girls who code

University of Michigan Department of Computational Medicine and Bioinformatics

Running of the Bots

Objective
Use Ozobot to perform an experiment. Concepts covered include probability, computer programming, conditional statements, and random number generators.

Introduction
If you toss a fair coin what is the **probability** it will land on heads? One-half. If you toss a fair die what is the **probability** it will land on 1? One-sixth. If you toss an unfair die what is the **probability** it will land on 1? You would have to perform an **experiment** to quantify the **experimental probability** because the **theoretical probability** is no longer one-sixth. How can we use Ozobot to quantify **experimental probabilities**? How does **computer programming** help us perform this experiment?

Experiment
Set up two Ozobots, one on each track. Have one Ozobot execute the Ozocode FairCoin and the other execute the Ozocode UnfairCoin. After inspecting the code, have students **hypothesize** what the **theoretical probability** should be for each Ozobot to choose left, right, or straight at the track intersection.

Begin the experiment and have students record the direction the Ozobot chooses at the intersection (left, right, or straight). Over time the **sample** will allow students to quantify the **experimental probabilities** for each direction. For UnfairCoin what are the values for P_{Right}, P_{Left} , $P_{Straight}$? What about for FairCoin? Have the students **infer** the **theoretical probabilities** based on the **experimental probabilities**.

Use the sample collected to identify **runs** in the data. Calculate the **probability** of Ozobot turning a given direction consecutively for run length L. When Ozobot is executing UnfairCoin, what is the probability Ozobot makes the following **permutatation** of directional choices: RRSSSSL? How many permutations of directional choices are possible when Ozobot traverses the track 100 times?

Extensions
Use fair die and unfair die to run a concurrent experiment with different **theoretical probabilities**. Have students **plot** the data from the experiments with Ozobot and the die. Have students hypothesize what **sample size** is needed to see accurate estimates of the **theoretical probability** (e.g. experimental probability converges on theoretical probability). Have students write Ozocode to create new probabilities of choosing left, right or straight.

Instructional Concepts and Definitions

- **Probability** is the likelihood that an event will occur and is a number between 0 and 1. The **theoretical probability** that a fair coin will land on tails (P_H) is 1/2 because there is only one way this event can happen but there are two possible outcomes (heads or tails).

$$Probability\ of\ an\ event = \frac{Number\ of\ ways\ \ the\ event\ can\ happen}{Total\ number\ of\ outcomes}$$

- The **theoretical probability** is what we expect to happen but the **experimental probability** is what actually happens. We expect a coin to land on heads half of the time so the **theoretical probability** is P_H=1/2. But if we flip a coin 10 times it may actually land on heads 6 times, so the **experimental probability** is P_H=3/5.
- When Ozobot is traversing the track without code, it will randomly choose a direction to proceed when it encounters the intersection. Ozobot uses a **random number generator** to choose the direction and each direction can be chosen with equal probability. For example, the probability of choosing left at a three-way intersection (P_L) is 1/3.
- A **random number generator** produces a sequence of numbers that cannot be reasonably predicted better than by a random chance. We use a **random number generator** in the Ozocode. In FairCoin a random number generator randomly chooses an **integer** between 1 and 3. If the number is 1 then Ozobot will turn left, if the number is 2 then Ozobot will continue straight, if the number is 3 then Ozobot will turn right.
- Technically most things we consider **random number generators** are **pseudo-random number generators.** Simply put, **pseudo-random number generators** are not truly random because they are determined based off an initial value which is called a **seed**. This is usually random enough, but to be a **true-random number generator** we would use hardware that generates random numbers from physical processes (e.g. thermal noise) rather than a computer program.
- A **sample** is a subset of individuals chosen randomly from a larger set (a **population**) to estimate characteristics of the whole **population**. Performing our experiment is like taking a **sample** from a **population** of people wearing t-shirts with right, left, and straight printed on them. When the Ozobot is executing the UnfairCoin code, the number of people wearing t-shirts with right is greater than those wearing t-shirts with left or straight. In our experiment, our **sample size** is 100.
- A **probability distribution** is a mathematical function that provides the probabilities of different possible outcomes occurring in an experiment. Our Ozobots are programmed with Ozocode to choose specific direction at an intersection with a given **probability**. UnfairCoin and FairCoin have different **probabilities** (e.g. P_R, P_L, P_S) and therefore different **probability distributions**.

$$U \sim f_u(u) = P_R \times\ I(R) + P_L \times\ I(L) + P_S \times\ I(S)$$

$$where\ I(R) = \begin{cases} 1\ if\ Ozobot\ turns\ right \\ 0\ otherwise \end{cases}$$

$$where\ I(L) = \begin{cases} 1\ if\ Ozobot\ turns\ left \\ 0\ otherwise \end{cases}$$

$$where\ I(S) = \begin{cases} 1\ if\ Ozobot\ continues\ straight \\ 0\ otherwise \end{cases}$$

- A **run** is an unbroken sequence of events that are the same. For example, if Ozobot turns right five times in a row that is a run. Humans typically underestimate the lengths of the longest runs in a sample of coin flips (e.g. consecutive heads or tails). The observation of run lengths in a random process is a good test of whether or not the process is truly random. Mathematical tests for runs are used to evaluate **random number generators**.
- Humans also typically believe if something happens more frequently than normally expected, such as a long run, then that event will happen less frequently in the future. This is known as **gambler's fallacy** or Monte Carlo fallacy. When events are **random** and **independent**, such as Ozobot's runs in FairCoin, every event is equally likely and there is no such balancing phenomena. For example, if Ozobot just turned right 5 times in a row, the probability of turning right on the 6th time is still 1/3.
- The **programming language** we are using is Ozoblockly. It is a **graphical user interface (GUI)** programming language. Other **programming languages** include Python, R, C++, and Java.
- Our code uses **conditional statements** which test a variable against a value and act in one way if the condition is met by the variable or another way if not. They are also commonly called **if statements**. Conditional statements evaluate variables known as **booleans**. A **Boolean** is a variable that can represent binary values: True or False. For example in our code if the **variable** Path is equal to 1, then the **boolean** is True. According to the conditional statement, when the equality is evaluated as true, then the Ozobot will turn left at the intersection.

```
If Path==1:
   Ozobot turns left
```

- Our code also uses a **for loop**. We use the for loop to perform a given action 100 times. In the example, below, we would print the numbers 1, 2, 3, …, 100. Another type of loop is a **while loop**. This loop continues while a variable, statement, or equality is evaluated as true and only when it is evaluated as False. In other words, while something is true, do something.

```
For i from 1 to 100 by 1:
   Print i
```

Answers

1. **Infer theoretical probabilities** of Ozobot's run.
For the FairCoin **the theoretical probability** of Ozobot choosing straight, right, or left is 1/3 for each direction. This is also what the Ozobot would do if left to traverse the track without executing the code.

Probability of choosing…	UnfairCoin (U)	FairCoin (F)
Right (P_R)	4/6	1/3
Left (P_L)	1/6	1/3
Straight (P_S)	1/6	1/3

2. Calculate the **probability** of Ozobot turning a given direction consecutively for run length L.
For FairCoin this is $(\frac{1}{3})^L$. For UnfairCoin the probability of a run of right is $(\frac{2}{3})^L$, and left and straight are each $(\frac{1}{6})^L$. This is because of the **Rule of Multiplication**. If event A and even B are **independent**, the probability of both events happening is $P_A \times P_B$.

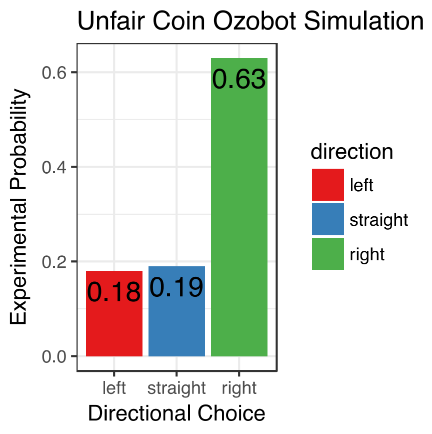
3. When Ozobot is executing UnfairCoin, what is the probability Ozobot makes the following **permutation** of directional choices: RRSSSSL?

$$\frac{4}{6}^2 \times \frac{1}{6}^5 = 0.1112397$$

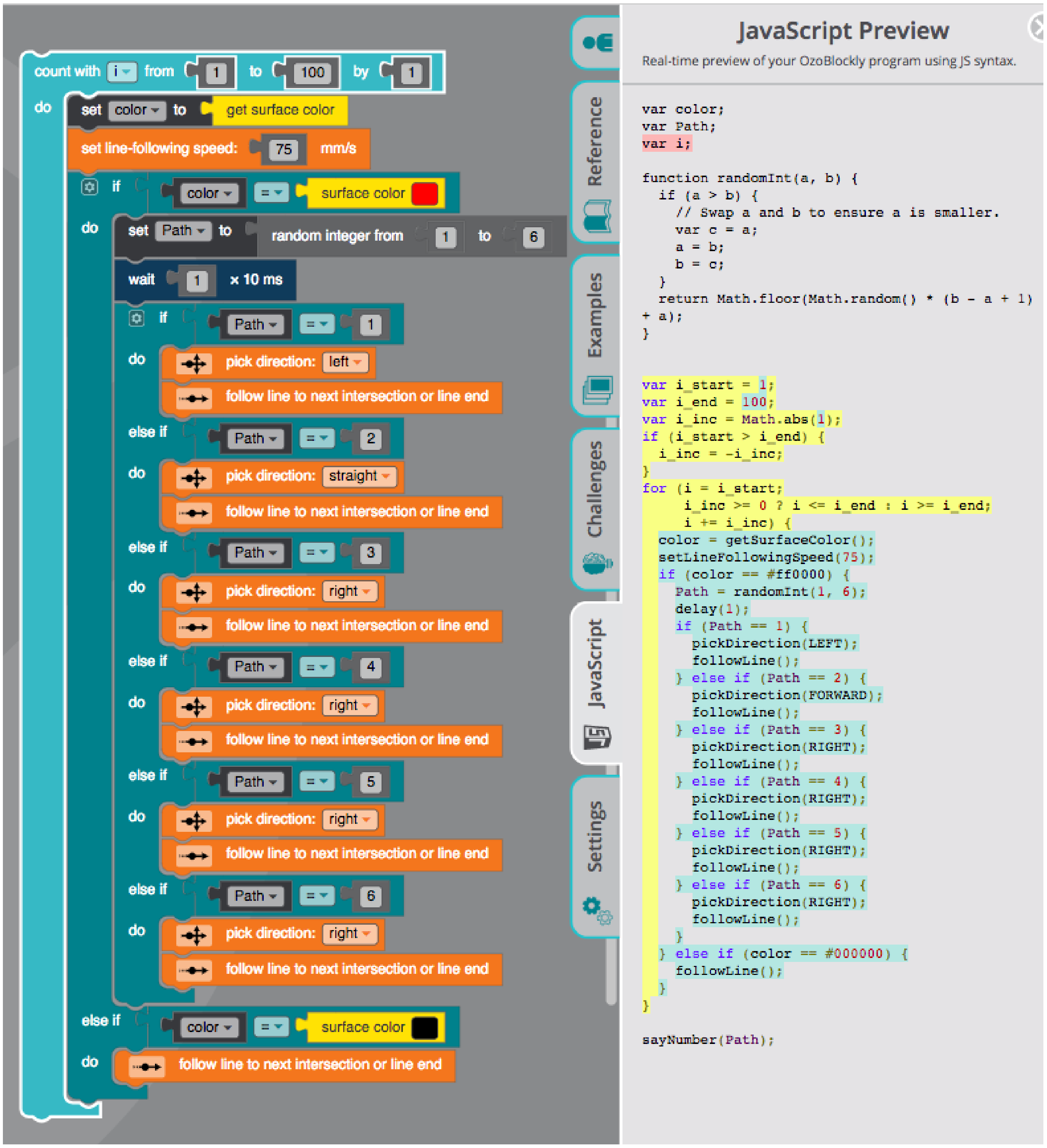
4. How many **permutations** of directional choices are possible when Ozobot traverses the track 100 times?

10 coin flips have $2^{10} = 1024$ possible permutations while 100 coin flips have $2^{100} = 1.267651 \times 10^{30}$ possible permutations. That number escalates quickly, **exponentially** actually. While executing FairCoin code, when Ozobot traverses the track 100 times there are $2^{100} = 5.153775 \times 10^{47}$ possible permutations of directional choices. This is because we have 3 equally probable outcomes for a single trial and we carry out N trials.

5. **Plot** of a simulated experiment (n=100) for UnfairCoin.



UnfairCoin Ozocode



```
count with i from 1 to 100 by 1
do  set color to get surface color
    set line-following speed: 75 mm/s
    if color = surface color 🟥
    do  set Path to random integer from 1 to 6
        wait 1 x 10 ms
        if Path = 1
        do  pick direction: left
            follow line to next intersection or line end
        else if Path = 2
        do  pick direction: straight
            follow line to next intersection or line end
        else if Path = 3
        do  pick direction: right
            follow line to next intersection or line end
        else if Path = 4
        do  pick direction: right
            follow line to next intersection or line end
        else if Path = 5
        do  pick direction: right
            follow line to next intersection or line end
        else if Path = 6
        do  pick direction: right
            follow line to next intersection or line end
    else if color = surface color ⬛
    do  follow line to next intersection or line end
```

JavaScript Preview

Real-time preview of your OzoBlockly program using JS syntax.

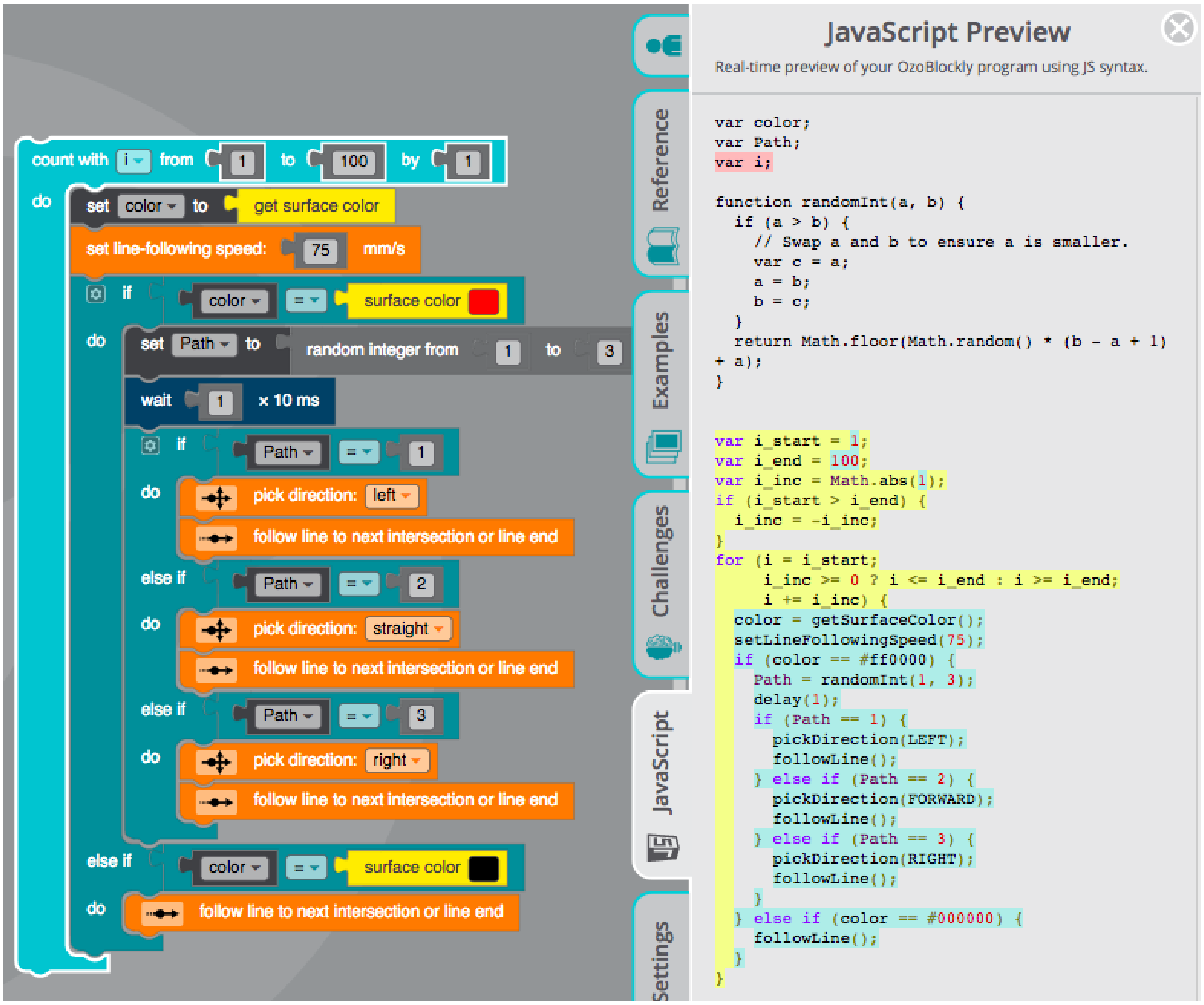
```
var color;
var Path;
var i;

function randomInt(a, b) {
  if (a > b) {
    // Swap a and b to ensure a is smaller.
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1)
+ a);
}


var i_start = 1;
var i_end = 100;
var i_inc = Math.abs(1);
if (i_start > i_end) {
  i_inc = -i_inc;
}
for (i = i_start;
     i_inc >= 0 ? i <= i_end : i >= i_end;
     i += i_inc) {
  color = getSurfaceColor();
  setLineFollowingSpeed(75);
  if (color == #ff0000) {
    Path = randomInt(1, 6);
    delay(1);
    if (Path == 1) {
      pickDirection(LEFT);
      followLine();
    } else if (Path == 2) {
      pickDirection(FORWARD);
      followLine();
    } else if (Path == 3) {
      pickDirection(RIGHT);
      followLine();
    } else if (Path == 4) {
      pickDirection(RIGHT);
      followLine();
    } else if (Path == 5) {
      pickDirection(RIGHT);
      followLine();
    } else if (Path == 6) {
      pickDirection(RIGHT);
      followLine();
    }
  } else if (color == #000000) {
    followLine();
  }
}

sayNumber(Path);
```

FairCoin Ozocode



count with i from 1 to 100 by 1
do
set color to get surface color
set line-following speed: 75 mm/s
if color = surface color
do
set Path to random integer from 1 to 3
wait 1 x 10 ms
if Path = 1
do
pick direction: left
follow line to next intersection or line end
else if Path = 2
do
pick direction: straight
follow line to next intersection or line end
else if Path = 3
do
pick direction: right
follow line to next intersection or line end
else if color = surface color
do
follow line to next intersection or line end
Reference
Examples
Challenges
JavaScript
Settings

JavaScript Preview

Real-time preview of your OzoBlockly program using JS syntax.

```
var color;
var Path;
var i;

function randomInt(a, b) {
  if (a > b) {
    // Swap a and b to ensure a is smaller.
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1)
+ a);
}


var i_start = 1;
var i_end = 100;
var i_inc = Math.abs(1);
if (i_start > i_end) {
  i_inc = -i_inc;
}
for (i = i_start;
     i_inc >= 0 ? i <= i_end : i >= i_end;
     i += i_inc) {
  color = getSurfaceColor();
  setLineFollowingSpeed(75);
  if (color == #ff0000) {
    Path = randomInt(1, 3);
    delay(1);
    if (Path == 1) {
      pickDirection(LEFT);
      followLine();
    } else if (Path == 2) {
      pickDirection(FORWARD);
      followLine();
    } else if (Path == 3) {
      pickDirection(RIGHT);
      followLine();
    }
  } else if (color == #000000) {
    followLine();
  }
}
```

Boolean value: binary variable with two possible values

```
If something is TRUE:
   Then do something
Else if something is TRUE:
   Then do something
Else:
   Do something
```

```
For i from 1 to 100 by 1:
   Do something


While something is TRUE:
   Do something
```