Teacher – Parent Manual

Thank you for participating in our Scratch Course for Teachers (Scratch), made by Delft University of Technology!

This manual provides you the information you need to successfully start but also finish this course, together with your children, nephews, nieces or students. As of now, we will say “kids” to cover all these target audiences. With this, we mean any kid you can help learning how to program in Scratch!

We will begin with explaining how this course material can be used throughout these classes. After that, we will introduce edX to you, the platform used for these online courses. After that, we will cover the concept of programming with (and without) Scratch. Finally, the last chapter covers all information which has been thought in this course material.

How do I use Scratch?

All course material is developed by Delft University of Technology, for children. Video’s and quizzes should be understandable for children with an age of 8 or older.

Videos are short and aimed to encourage children to experiment with coding themselves. After a quiz, the next video will elaborate on what was being discussed during the quiz and how to proceed with the classes. Each video provides an explanation on what the correct answers are and points them towards additional / earlier course material where needed.

As a teacher or parent, you do not have to do a lot within this course. What is necessary is as follows:

- Create an account on edX for each student, “enrol” each student for this course. EdX was not intentionally made for children. It is not an intuitive website
- Install Scratch or go to scratch.mit.edu (more information at “Scratch”).

What would be handy, but not mandatory:

- Play around in Scratch a little bit. You could watch one of the videos in this course for Kids, or experiment on scratch.mit.edu.
- Get familiar with a few programming concepts (more information at “Programming”).
- Read the course overview for you to know what the kids will learn (more information at “Scratch”).

Attention! This is a message for parents or teachers who are already known with programming: This course material is not only specially made for children who need to learn how to program, but also how to reflect their own work and how to code neatly. Some videos hardly contain explanations on how to code but go more in-depth in how children can organize their work in a clear way. Please, as an adult, do not interfere in this, as we will elaborate on these matters in a later stage.

Next to the video's, there is additional **course material** on a weekly basis. These can be used instead of the videos. There are similar exercises in this course material, but, as a student you cannot receive feedback in the way one can receive this in the online environment. Please keep this in mind should you decide to use this offline course material and provide more help to the student where necessary.

edX

For this MOOC, we make use of edX, an online educational platform. The Delft University of Technology (TU Delft) has made a strategical agreement with edX. This platform is an initiative of the American universities Harvard and MIT.

edX's main target audience are actual adult 'learners' as they call it. Publishing a MOOC for children is something quite new. Should you have any feedback, please let us know!

There are a few reasons for us to make use of edX rather than writing a book or publishing some videos on YouTube for example.

Out of experience in programming education, we noticed that it is hard to grasp what children understand from this and what not. With trial and error, children get these programming blocks to work eventually, but, we wonder whether they understand the concepts and deeper thoughts behind it. So, to investigate whether this logical reasoning is being understood, we post little quizzes after (almost) every video as well as some puzzles or programs for them to

practice with. This way, children test their own knowledge, but you as a teacher or parent will also get an idea with what the child has difficulties (or with what not).

We also encourage you to please create a separate account per child. This way each student can learn in their own pace.

Our MOOCs are part of a long-term research by TU Delft, to investigate children's ability to learn a programming language. This is another reason why we use Scratch, as we can analyse by the quizzes where they score high and where they score low. We learn from that data! What concepts are easy to understand? Which one are more difficult?

We will never publish any data without your consent.

Creating an account

Before you can register, you must make an account on edX. As explained in the above, we prefer you create a personal account per child.

Working with the materials

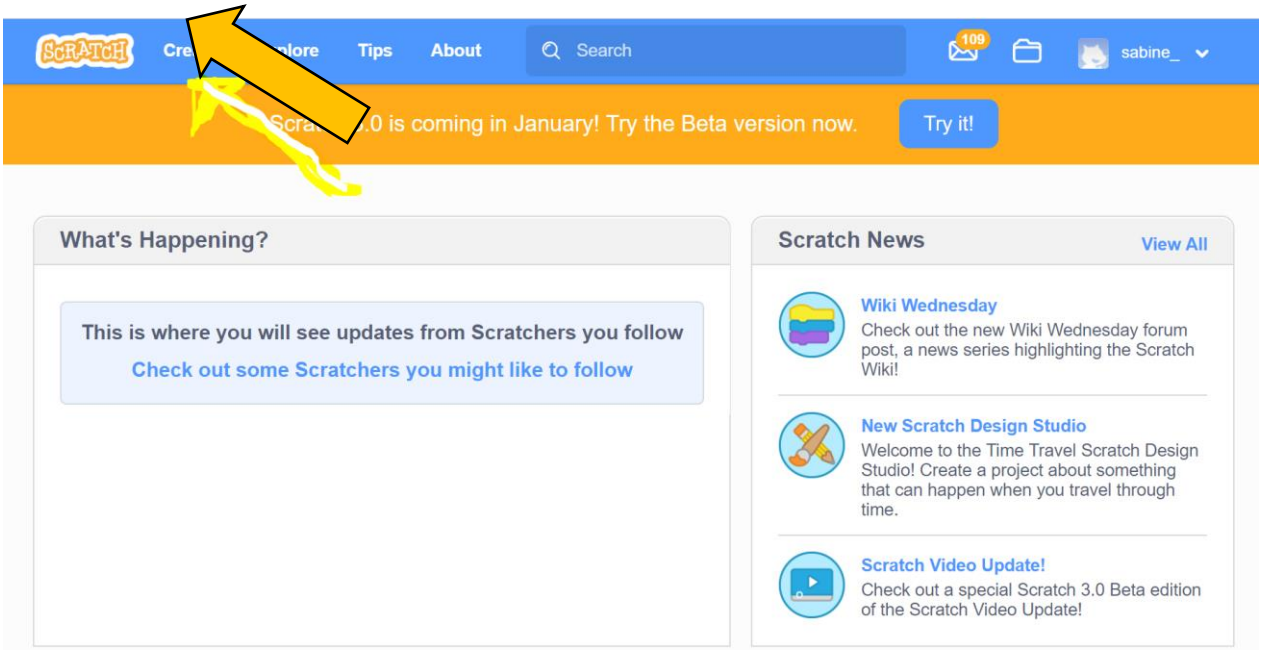
The materials provide via this MOOC are through videos, links to Scratch and sometimes additional materials, such as .pdf files.

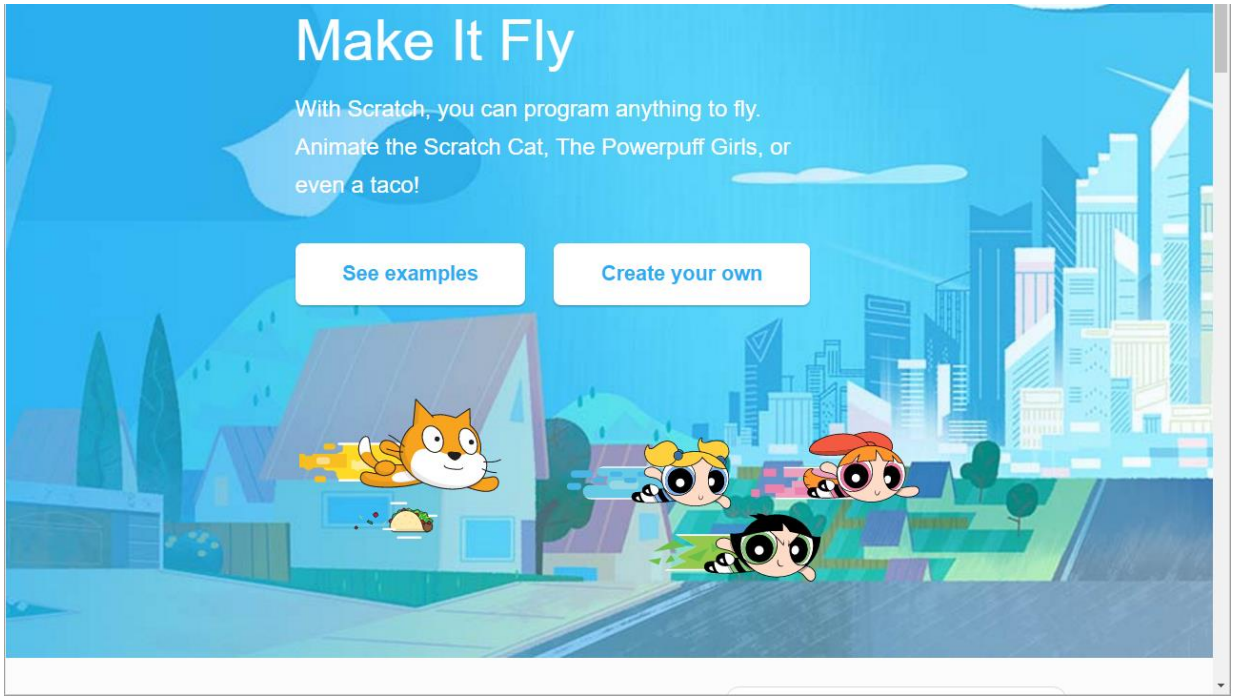
Online discussion forum

Should you or a student have any questions, these can be posted on our forum, called the Discussion section.

Scratch

Scratch, made by Massachusetts Institute of Technology (MIT), is a programming language especially developed for children. Kids can easily make programs, games and animations with this. The interface is colourful and invites kids to try and experiment themselves.





Starting with Scratch

You don't have to install anything to start with Scratch, you can start programming directly via the browser. There is an offline app, in case your laptop or the computer of the student does not have internet. You can download the offline version by clicking on this link.

In the video lecture, we provide links to the website version. You will go to this website version by clicking on "create" as the arrow points at in the image above.

Scratch 3.0

The newest version of Scratch will be launched in January 2019. Already a beta version is published, for you to get familiar with it. We used Scratch 3.0 in the Scratch: Programming for Teachers edition. Scratch 3.0 differs in terms of interface a bit from Scratch 2.0, and it also has very nice new features.

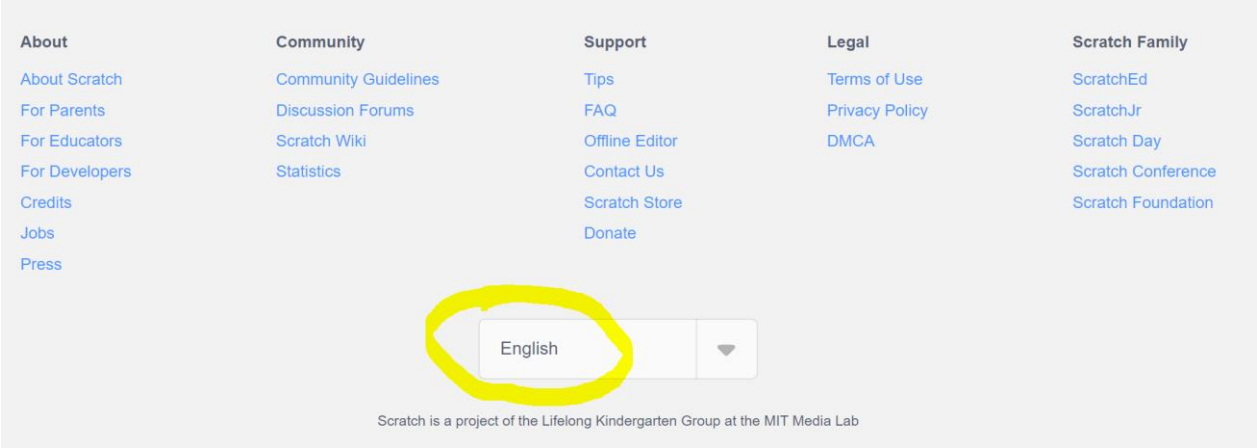
The concepts are of course the same, so, in case you want them to work with the latest version, you could also encourage the kids to work in the beta Scratch 3.0 version.

Please bear in mind that this beta version doesn't support saving project yet, so, let this know to the students!

Language settings

Make sure that the language is set to English. You can do this manually, in case a different language is displayed when opening Scratch.

For this, you scroll all the way down on the homepage. See the image below to see the button in which you can adjust the language:

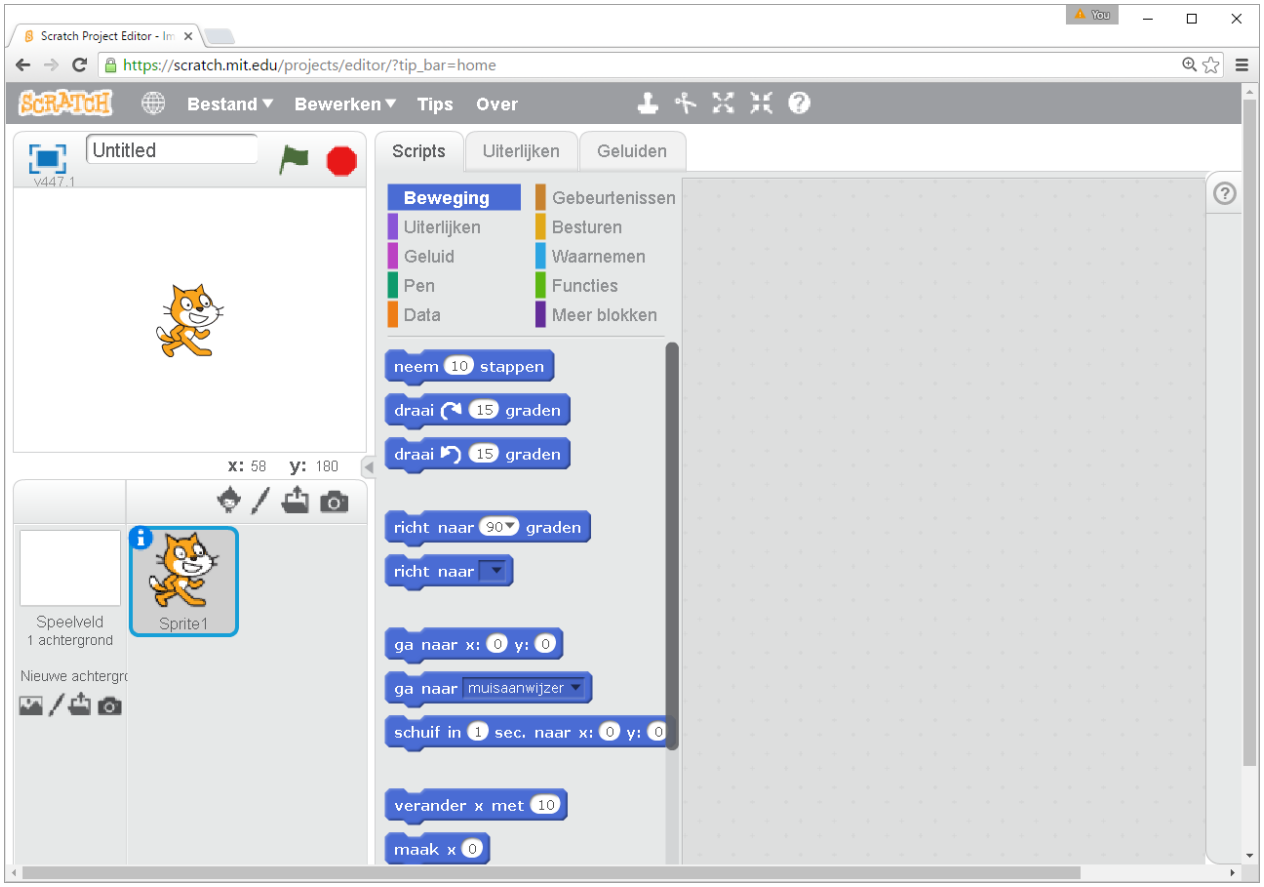


In the following paragraphs, we will elaborate on some of the concepts and terminology of these video lectures and relevant to the Scratch programming language.

Sprites

Programming in Scratch means meeting some “sprites”. Sprites are the animations – so little animals or fantasy cartoons. In the videos, these things are called either “sprites” or by their names (like “Giga” or “Devon the Dinosaur”).

After you’ve clicked “create” on the Scratch website, a new program will open in which you can code. The default cartoon (sprite) will be the lovely cat. This is also the logo of Scratch. In our course, we refer to the cat as Carl.

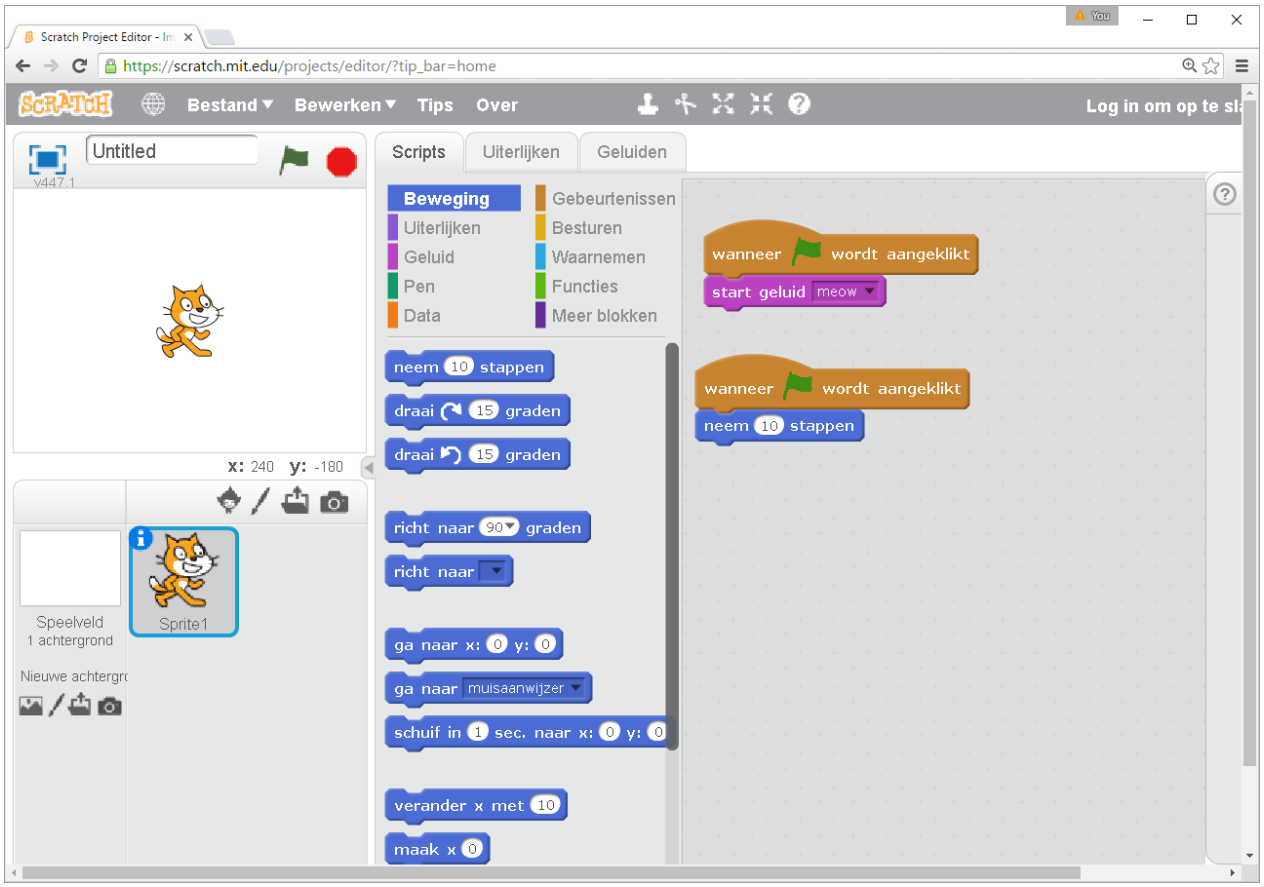


A program has many sprites. Scratch has a big library of sprites, but students can also upload their own images! We will show them how they can do this in one of our videos.

You can also change the background of the program. This has its own coding area, so the student can adjust this background as well.

Scripts

Every sprite can have its own piece of code, which we call a script. For example: a script for making sounds per sprite. Or a script for one sprite to make it move. All scripts are “parallel”, so they will be run at the same time.



Programming

As a teacher or parent, you might wonder “what exactly is programming?”. Your students might ask you the very same question. There are many possible answers to this question.

The simplest answer to this is according to us, is: to make a computer do what you want it to do. Unfortunately, computers don’t really speak human language, so we must come up with a special computer language.

Scratch is a programming language for children, but this doesn’t mean it is not a strong, well-developed language. All “adult” programming languages contain the same important programming concepts which Scratch also covers.

In this chapter, we will cover some of these important programming concepts. We do this, so you can talk along with the students when they ask questions about certain concepts, or to just have a discussion with them about it.

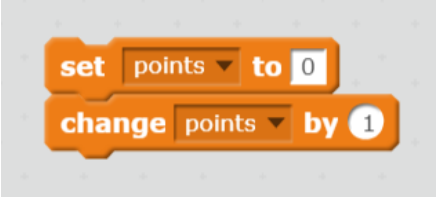
There is nothing wrong with putting this manual down now and watch the videos yourself, do our quizzes and the homework! This chapter won’t only cover Scratch, but also Python. Python is a programming language for professional programming. There even exist a book for children about Python!

I will show you some examples for both languages, so you can spot the differences easily.

Variables

Variables is a tiny storage for data. Think of a bank account, on which a certain number are stored. Or a white board on which you can write down the name of a kid that is his or her birthday. You can write and read a variable.

The simplest example of a variable in the Scratch lessons, are the amount of points in a game. Imagine that if you would get a coin, this will give you points. This doesn’t have to be a number per se, we can also store this as text in a variable.



This code makes “points” for 0 and raises it with 1.

In Python, we would code it like this:

```
points = 0
points = points + 1
```

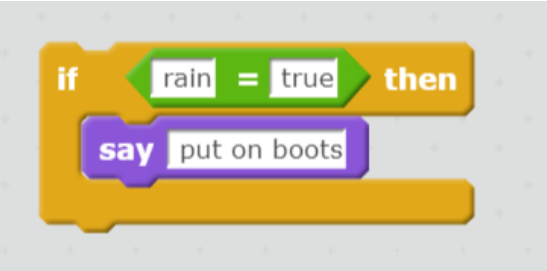
If-then-(else)

One of the central concepts in programming is the “if-then” function. If “X” then “Y”. You can let your students think about this also without any computer involved (we would call this working *unplugged).* We will be doing this in one of our quizzes!

If it rains, I will put on my boots.

If I get a baby brother, I will paint his room blue.

In Scratch, we would code that like this:



Sometimes, the *if-then* will be combined with an *else:* **If it rains, I will put on my boots, else, I will put on my sandals.**

In Scratch:

In Python, it would look like this:

```
if rain:
  print (‘Put on boots.')
else:
  print ('Put on sandals.')
```

Loops

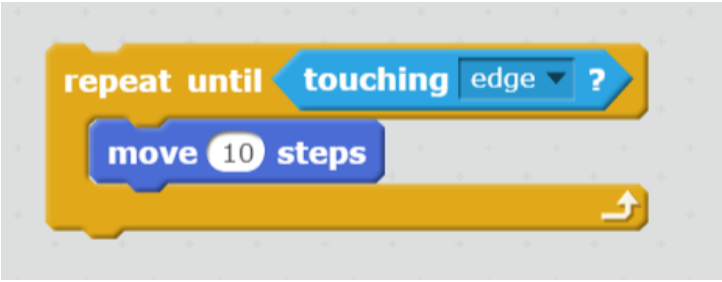
A loop is a piece of a programming language that repeats other assignments the computer gets via code. In the videos we will either address this as “repeat”, “forever” or just as *loop*. In its simplest form, a loop without another condition:

In Python it would be coded like this:

```
while True:
  move (10)
```

This piece of code will make sure the sprite of relevance will endlessly move, in theory of course. In practice, the sprite will move until it touches the edge and then it will get “an error”, because it cannot move anymore.

You can also program it in such a way, that it contains another condition. A “repeat until…” for example. Below, we will show you how that could look:



Following this code, the sprite would move until it would touch an edge. After that, it will stop moving.

Loops can occur also in daily life. For example, “walk straight until you are at the crossover”.

Python does not contain a “repeat until”, but for this it would use a “while” function. That would look like this:

```
while touching border <> true:      ← the <> means “not equal to”
  move (10)
```

Note: the following concepts are a bit more Scratch-specific than the one in the above

Events

Scratch is “event-based”. This means that the programs made in Scratch can react to events in their surroundings. For example: when the green flag is clicked, or when the space bar is being clicked. Every *script* needs to start with an event-block. You can recognize these by their “hat” shape:

Signals

When sprites want to communicate with each other, you can do this by the function “signals”. For example, when a sprite wants to let inform the player that the game has finished.

Cloning

If you make use of a lot of sprites with the same functionalities, for example the tiny ghosts in Pacman, then it would not be handy to code each script per ghost. That costs a lot of effort! Scratch supports a “cloning” function, so basically a copy/paste of a sprite’s code.