



# GETTING STARTED WITH LYNX

BASIC TECHNIQUES TO GET YOU STARTED



# Table of Contents

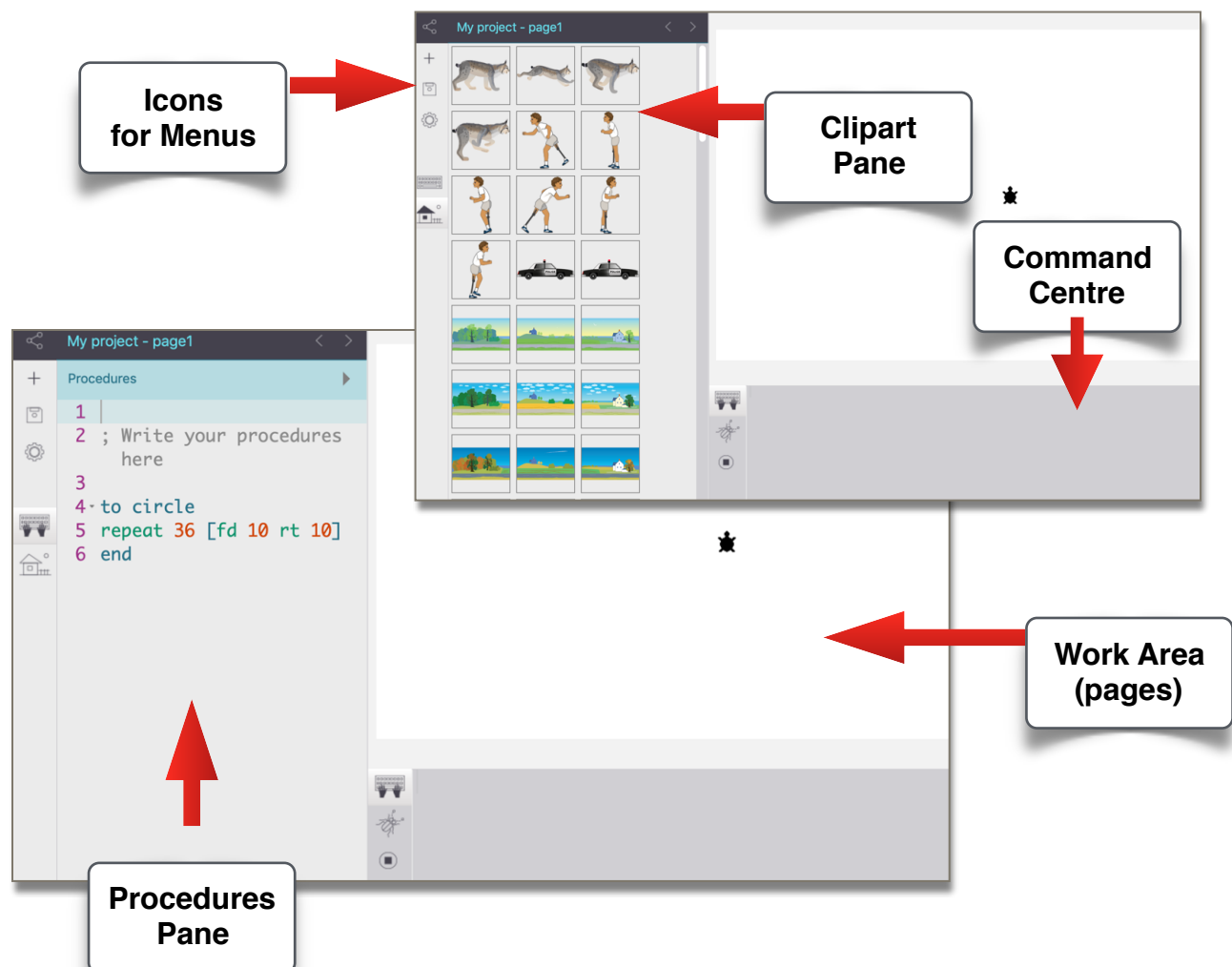
<b>Introduction</b>	<b>3</b>
<b>Creating an account and creating your first project</b>	<b>4</b>
<b>Explore settings: UI colour, font, auto-complete, learner mode</b>	<b>5</b>
<b>Need help using Lynx?</b>	<b>6</b>
<b>How to use samples</b>	<b>7</b>
<b>Exploring the " + "</b>	
<b>Adding, moving, turning and stopping turtles</b>	<b>8</b>
<b>Adding text</b>	<b>12</b>
<b>Creating procedures</b>	<b>14</b>
<b>Adding and using buttons</b>	<b>15</b>
<b>Adding and using sliders</b>	<b>16</b>
<b>Adding sound and music</b>	<b>17</b>
<b>Adding and switching pages</b>	<b>18</b>
<b>Clipart</b>	
<b>Sample clipart</b>	<b>19</b>
<b>Adding your own clipart</b>	<b>20</b>
<b>Using clipart for a turtle shapes</b>	<b>22</b>
<b>Using clipart to make a nice background</b>	<b>24</b>
<b>More about turtles: event driven programming</b>	
<b>Making a clickable turtle</b>	<b>25</b>
<b>Colour detection</b>	<b>26</b>
<b>Collision detection</b>	<b>28</b>
<b>More about procedures</b>	<b>29</b>
<b>Your project</b>	
<b>Saving and retrieving your project</b>	<b>30</b>
<b>Sharing your project with your friends</b>	<b>32</b>
<b>List of most common primitives</b>	<b>33</b>

# Introduction

Lynx is a robust programming platform based on the Logo language. It is a more mature programming language than *coding with blocks*, but it is a lot friendlier than professional languages like Javascript or Python. If you are somewhere between a block coder and a professional programmer, Lynx is for you.

You work on your Lynx projects in a browser window and you save them in the cloud, or locally on your computer. You can share your projects and even let your friends play or modify them.

The programming platform comprises a Work Area (you can have multiple pages in your project), a Command Centre, a Procedures Pane and a Clipart Pane.



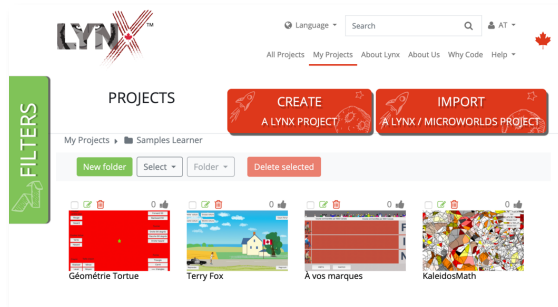
Note: In this document, "[clicking](#)" refers to computers. If you are using a tablet, take that as a "[touch](#)" gesture.

# Creating an account and creating your first project

Lynx lets you create projects and also provides a space for your projects. By default, it is a private space, though you can choose to make your projects public or share them with others. More about this later.

Go to [www.lynxcoding.org](http://www.lynxcoding.org). If you see your [nickname](#) in the top-right corner, you are already registered and logged in. You can now go to [All Projects](#), [My Projects](#), [Create a Lynx project](#) (see pic at bottom of this page) or any other page of the Lynx web site.

If you see [Login/Register](#), click on it to create an account or login. In the Help Section of our website there is a PDF called [How to Create a Lynx Account if you do not have one](#). Use that PDF to create your account. We offer third-party account registration and login using [Google](#), [Microsoft](#) and [Facebook](#)\*. Once your account is created, click on Login/Registration page and login.



Welcome aboard! Click on [My Projects](#) and see your private area in the Lynx cloud. Here, you will see projects that you have created (none if this is a new account), and a big red button allowing you to create your first project.

If this is your first time with Lynx, and in order to follow the instructions in the following pages, click on



Note: You can only select the Work Area size when you start a new Project.

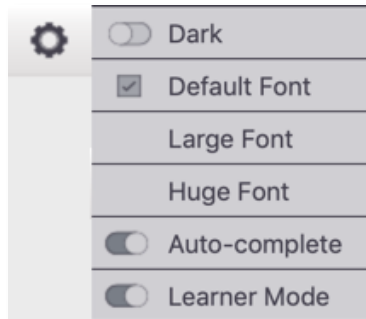
\* Login using [Apple/iCloud](#) will be available in November, 2019.

# Explore Lynx's settings

(UI colour, font, auto-complete, learner mode...)

Customize Lynx. Click on the [Settings](#) button:


From this menu, you can...



1. Toggle the [Dark mode](#) for a different look.

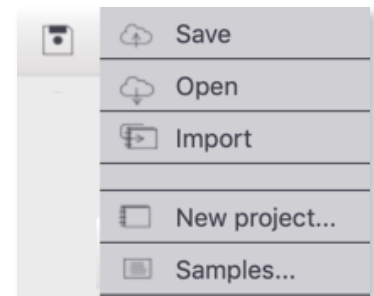
2. Choose one of 3 [font](#) sizes for the Procedures Pane and the Command Centre.

3. Toggle the [Auto-complete](#) feature. Lynx suggests primitives as you type in the Procedures Pane or in the Command Centre. A primitive is a command you use in Lynx.

4. Toggle [Learner Mode](#). In Learner Mode, the Help page contains only the 40 easiest and most popular primitives (commands). After setting the Learner Mode on or off, click on the [Help](#) button again to display the Help page again. The Help button is the  in the bottom left corner

## SAVE YOUR WORK OFTEN!

Please tell your students to save their project regularly. There is NO Autosave! To Save, go to the Diskette icon on the left side of the Editor and SAVE is the first choice in the submenu. There is a longer explanation about saving near the end of this PDF.



# Need help using Lynx?

Lynx will help you in many ways.

## TOOL TIPS

When you type a primitive in the Command Centre or in the Procedures Pane, briefly leave your mouse pointer on the primitive and Lynx will display a

floating tool tip with a short definition, the inputs required, and an example of how it could be used. The last page of this PDF has a list of the 40 most-used primitives.

repeat


REPEAT number list-of-instructions  
Runs the list of instructions the specified number of times.  
  
repeat 90 [bk 40 fd 40 rt 4]

## AUTO-COMPLETE

set  
seth  
setheading  
setpos  
setsize

When you start typing in the Command Centre or in the Procedures Pane, Lynx suggests primitives that match what you are typing. You can turn this feature *off* in the [Settings](#) (gear) menu.

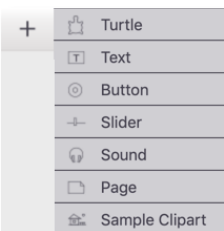
## HELP PAGE

In the bottom left corner, Click on the [Help](#) button  to open a floating help page. The Help page matches your current UI language and mode: Learner mode has a shorter Help. See an explanation of [Settings](#) earlier in this guide.

## SAMPLE PROJECTS

Choose [Samples...](#) in the [Disk](#) menu to see a list of samples that matches your mode: Learner mode has simpler samples. You can try the sample and read the explanations written by us, in grey, in the Procedures Pane to understand how we created it. You can change it and save it as your own project.

## SAMPLE CLIPART

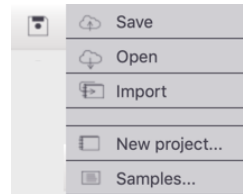


Want to try something quick without finding some clipart on your own? Choose [Sample Clipart](#) in the "+" menu. Lynx will populate the Clipart Pane with some clipart, made by us, that you can use immediately in your project.

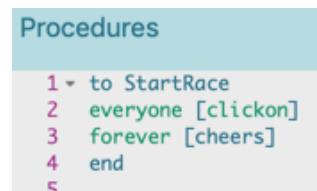
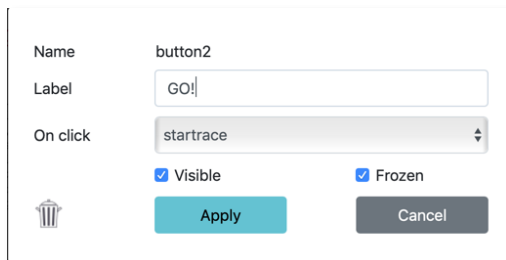
# How to use the samples

Here's how you can use our samples to get ideas, learn tricks, "borrow" some code and just tweak them to your heart's content.

There are two sets of samples: Learner Mode (easy) and Advanced (challenging). Choose [Samples...](#) in the [Disk](#) menu and choose one of the samples in the list of your choice.

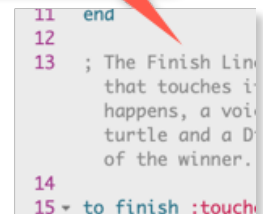


Play with the sample. Right-click on buttons, turtles, text boxes, anything you find. Buttons and turtles often refer to procedures in the Procedures Pane, like this one: the button named [Go!](#) calls the procedure [startrace](#):



## Comments

Don't forget to read our comments in the Procedures Pane. When you see a semi-colon, these are our comments, *not part of the actual code*. You will learn a lot from reading comments and looking at the Procedure (usually immediately below it). And don't forget to add comments to your code as well! Programmers use comments to explain, in clear language, what the code does. It's a good habit for coders!



You can try 3 things with Samples:

- ▶ Make changes here and there just to see if you understand what's going on.
- ▶ Copy some code, close the sample project and paste the code in your own project (you will certainly have to make adjustments, that's good!)
- ▶ Or make some changes and save the modified project as yours! Saving a project was explained on Page 5 and more Saving info is found near the end of this PDF.

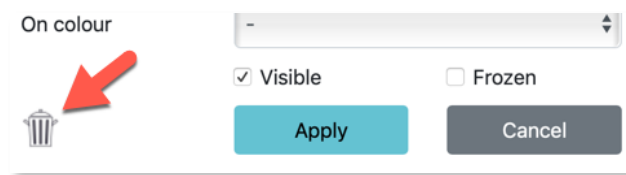
# Adding, moving, turning and stopping turtles

## ADDING TURTLES

A new project comes with a turtle in the centre of the page. If it's gone or if you need more turtles, simply choose Turtle in the "+" menu. If you already have a turtle in the centre, the new turtle will appear on top of it. Drag it away to see both turtles. Turtles are named t1, t2, t3 and so on, but, in a couple of pages, we'll show you how to change the name of the turtle.

## REMOVING TURTLES

Too many turtles on your page? Right-click on a turtle to open its dialog box, then click on the trash can.



## MOVING AND TURNING THE TURTLE, PEN UP AND PEN DOWN

Now that you have a turtle, type these commands in the Command Centre (the grey area below the Work Area). After you press Enter/Return the code will execute.

```
forward 50
right 90
fd 100      (fd is short for forward)
rt 90       (rt is short for right)
pd          (means "pen down" - now the turtle will
            draw lines as it moves)
back 100    (or bk if you wish)
```

Now you know about `fd`, `bk`, `rt`, and `pd`. Experiment with these commands, you can figure out what they do:

```
setcolour "red" (setc is short for setcolour)
fd 50
setpensize 20
glide 200 1      (200 is the distance, 1 is the speed)
glide 200 10
pu              (means "pen up" - now the turtle won't draw)
setsize 80      (the turtle gets twice its size. setsize 40 returns it to normal
                size)
```



# Adding, moving, turning and stopping turtles (cont.)

These instructions can also be put in procedures, and procedures can be called in many ways. Look below for the sections about creating procedures, buttons, clickable turtles, etc.

The colour names for `setcolour` are: `black`, `blue`, `brown`, `cyan`, `grey`, `green`, `lime`, `magenta`, `orange`, `pink`, `red`, `sky`, `turquoise`, `violet`, `white`, `yellow`.



## WHERE ARE YOU HEADING TO?

Besides right and left, there is another way to set the turtle's heading: `setheading` (or `seth` for short). Try these commands and see if you can spot the difference.

`cg` (stands for clear graphics - the work area is cleaned and the turtle is back home)

```
rt 90
```

```
rt 90
```

```
rt 90
```

```
cg
```

```
setheading 90
```

```
seth 90
```

```
seth 90
```

You see, each time you run `rt 90`, the turtle turns 90 degrees, starting from its current heading. Always 90 more degrees. Turn 90 degrees, *relative* to your current heading.

Each time you run `setheading 90`, you tell the turtle to "face East". You can tell it that several times, *it will always face East*. This is the *absolute* heading.

The input for `setheading` corresponds to the degrees on a compass.

# Adding, moving, turning and stopping turtles (cont.)

## TALKING TO TURTLES (Hey! I'm talking to you!)

If you have only one turtle on the page, it will listen to every command you type. If you have more than one turtle on the page, only one will execute your commands: it is the last one you created or the last one you clicked on.

BUT... you can decide otherwise. Say you have three turtles on the page, named t1, t2 and t3 (to see the turtle's name, right-click on each turtle and check the top of the dialog box:



By knowing the turtle's name, you can "talk to" any turtle (or turtles), any time. (You can also choose another 1 word name for the turtle here). With t1, t2, and t3 on the page, you can use instructions such as these:

```
t1, forward 50
t2, glide 200 1
t3, setsize 80 wait 10 setsize 40 (bigger, wait one second, smaller)
```

If you want to talk to several turtles at the same time, use the command `talkto`:

```
talkto [t1 t2]
setheading 90
```

There is also a way to talk to every turtle present on the page.

```
everyone [setheading 90]
```

Or, if you think you've lost a turtle:

```
everyone [st]
```

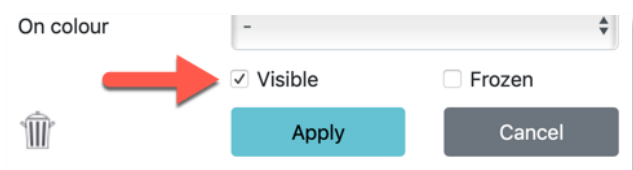
`Everyone` is followed by a pair of square brackets. Inside the brackets, type the instructions that every turtle on the page must execute. `st` means show turtle.

## HIDING, SHOWING, FREEZING

A turtle can be invisible! But before you hide it, make sure you know its name, so you can show it again.

```
t1, ht (stands for hide turtle)
t1, st (stands for show turtle)
```

You can also use the `Visible` check box in the turtle's dialog box.



# Adding, moving, turning and stopping turtles (cont.)

Sometimes, when you make a game, you want to freeze the turtle so it's impossible to drag it around (to cheat 😏). Try this:

`freeze "t1` (that's the word freeze, a space, a quotation mark, and the name of the turtle)

Try to drag it around now. Good luck!

`unfreeze "t1` (you can drag it around now)

You can also freeze and unfreeze a turtle from its dialog box. Note that a frozen turtle will still execute your commands... You just can't drag it around.

# Adding text

## ADDING A TEXT BOX

You can't type text directly in your work area, you must create a text box for that. Simply choose Text in the "+" menu, and a text box appears on the page. It is named "text1".

## TYPING TEXT USING THE KEYBOARD

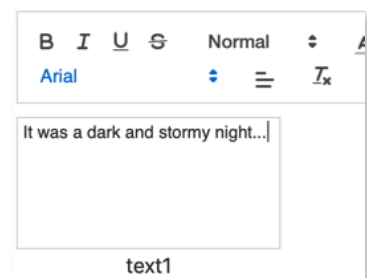
Naturally, you can just click in the text box and start typing. Pretty obvious, right!

## FORMATTING YOUR TEXT USING THE FORMATTING PALETTE

If you click inside a text box, above it you will see a box with formatting options. Try them! If you have selected text, the formatting applies to it. Otherwise, it sets the format for the text you are about to type at the insertion point.

Type some text in a text box and try different formats.

If you can't see the Formatting options, click on the Text Box name and drag the Text Box lower in the Work Area.



## TYPING TEXT USING COMMANDS

There are also commands that let you add text to the text box. Try these instructions in the Command Centre. Remember that all these commands can also be part of procedures you make (more on that further down).

```
print "hello
```

Use a quotation mark to indicate that hello is just a word you wish to print, not a command. `Print` will print the text, and bring the insertion point to the next line.

```
print "there
```

See, the insertion point was on the next line. You can use `pr` for short.

```
cleartext
```

This empties the text box. You can also use `ct` for short.

```
print [hello there]
```

```
print 'hello there'
```

Use the square bracket to print several words at once (it's called a list). Replace a bracket with a single quotation mark to get the same result, if you like.

```
insert [My name is]
```

`Insert` works like `print`, but the insertion point *stays on the same line*, ready to add more text.

```
insert "| |
```

Use vertical bars to enclose anything you wish to print "as is". In this example, it is inserting a blank space.

```
insert "Kim
```

Now *My name is Kim* should be visible in the text box.

# Adding text

(cont.)

## MULTIPLE TEXT BOXES

If you have several text boxes on your page, you can use this method to talk to a specific text box:

```
text1, print "hello  
text2, print "goodbye  
cleartext
```

The text box that "listens" to you is the last text box that you have created, or the last text box that you used (clicked in), or the last text box that you "talked to" using the comma syntax. In the example above, the `cleartext` command will be executed by the text box `text2`.

## MOVING TEXT BOXES, VISIBLE, TRANSPARENT, WITH OR WITHOUT A LABEL

To move a text box, simply drag it by its label.

Also, right-click on the text box to open its dialog box. Here, you can hide or show the text box or its label, and make the text box transparent. You can also change its name - make sure you use a single word (no spaces) for the name. You will see why later.

Oops, if you make the text box invisible, you can't open its dialog box again to make it visible. No worries, type this in the Command Centre:

```
showtext
```

```
text1,
```

```
showtext
```

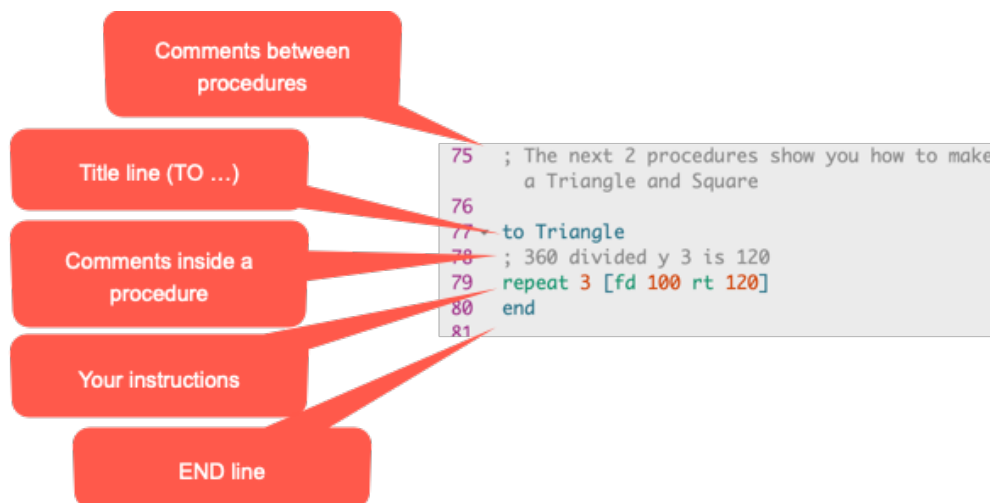
The Text Box reappears. If you want to show or hide another text box, you will have to call it by its name like this:  
Use its name, followed by a `comma`. This is why it is important to use a single word (no spaces) for naming things. Now, this text box will listen to your commands.  
`Hidetext` does the opposite.

# Creating procedures

You will see procedures all over this Getting Started guide. A procedure *adds a new command* that Lynx will understand but *only in the specific project you are working on*. Procedures not only help you organize your code but they will make debugging easier.

Here are the first things you have to know about them:

- A procedure must start with the word `to`, followed by a space, and the name of the procedure. The name must be a single word with no spaces. The name can't be a Lynx primitive already like `Forward` or `Right`.  
These are good names: `start`      `TurnAround`      `turn_around`.
- A procedure must finish with the word `end`, all by itself on the last line.
- A procedure includes primitives and can include other procedures!
- You can (and should) add comments before or inside your procedures. All you have to do is start a line with a semi-column (;). Check our samples!



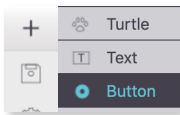
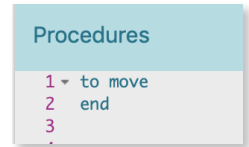
- When a procedure is created, you can use it like any Lynx primitive:
  - In the Command Centre
  - In a clickable turtle. See example on page 25
  - In a button. See examples on pages 15 and 18
  - In another procedure
- For colour and collision detection, Lynx will prepare the procedure for you, as described further down.
- See Page 29 for More Info on Procedures

# Adding and using buttons

A button is always linked to a procedure found in the Procedures Pane. There are two ways to create a button and link it to a procedure.

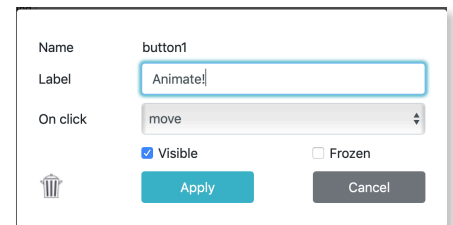
## FIRST CREATE THE PROCEDURE

This is the quickest method: start by creating a procedure. All procedures must start with **To** followed by a space, then a single word. Procedures must finish by the word **End** on the last line by itself. If you are not sure yet how your procedure is going to work, just create the title line and the end line.



Now create a button: choose **Button** in the "+" menu.

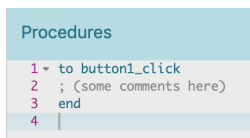
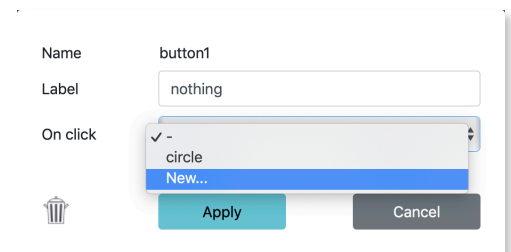
A button named **Nothing** appears on the page. Right-click on the button to open its dialog box. In the Label field, type something *meaningful* that you want to see on the button. Then, in the **On Click** drop down menu, select the name of the procedure you have just created. Click on **Apply**.



Now work on your **move** procedure and use your button to test it. If you want to change the size of the button, drag the arrow in its bottom right corner. Same for Text Boxes.

## FIRST CREATE THE BUTTON

You can start by creating the button even if the procedure does not exist. Choose **Button** in the "+" menu. A button named **Nothing** appears on the page. Right-click on the button to open its dialog box. In the Label field, type something *meaningful* that you want to see on the button. Then, choose **New...** in the **On Click** drop down menu. Click on **Apply**. This creates a procedure named **button1\_click**.



Now work on your procedure, the button is linked to it. Give your procedure a useful name. If your procedure starts an animation, you may want to call it **move** or something like that. If you rename your procedure (which is a good idea!), *you need to re-link the button to the procedure*. Open the button's dialog box and choose your new procedure in the **On Click** drop down menu.

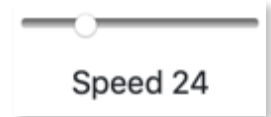
# Adding and using sliders

## CREATING A SLIDER

A slider is just like a variable number that you can control using your mouse. To create a slider, simply choose **Slider** in the "+" menu, and a slider appears on the page. It is named "slider1", it goes from 0 to 100, and we set the current value at 50. You can move the slider by dragging its name.

## SET THE RANGE OF THE SLIDER

Right-click on the slider to open its dialog box. Here, you can change its name (always use a single word, no space), its min and max values, and its current value (which must fall between the min. and max. values). You can also make it invisible, show or hide its name, and freeze it. When frozen, a slider can be used but it cannot be relocated.



## USING YOUR SLIDER (GET AND SET ITS VALUE)

The name of the slider reports its value. Type this in the Command Centre:

```
show slider1
```

Assuming you have a slider with that name on your page, this will print the current value of the slider in the Command Centre.

```
setslider1 10
```

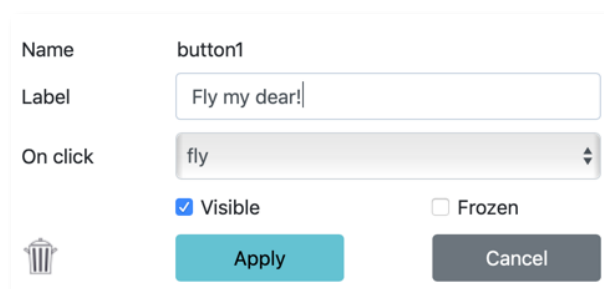
The word set, with the name of the slider (no space), will set the value of the slider without having to touch it.

## USING YOUR SLIDER (TO CONTROL THINGS)

Make sure you have a turtle on the page, and configure a slider with a min of 0 and a max of 10. Also change the name of the slider to **speed**.

Create a procedure such as this one. The input to **forward** is the value of the slider called **speed**, divided by 10 (otherwise, the turtle would fly too fast).

```
to fly
  forever [forward speed / 10]
end
```



Create a button and set it to run the procedure **fly**. Click on the button, and use the slider to control the speed of the turtle.

# Adding sound and music

## BRING A SOUND BITE INTO YOUR PROJECT

The supported sound formats, using Chrome and Firefox, are: WAV and MP3.

To bring a sound file into your project, simply choose **Sound** in the "+" menu. The Import Sound dialog box appears. Choose a sound file on your device or one of the other 3 options in the grey boxes. The Sound file you chose will now be visible in the URL field. Click on Create. A sound icon appears on your page.

You can right-click on the sound icon to edit its name. Again, use a single word, with no spaces. If you wish, you can also hide this icon.

## USING SOUND IN YOUR PROJECT

You can click on the icon to play the sound, but sometimes, you will want to play the sound as your program runs (when there is a collision or a winner for example). The name of the sound clip is a command that plays the sound. Say you have a sound named **cheers** on your page (visible or not). You can type **cheers** in the Command Centre, or include it as an instruction in a procedure.

Here is an example from our **Get Set, Go** sample. When the race starts the sound (**cheers**) begins.

```
to StartRace
  everyone [clickon]
  launch [cheers]
end
```

In the procedure above: **Launch** will run the instruction (play the sound) once, as an independent process.

If you want tons of sound effects that are free for educational purposes, go to this BBC site (BBC Sound effects): <http://bbcsfx.acropolis.org.uk>.

# Adding and switching pages

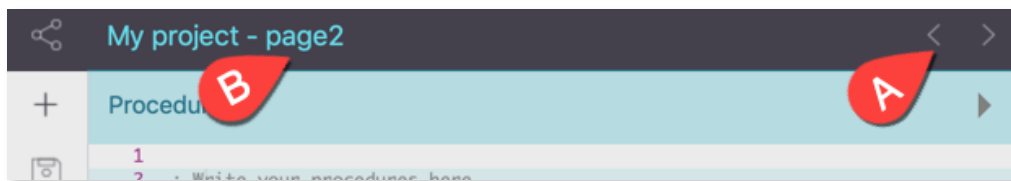
## ADDING A PAGE

One page is nice but you may want more pages if your project is a presentation, story, or a multi-level game, or if you want to write instructions for your project on a different page.

To add a page to your project, simply choose **Page** in the "+" menu. You immediately get a new, empty page.

## SWITCHING PAGES MANUALLY

To switch pages manually, simply click on the left and right arrows at the top of the Procedures or Clipart pane (A). The name of the page appears directly to the right of the project name (B):



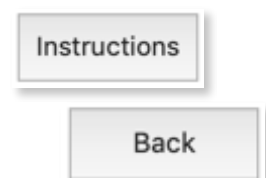
## SWITCHING PAGES UNDER PROGRAM CONTROL

The name of a page is also a command that goes to that page. Here is an example in which a button on **Page1** leads to the Instructions page, and a button on **Page2** returns to the Lab page.

Create procedures such as these:

Then, create a button on **Page1** that runs the procedure named **Instructions**, and a button on **Page2** that runs the procedure named **BackToLab**.

```
23 to Instructions
24 page2
25 end
26
27 to BackToLab
28 page1
29 end
```



You can also use page names as commands in a clickable turtle, or in a collision detection, or in a colour detection. You can place a turtle near the bottom of your work area and, in the Sample Clipart, there are 2 images of arrows to use for turning a page.

## RENAMING OR REMOVING A PAGE

This is how you **rename** or **remove** a page in your project:

```
rename "page1" "intro"
remove "page3"
```

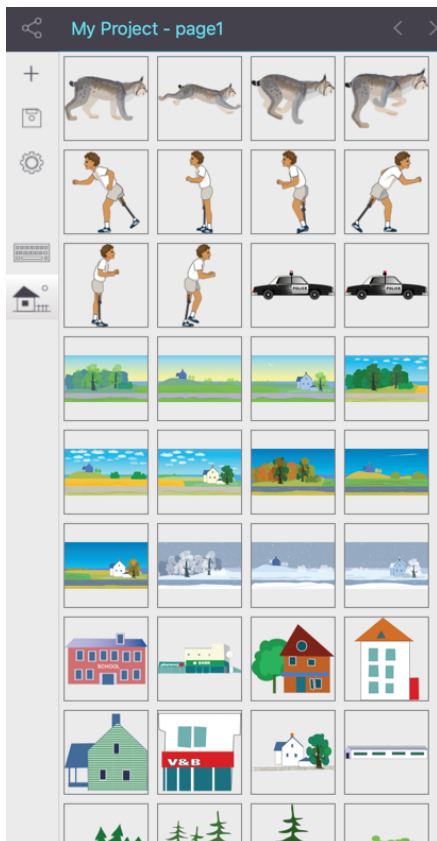
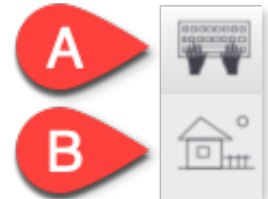
# Sample clipart

## INTRODUCING THE CLIPART PANE

Most projects need some sort of clipart, either as a full background image, as shapes for your turtles or some other graphical elements.

A background image is a large clipart. Turtle shapes and graphical elements (a cloud for example) will be smaller. We included some sample clipart to get you started quickly, if you just want to play around with Lynx. On the next page, you will learn how to get your own clipart.

Click on the [House](#) icon (B) to open the [Clipart pane](#) (the [Keyboard](#) icon (A) will bring you back to the [Procedures pane](#)).



## ADDING OUR SAMPLE CLIPART

The Clipart pane is now open and you see many empty boxes. Next, choose [Sample Clipart](#) in the "+" menu. The [Clipart Pane](#) will be populated with clipart (to create animations), very large clipart (to use as backgrounds) and mid-size clipart (to be used as graphical elements on your page).

The Clipart pane will include *both* our Sample Clipart and your own clipart. It is the Clipart storage unit!

See "[Adding your own clipart](#)" and "[Using clipart...](#)" further down.



# Adding your own clipart

## WHERE DO YOU GET CLIPART?

You can get your own clipart from three places:

- Use any paint program and draw your own images. Save your art as a JPG or PNG (PNG supports transparency around your objects if you need it).
- Download images from any image search web site. Respect copyright issues!
- Use the camera on your smartphone or tablet.

In any case, you may want to process your image before importing it into Lynx, so it has the right size (see below) or maybe for trimming the edges around objects to make them transparent (PNG only).

## HOW TO IMPORT CLIPART (COPY-PASTE)

This is the *easiest* method. First, go **copy some clipart** in a paint program of your choice or on a web page, or by doing a screen capture. Press **Ctrl-C** (**Command-C** on a Mac)

Now click on the **House** icon to open the Clipart Pane. Click on an empty box to reveal a "+" sign. Now press **Ctrl-V** (**Command-V** on a Mac).



You can use a spot that is not empty if you do not want to keep the existing clipart. Click on the clipart, then click the **trash can**. Click again and you will get the "+".



## HOW TO IMPORT CLIPART (IMPORT A FILE)

Click on the **House** icon to open the Clipart Pane. Click on an empty box to reveal a "+" sign in the bottom right corner, and click on that "+". Then use the dialog box to locate a clipart file on your device or online. Finish by clicking the **Create** button.



# Adding your own clipart (cont.)

## LARGE CLIPART FOR BACKGROUNDS

Run this instruction in the Command Centre:

`show projectsize` This will print a pair of numbers such as 800 450 in the Command Centre. This is the **width** and **height** of your project in pixels, or turtle steps. Your numbers may be different.

So when you search for images to be used as backgrounds, keep these numbers in mind.

**Too small:** If you find images that are too small, you will have to stretch them to cover the entire background, and they will *not* look good.

**Too large:** If you find images that are way too large, that's OK, because as you import them, Lynx will resize them to match the project size.

Later on we will show you how to use your newly acquired clipart.

## SMALLER CLIPART TO USE AS TURTLE SHAPES OR GRAPHICAL ELEMENTS

Same technique, but here, if you just need a cat, a ball, a tree or an atom, try to find images that are small, relative to your project size. In this example, the tree is a 100-pixel clipart. How much is 100 pixels? Add a turtle to the work area and run `pendown forward 100`. That's 100 pixels.

You can use `setsize` to resize the turtle, but there is no point having 5-pixel or 1,000-pixel clipart for turtles. So do the math. If your project is 800 x 450 (so 450 pixels vertically), and you need an object that's half of that, any image around 225 pixels is good.

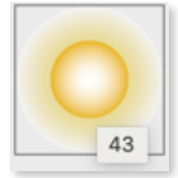


# Using clipart as turtle shapes

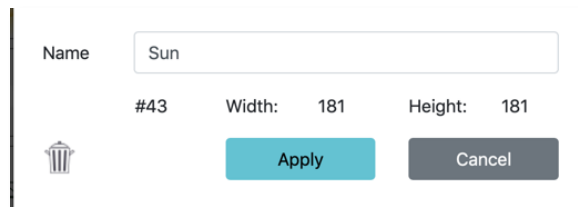
Assuming you added our Sample Clipart or imported your own clipart, now add a turtle to the page.

## A SINGLE CLIPART

In the Clipart Pane, leave your mouse pointer over a box that has clipart. You will see its **number**.



Right-click on the clipart. You can give it a name in its dialog box. Use a one-word name, with no spaces.



Now in the Command Centre, try these instructions

```
setshape 43
```

Or use the number you got. You can use `setsh` for short.

```
setsh 0
```

0 is the original turtle shape.

```
setsh "sun"
```

Include quotation marks before it, because `sun` is just a name, not a command that does something.

Note that only the normal turtle shape rotates (when you use `right`, `left`, or `setheading` commands). This means the head of the turtle will be pointing in the direction you chose.

# Using clipart as turtle shapes (cont.)

## ANIMATION USING A SINGLE CLIPART

This example uses the Sample Clipart. It is a *really* good idea to test your setup and procedure using the normal turtle shape because you can see where the turtle is heading. Once you are happy with it, add a clipart shape.

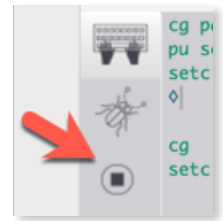
Make sure you have a turtle on the page. Then type these instructions in the Command Centre:

```
setheading 90  
forever [forward 1 wait 1]
```

Click on the **Stop All** button to stop the animation. It is the button below the bug just to the left of the Command Centre. Run the same line, change the values after `forward` and `wait`. See what happens.

Now type:

```
setshape 11 (the police car shape from the Sample Clipart)  
forever [forward __ wait __ ] Use whatever numbers you like.
```



## ANIMATION USING SEVERAL CLIPART

This example uses the Sample Clipart again with Terry Fox running to the right. These are shapes 8, 9 and 10. Try this:

```
setshape 0  
setheading 90  
setshape [8 9 10]  
repeat 100 [forward 10 wait 3]
```

Here, `setshape` uses a list of numbers as input. When you do that, Lynx switches shapes *each* time the turtle moves (`forward`, `back`). You should play with the numbers (`fd` and `wait`) to create a realistic animation. Have fun!

# Using clipart to make a nice background

Creating a background is a four-step process:

1. Acquire a large clipart
2. Give that shape to the turtle
3. Stamp the turtle
4. Set the turtle back to its original shape

To create a nice background, you can use a paint program to draw an image, or you can just make a screen capture of something you like. Remember (see "[Getting your own clipart](#)" above) to use graphics that are about the size of your project, or a bit larger.

Then import the file into an empty clipart box. This example uses the Sample Clipart, which has nice backgrounds from spots 13 to 24. Type this in the Command Centre:

```
home          Bring the turtle to the centre of the page.  
setshape 17   Use the number that corresponds to the clipart spot of your choice.
```

If the image does not fill the page, you can use commands such as:

```
setsize 60
```

You can also drag the turtle around to frame the image as you like. Yes, the image is a turtle with a huge shape. Now...

```
stamp
```

The turtle stamps its shape as an image on the background of the page. [However, the turtle is still present, with its huge shape.](#) Drag the turtle around. You will see a nice background fixed or stamped in the work area *and* a turtle that is moving around with the same background.

You probably want to get rid of the second background shape that moves with the turtle. Here is the solution:

```
setshape 0
```

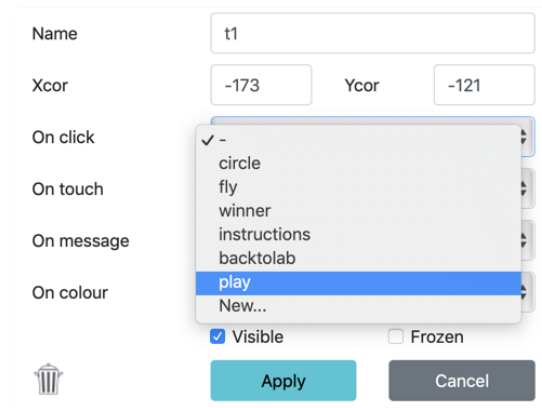
Set the turtle back to its normal shape. Now you have a normal turtle and a stamped image in the background.

# More about turtles: Making a clickable turtle

A clickable turtle is much like a button, with a nicer shape (if you added clipart to it!). Making a clickable turtle is a three-step process:

1. Create a procedure for what the turtle should be doing when you click on it. If you are not sure yet, just create an empty procedure such as this:
2. Create a turtle and right-click on it. In the turtle's dialog box, click on the On Click menu and choose the procedure that you just created, then click Apply to close the dialog box.
3. Now complete your `play` procedure by adding some instructions. Next, click on the turtle to test your procedure.

```
31 to play
32 end
```



## Different examples of what you can instruct the Turtle to do with a simple click

Make a clickable turtle to turn pages. Start by adding Sample Clipart and give shape 51 to the turtle. Follow the other 3 instructions directly above. And have a 2nd page!

```
to continue
page2
end
```



Make a clickable turtle to start a game. Note: this "GO!" button is an imported pic.

```
to start
everyone [clickon]
end
```



A clickable turtle that displays instructions. This "Question Mark" button is also an imported pic. Assuming that `text1` is the name of an existing text box containing the instructions:

```
to instructions
text1, showtext wait 30 hidetext
end
```



# More about turtles: Colour detection

You use colour detection when you want a turtle to do something when it moves and passes over a certain colour. You can use colour detection to create a beep sound, force a bounce, go to the next page, declare a winner in a race, etc.

Programming a turtle for colour detection is a five-step process.

1. Add a turtle that's going to be moving - the one that is going to perform the colour detection.
2. State that it is going to perform a colour detection: open its dialog box, and in the On Colour drop down menu choose New, and click on Apply.

3. This creates, in the Procedures Pane, a procedure such as this one:

```
to t1_oncolour :prevColour :newColour
; Use an instruction like this to do colour detection every time.
; Moving from any colour to red, even red to red, will trigger the action.
; Pick your own colour name and instructions instead of [bk 10 rt 180]
; if :newColour = "red [bk 10 rt 180]handle the event
end
```

4. The lines starting with a ";" are just comments describing the process. Read the comments and delete them if you wish. At this point, you want to know "what is the new colour" that will trigger an event. Try this:
5. Draw a region and colour it. For example:

```
cg pu setx 200 pd fd 2000
```

Clear the graphics, pen up, set the x coordinate at 200 (move to the right), put the pen down, and forward a lot to make a vertical line that splits the page.

```
pu setx 210 setc "red fill
```

Pen up, move a bit to the right, set the colour to red and fill the area.

6. You must place the turtle "not on red", and make sure it is pointing towards the red area. Now type this in the Command Centre:

```
setc "black
```

The Turtle is black again so you can see it

```
pu
```

Put the pen up.

```
home
```

Places the turtle at the 0 0 position

```
setheading 90
```

Point towards the red wall.

7. Now that you know the turtle will detect the colour red, it's time to complete the procedure that was "prepared" in Step 3 above. In this example, I have removed the comments:

```
to t1_oncolour :prevColour :newColour
if :newcolour = "red [back 10 right 180]
```

# More about turtles: Colour detection (cont.)

`end`

8. Final Step: Get the turtle moving! Type this in the Command Centre:

`forever [fd 2 wait 1]`      Go hit the wall.

The turtle will bounce when it touches the red area, then continue moving and bounce again when it touches red.



This example uses "red as the colour to be detected. These are other colour names you can use: `black`, `blue`, `brown`, `cyan`, `gray`, `green`, `lime`, `magenta`, `orange`, `pink`, `red`, `sky`, `turquoise`, `violet`, `white`, `yellow`.

In short: if you are drawing a target yourself, use `setcolour "red` (or any other colour name) and `fill` to draw the target, and then use that colour name in your colour detection procedure.

Important: If you are importing a background image, you should use a *solid colour* for the colour that the turtle will touch and detect.

# More about turtles: Collision detection

Collision detection is similar to colour detection, at least for the first steps:

1. Add a turtle that's going to be moving - the one that is going to collide with another turtle.
2. Instruct it to perform a collision detection: open its dialog box, and choose New in the **On Touch** drop down menu, and click on Apply.
3. This creates, in the Procedures Pane, a procedure such as this one:
4. The lines starting with a ";" are our comments. The name of the procedure (**t1\_touch**) means that this procedure is applicable to the turtle named t1. That's the one that has to be moving. The variable (**:touchedturtle**) lets you have different things happen, depending on "which turtle" is hit.
5. Create another turtle and find out its name (right-click on it). That is the turtle you want to touch / hit.
6. Assuming you have 3 turtles in the work space, complete the procedure that was "prepared" in Step 2 above:

```
7 to t1_touch :touchedturtle
8 ; Here you describe what does the turtle do when
  it touches another turtle.
9 ; The variable :touchedturtle contains the name
  of the other turtle.
10 ; Example: say word 'Hello! ' :touchedturtle
11 end
```

```
to t1_touch :touchedturtle
if :touchedturtle = "t2 [say "hi!]
if :touchedturtle = "t3 [stopall]
end
```

Now, make sure you write code to cause t1 to touch t2 and t3.

In the Terry Fox sample, Terry is obviously a turtle, and the rock, on the right hand side, is also a turtle named "rock". Terry is programmed for this collision detection:

```
to terry_touch :touchedturtle
if :touchedturtle = "rock [stopall]
end
```



Name	terry	
Xcor	295	Ycor
On click	-	
On touch	terry_touch	

# More about procedures

You have seen procedures all over this PDF guide. You know almost everything about them, here's a recap:

- A procedure must start with the word **To**, followed by a space, and the name of the procedure. The name must be a single word with no spaces. These are good names: `start`      `TurnAround`      `turn_around`.
- A procedure is completed with the word `end`, all by itself on the last line.
- You can (and should) add comments before or inside your procedures. All you have to do is to start a line with a semi-column (;). Check our samples!
- A procedure may have an input, such as this one containing a variable:

```
to square :size
repeat 4 [fd :size rt 90]
end
```

In this case, you can't use just `square` as a command. Like `forward`, you HAVE to indicate "how much". You must use this procedure like this:

```
square 50
or
square 100
```

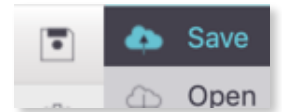
- When a procedure is created, you can use it:
  - In the Command Centre
  - In a clickable turtle
  - In a button
  - Even in another procedure!
- For colour and collision detection, Lynx will prepare the procedure for you, as described in previous pages.
- To better organize your procedures, you can create several Procedures tabs. Click on the arrow to create a new tab, then click on the arrows to switch from tab to tab.
- You could think about organizing your procedures in a different tab for each page and giving each Procedures Tab a meaningful name.



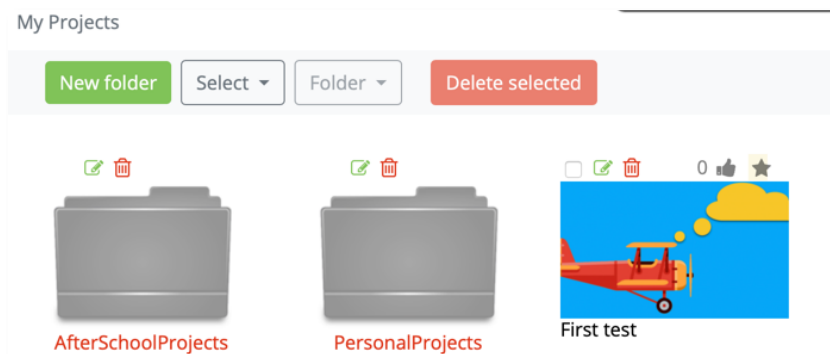
# Saving and retrieving your project

**Important:** There is **no autosave**, not even when you leave your project to open another one, or when you close or you leave your browser window. *Save your project as you are working on it*, and **before** leaving the project editor.

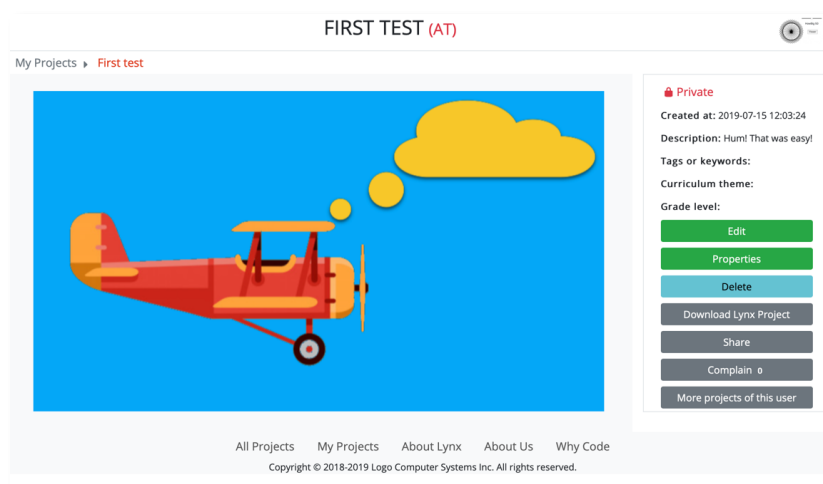
Saving is easy! You gave a name to your project when you created it. As you work, simply click on the Disk icon regularly and choose Save in the menu.



When your project is saved, you can go back to [www.lynxcoding.org/projects](http://www.lynxcoding.org/projects) (create a bookmark for that page, that's your Lynx personal place in the cloud). Here is an example, with two folders and a project. Projects have a generic icon until you give them a preview image.



When you want to work again on your project, come back here and click on your project (Look inside a folder if you've put it there). You will see your project in **PLAY MODE**. If your project has buttons and clickable turtles, you can play with it right here.



# Saving and retrieving your project (cont.)

## ABOUT THE BUTTONS ON THE RIGHT

Click on **Edit** to open this project in the editor (and keep working on it).

Click on **Properties** if you want change the name of the project, move it into a folder you have created, set keywords, or give it a preview image like the example above. To create a **Preview** image, make a screen capture of your project and save it on your computer. Then use the Browse button to upload it and scroll down and Save the change.

Click on **Delete** to remove (forever) this project from your personal space.

Click on **Download** if you want a local copy (on your computer). It will be called **ProjectTitle.js**, probably in your Downloads folder. Note: *Some* email servers will reject an attachment with a js extension.

Click on **Share** to obtain a link which you can copy and paste into an email or elsewhere. There is also a button to post the project in your Facebook account.

# Sharing your project with your friends

There are several ways to share your Lynx project, from within the Lynx editor, or from your Lynx personal space in the cloud.

## FROM THE LYNX EDITOR

To share a project from within the Lynx editor, simply click on the Share icon in the top-left corner of the editor.



In the dialog box that comes up, you can choose an image file to use as a preview (use the buttons to fetch a file from your computer/device, OneDrive, Dropbox or GoogleDrive). Enter a title and a description, then click on one of the Share sites: Twitter, Facebook, or copy the link to paste it where needed, or click on E-Mail to send the link by email.

Share project...

Preview

My device OneDrive Dropbox Google

Title: Zig and Zag

Desc.: I changed all the parameters for Kaleidoscope

☐ Private

URL: https://lynxcoding.org/share/DHDAommenD

Twitter Facebook Copy link E-mail


## FROM WITHIN YOUR LYNX PERSONAL SPACE IN THE CLOUD

From your Lynx personal space, click on your project to open it in **PLAY MODE**. Then click on Share to obtain a link which you can copy and paste into an email or elsewhere. There is also a button to post the project in your Facebook account.

## ENJOY (AND EDIT) MY PROJECT

Before sharing a project, go to its Properties in your Lynx personal space in the cloud, and uncheck the **Private** check box and scroll down and **Save** the change. Next, look for the **Share** Button and Copy the Link. Then send the link to a friend. Using the link, he / she will be able to make changes to your project but it will be saved, by your friend, under a new name. Your original Project will be preserved as it was.

# List of most common primitives

Remember: you will see them all in Learner Mode Help (click on the  in the bottom-left corner of the Lynx Editor), or for a quick description and example, just type the name of the primitive in the Command Centre or in the Procedures Pane, and leave your mouse pointer over the name of the primitive.

```
repeat
  REPEAT number list-of-instructions
  Runs the list of instructions the specified number of times.
  repeat 90 [bk 40 fd 40 rt 4]
```

## TURTLE MOVEMENT

forward (fd)	back (bk)	right (rt)	left (lt)	home
glide	setheading (seth)	setpos	setshape (setsh)	pos
setx	sety	xcor	ycor	

## TURTLE STATE

st	ht	setsize		
----	----	---------	--	--

## TURTLE DRAWINGS

pendown (pd)	penup (pu)	penerase (pe)	clean	cleargraphics (cg)
setpensize	setcolor (setc)			

## TEXT STUFF FOR TEXT BOXES

print (pr)	insert	cleartext (ct)		
------------	--------	----------------	--	--

## OTHER TEXT STUFF

announce	question	answer	show	say
----------	----------	--------	------	-----

## OTHER STUFF FOR MAKING THINGS HAPPEN

if	ifelse	repeat	wait	forever
stopall				

## OTHER STUFF FOR TURTLES

everyone	ask	talkto (tto)	clickon	clickoff
----------	-----	--------------	---------	----------

## OTHER "RANDOM" STUFF

random	pick			
--------	------	--	--	--