Program in the obstacle course. Give them one minute to gain as many points as they can!

Evaluation:

- In a journal or on a worksheet have students answer the following questions. Examples of engineering portfolios are available on Google Sites and ProjectBoard.

1. Come up with an idea for a program that would include user input. Use pseudocode to explain how it would work.
2. How are conditionals and loops used in user input programs?
3. Explain your Super Program using pseudocode. Why did you choose to make your program the way you did?
4. Did you have any problems running your codes? If so, what did you do to fix them?
5. What did you learn?

Related Vocabulary: break, delay, drone, input, landing, library, loop, menu, movement command, negative, parameter, pitch, positive, print, Python, roll, takeoff, throttle, variable, yaw

Standards:

CCSS:
ELA-LITERACY.RST.6-8.3: Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.
ELA-LITERACY.RST.9-10.3: Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.
ELA-LITERACY.RST.11-12.3: Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks; analyze the specific results based on explanations in the text.
MATH.PRACTICE.MP1: Make sense of problems and persevere in solving them.
MATH.PRACTICE.MP5: Use appropriate tools strategically.
MATH.PRACTICE.MP7: Look for and make use of structure.

NGSS:
MS-ETS1-1: Define the criteria and constraints of a design problem with sufficient precision in order to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions.
MS-ETS1-2: Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.
HS-ETS1-1: Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.
HS-ETS1-2: Design a solution to a complex real-world problem by breaking it down into

smaller, more manageable problems that can be solved through engineering.

CSTA:
2-CS-03: Systematically identify and fix problems with computing devices and their components.
2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-19: Document programs in order to make them easier to follow, test, and debug.
3B-AP-16: Demonstrate code reuse by creating programming solutions using libraries and APIs.

ISTE:
5D: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.
6A: Students choose the appropriate platforms and tools for meeting the desired objectives of their creation or communication.