**Drone Programming Activity Compilation**

Objective: Engage participants in creating interactive programs using DroneBlocks that incorporate variables, flips, loops, and logic to explore the capabilities of drones and enhance programming skills.

**Variable Activity**

**Objective:** Create a program using DroneBlocks that utilizes variables to control flight parameters.

**Key Concepts:** Variables

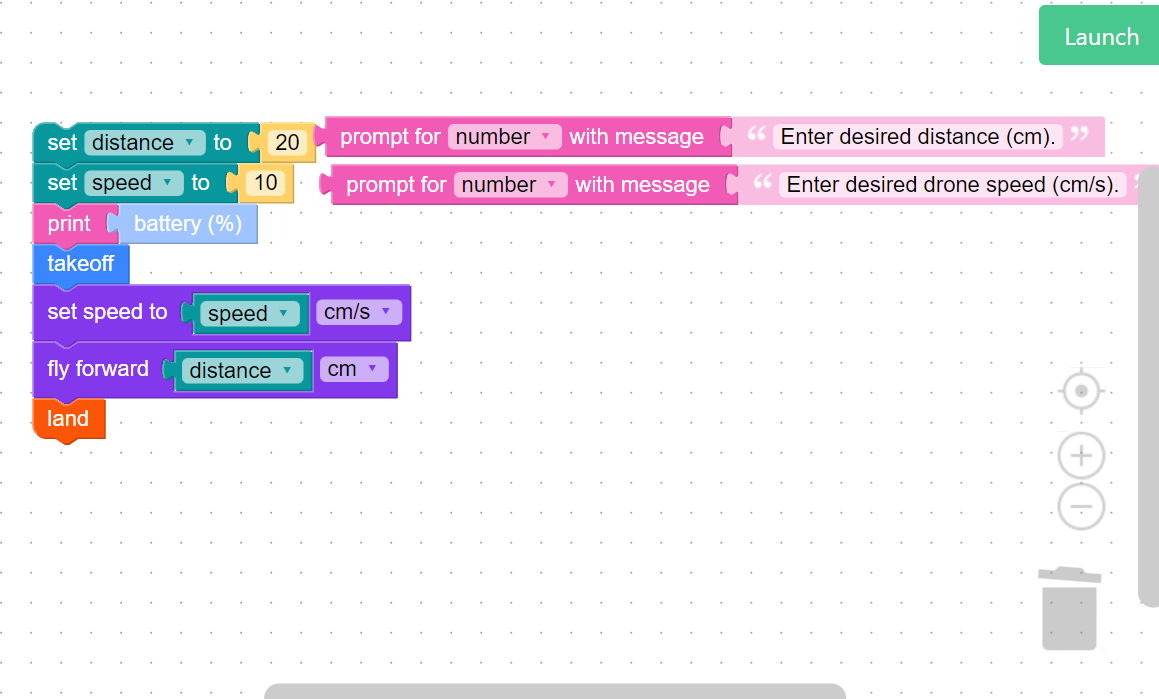
**Description:** Participants will utilize variables in DroneBlocks to control flight parameters such as altitude, speed, and duration, allowing for dynamic and customizable drone flights.

**Instructions:**

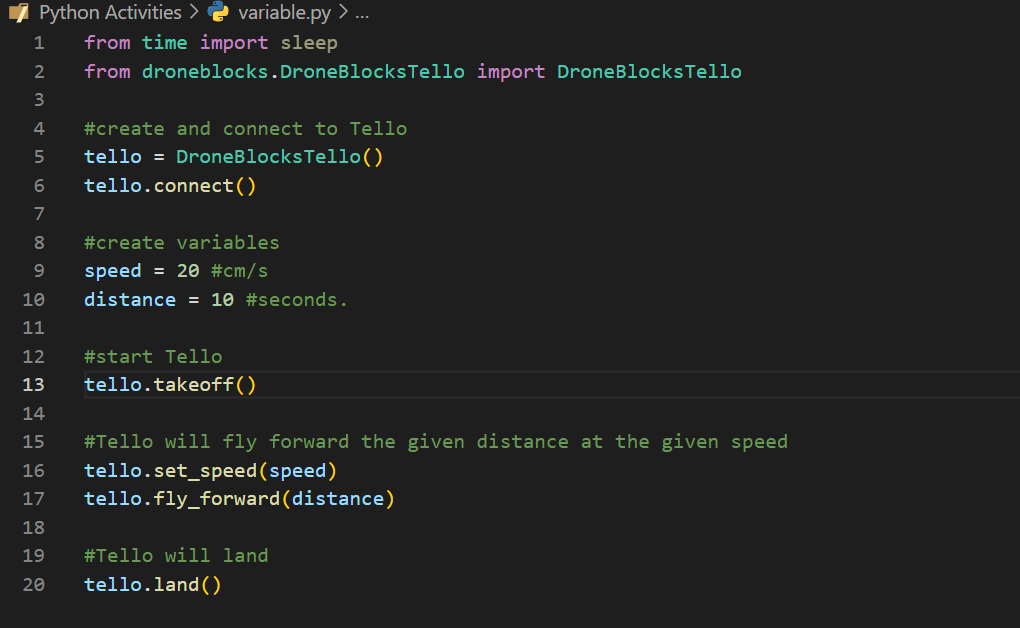
1. Create a new mission or program in DroneBlocks.
2. Define variables to control flight parameters such as altitude, speed, and duration.
3. Add blocks to set the initial values of these variables. For example, you can set the altitude to 10 meters, speed to 2 meters per second, and duration to 10 seconds.
4. Use the variable blocks to dynamically control the drone's flight based on these parameters.
5. Experiment with different flight maneuvers, such as ascending to the specified altitude, flying at the specified speed for the specified duration, and descending back to the ground.
6. Challenge yourself by adding user input blocks to allow for dynamic changes to the variable values during the flight. For example, you can prompt the user to enter a new altitude or speed value before the drone performs a specific maneuver.
7. Test and refine your program to ensure the drone performs the desired flight actions based on the variable values.

**Variable Activity SOLUTIONS**

**Sample Block Code:**



**Sample Python Code:**



**Flip Activity**

**Objective:** Create a program using DroneBlocks that enables the drone to perform multiple flips in the air.

**Key Concepts:** Flips

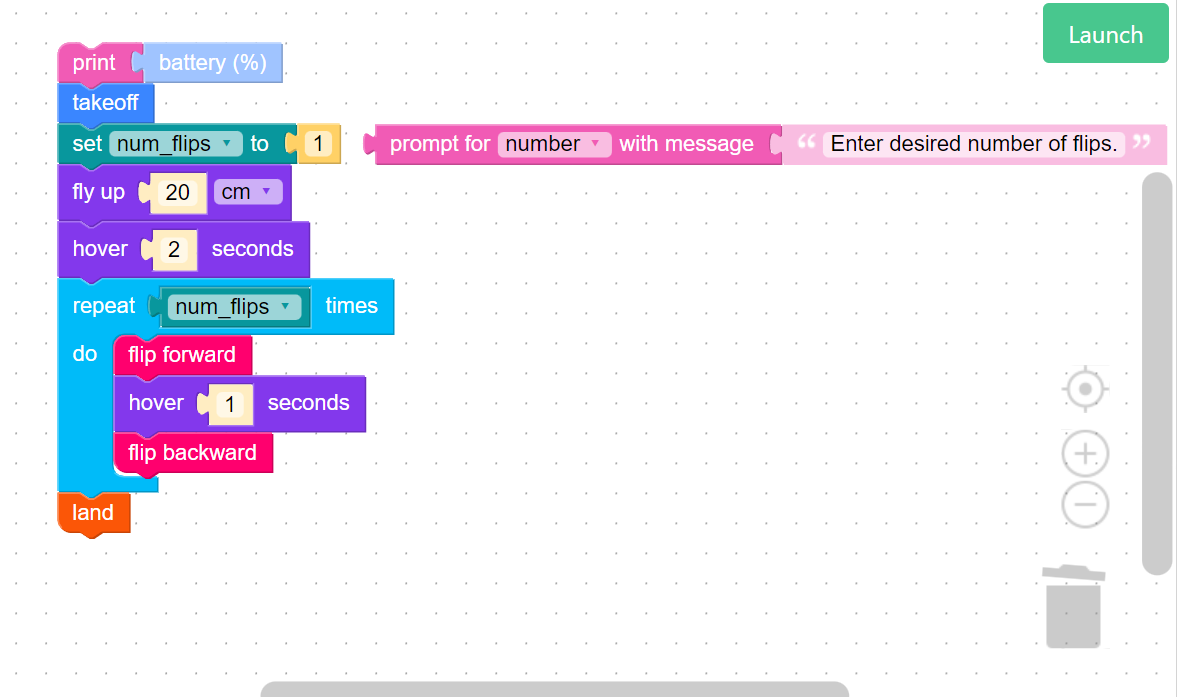
**Description:** Participants will program their drone to perform flips using DroneBlocks, exploring different flip maneuvers and incorporating them into their programs.

**Instructions:**

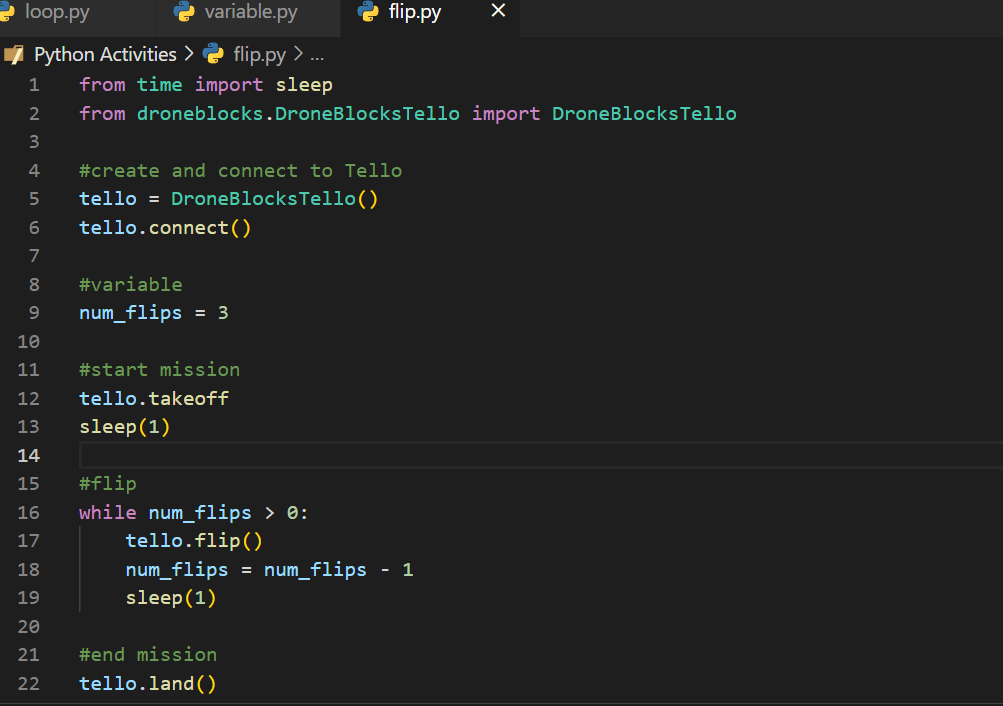
1. Create a new mission or program in DroneBlocks.
2. Add a "Takeoff" block to the beginning of the program to make the drone take off from the ground.
3. Add blocks to make the drone hover at a suitable altitude for performing flips. You can set the altitude based on your available space and the capabilities of your drone.
4. Add a "Flip" block to instruct the drone to perform a specific flip maneuver.
5. Experiment with adding multiple flip blocks to perform a sequence of flips. You can also control the timing between flips using delay blocks to add a sense of rhythm to the drone's movements.
6. Consider using variables to allow for user input to control the number of flips or the direction of the flips dynamically.
7. Add a "Land" block at the end of the program to safely bring the drone back to the ground.

**Flip Activity SOLUTIONS**

**Sample Block Code:**



**Sample Python Code:**



**Loop Activity**

**Objective:** Using loops, create a program in DroneBlocks that allows the drone to fly in the shape of a regular polygon.

**Key Concepts:** Loops, Variables, Math

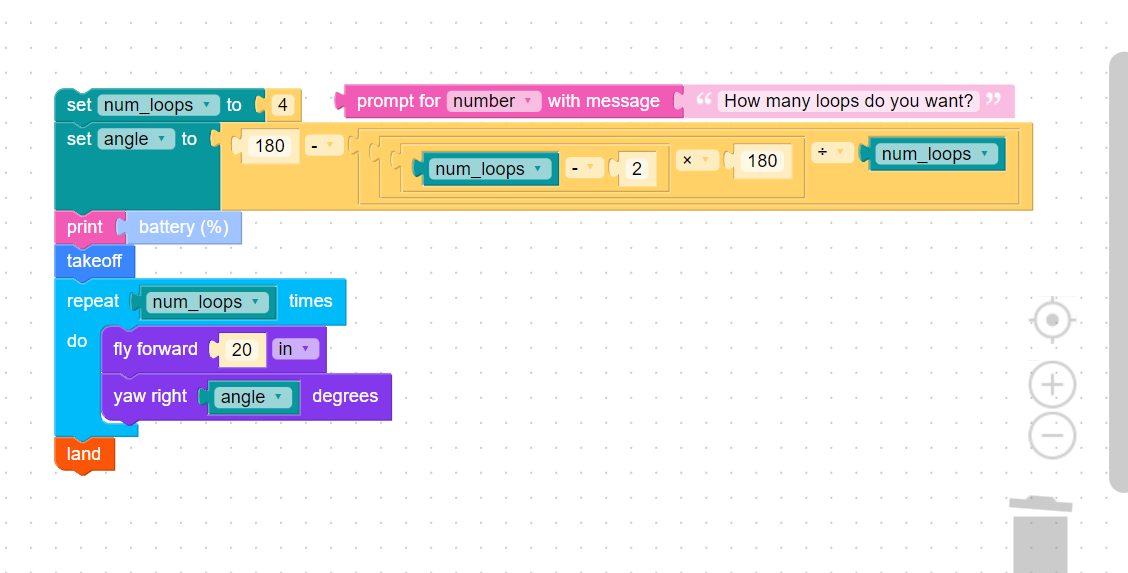
**Description:** Participants will use DroneBlocks to create a program where the drone can fly in the shape of any regular polygon. To accomplish this, participants will use loops, flight commands, math, and variables.

**Instructions:**

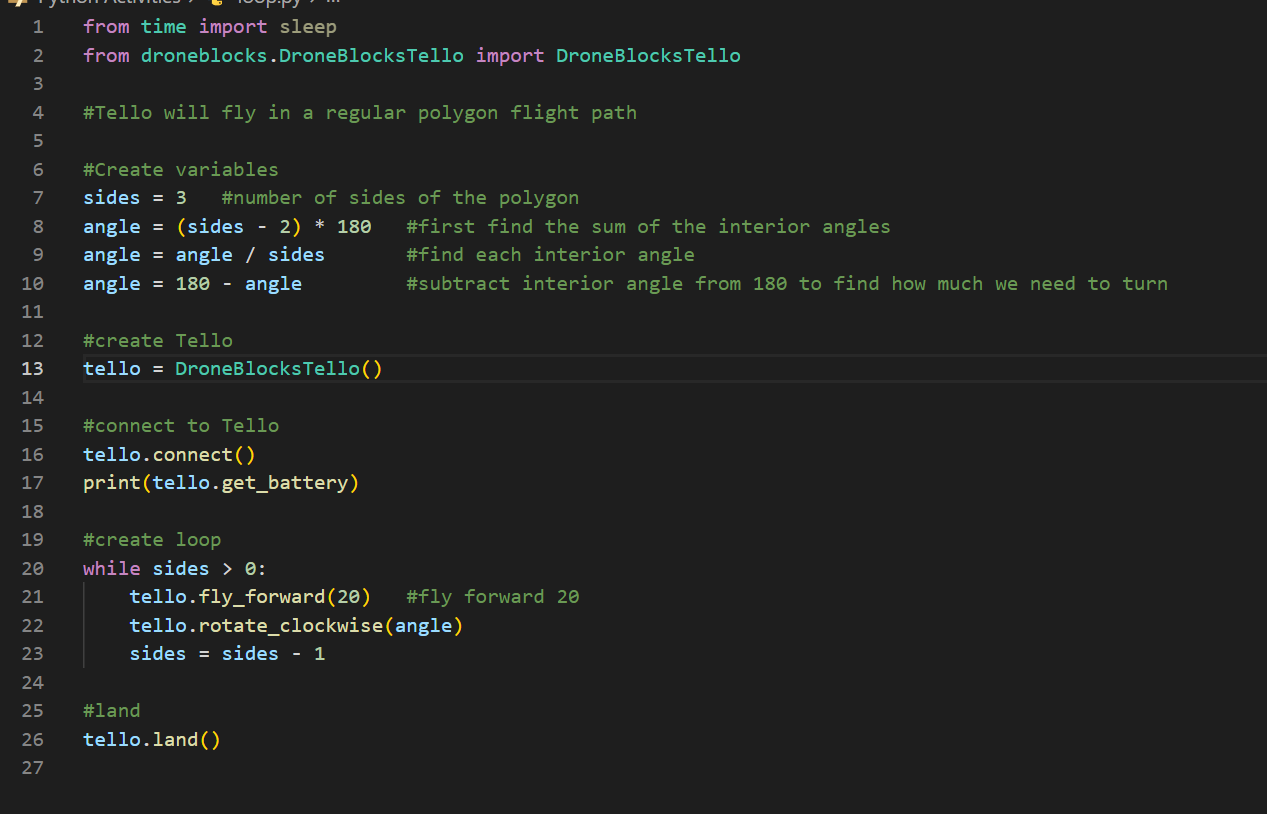
1. In DroneBlocks, create a new mission or program.
2. Add a "Takeoff" block to the beginning of the program to make the drone take off from the ground.
3. Define helpful variables, such as number of sides, side length, and turning angle.
4. Figure out what blocks of code need to be in a loop to effectivlely create a polygon flight path.
5. Adjust the loop settings to specify the number of times the drone should perform the loop.
6. If you need help figuring out code that works for every regualar polygon, start with figuring out how to create basic shapes such as a triangle or square.
7. Finally, add a "Land" block at the end of the program to safely bring the drone back to the ground.

**Loop Activity SOLUTIONS**

**Sample Block Code:**



**Sample Code:**



**Logic Activity**

**Objective:** Create a program using DroneBlocks that incorporates logic statements to control the drone's behavior based on certain conditions.

**Key Concepts:** Logic

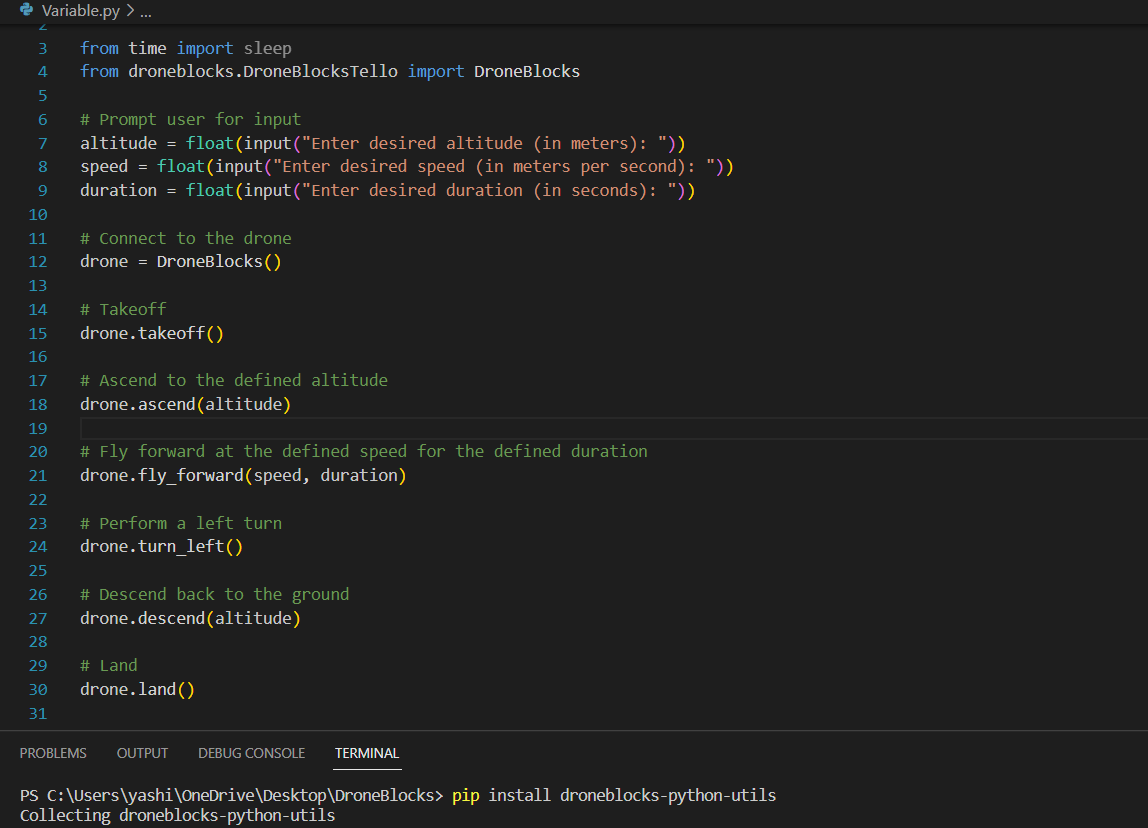
**Description:** Participants will utilize logic statements in DroneBlocks to control the drone's behavior, incorporating conditional statements, sensors, and user inputs to create intelligent behaviors.

**Bonus:** Ask the 3 variables as user Input

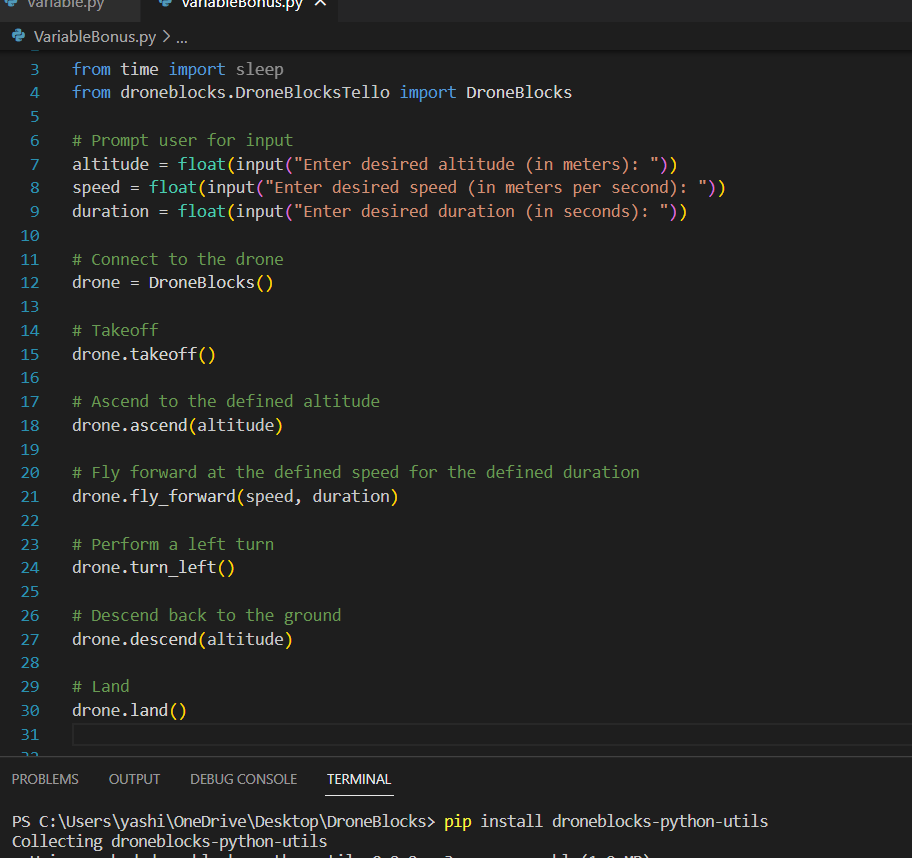
**Instructions:**

1. Create a new mission or program in DroneBlocks.
2. Identify a specific condition or criteria that you want to use to control the drone's behavior. For example, you can use the drone's altitude, battery level, or user input as the basis for the logic statements.
3. Add blocks to retrieve the necessary information from the drone. For instance, if you want to use altitude as a condition, include a block to get the current altitude value.
4. Integrate a logic statement such as an "if-else" block to evaluate the condition and determine the appropriate action for the drone.
5. Inside the logic statement, add blocks to specify what should happen if the condition is true or false. This can include various flight commands, maneuvers, or even notifications.
6. Repeat steps 3 to 6 for additional conditions or logic statements that you want to incorporate into your program.
7. Test your program by running it and observing how the drone behaves based on the specified logic statements.
8. Adjust and refine your program as needed to ensure that the logic statements and actions align with your intended behavior.

**Logic Activity SOLUTIONS**

**Sample Code:**  


**Bonus code:**



**Racing Activity**

**Objective:** Create a racing game using DroneBlocks that incorporates loops, variables, flips, and logic to engage middle school and high school students in programming and drone control.

**Key Concepts:** Loops, Variables, Flips, Logic

**Description:** Participants will work in groups to program their drones for a racing game, utilizing loops, variables, flips, and logic to navigate through a racecourse, track completion times, and determine the winner.

**Bonus:** Add a countdown timer for 3 seconds before the start of each lap.

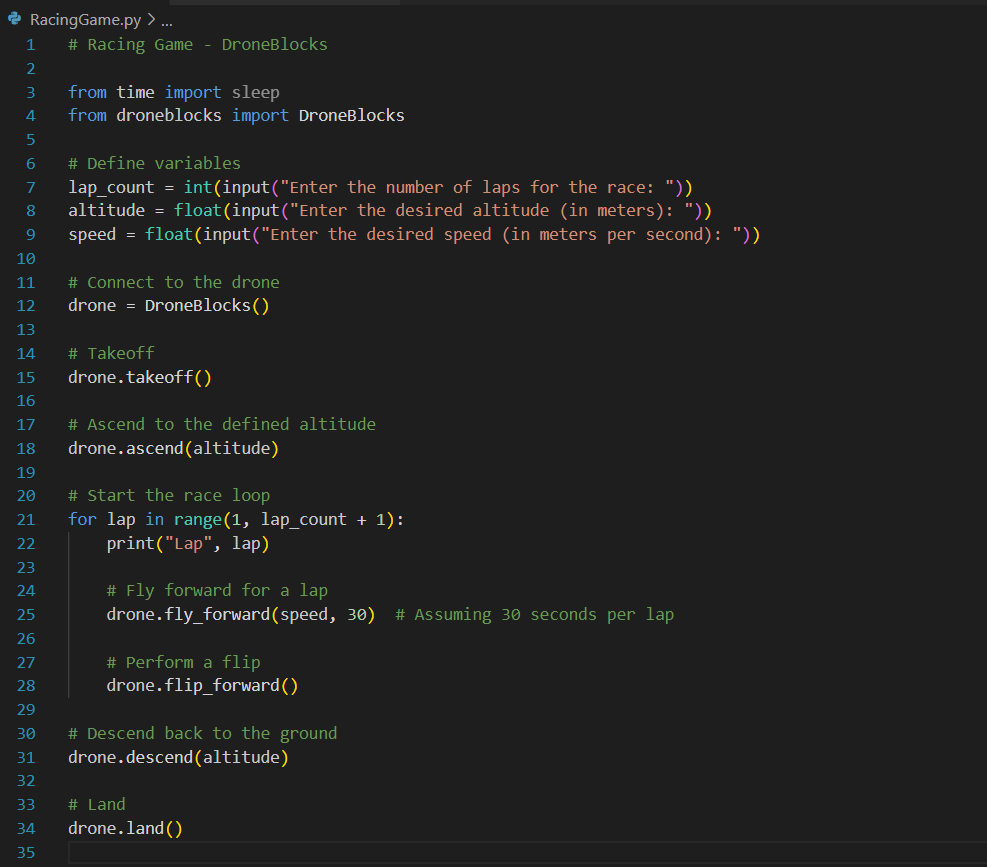
**Instructions:**

1. Set up a racecourse using cones, markers, or other designated boundaries.
2. Create a new mission or program in DroneBlocks.
3. Define variables to control game parameters such as altitude, speed, duration, and player keys.
4. Set initial values for the variables, including the starting altitude, desired speed, duration for completing the racecourse, and scores.
5. Utilize loops to repeat the game rounds until a certain condition is met (e.g., completing a specific number of laps or reaching a time limit).
6. Within each game round loop, include the following components:
   1. Start by displaying a message to indicate the start of the race round.
   2. Use logic statements (if-else blocks) to control the drone's movements and ensure it stays within the racecourse boundaries.
   3. Incorporate flips as a challenge or bonus feature during the race. For example, students can include a specific sequence of commands that triggers a flip if performed correctly.
7. Update the scores or timing variables to track the progress and completion time of each group.
8. Display the final scores or completion times at the end of the race and announce the winning group.

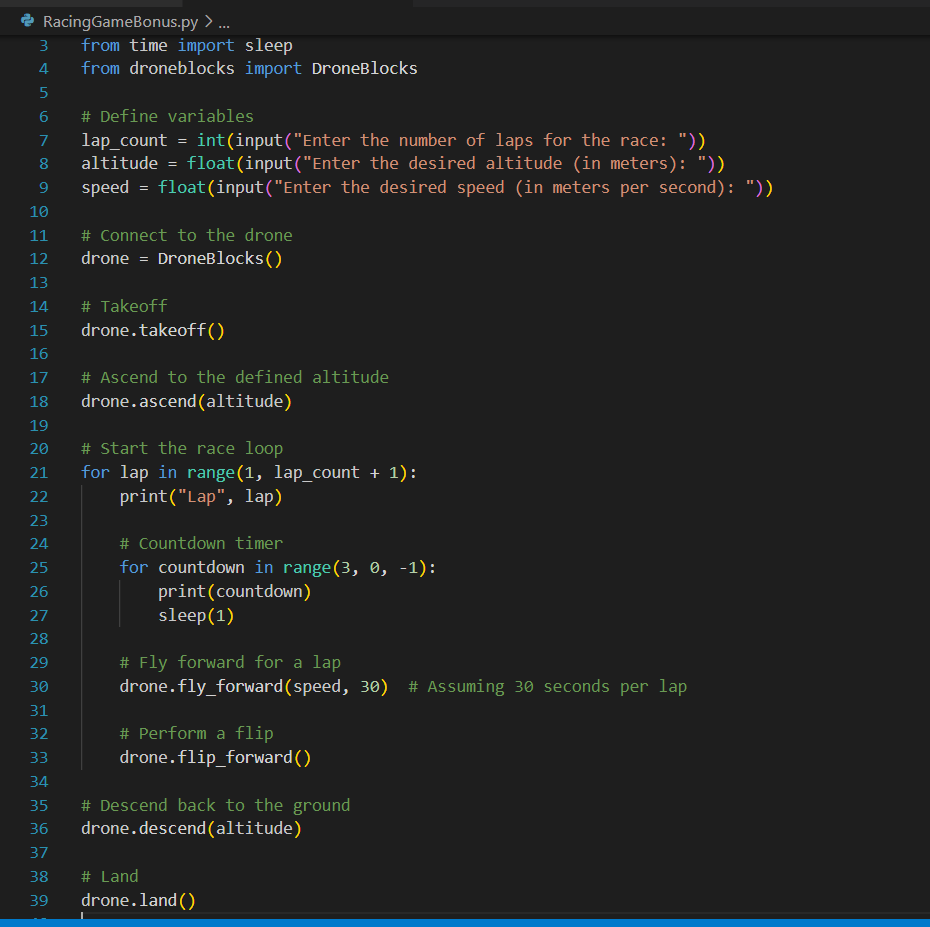
9. Add a "Land" block at the end of the program to safely bring the drone back to the ground.

**Racing Activity SOLUTIONS**

**Sample Code:**



**Bonus Code:**



**Ultimate Drone Activity (A bit more difficult)**

**Objective:** Create an interactive game using DroneBlocks that incorporates loops, variables, flips, and logic to challenge players and showcase the capabilities of the drone.

**Key Concepts:** Loops, Variables, Flips, Logic

**Description:** Participants will create a game using DroneBlocks that combines loops, variables, flips, and logic to engage players in competitive challenges, incorporating elements like reaction times, scoring, flips, and customizations.

**Instructions:**

1. Create a new mission or program in DroneBlocks.
2. Set up a safe flying area outdoors or indoors with enough space for the drone to maneuver.
3. Define variables to control game parameters such as altitude, speed, duration, scores, and player keys.
4. Add blocks to set the initial values of these variables. For example, set the altitude to a suitable height, speed to a moderate level, duration for each game round, and scores to 0 for both players.
5. Display instructions or game rules using text blocks so that players understand how to play the game.
6. Use loops to repeat the game rounds until a certain condition is met (e.g., a specific score is reached or a time limit is reached).
7. Within each game round loop, include the following components:
   1. Start by displaying a message to indicate that the round is starting.
   2. Use a delay block to introduce a random wait time before the LED turns on, signaling the players to react.
   3. Utilize logic statements (if-else blocks) to determine which player presses their designated key first upon seeing the LED.
   4. Based on the outcome of the logic statements, award points to the respective players, update their scores, and display a message indicating the winner.
   5. Incorporate flips into the game as a challenge or bonus feature. For example, you can instruct the drone to perform a flip if a player achieves a certain score or completes a specific action.
   6. Implement a delay block to pause the game for a few seconds before starting the next round.
8. Continue the game loop until the desired condition is met (e.g., a specific number of rounds are played, or a player reaches a predefined score).
9. Display the final scores of both players and announce the winner of the game.
10. Add a "Land" block at the end of the program to safely bring the drone back to the ground.

**Ultimate Drone Activity SOLUTION**

**Sample Code:**

**Bonus Code:**