



CREATE A SIMULATION BIRTHDAY MATCH

with



lynxcoding.club

With funding from



CODE to LEARN.ca



DESCRIPTION

There are usually 365 days in a year. How many students must you have in a classroom in order to be almost certain that two students have the same birthday? 365? Half of 365? Less?

This project lets you find out by experimenting with statistics.



Description

Coding a Simulation

Students (grades 5-8) will create a computer simulation to test the probability of an event, in this case of two people in a class having the same birthday.

This activity can be modified to demonstrate the probability of other events, such as the chance of rolling the same number on a die or on multiple dice.

Students will code this app using Lynx at **lynxcoding.club**.

Students will code the computer to:

- Stamp a shape for each person's birthday in a randomly determined location.
- Create variables using sliders, and text boxes.
- Create conditional statements to test conditions and control action.

Prerequisite Skills - Students should know how to:

Define super- and subprocedures

Add buttons, text boxes, sliders, and turtles

Use basic turtle graphics commands

Success Criteria

Co-construct success criteria with your students but the mathematics of the simulation should work correctly.



LEARNING GOALS

Students will learn, and use, these...

BIG IDEAS IN CODING			
	MAIN IDEAS		
CODE & CONCEPTS	<i>colourunder</i> Event handling and colour detection	<i>if</i> Create conditional statements	<i>settextboxname, setslidername</i> Change values with commands
	<i>random</i> Use a random number generator.	<i>textboxname, slidername</i> Use text box and slider names in commands	<i>stop, stopall</i> Stop single procedures or all action under program control



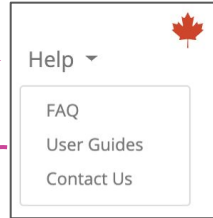
GETTING STARTED

Get a LYNX Account and Understand the Layout



Get a Lynx Account

Details at lynxcoding.club



We suggest:

- teacher gets a **School Administrator Account**
- students get **permanent Individual Accounts**
- teacher creates a '**club**' and invites all students

NO Account

You can try Lynx for free without an account, by clicking on **Create a Lynx Project** on the home page at lynxcoding.club.

FREE TRIAL Account

For full access, register (click **Login/Register** located at the top, right side of the Lynx web page).

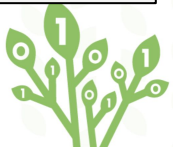
INDIVIDUAL Account

Convert your trial account to a permanent individual account before end of trial period.

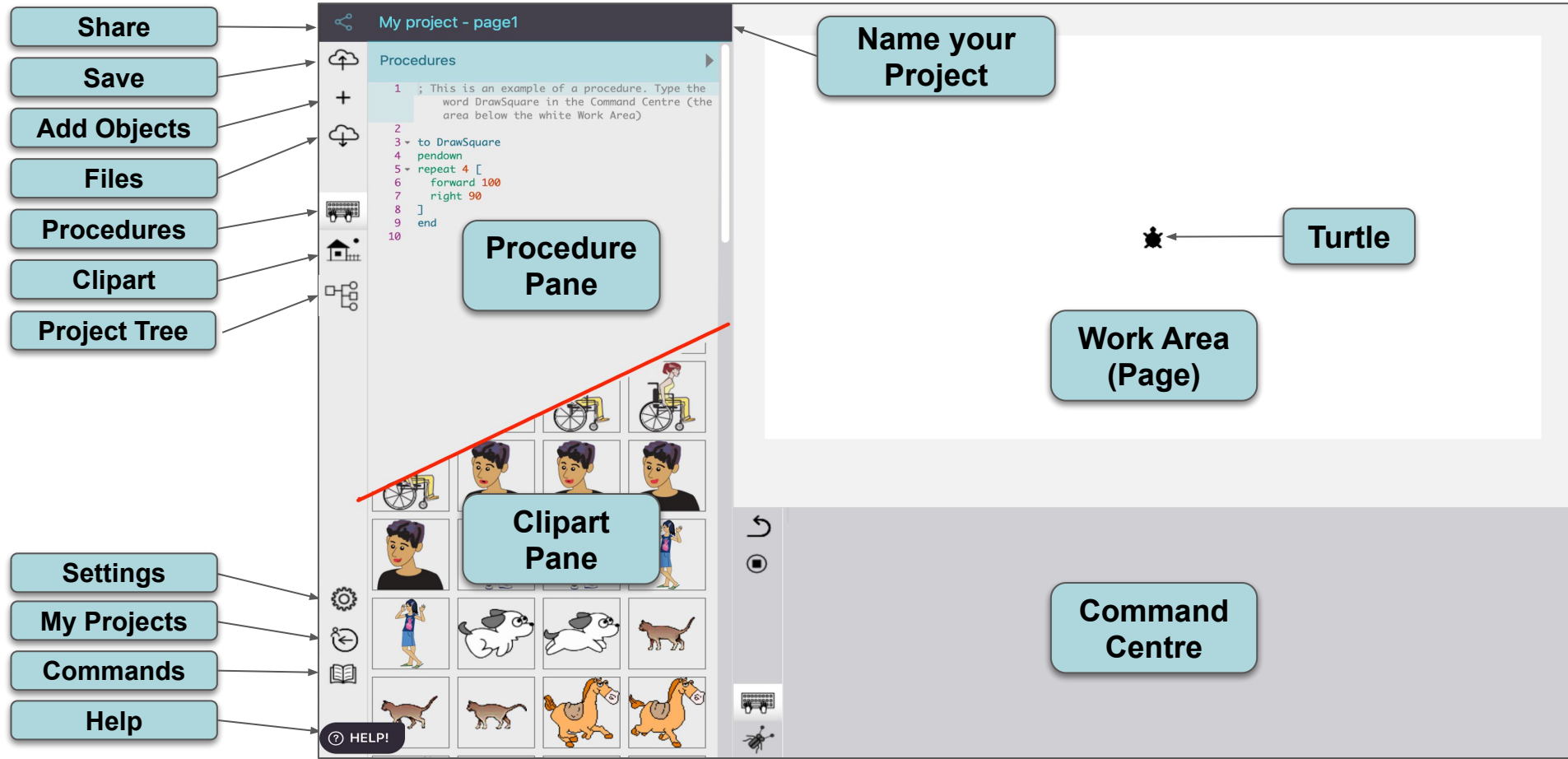
SCHOOL ADMINISTRATOR Account

Convert your trial account to a School Administrator account before end of trial period.

Accounts are free for Canadians thanks to a subsidy by the Government of Canada.



Layout



The BIG Picture! (*Table of Contents*)

1. **Log in** and create a new project.

2. Add a *turtle* and text box.

3. Pick a *random* number.

4. Use **colourunder** to check dates.

5. Write a **setup** procedure.

6. Add a **students slider** to run the experiment.

7. Add *instructions* for users.

8. Add features.

9. Bonus - Average (optional)

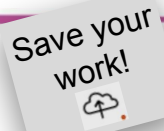


LOGIN AND GET STARTED

Log in, **CREATE** a NEW PROJECT,
NAME it and **SAVE** it.



Get Started



The turtle in this project is used to find "date" matches.

- The first time it moves, it moves forward a random number from 1 to 365 and makes a small red dot, then returns to its starting point.
- The next time it moves, it first checks if the colour under it is red (meaning a red dot was already there). If yes, the procedure stops and "Match" is printed in a text box. If the colour under isn't red, it leaves a tiny dot and returns to the starting point.

This process is repeated until either a match is found or there are no more students to check.

1. **Log in.**

2. Click on **Create a Lynx Project.**

3. **Name** it something personal
e.g., *Talia's Birthday Match*



4. **Save**
Click this icon



ADD A TURTLE AND TEXT BOX

Add Essential Objects



Add Essential Objects - a Turtle and a Text Box

1. If there is no turtle in the Work area, click on the **+** sign and select **Turtle**.

2. Make sure the turtle's pen is up. In the Command Centre, type:

pu

Press Enter.

3. Set the turtle's colour to red. Type:

setc 15

Press Enter.

Each colour has a number. Red's number is 15. There is a Lynx Colour Chart in the User Guides section in the Help section at lynxcoding.club

4. Click on the **+** sign and select **Text**. Move the text box to the side of your Work Area. The Text Box will be used to keep track of the matches.

5. Change the name of the text box to **Matches**.

To change the text box's name, right-click on the text box and type **Matches** in the top of the dialog box. Click **Apply**.

A text box name must be one word!



TEST A RANDOM NUMBER

Explore How to Use Random



Save your work!

Random picks a number from 0 to one less than its input, so 0 to 9 in this example.

- ## show random 10

Try this several times. You may not always get a different number, but you won't be able to predict what number you'll get each time.

- forward 1 + random 365*

Random 365 represents a random day in the year, and the turtle moves forward that number of days.

If you repeat this action many times, does the turtle always move the same amount?

Adding a 1 means you won't get a 0 and the turtle moves forward from 1 to 365 pixels. The 1 is added before the random command.



CREATE EVENTS WITH COLOUR

Use Colourunder to trigger an action



Check the Dates with Colourunder

To see where the turtle moves have the turtle leave a red dot each time it moves and then return to its original position.

1. *Move the turtle to the centre of the screen. Type:*

`home`

2. *Then type:*

`repeat 10 [pu fd 1 + random 365 pd fd 0 pu`

`home]`

`clean`

The sequence “`pd fd 0 pu`” leaves a tiny dot on the page, then the turtle returns to the centre of the work area. It is a bit “magical”, but yes, the turtle leaves a dot when it runs `fd 0`.

Do you get different patterns of dots each time you run the instruction?



A Colour Checking Procedure

Save your work!

This procedure checks if the colour under the turtle is colour number 15 (red).

The next time and all the other times the turtle moves, it first needs to check if there is a colour under it. To do this, use **colourunder**. **Colourunder** checks what the colour under the turtle is.

1. Write in the Procedures Pane:

to OneStudentOneDot

pu setpos [-375 -90] ;turtle goes to this X Y coordinate

fd 1 + random 365

if colourunder = 15 [matches, pr 'MATCH!' stopall]

pd fd 0 pu ;this leaves a dot

end

Using **stopall** once there's a match stops the all the action.

Remember: You must use a colour **number** with **colourunder**.

If **colourunder = 15**, it means that a random birthday is the exact same date as another random birthday, therefore we have a "match".

2. Type **OneStudentOneDot** in the Command Centre.

Try it several times, until you get a "match"!



SET UP

Set Up Your Experiment



Get the turtle into position



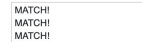
- Your numbers may be different*

- ## setheading 90

Setheading or **seth** sets the turtle to a specified direction in degrees.

- ```
setc 15
```

- clean**



### Matches



# Write a Setup Procedure

Save your  
work!

Put all the instructions from the previous cards together!

1. Put these instructions altogether in a **Setup** procedure. In the Procedures Pane, type:  
**to setup**  
**clean**  
**t1, pu**  
**setpos [-375 -90]**  
**setheading 90**  
**setcolour 15**  
**end**

**Clean** will clear the graphics without moving the turtle.

Use your turtle's name followed by a **comma**. This ensures the turtle on this page follows your instructions. This is useful in case you decide to extend your experiment with one or more turtles later.



# ADD A SLIDER


---

Set the number of students



# Add a Slider

## Set how many students are in a class

Save your work!  


1. How many times should you run *OneStudentOneDot*? Once for each student in the class until a birthday match comes up.

2. Click the **+** sign and select *Slider*  
A slider appears.

3. Right-click on the slider and name it *Students*.

4. Set the **minimum** to 1, the **maximum** to 50, and the current value to the number of students in your class.  
Click *Apply*.

5. Drag the slider to where you want it in the work area. Right-click on it again and select *Frozen*.  
Click *Apply*.

|       |          |
|-------|----------|
| Name  | Students |
| Min   | 1        |
| Max   | 50       |
| Value | 30       |



# Using the Slider

Repeat the **OneStudentOneDot** procedure for each student in the class

1. Set your slider to the size of your class.

2. In the Command Centre, type:

```
setup
repeat students [OneStudentOneDot]
```

**Students** reports the number on the slider.

3. Write a procedure to run your experiment.

```
to OneClass
 setup
 matches, cleartext
 repeat students [OneStudentOneDot]
end
```

**Matches** followed by a comma tells the matches text box to follow the next text direction: **cleartext** or **ct**.

4. Test your procedure!

# ADD USER INFORMATION

---


**Add a button and instructions**





# Add a Button and Text Boxes

*Help your users run the simulation.*

Save your  
work!  


**1. Click on the + sign and select *Button*.**

A button named “Nothing” appears.

**2. Right-click on the button and give it a clear label, such as *Start the Experiment*.**

A button’s label does not have to be one word. It’s just a label in plain English, not a command.

**3. Click on the arrow in the box next to *On click* and select *oneclass*.**

Drag the corner of the button to show the whole label.

Drag the button to where you want it.

Once in place, right-click on the button, select *Frozen*, and click *Apply*.

Test the button several times. How many students do you need (slider value) to get frequent matches?



# Add Instructions

## Add one or more information text boxes

1. Click on the **+** sign and select **Text**.

2. Add instructions for the user, for example:  
**Set the number of students in the class**

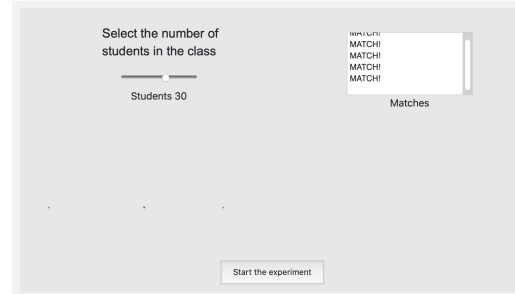
3. Click inside the text box and use the formatting tools to write an explanation.

4. Move the text box to where you want it.

5. Right-click on the text box.

6. Check **Transparent** and **Frozen**.  
Click Apply.

7. Repeat for any additional information text boxes you add.



# Share your Project

Send it to friends or publish it!

1. Click the **Share** icon.



2. Click **Create**.  
A URL will be generated here.

3. Click on **Copy Link**  
and send it to friends.

Share this project...

Sharing Options **Project Properties**

Link Sharing is OFF. [Create](#) a link to share.

URL

To create a link to share, click Create

[Copy link](#) [E-mail](#) [Twitter](#) [Facebook](#)

Embed on your site

To see the code, click Create

[Copy](#)

Share this project... x

Sharing Options **Project Properties**

Preview Image

Select...

Title Talia's Birthday Match

Desc.

☒ Private only author and teacher can edit and download

4. Add a Preview Image to give your project a Visual identifier.

5. Will you allow others to copy and modify your project? Your original will remain!



This section teaches you how to code additional features. New procedures show how many matches there are in many classes of the same size.

It requires adding new procedures, similar to previous ones, but with important differences. Just type the procedures **exactly** as they are, and you will succeed!

The program you write will:

- Use a slider to test many classes
- Keep track of the number of classes with birthday matches
- Calculate probability.


## ADVANCED CODING!

Extend Your Simulation!



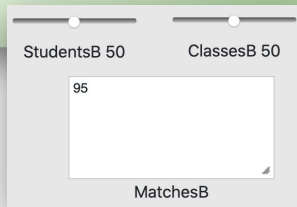
# Set Up a New Page

## Set up a second page for Experiment "B"

Save your work!  


1. Go to the **+** icon and select **Page**. Then add a **turtle** and a **text box** to the page. Name the text box **MatchesB**.

2. Add two sliders. Name one slider **StudentsB** and the other **ClassesB**.



Remember, use your turtle's name followed by a comma. This ensures the turtle on this page follows your instructions.

All slider, text box, turtle, and procedure names must be different from the ones you used before. In this experiment "B", we are adding "B" to everything.

3. Write a **setupB** procedure for this page. First, check what your new turtle's name is by right-clicking on the turtle.

**to setupB**  
**clean**

This must be a unique name.

**t2, pu**  
**setpos [-375 -90]**  
**setheading 90**  
**setcolour 15**  
**end**

# Write a New OneClassB Procedure

## Test Each Individual Class

1. *Write these three procedures in the Procedures Pane.*

*to OneClassB*

*repeat StudentsB [OneStudentOneDotB]*

*clean*

*end*

*to OneStudentOneDotB*

*t2,*

*pu fd 1 + random 365*

*if colourunder = 15 [IncreaseMatchCount]*

*pd fd 0 pu ;this leaves a dot*

*setpos [-375 -90]*

*end*

*to IncreaseMatchCount*

*SetMatchesB MatchesB + 1*

*end*

Although similar to the **OneClass** procedure, note the differences. It uses **StudentsB** and **OneStudentOneDotB**, because all these are “unique names” for your second experiment.

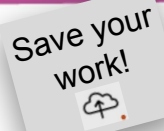
- Add **set** to the front of the text box's name to change the value of a text box, **SetMatchesB** sets the value of **MatchesB**. **MatchesB** reports the value in the text box **matchesB**. **SetMatchesB MatchesB + 1** sets the number in the MatchesB text box to one more than its previous value.

Save your work!



# Add a ManyClasses Procedure

## Add a new Procedure and a Button to run it



1. Add a new procedure to run **oneclassB** once for each class in the experiment.  
to ManyClasses  
setupB  
SetMatchesB 0  
repeat ClassesB [OneClassB]  
end

Set your **StudentB** and **ClassesB** sliders to different values and test **ManyClasses**.

2. Add a Button. Give it a label like “**Large experiment**”. Select **ManyClasses** in the **On Click**. Click **Apply**.

3. Add a text box with information on how to run the experiment.

Resize your button, move it to a good location.  
Then right-click on the button and select **Frozen**.  
Click **Apply**.

Select the number of students in a class and the number of classes (number of times you run the experiment).

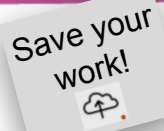
A screenshot of a software interface. At the top, there are two sliders: "StudentsB 35" and "ClassesB 50". Below them is a text box containing the number "56". Under the text box is the label "MatchesB". At the bottom is a button labeled "Large Experiment".

In this example, we ran the experiment 50 times with classes of 35 students. We had a total of 56 matches, which means that in some classes, we had more than one match.



# Calculate the Average

## What Are The Chances



1. Add a **text** box. Name it **Average**.

2. Write a **CalcAverage** procedure in the Procedures Pane.  
*How do you calculate the average?*

*In this case, it's the number of matches that occurred (**MatchesB**) divided by the total number of **ClassesB**.*

*to CalcAverage \_\_\_\_\_  
average, cleartext  
print 'Average number of matches for classes of this size:'  
print MatchesB / ClassesB  
end*

3. Add a **button** to run this procedure. Remember to give it a meaningful label, like **Average**.

4. Always **TEST** your buttons and procedures!

The procedure name must be different from the text box name.

Select the number of students in a class,  
and the number of classes (the number of  
times you run the experiment)

A screenshot of a Scratch interface. At the top, there's a text box with the instruction "Select the number of students in a class, and the number of classes (the number of times you run the experiment)". Below this are two sliders. The first slider is labeled "StudentsB 36" and has a value of 68. The second slider is labeled "ClassesB 50" and has a value of 1.36. Below the sliders are two text boxes: "MatchesB" and "Average". The "Average" text box contains the text "Average number of matches for classes of this size: 1.36". At the bottom, there are two buttons: "Large Experiment" and "Average".

StudentsB 36

ClassesB 50

68

Average number of matches for classes of this size: 1.36

MatchesB

Average

Large Experiment

Average

How many students must you have in a class to have an average of "close to 1" match per class?





# CODEtoLEARN

## Credits

**Principal Writer..... Susan Einhorn**  
**Contributors.....Alain Tougas**  
**Elena Yakovleva**  
**Sergei Soprunov**



Create a Birthday Match Simulation by [Code To Learn](#) is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).

## Partners



A program of



Connected North

With funding from

