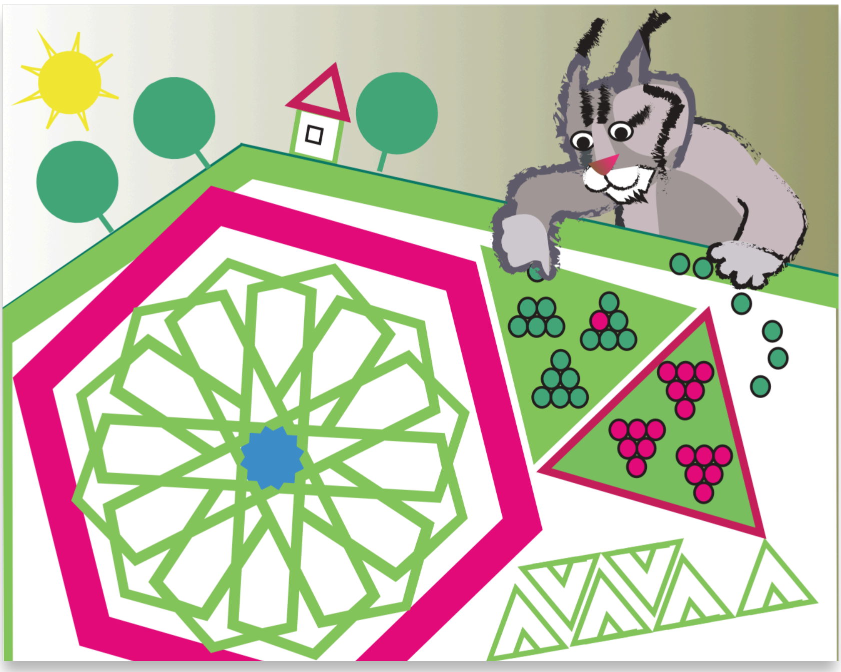
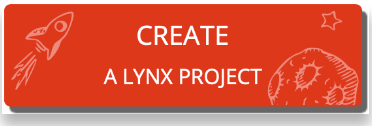
Project 1 - Geometric fun



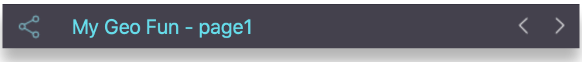
This project will introduce you to Lynx. You will learn about the Command Centre and the Work area (or page), a few Lynx primitives / commands, some geometry concepts like angles, polygons, the total turtle trip concept (the big 360), and the basic foundation of coding: creating your first procedure!

First things first, start a new project

Go to the Lynx web site (`lynxcoding.club`) and log into your account. [If you don't know how to do that or don't have an account, see **Create an account** on page 4]. Make sure you are on **My projects** then, click on



As a good habit, start by **naming your project** in the **top left corner**. Change **My project** to a name that will make your project recognizable from the list of projects on your **My Projects** page. If you don't give it a name, all your projects will be called **My project** and you will have a hard time finding out which is which!



Second good habit: save your project often. There is NO autosave. Simply click on this icon:

If you have to leave the project before it is completed, just remember to save before you leave. You will find your new project on the page **My Projects** when you are ready to continue.

Your first instructions: turtle's pen, moving and turning

No turtle on your page? Choose Turtle in the + menu!

You're looking at a new, blank project, with a turtle in the centre of the work area (or the page).

In the grey Command Centre below the white Work Area, type:

`pendown`	`PD` IS THE SHORT FORM. PRESS ENTER
`forward 100`	`FD` IS THE SHORT FORM. PRESS ENTER

`Pendown` puts the pen down so the turtle leaves a visible line. `Fd 100` is a command telling the turtle to move forward 100 turtle steps (pixels). We say that `fd` is the **command**, and `100` is the **input**.

Try these:

`right 60`	TURN RIGHT 60 DEGREES. `RT` IS THE SHORT FORM.
`back 150`	MOVE BACK 150 STEPS. `BK` IS THE SHORT FORM.
`left 145`	TURN LEFT 145 DEGREES. `LT` IS THE SHORT FORM.

Type many commands on one line, then press **Enter**. Remember that primitives like `forward`, `back`, `right` and `left` always require an Input. You need to tell the turtle how far to move or how wide to turn! Don't forget to leave a space between the **command** and the **input.**

Project 1 - Geometric fun

Experiment with putting the turtle's **penup** and **pendown** before you use **forward** or **back** commands.

Make a dotted or dashed line.

Print your initials.

Find out how many pixels (turtle steps) wide or high the work area is?

PLAY! Try small and big numbers! Try numbers less than 1. What does `forward -150` do?

Try a number larger than 9999, you will see your first **Lynx error message**. Error messages are important. Read them, they tell you how to fix the small problem you are having.

Now that your turtle is moving…

..add some colour and different line sizes! Here are some graphics commands you can use:

`setcolour` (or `setc` for short)	Sets the colour of a turtle and Pen
`setpensize`	Sets the turtle's pen width
`setbg`	Sets the colour of the background in the work area
`cg`	Stands for **C**lear **G**raphics. The turtle also goes home
`clean`	Cleans the page, without moving the turtle
`home`	Moves the turtle back home, in the centre of the page

Try these instructions, change the input values, make your own combinations:

`cg`	
`setpensize 20`	The default pen size is 1
`fd 50`	You can use the long form `FORWARD`
`clean`	
`setcolour 'blue'`	The default colour is 'black'
`rt 90`	
`fd 30 pu fd 10 pd fd 30 pu fd 10 pd fd 30`	

GOOD TO KNOW

Be careful with the single straight quotes around the colour names! You will find other colour names in Appendix C, at the end of this book.

You can change the background colour of the page too:

```
setbg 'yellow'
```

Unlike the turtle graphics, the background colour remains even after a `clean` or a `cg` command. You will have to use `setbg 'white'` to return to the original background colour.

GOOD TO KNOW…

Did you notice that SOME commands work as "just one word", while some other require a value?

`Cg`, `clean`, `penup`, `pendown` work "as is"- 1 word

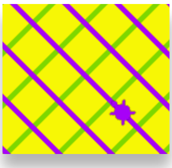
`Forward`, `back`, `right`, `left` , `setpensize` all want a number as an input. `Setcolour` and `setbg` want a colour name or number.

Explore these commands some more:

```
cg
pd
setpensize 5
rt 45
setcolour 'green'
fd 9999 fd 9999
lt 90
setcolour 'violet'
fd 9999 fd 9999
setpensize 1
```

9999 IS THE LARGEST NUMBER YOU CAN USE

BACK TO NORMAL PEN SIZE



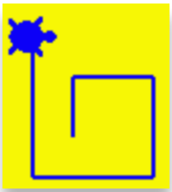
Usually, you can undo the effect of *a few previous commands* by clicking on the **Undo** button immediately after the action.

You can execute instructions many times without retyping: Type this instruction all on one line in the Command Centre:

```
cg
setc 'blue' forward 30 rt 90
```

YOU CAN TYPE SEVERAL INSTRUCTIONS ON A LINE

Press **Enter** and see the results. Now click the **Up Arrow** on your keyboard to return to the line with the Lynx code. Change `forward 30` to `forward 60` and press **Enter** again. Using Arrow keys you can re-execute any line already present in the Command Centre. Change `forward` again with `90` or `180`.



When you want a clean empty Command Centre, type:

```
cc
```

STANDS FOR **C**LEAR **C**OMMAND **C**ENTRE

Can you repeat that?

Type the following in the Command Centre and press **Enter**:

```
cg
repeat 10 [fd 100 bk 80 rt 6]
```

CG STANDS FOR CLEAR GRAPHICS

In plain English, this means,
Repeat ten times whatever instructions are inside the 2 square brackets. In this case, go forward 100, go back 80, turn right 6 degrees. Find square brackets to the right of the letter P on the keyboard.

Try these (use `cg` before each instruction, if you wish):

```
repeat 20 [fd 100 rt 165]
repeat 8 [fd 70 bk 60 rt 45]
repeat 10 [fd 100 rt 140 bk 100 rt 45]
repeat 6 [fd 80 rt 60 bk 80 lt 120 wait 2]
repeat 20 [fd 80 rt 18 wait 2 bk 80 fd 10 wait 2]
```

LOOK! A `repeat` inside a `repeat`!

```
repeat 10 [repeat 15 [fd 4 rt 15] rt 120]
repeat 9 [repeat 10 [fd 4 rt 20] rt 120]
```

Add `wait` commands to slow down how fast your commands are executed. it will be easier to see each command running and then understand what is going on.

Something like:
```
repeat 8 [fd 70 wait 5 bk 60 wait 5 rt 45]
```

Try changing one input at a time in some of those examples.

Try putting `pu` and `pd` in various spots inside some of those examples.

Try doing the same line over and over again. Pretty cool, eh? Can you write a `repeat` command to do that automatically?

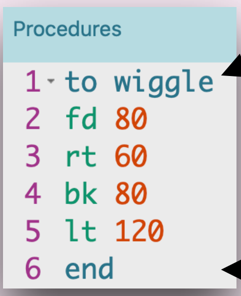
Your first procedure

A **procedure** is a group of Lynx instructions with a name you choose. The **procedure's name** becomes a **new command**, just like `forward` and `right`. Procedures that you create *only work inside the project you are working on now*!

Click on the **Procedures** icon to open the **Procedures Pane**.

Click in the Procedures Pane, where the lines are numbered, and type this procedure:

A procedure has three parts

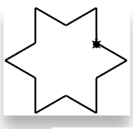


```
1 to wiggle
2 fd 80
3 rt 60
4 bk 80
5 lt 120
6 end
```

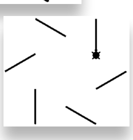
The title line: the word `to`, a space, and the `name` of the procedure as a single word (no space)

Instructions: As many as you want, one or many instructions per line.

The word `end`, alone on the last line



Type `wiggle` in the Command Centre. Try it several times actually. Make sure your Pen is down (`pd`). Change the inputs (values) in the procedure. Try it again.



Can you write a procedure to create this design by using `pu` and `pd`?

Project 1 - Geometric fun

Many squares, many styles

Create this procedure in the Procedures Pane.

```
to square
repeat 4 [forward 100 right 90 wait 2]
end
```

If you want to make a larger or smaller square, which number would you change?

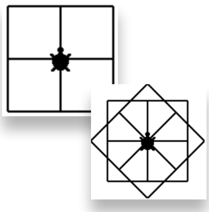
Yes. The input of the `forward` command.

If you want to draw the square on the other side, what command do you need to change?

Yes. Change `right` to `left`. Or use `back` instead of `forward`.

Make the smallest square you can. Make the largest square you can.

Can you make these patterns?



Hint: Make a square, make the turtle turn a bit (using a `rt` instruction in the Command Centre), then make another square, and so on.

Make other patterns that you dream up! :-)

Total turtle trip, let's do triangles!

Look at your `square` procedure again. How much does the turtle turn in total? **Four times 90 degrees**, right? Do the math, **4 x 90 is 360 degrees**. Like doing a 360 on a skateboard or a snowboard. That is the **Total Turtle Trip**, or **TTT** for short!

The same is true for a triangle (or any polygon).

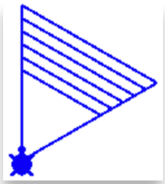
GOOD TO KNOW

In order to make a closed figure, the turtle must turn a total of 360 degrees.

Think about the procedure below - what inputs would you use to create an equilateral triangle? Do the math and try!

```
to triangle
repeat ?? [fd 100 rt ??]
end
```

Can you make an equilateral triangle that goes to the left? Or an equilateral triangle with shorter (or longer) sides?



Can you make a pattern similar to this?

Make a triangle pattern of your choice.

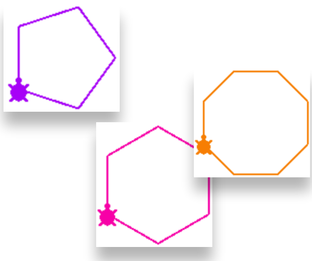
Think about using `pu`, `pd`, `setcolour`, `setpensize`

Some polygons now!

Following the "**total turtle trip**" rule above, can you figure out how to make these other polygons?

```
to pentagon
repeat 5 [fd 100 rt ??]
end

to hexagon
repeat ?? [fd 100 rt 60]
end
```



What is the `rt` angle and the number of `repeat` for each of these?

Make a polygon with as many sides as you can!

Make a polygon but have the computer do the arithmetic for you to figure out the angle!

Note: You may have to make your sides shorter!

And for the grand finale, circles!

Think about the circle procedure below - what inputs would you add to create a circle? Ready to try it? Remember the TTT!

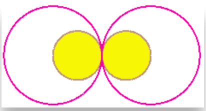
```
to circle
repeat ?? [fd ?? rt ??]
end
```

What is a circle after all? **Hint**: you may have to use a very small input for forward, otherwise your circle will go *wayyyy* over the edge of the page.

Once you discover the "trick" of the circle, you can have all sorts of fun!

Can you make:

- a very small circle? A very large circle?
- some "googly eyes" (circles to the right, circles to the left)?
- a snowman?



Create other shapes!

About the filled-in googly eyes: Drag the turtle **inside** one eye, run `setpensize 2 setcolour 'yellow'` and `fill` in the **Command Centre**. `Setpensize 2` makes a thicker line so the colour doesn't leak through the edge of the circle. Not sure what `fill` does? Remember to use Tooltips! Place your cursor over the word `fill` for 2 seconds. Now you can't see the turtle, because it is a yellow turtle inside a yellow eye. Run `setcolour 'black'` to see the turtle again, and drag it to the other eye to fill it too.

Show these to your friends and family!

```
to circle1
repeat 360 [fd 1 rt 1]
end

to circle2
repeat 180 [fd 0.5 rt 2]
end

to circle3
repeat 720 [fd 1 rt 0.5]
end
```

```
to circle4
repeat 720 [fd 0.5 rt 0.5]
end
```

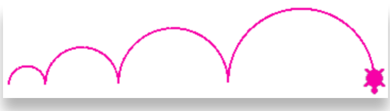
More advanced ideas

Why do `circle1` and `circle4` look the same? Ask your friends to explain it!

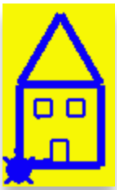
Can you make the same size circle several different ways?

Can you make semicircles? Of different sizes?

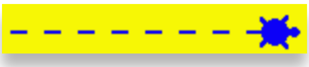
Try to make a pattern like this one. Hint: You'll have to turn your turtle between semicircles.



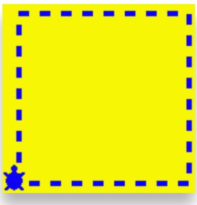
Can you make this using a `square` and a `triangle` procedure?



Can you make a `DashedLine` procedure to produce this?



Good! Now can you use this procedure to create a dashed square?



Curriculum Links for Ontario

MATH C3 - Solve problems and create computational representations of mathematical situations using coding concepts and skills.

C3.1 - Solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential, concurrent, repeating, and nested events.

C3.2 - Read and alter existing code, including code that involves sequential, concurrent, repeating, and nested events, and describe how changes to the code affect the outcomes.