Lab 01

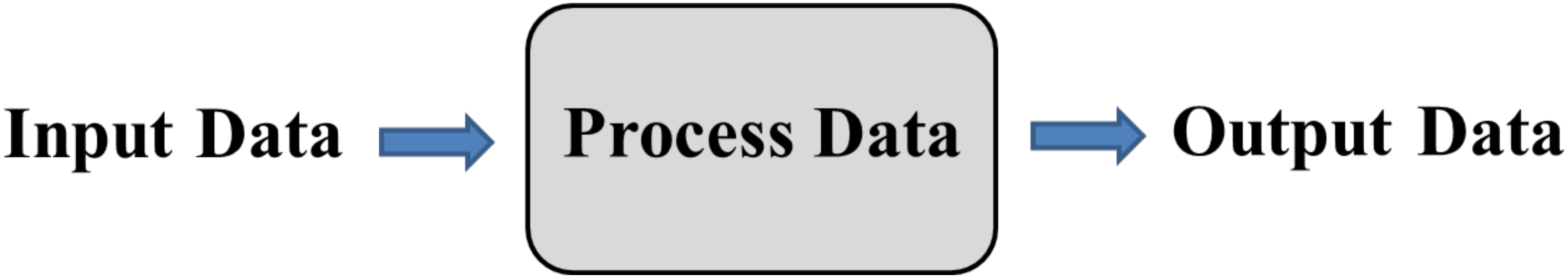
Introduction to Problem Solving & Flow of Control

Basic Programming via code.org

Problem Solving & Flow of Control

Computer Operation: consists of three main parts.

- Input
- Process
- Output



Problem Solving & Flow of Control

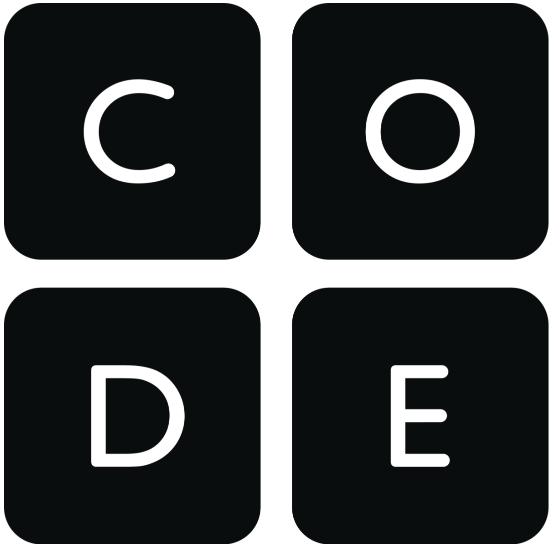
- **Processing Data:**
  - Accepts and gather data. (**INPUT**)
  - **Processes** data using problem solving methodology.
  - Achieve and present required result(s). (**OUTPUT**)
- **Flow of Control:** sequences/steps of program instructions used to solve a problem.
  - **Sequential**
  - **Selection or Conditional**
  - **Repetition or Loop**
  - **Invocation or Calling Function**

Problem Solving & Flow of Control

Flow of Control

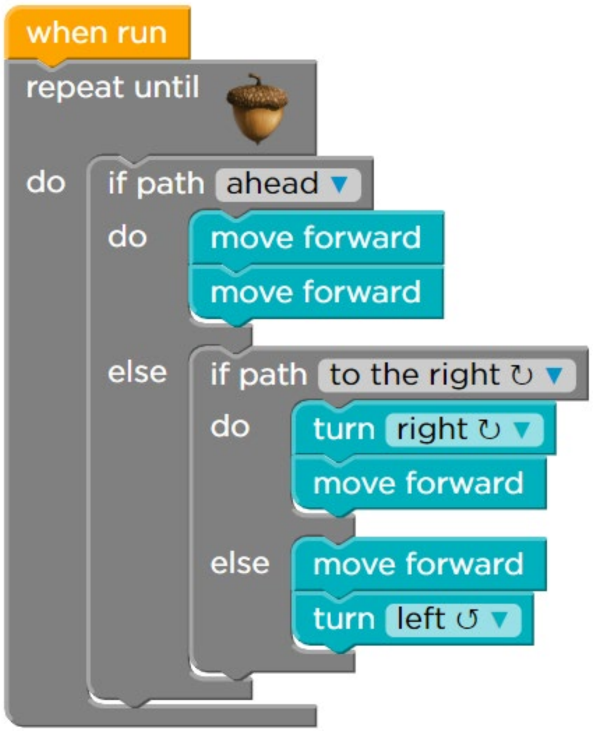
- **Sequential:** Normal flow of control for all programs.
- **Selection:** is used to select which statements are performed next based on a condition.
- **Repetition:** is used to repeat a set of statements.
- **Invocation:** is used to invoke a sequence of instructions using a single statement, as in calling a function

code.org



Introduction

- Today, we will work with one of easier approach to create a program – Block Programming
- Instead of typing up codes, you add instructions to the program by dragging and dropping blocks into the program



```
when run
repeat until
do  if path ahead ▾
    do    move forward
          move forward
    else  if path to the right ↻ ▾
          do    turn right ↻ ▾
                move forward
          else  move forward
                turn left ↺ ▾
```

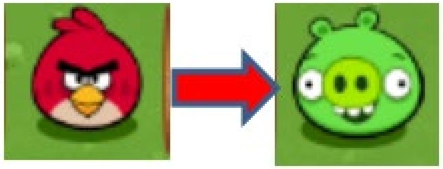
- In this lab we will work on a game called “**Classic Maze**” from **Hour of Code**.

- The main goal is to code different characters through the maze using available instructions.

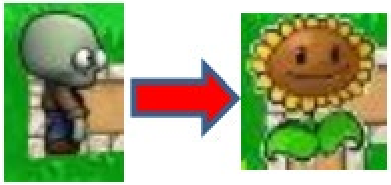
Objectives

- The goals of Classic Maze are:

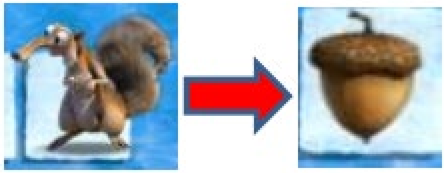
1. Lead Angry Bird through the maze to the Green Pig, using available commands



2. From level 12, you now will need to lead Zombie to the Sunflower

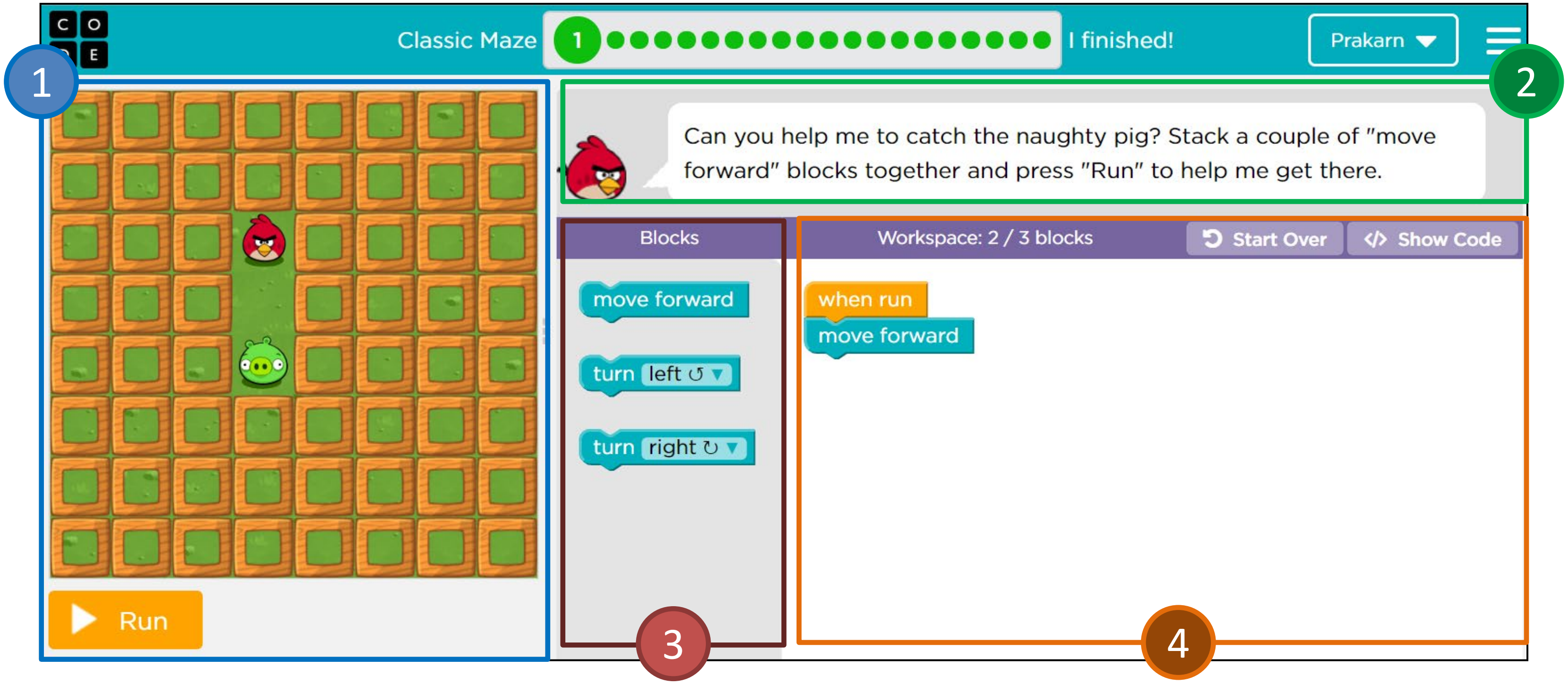


3. From level 16, you will instead need to guide Scrat to the Acorn



Note: Pay attention to the direction your character is facing!

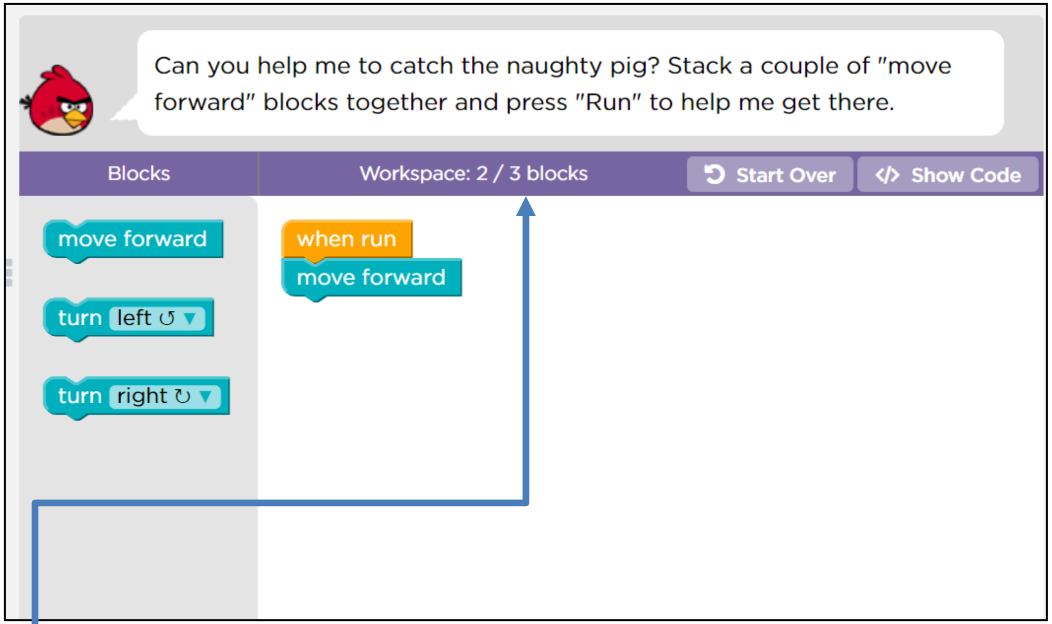
What's on the Screen?



Can you help me to catch the naughty pig? Stack a couple of "move forward" blocks together and press "Run" to help me get there.

1. The Simulation (The result of your code)
2. The Instruction and objective
3. Available Blocks (Operations you can do)
4. Workspace (where you use operation)

Right Portion – Objectives and Your Program

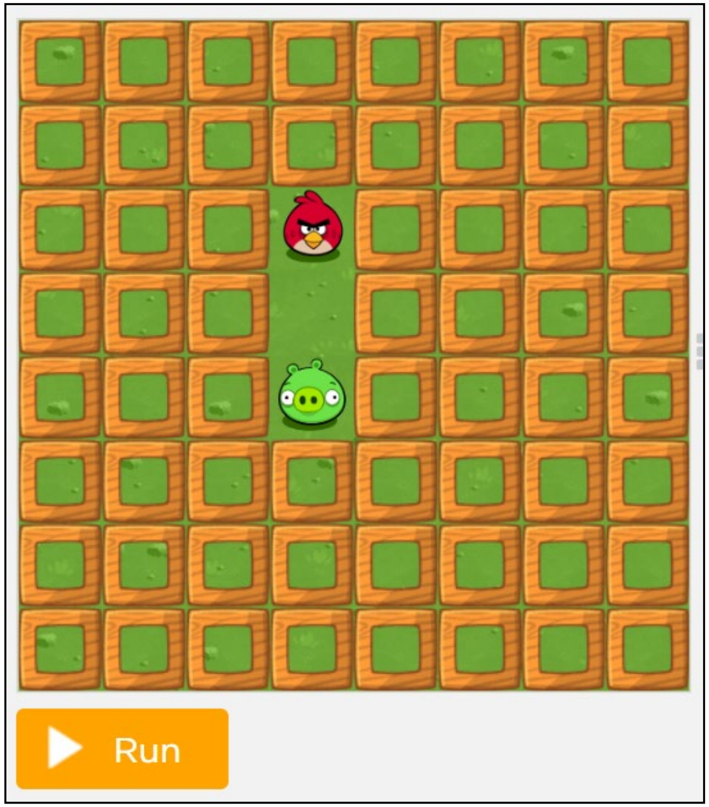


Can you help me to catch the naughty pig? Stack a couple of "move forward" blocks together and press "Run" to help me get there.

- To create a program, drag a block from the available blocks and add them to your program in the workspace.

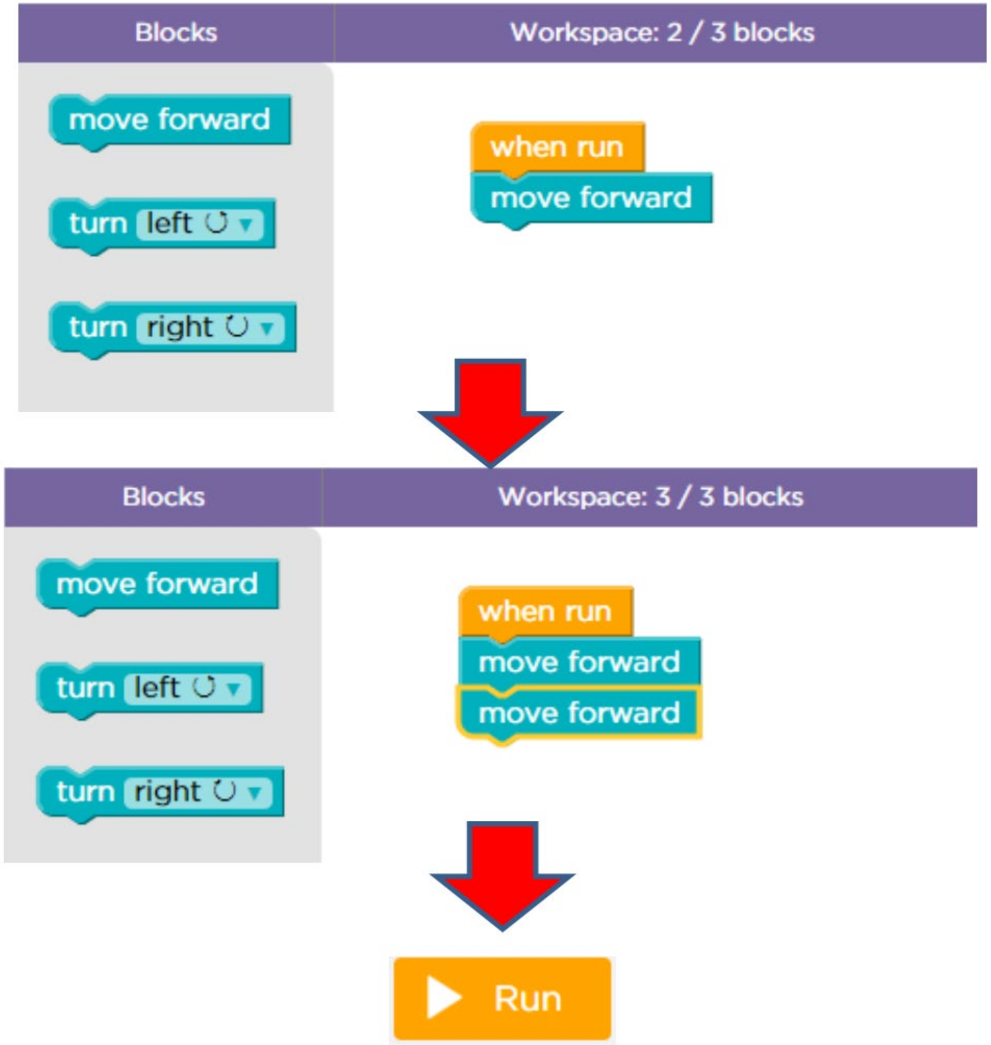
- When the program is run, it will start from “when run” block, the move down to the block below it (**sequential manner – from the top to the bottom**)
- You can use as many blocks of any type as you want, but the workspace will show the minimum number of blocks that can be use to finish the level
- “Start Over” button will reset your program
- “Show Code” will show the code of your program

Left Portion – The Simulation



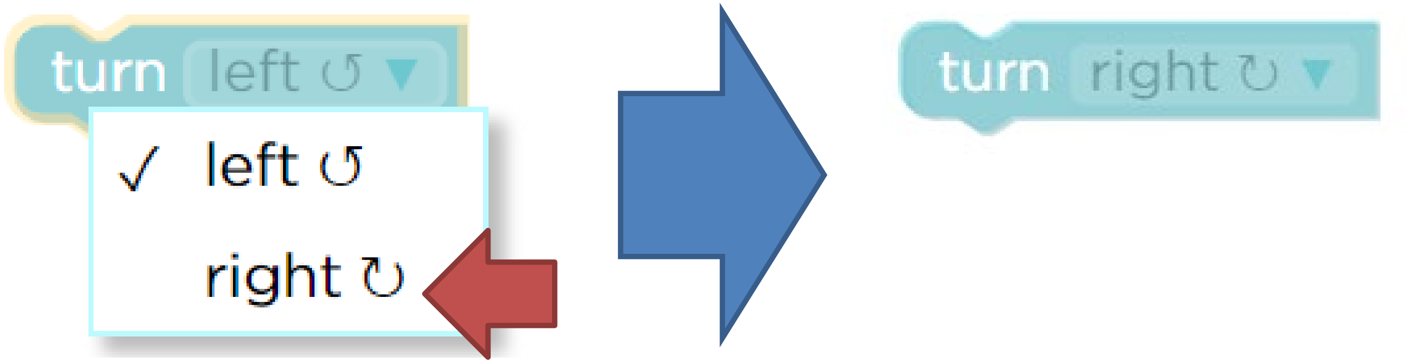
- Once your program is ready, click “Run” will run the program, showing the result on the maze
- While the program is running, the block currently running will be highlighted

Example – First level



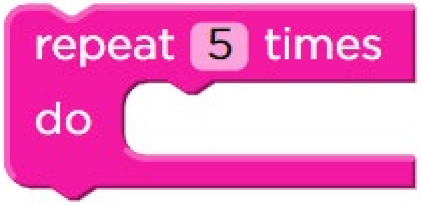
Changing the Argument

- Some blocks, like turn, has a value you can change (left and right for turn). This is call **argument**, which will make the block behave differently



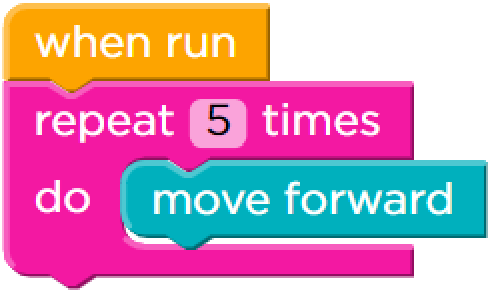
Loops

- Starting at Level 6, you will get a new block
  - “**repeat ... times**”



```
repeat 5 times
do
```

- “**repeat ... times**” block will repeat everything inside it for the number of times chosen as its argument.
- For example, the repeat block below will repeat “move forward” 5 times

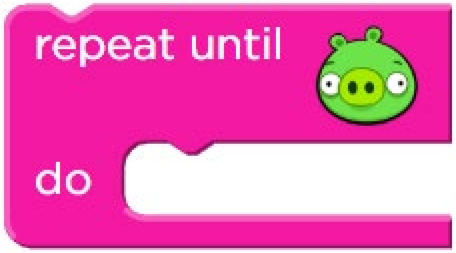


```
when run
repeat 5 times
do  move forward
```

You can put more than one block in here!

Loops: Repeat Until

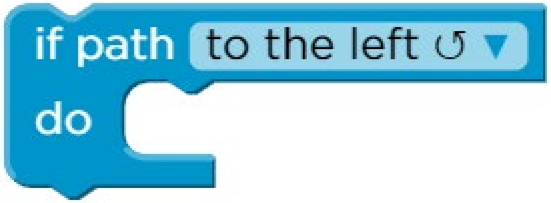
- At level 10, you will get to use “**repeat until**”



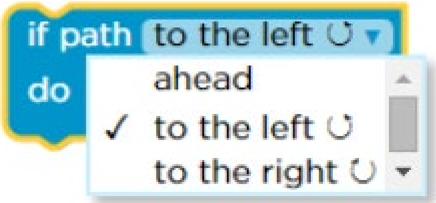
- “**repeat until**” will execute the code inside until the condition is met.
- In this case the block will repeat the code (can be more than one block) inside it until the pig is found.

If: Conditional

- At level 14, you will get an "**if**" block

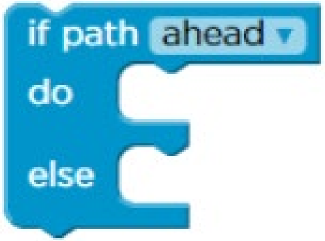


- "if" block will only execute the code inside it if the condition in the argument is true
  - It reads "**If there is a path to the left, do…**"
- As mention, if the condition is not what you want, you can change the condition by changing the argument



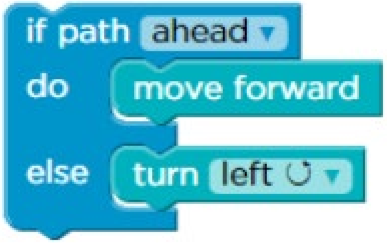
If-Else: Choosing between Two Choices

- From level 18, you will get the last block, “**if-else**”



```
if path ahead ▾
do
else
```

- Similar to “**if**” block mentioned before, with addition “**else**” part.
- If the condition is true, the “**do**” part will be executed, if not, the “**else**” part will be executed

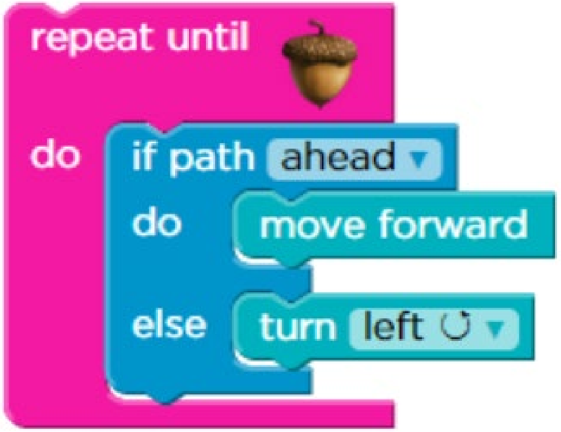


```
if path ahead ▾
do    move forward
else  turn left ↺ ▾
```

- For example above, the there is a path ahead, the character will move forward. Otherwise, the character will turn left.

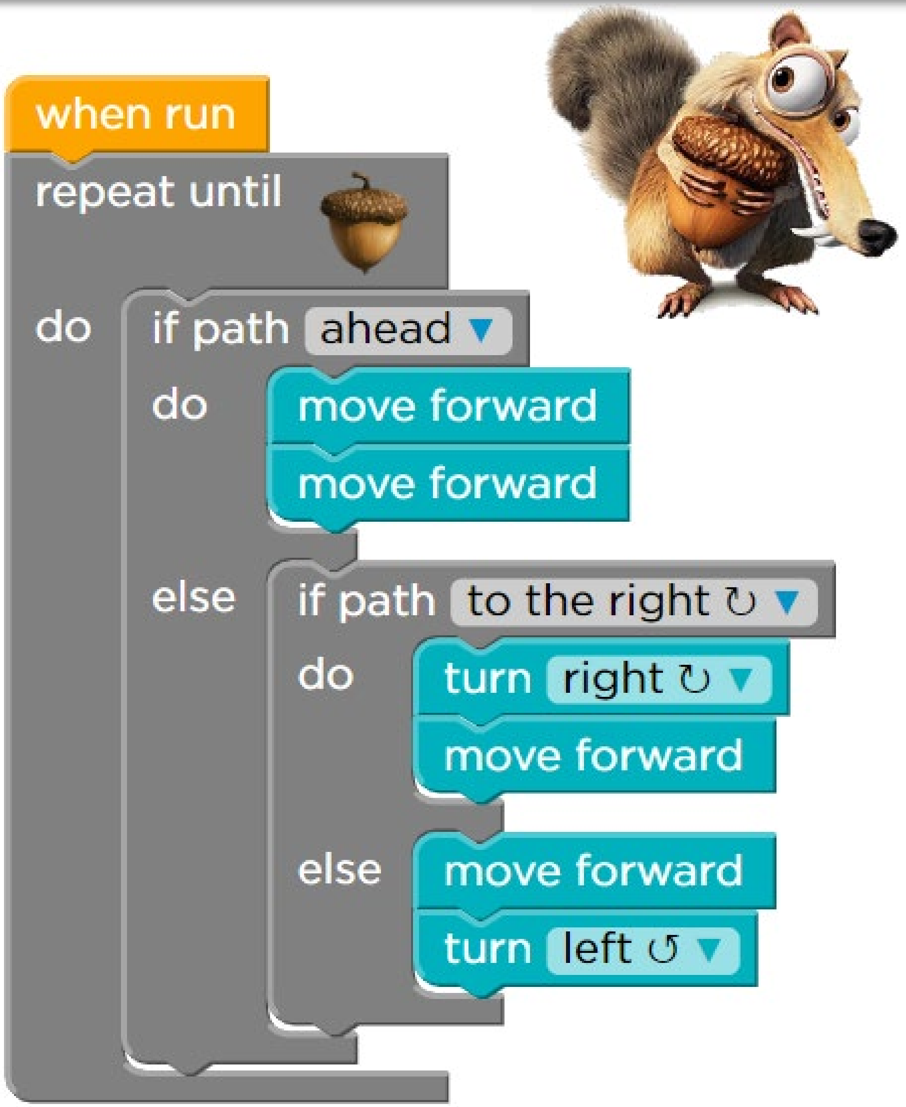
Nested Blocks

- Not only can you use as many block as you'd like, you can also put any inside another, creating more complex program



```
repeat until
do  if path ahead ▾
    do   move forward
    else turn left ↺ ▾
```

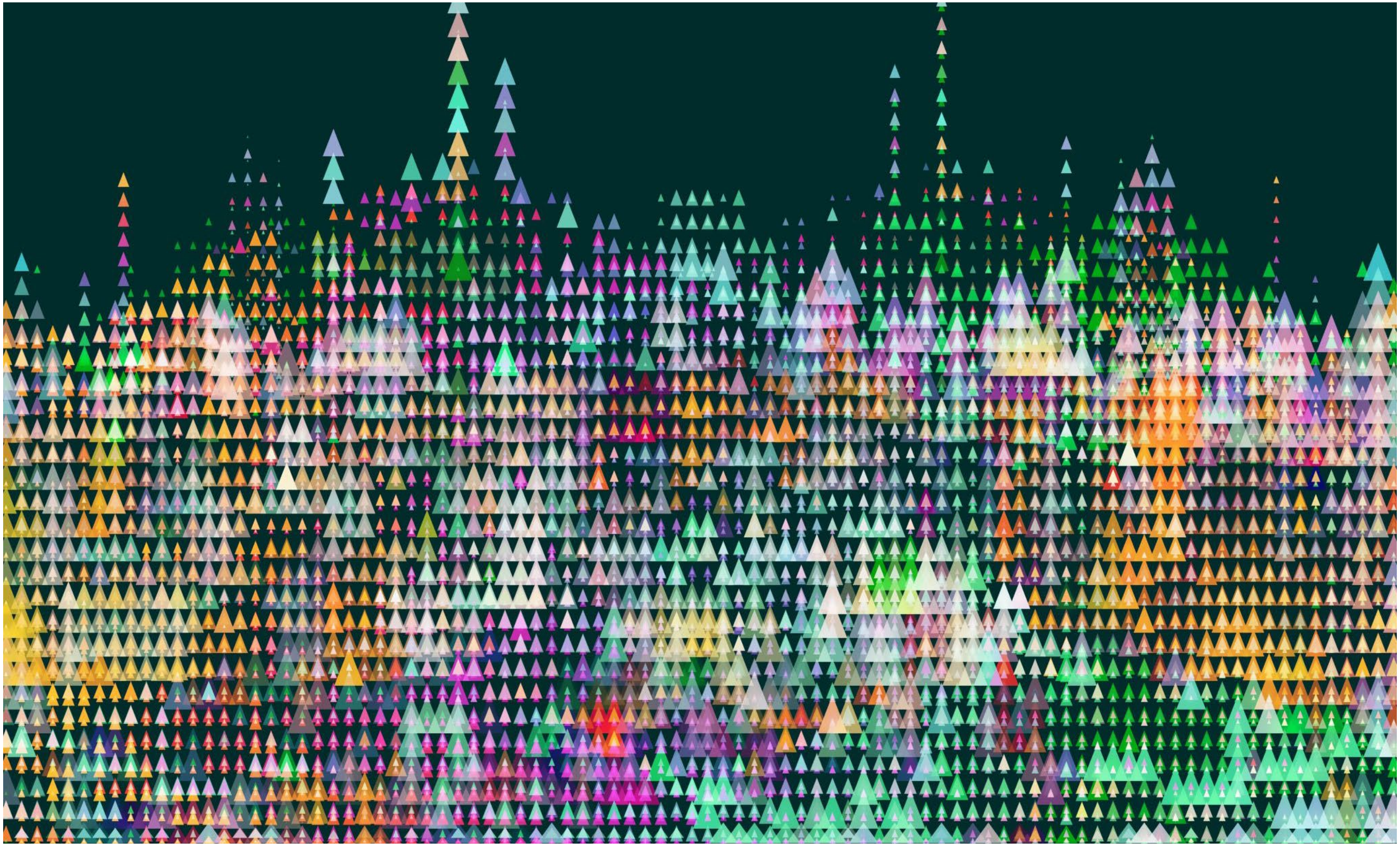
Let’s Analyze This



```
when run
repeat until 🌰
do  if path ahead ▾
    do    move forward
          move forward
    else  if path to the right ↻ ▾
          do    turn right ↻ ▾
                move forward
          else  move forward
                turn left ↺ ▾
```

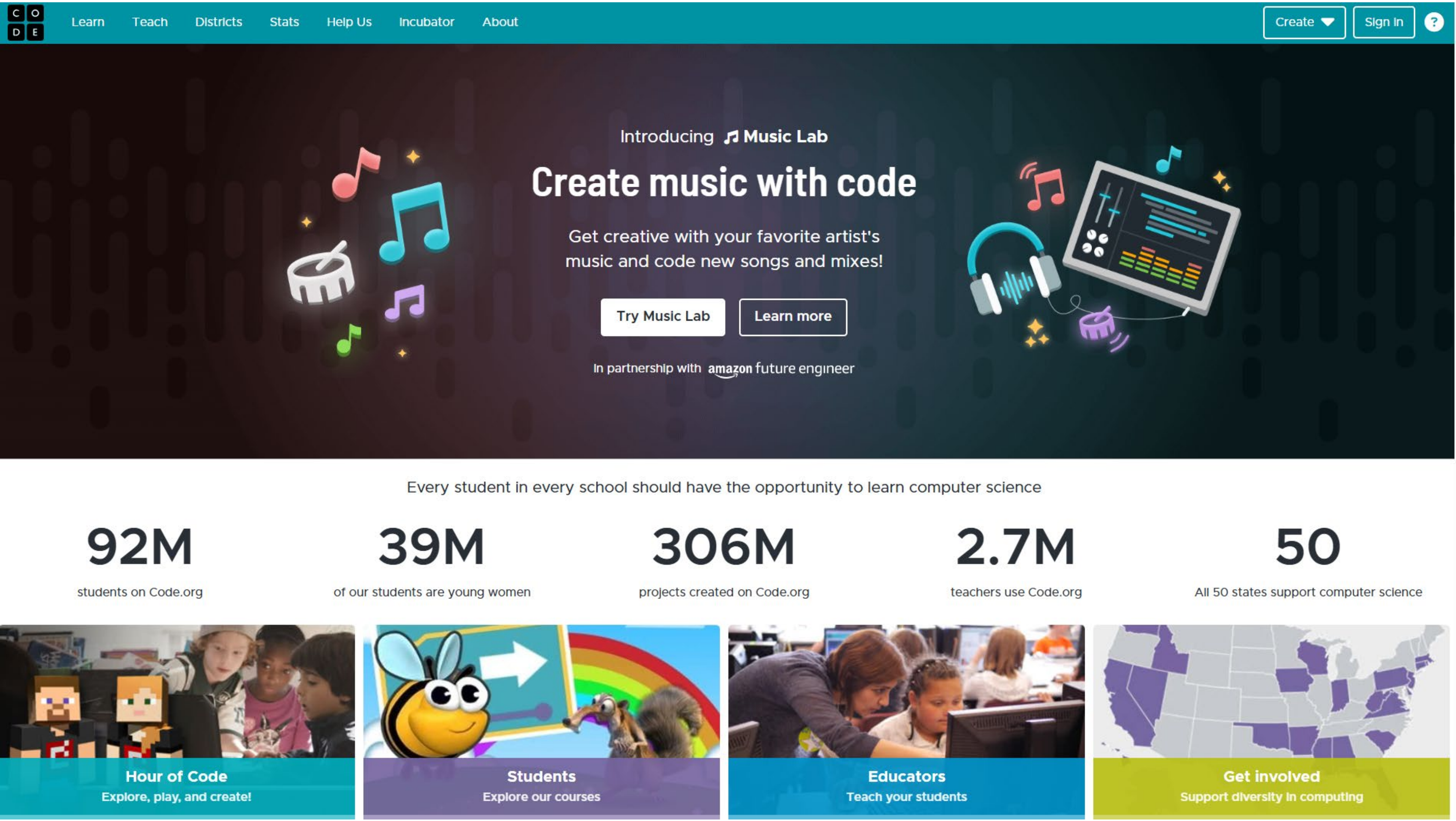
- Let’s take a moment to analyze this:
- What is this supposed to do?
- The logic isn’t entirely correct, can you tell me what is wrong with it?

Let Lab 1 begin



Accessing code.org

- From your web browser, go to https://code.org



Introducing ♫ Music Lab

Create music with code

Get creative with your favorite artist's music and code new songs and mixes!

In partnership with amazon future engineer

Every student in every school should have the opportunity to learn computer science

92M
students on Code.org

39M
of our students are young women

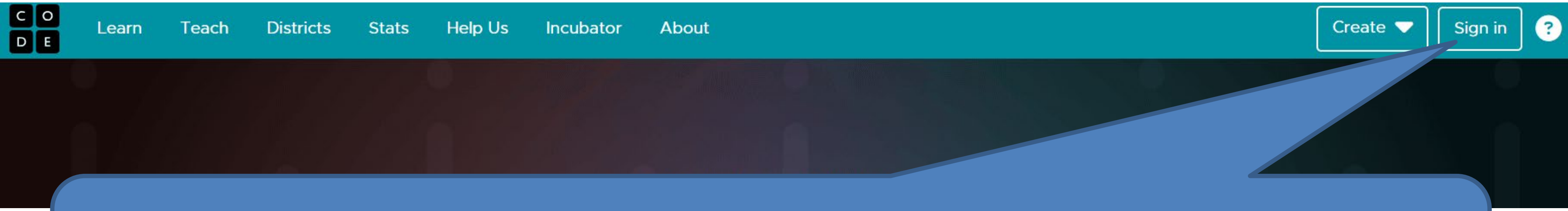
306M
projects created on Code.org

2.7M
teachers use Code.org

50
All 50 states support computer science

Signing In

- If you are signed in on the code.org, your progress will be saved



- To sign in, select “Sign in” at the top-right corner of the page

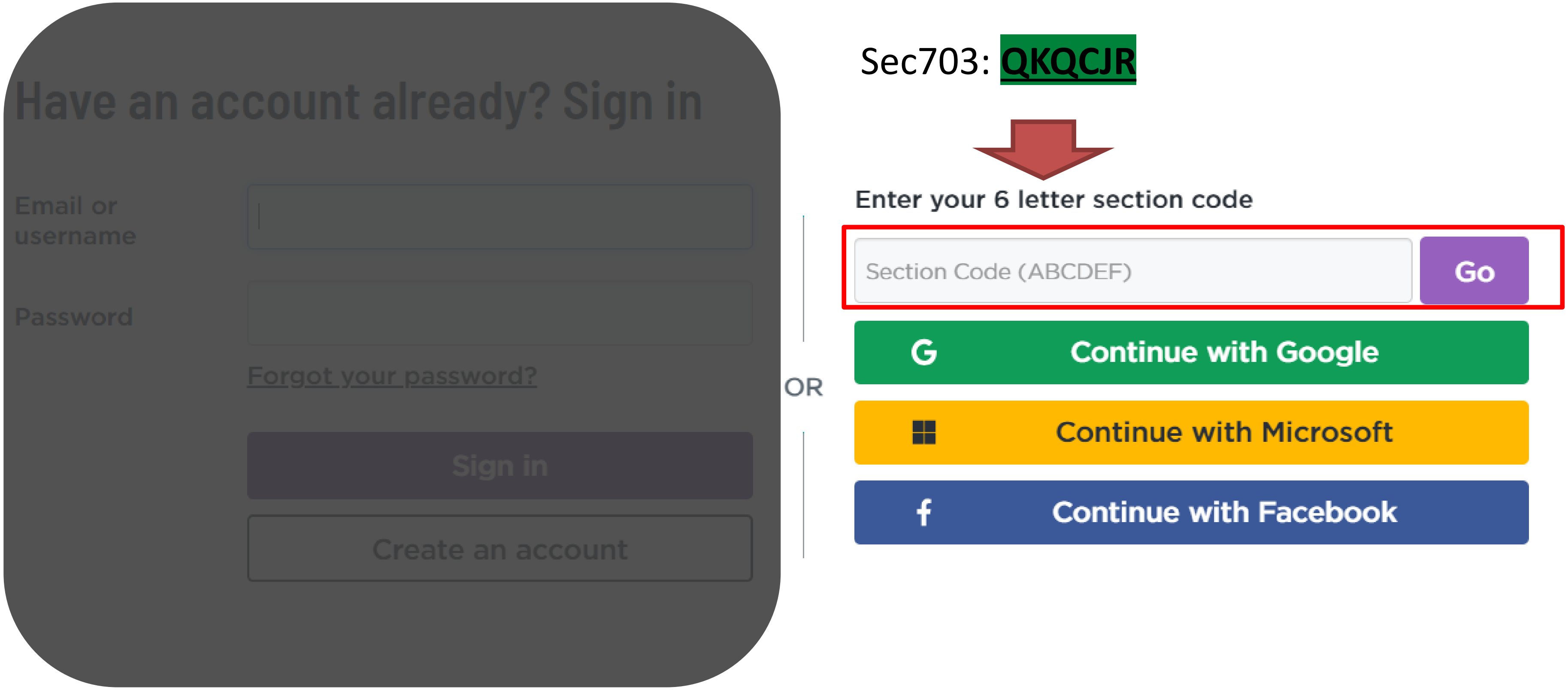
Signing In for the first time

input section code FIRST

Sec701: **VSVJFF**

Sec702: **HRZKML**

Sec703: **QKQCJR**



Have an account already? Sign in

Email or username

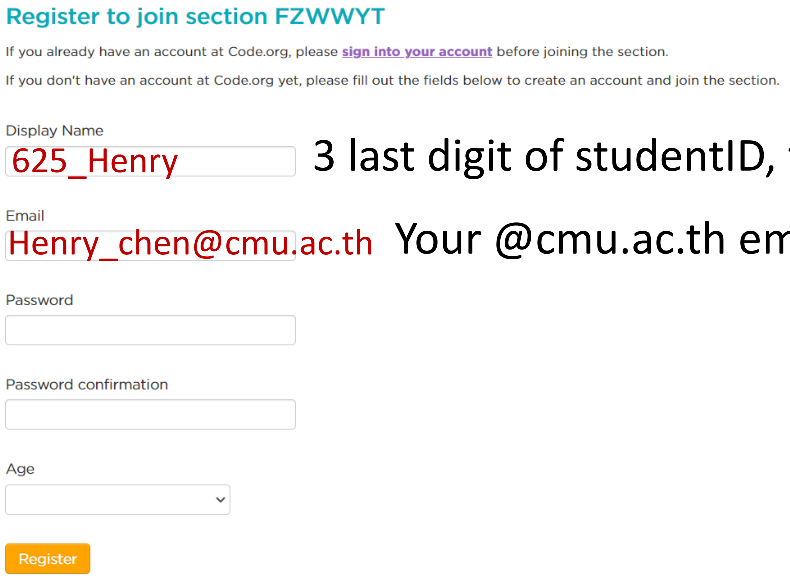
Password

Enter your 6 letter section code

Section Code (ABCDEF)

Creating Account

- For the first time, you will then be asked to fill in the details.



Register to join section FZWWYT

If you already have an account at Code.org, please **sign into your account** before joining the section.

If you don't have an account at Code.org yet, please fill out the fields below to create an account and join the section.

Display Name

625_Henry

3 last digit of studentID, followed by underscore and your first name

Email

Henry_chen@cmu.ac.th

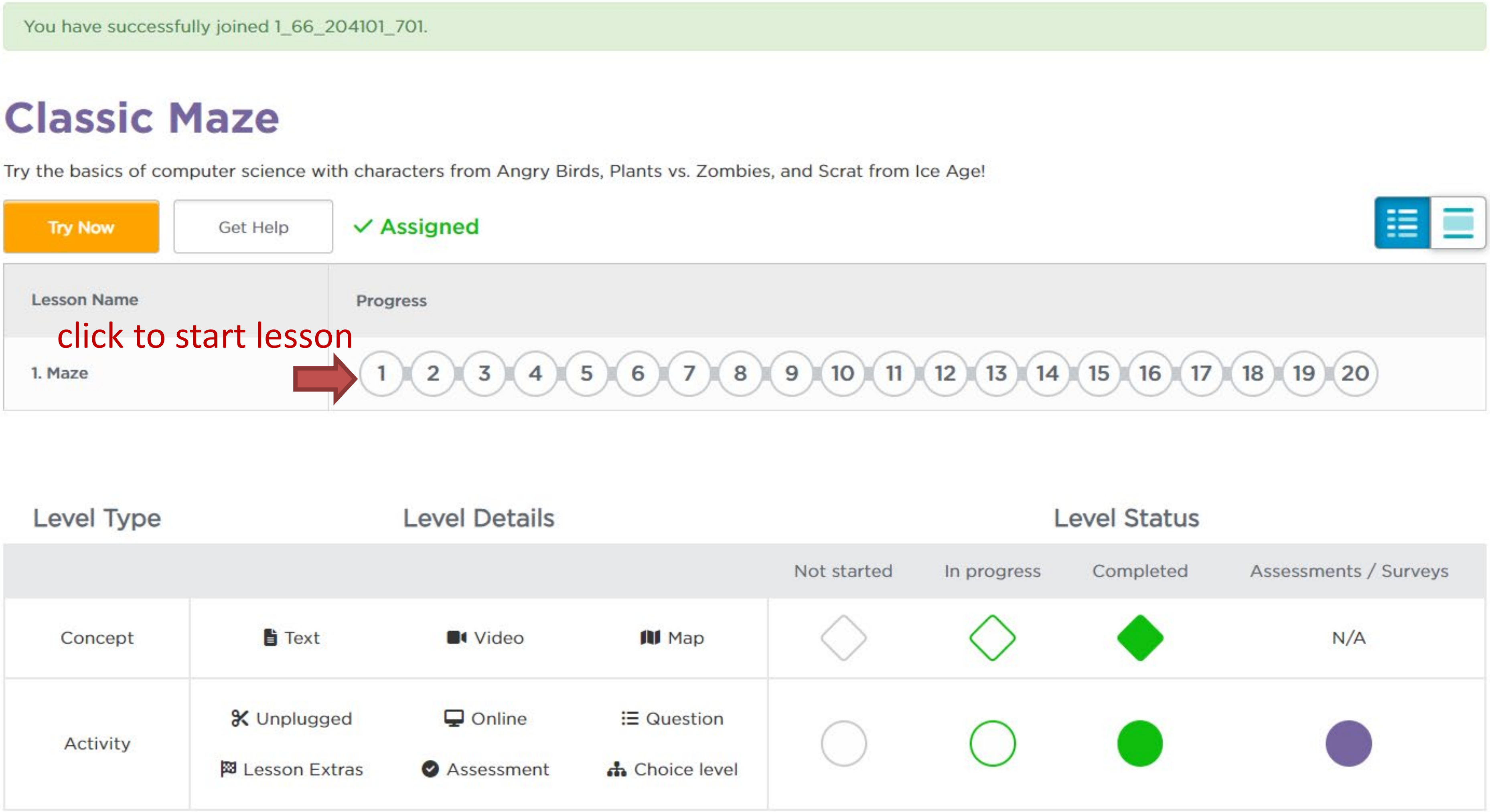
Your @cmu.ac.th email

Password

Password confirmation

Age

Already join the class



You have successfully joined 1_66_204101_701.

Classic Maze

Try the basics of computer science with characters from Angry Birds, Plants vs. Zombies, and Scrat from Ice Age!

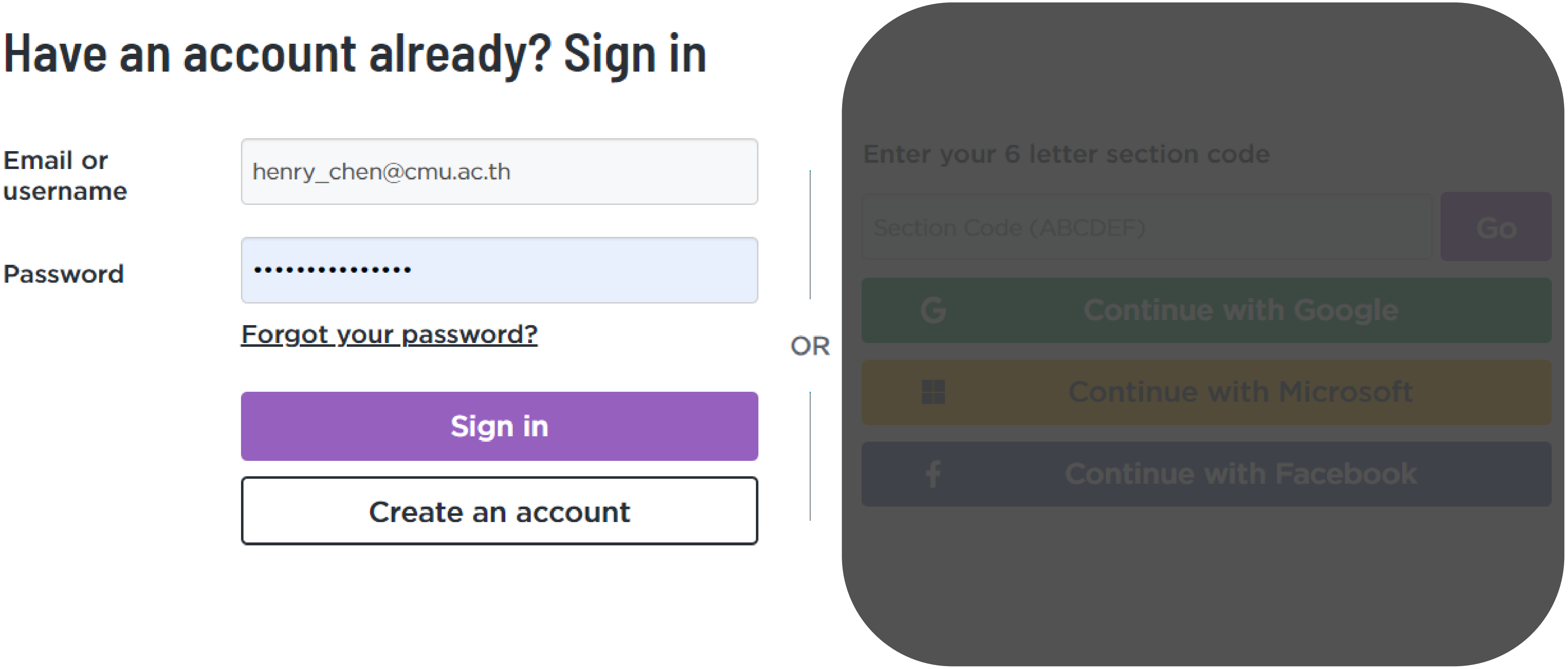
Lesson Name	Progress
1. Maze	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

click to start lesson

Level Type	Level Details			Level Status			
				Not started	In progress	Completed	Assessments / Surveys
Concept	Text	Video	Map	◇	◇	◆	N/A
Activity	Unplugged Lesson Extras	Online Assessment	Question Choice level	○	○	●	●

Login in another time…

- Remember username and password to login next time to keep your progress



Have an account already? Sign in

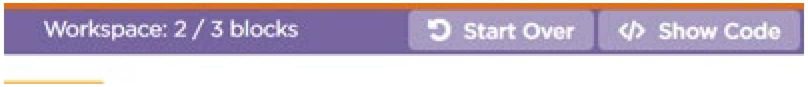
Email or username: henry_chen@cmu.ac.th

Password: •••••••••••••••

Enter your 6 letter section code

Lab01

- Finish HOC Classic Maze following level **in class***:
  - Level 1-5 (Sequential Programing)
  - Level 6 (Repeat with the number of times)
  - Level 10 (Repeat with until condition)
  - Level 14 (Selection with condition)
  - Level 18 (Selection with condition and alternative path)
- Study the code with Show Code button



- Challenge yourself by make sure to use blocks as allowed

Homework 1

1. Finish all 20 Levels on HOC
2. Upload Certificate from code.org to Mango assignment

- Deadline*: For all sections
  - 701, 702 and 703: July 15, 2025 (Tuesday)

*Your homework assignment will be graded by midnight on the due date
We don't accept late submissions.