Exercise 1: Introduce Yourself

Creative Programming 2026

This exercise serves as an introduction to coding by presenting yourself in a short p5.js sketch. The first 4 parts cover what was introduced during the lectures. Keep in mind that the second half of this exercise (part 3 onwards) sheet builds on what was discussed in the second lecture.

- **Time:** 12:15-16:00
- **Room:** 5A14-16
- **Teaching Assistants:**
  - Axel Døring
  - Kirstine Lund Hansen
  - Line Larsen
- **Hand-in:**
  - Sunday the 8th of February 20:00 on LearnIt
  - Submit the sketches with clear labels on which sketch belongs to what exercise step.

1 Concept of Coding

Algorithms are well-defined procedures with a set of instructions to complete a task or solve a problem. They usually take an input and transform that input into an output. They can be simple or complex depending on the output. Algorithms do not only exist in mathematics or programming, but also in real life.

1. For this first part of this exercise, pick only one of the following prompts and write down an algorithm to complete the task:

- Pick your favorite dish and write down its recipe line by line. Include each small steps such as “turn on the tap,” “fill the pot with water,” “turn on the stove,” or “heat up until the water boils.”
- Describe your morning routine step by step until you arrive to ITU.

- Once you are in ITU, describe each step of a common activity. For example, taking the elevator to come to classroom or ordering coffee in Analog.

2. Write down each step as an instruction in pseudo-code.
3. In Processing, write down the steps using multiple `console.log()` so that your list shows up in the console, line by line.
4. Now draw them on the canvas using `text()`. Don't forget to pass `x` and `y` coordinates!
5. Style your list! Try to experiment with:
   - `background()` for background colour
   - `fill()` for text colour
   - `textSize()` and `textFont()` for further text styling.
6. What happens when (differently coloured) lines overlap? Which one ends up on top?
7. Save the final sketch. We will come back to it in part 3 of this exercise.

2 Drawing

In this part, we'll start with a fresh sketch and experiment with drawing some shapes.

1. In a fresh sketch, set the background colour to something distinct (using `background()`).
2. Draw a rectangle (`rect()`) somewhere on the canvas.
3. Now add an ellipse using `ellipse()`.
4. Add a few more of either. What happens when shapes overlap?

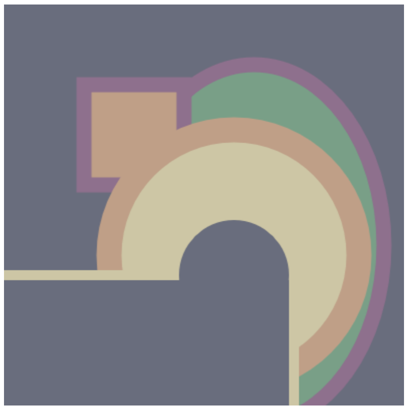


Figure 1: By overlapping shapes and using colour creatively we can create interesting complex shapes.

5. The sketch does not need to be still. Randomise some of the coordinates using `random()`.
6. We can add some basic interactivity by using the position of the mouse. Draw a shape using the mouse coordinates (`mouseX` and `mouseY`).

7. Let's try to make a nice logo or painting!
    - `fill()` can be used to change the colour of shapes (similar to text)
    - `stroke()` controls the colour of the stroke (line) around shapes
    - similarly, `strokeWeight()` controls how thick that line is
    - and don't forget to adjust your background and shapes accordingly!

3 Loops and Arrays

From this point forward, the exercise focuses on the material covered in Lecture 2. We will now continue with the sketch we made in part 1, using what we have seen in the second lecture to enhance the sketch.

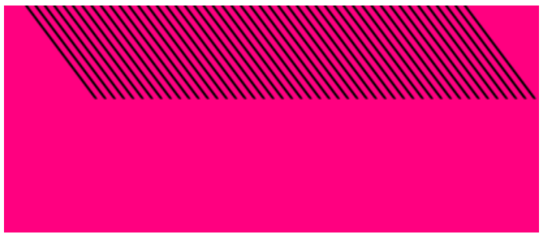
1. Load the sketch from part 1.
2. In part 1, we drew a bunch of text by repeatedly writing lines with `text()` and the string that we want to write. Define all of the text in an array definition, and then write the text on the canvas by accessing the individual lines, as shown below:

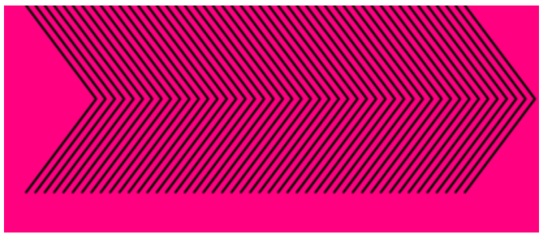
```
...
text(text_array[0], ...);
text(text_array[1], ...);
...
```

3. Randomise! Instead of drawing all lines at once, draw a random line with random styling (such as `fill()` and `fontSize()`) on a random position every time the `draw()` function is called. HINT: Too fast? You can control the speed by adjusting `frameRate()`.

4 Patterns

(Nested) for-loops are very powerful and can create some amazing visuals with few lines of code. In this exercise, we will work on designing and repeating simple graphics to create patterns.


(a) Possible result of step 1.


(a) Possible result of step 2.

1. By using for loops, start by repeatedly drawing 30 lines on the canvas, such as shown in the figure. HINT: Keep the lines and the distance between the lines shorter. Say, less than 1/10th of your canvas dimensions.
2. Next try to add lines below the first line in the opposite direction.
3. Now, repeat the two lines (try copy pasting it first) several times by offsetting the lines start and end values. Try to see the patterns of repetitions.
4. How do increases in each of the arguments (x, y, w, h) affect the result? What should be the increments of each iteration?
5. Try to create repetitions using loops, instead of copy-paste. One of the loops will be embedded in another loop, a nested loop. Nested loops create multi-dimensional iterations. See below for an example.

```
for (let i = 1; i < 6; i++) {
    let row = [ ];
    for (let j = 1; j < 6; j++) {
        row [i][j] = (i * j);
    }
    console.log(row);
}
```

HINT: The iterations multiply by the loop size, i.e. `5*5 = 25`. The code will be executed 25 times.

6. Play with arguments of the line function and the iteration values and see how uneven lines appear in a pattern. Use different offsets for the line's starting and end point coordinates.
7. Swap out the lines for rectangles or ellipses to generate interesting patterns, or even mix some of them. If you generate something interesting you could use that as the background for your final sketch.

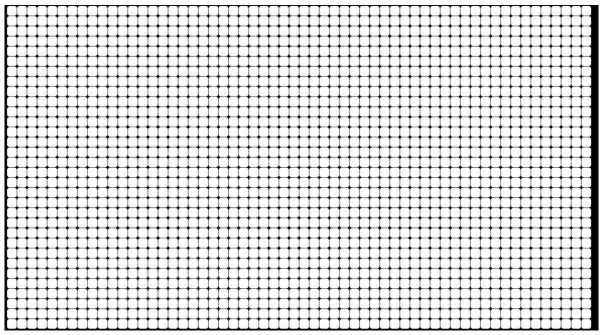


Figure 4: A basic pattern composed of square rectangles.

5 Hello World!

In this final part we will make a card that introduces ourselves. Combine the previous techniques to create something interesting. Think of what you wrote down about yourself in the first lecture as inspiration. Some ideas:

- Combine shapes and text to make interesting visuals. Draw objects using shapes representing your life, views and/or hobbies.
- Use patterns to get a striking background.
- Add some interactivity with mouse tracking. You can also create some 'stereoscopic' visuals by adding a very small portion of the mouse coordinates (e.g. `mouseX/100`) to shapes.
- Use a colour palette to create a strong identity (e.g. https://coolors.co/).