

# Cannyblocks

[Introduction](#)

[Programming Constructs](#)

[Logic - If](#)

[Logic - Boolean](#)

[Loops](#)

[Math](#)

[Text](#)

[Lists](#)

[Variables](#)

[Library Functions](#)

[Cannybots](#)

[Turtle](#)

[Example Code](#)

[Turtle](#)

[PiMazing](#)

[Code Generation](#)

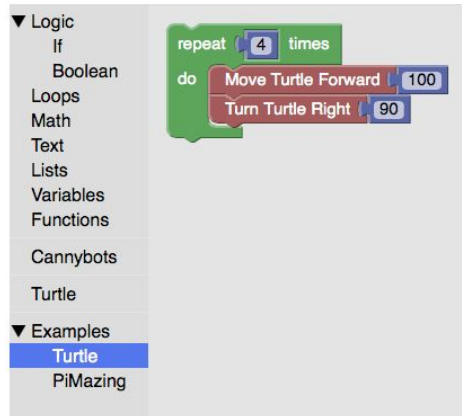
[JavaScript](#)

[Python](#)

# Introduction

Cannyblocks is a graphical drag and drop programming environment similar to Scratch but based on [Google Blockly](#).

Here's is a simple example of some code that controls the Cannybot Turtle



Cannybots (Blockly) supports the standard programming concepts in addition to some custom functionality. The key features are:

1. Conditional & boolean Logic
2. Loopings
3. Lists
4. Functions
5. Math
6. Text
7. Python and JavaScript code generation
8. Cannybots Extensions

Cannyblocks extends Blockly with custom blocks to support interacting with a Cannybot over the air using Bluetooth LE. The custom functions support:

- general purpose sending and receiving of message and
- Logo Turtle library.

The Cannyblocks webapp runs within a webserver hosted on RaspberryPi, iOS or Android tablets; smartphone screens are too small to use interactively directly, but the server will still run, which has it's uses described below.

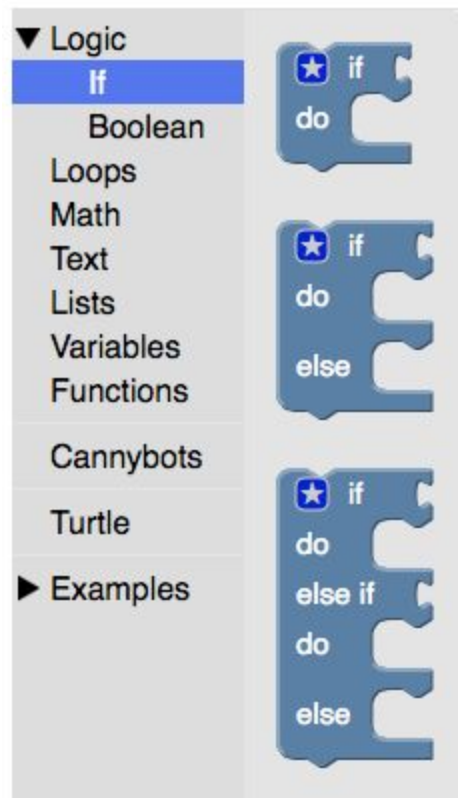
On all the platforms you can either directly interact with the Cannyblocks app on the local device, or, you can browse to the device from another machine on the same network.

For example, you could pair the iOS or Android app with a Cannybot over Bluetooth LE as normal and then access the Cannyblocks web app (served up by an embedded webserver on the device) from multiple RaspberryPi or PC/Mac browsers.

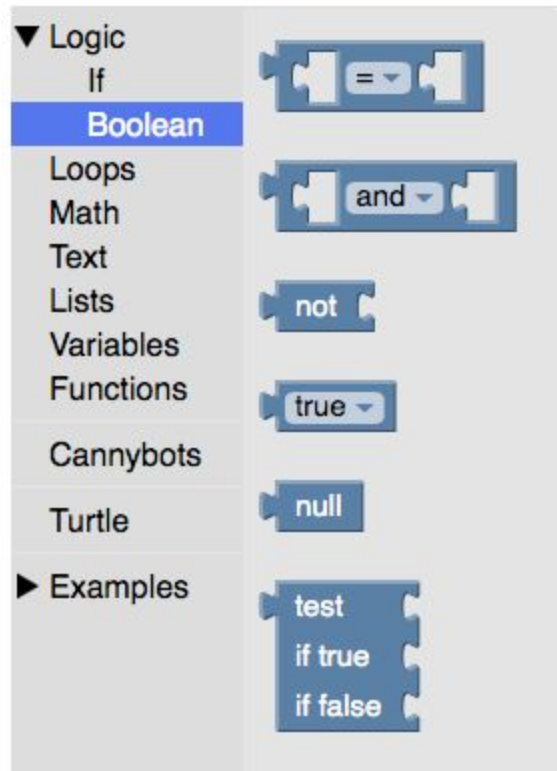
The reverse is also possible and the combinations are too many to list here.

# Programming Constructs

## Logic - If



## Logic - Boolean

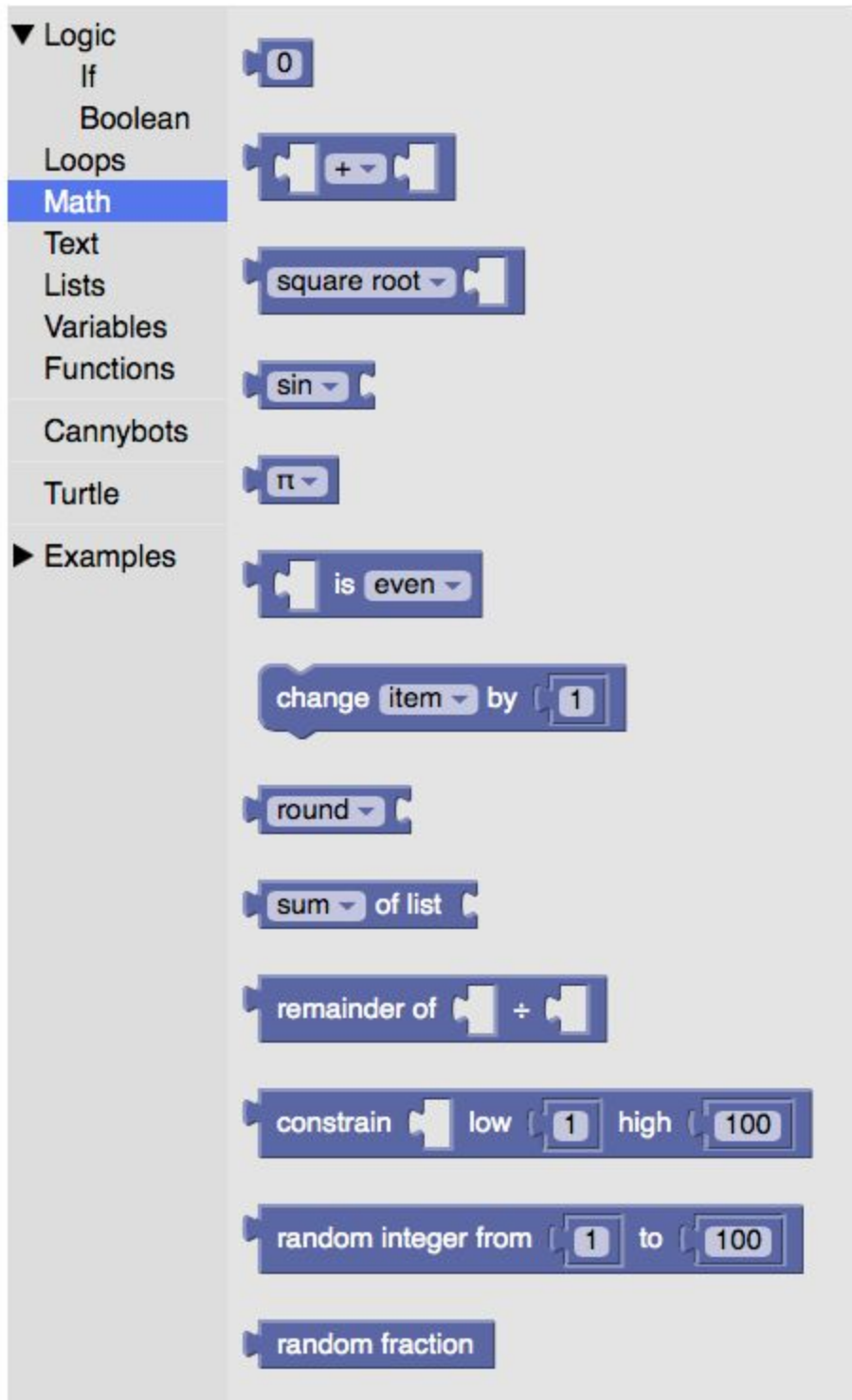


## Loops

The image shows the Scratch interface for the Loops category. On the left is a sidebar with a tree view containing: Logic, If, Boolean, Loops (highlighted in blue), Math, Text, Lists, Variables, Functions, Cannybots, Turtle, and Examples. The main workspace on the right displays several loop blocks:

- A "repeat" block with a dropdown menu set to "times" and a numeric input field containing "10". Below it is a "do" block.
- A "repeat while" block with a dropdown menu set to "while" and a "do" block below it.
- A "count with" block with a dropdown menu set to "i", followed by "from" with a numeric input field containing "1", "to" with a numeric input field containing "10", and "by" with a numeric input field containing "1". Below it is a "do" block.
- A "for each item" block with a dropdown menu set to "i" and "in list". Below it is a "do" block.
- A "break out" block with a dropdown menu set to "of loop".

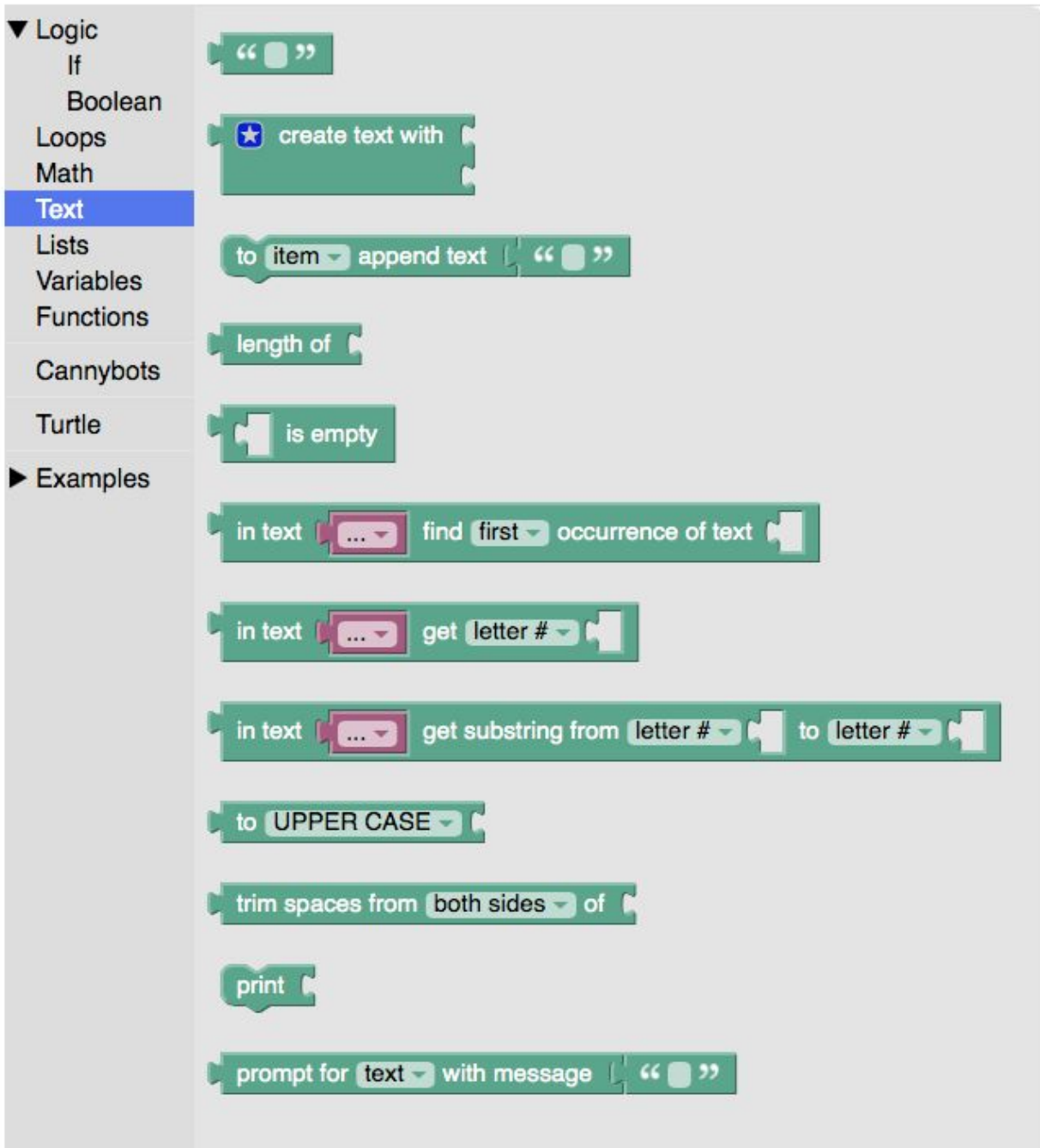
# Math



A screenshot of the Scratch 'Math' block palette. The left sidebar shows a tree view with categories: Logic, If, Boolean, Loops, **Math** (highlighted), Text, Lists, Variables, Functions, Cannybots, Turtle, and Examples. The main area displays various math blocks:

- 0
- +
- square root
- sin
- $\pi$
- is even
- change item by 1
- round
- sum of list
- remainder of  $\div$
- constrain low 1 high 100
- random integer from 1 to 100
- random fraction

## Text

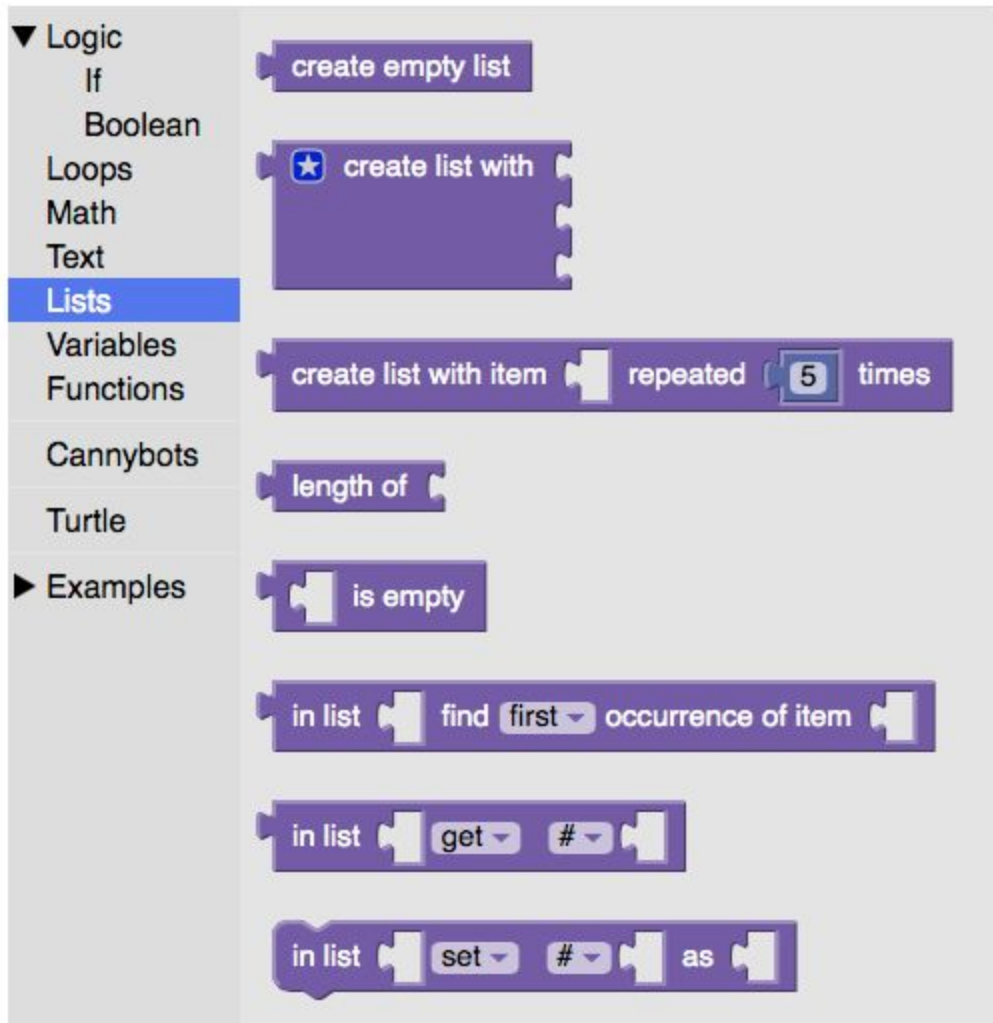


The image shows the Scratch Text block palette. On the left is a sidebar with categories: Logic, If, Boolean, Loops, Math, Text (highlighted in blue), Lists, Variables, Functions, Cannybots, Turtle, and Examples. The main area displays various text-related blocks:

- “ ”
- ★ create text with
- to item append text “ ”
- length of
- is empty
- in text ... find first occurrence of text
- in text ... get letter #
- in text ... get substring from letter # to letter #
- to UPPER CASE
- trim spaces from both sides of
- print
- prompt for text with message “ ”



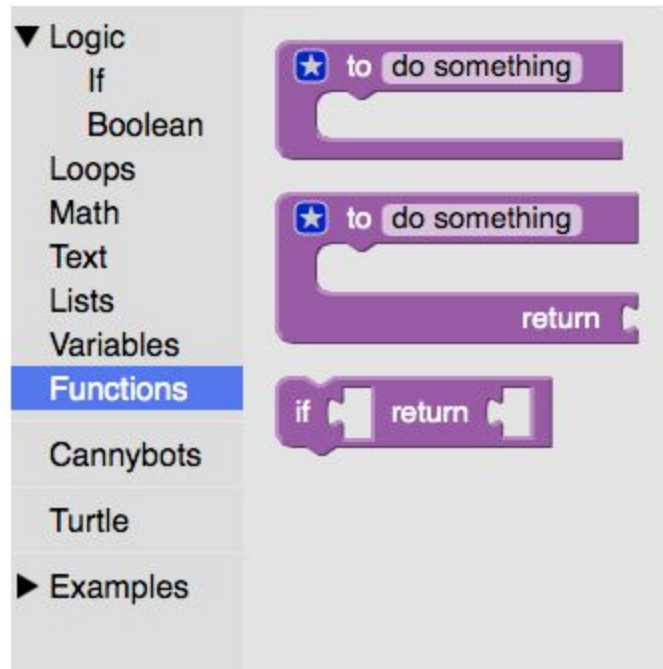
## Lists



A screenshot of the Scratch Lists block palette. The left sidebar shows a tree view with categories: Logic, If, Boolean, Loops, Math, Text, Lists (highlighted in blue), Variables, Functions, Cannybots, Turtle, and Examples. The main area displays several list-related blocks:

- create empty list**: A simple block to create an empty list.
- create list with**: A block with a star icon and a large empty input field for creating a list.
- create list with item repeated 5 times**: A block with a dropdown menu for an item and a numeric input field set to 5.
- length of**: A block to determine the length of a list.
- is empty**: A block to check if a list is empty.
- in list find first occurrence of item**: A block with a dropdown for an item and a return value field.
- in list get #**: A block with a dropdown for an item, a numeric input for an index, and a return value field.
- in list set # as**: A block with a dropdown for an item, a numeric input for an index, and an input field for a value to set.

## Variables



The image shows a screenshot of the Scratch Functions palette. The left sidebar contains a list of categories: Logic, If, Boolean, Loops, Math, Text, Lists, Variables, Functions (highlighted in blue), Cannybots, Turtle, and Examples. The main area displays three function blocks:

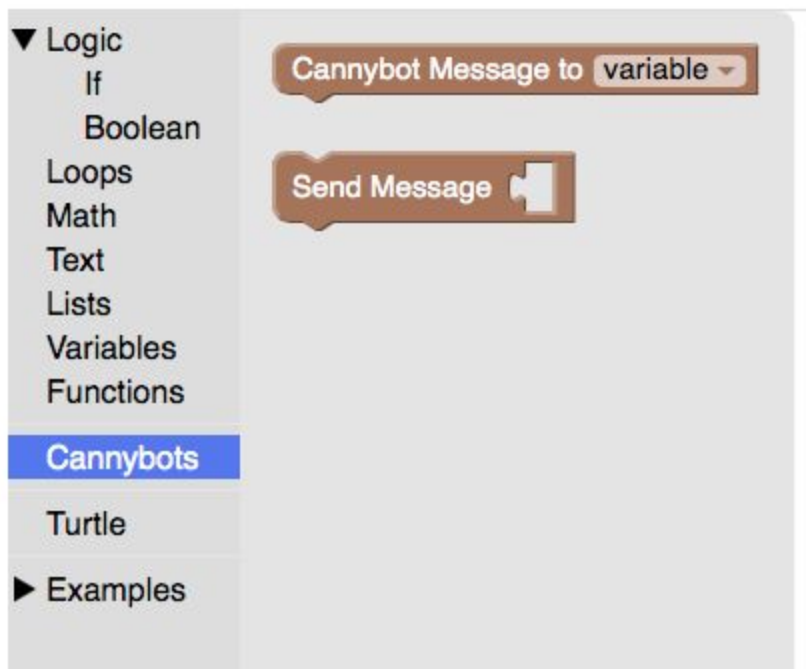
- A purple block with a star icon, labeled "to do something".
- A purple block with a star icon, labeled "to do something", with a "return" block attached to its bottom-right connector.
- A purple block labeled "if" with a "return" block attached to its right connector.

# Library Functions

## Cannybots

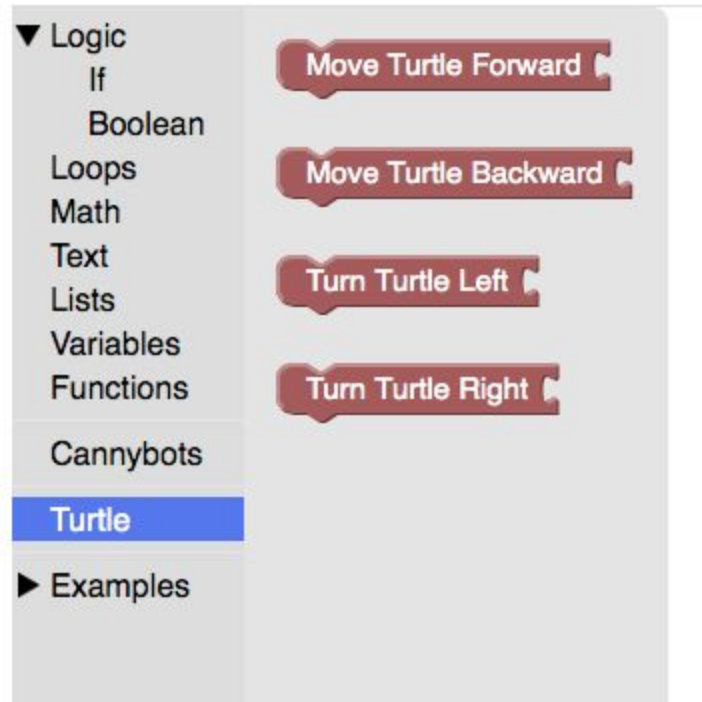
The “Cannybot Message to [variable]” will place the message received from a Cannybot into the named [variable].

The “Send Message [string]” will send a user define “[string]” or variable value to a Cannybot.



## Turtle

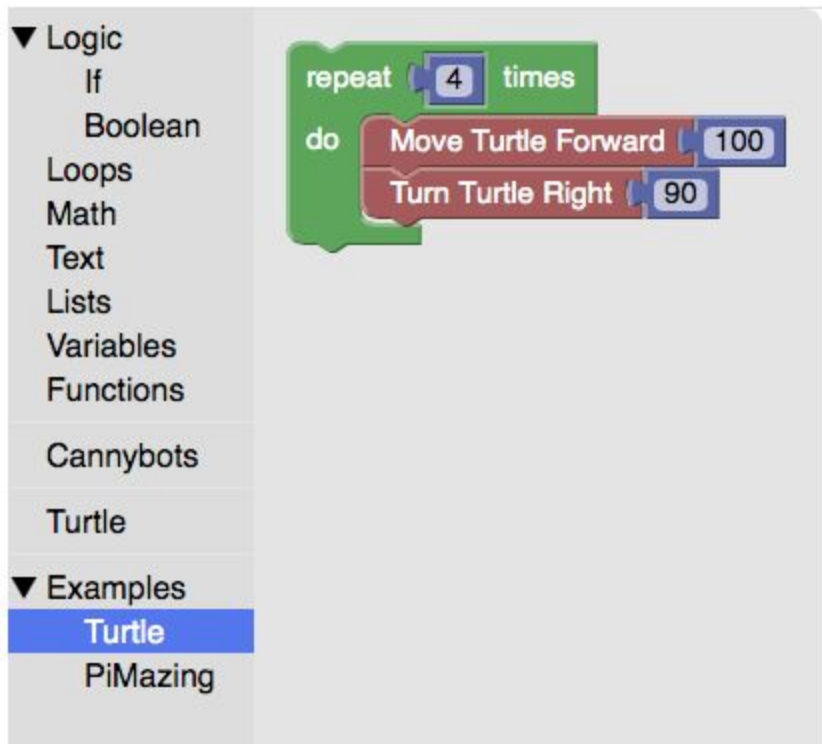
The Cannybots Turtle library provides basic Turtle commands for moving specified distances and angles.



## Example Code

### Turtle

The code below will cause the Cannybots Turtle to trace out a square.

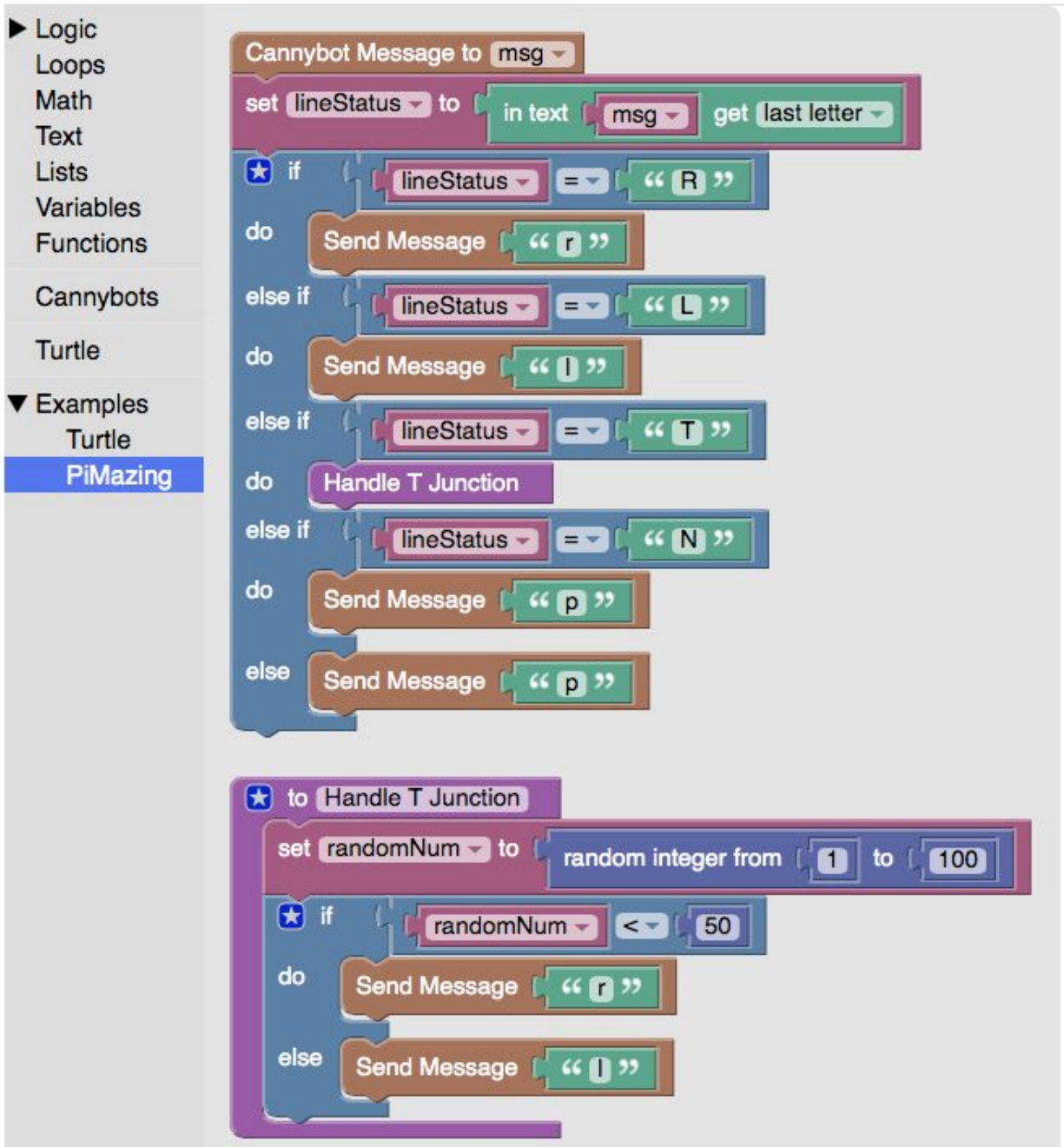


The image shows a Scratch code editor interface. On the left is a sidebar with a menu. The menu items are: Logic (expanded), If, Boolean, Loops, Math, Text, Lists, Variables, Functions, Cannybots, Turtle, Examples (expanded), Turtle (highlighted), and PiMazing. The main workspace contains a Scratch script with the following code:

```
repeat 4 times
do
  Move Turtle Forward 100
  Turn Turtle Right 90
```

## PiMazing

The following



The image shows a Scratch script for a Cannybot. The script is organized into two main sections. The first section, titled "Cannybot Message to msg", processes incoming messages. It starts with a "set lineStatus to" block that takes the last letter of the message. This is followed by a series of "if" blocks that check the value of "lineStatus". If it is "R", it sends the message "r". If it is "L", it sends "l". If it is "T", it calls a "Handle T Junction" function. If it is "N", it sends "p". If none of these conditions are met, it also sends "p". The second section, titled "to Handle T Junction", generates a random integer between 1 and 100. If this number is less than 50, it sends "r"; otherwise, it sends "l".

```

Cannybot Message to msg
set lineStatus to in text msg get last letter
if lineStatus = " R "
do Send Message " r "
else if lineStatus = " L "
do Send Message " l "
else if lineStatus = " T "
do Handle T Junction
else if lineStatus = " N "
do Send Message " p "
else Send Message " p "

to Handle T Junction
set randomNum to random integer from 1 to 100
if randomNum < 50
do Send Message " r "
else Send Message " l "

```

## Further Examples

More examples and apps can be found on the Google Blockly Site [here](#) and [here](#)

# Code Generation

Cannyblocks (Blockly) can generate underlying code for JavaScript and Python that represents the Blocks. The code generated is one-way (i.e. you can't create Blocks from Python code) and is standalone in the sense that it's intended to be used outside the Cannyblocks app. (More on that in another document)

## JavaScript

Add screenshots

## Python

Add screenshots