LYRE

composing code

code

usercode

https://assemblyre.glitch.me
Bauhaus-Universität Weimar

The educational materials presented in this zine are excerpts from the book *Aesthetic Programming: A Handbook of Software Studies* by Winnie Soon and Geoff Cox published by *Open Humanities Press 2020*. This zine is licensed under the Creative Commons Attribution. (HTTPS://CREATIVECOMMONS.ORG/LICENSES/BY-SA/4.0/)

Feel free to visit their site (HTTPS://AESTHETIC-PROGRAMMING.NET) to see the full course alongside practical programming tasks using the p5.js framework.

preface(){

Thank you for participating in this study on teaching computer science principles. In the following, you will encounter text fragments about programming and its societal and political implications. This zine is yours. Hence, you may highlight or comment the chapters to your liking. At the end of each chapter, you will find a section for your notes and the chapter's KEY CODE which will look like this:

Visit HTTPS://ASSEMBLYRE.GLITCH.ME to enter your user code from the front page to enter your personal course hub. With each code, you will unlock digital resources and an online editor to work on your personal PSEUDO-PROGRAMS. There, you will SUBMIT ONE PROGRAM PER CHAPTER.

Pseudo-Programs: In this course, you will not be coding in a conventional way. You will be provided with a graphical editor which will allow you to write and customize texts and images. Your task is to write programs that would represent your ideas regarding the chapter. You may come up with your own syntax, add related images, or write a plain text to express yourself. In the end, you will be asked to reflect on your work. Pseudo-Programs do not execute code, but convey its concepts and ideas. You will find more information on that later.

If you happen to have issues with your task, feel free to contact me. Happy coding!

}

setup(){

It has become commonplace to include programming in educational programes at all levels and across a range of disciplines. Yet this still remains relatively uncommon in the arts and humanities, where learning to program does not align explicitly with the related career aspirations. This raises questions about what does or doesn't get included in curricula, why this may be the case, and which knowledge and skills are considered essential for some subjects and not others. Certain forms of privilege (related to class, gender, race) are clearly affirmed in these choices. For instance, in very general terms, "high culture" has traditionally been described as the domain of university-educated (wealthy, white) people, whilst "low culture" the domain of non-university-educated (working class) ordinary people. Neither high nor low culture, programming cuts across this class divide as both an exclusive and specialized practice that is also one rooted in the acquisition of skills with applied real- world use in both work and play. Yet, despite its broad applicability, access to the means of production at the level of programming remains an issue all the same.

[Learning] to program allows us to think in new ways by introducing different methods and perspectives to raise new questions; programming offers us a better understanding of culture and media systems, subsequently allowing us to learn to develop better, or better analyze, cultural systems; and, finally, programming can help us improve society by creating, designing, and discovering programs. […] [We] see this as a means to

open up different working methods, and, using programming as a basis for our thoughts, to speculate on alternative forms and political imaginaries of programming practice.

[…] [The] argument for programming or coding as a necessary skill for contemporary life seems indisputable, and there are plenty of examples of initiatives related to computational literacy and thinking, from online tutorials to websites such as Codecademy and Code.org. [...] Vee's Coding Literacy also explores these connections, arguing how the concept of literacy underscores the importance, flexibility, and power of writing for and with computers.[1] An important aspect of this is that, not only does this help us to better understand the social, technical and cultural dynamics of programming, but it also expands our very notion of literacy and its connection to a politics of exclusion (as with other non-standard literacies). Furthermore, and given that programming, like other forms of writing, performs actions, it presents itself as a way to reconceive politics too: not simply writing or speaking, arguing, or protesting in public, but also demonstrating the ability to modify the technical processes through which the action is performed, in recognition of the ways in which power and control are now structured at the level of infrastructure.[2]

}

setup(notes){

}

This is your code for additional digital resources.
Visit your personal hub to start coding!

seed

data(){

Variables are used to store data and information in a computer program. You can think of variables as a kitchen container, in which you can put different types of things (like food, kitchen utensils, etc.) in a given container, replace them with other things, and store them for later retrieval. There are two main types of variables: "local variables" that are defined within a structure or a function, can only be used within that block of code; and "global variables" that can be used anywhere in the code. Global variables need to be defined before the setup of the program, usually in the first few lines of code.

Another reason for using variables is that if you have longer lines of code, it is easier to have all the variables that you have declared for the program in an overview. If a variable is used in different parts of a complex program, you can simply change the value of the global variable instead of changing the multiple parts in the entire program, and this is useful for testing/refining the program without locating specific and multiple lines of code for modification. This leads to the reusability of variables. Variables can be used in different functions and more than once.

[In] the era of big data, there appears to be a need to capture data on everything, even from the most mundane actions like button pressing. Moreover a button is "seductive,"[3] with its immediate feedback and instantaneous gratification. It compels you to press it. Similarly in software and online platforms like Facebook, a button calls for

interaction, inviting the user to click, and interact with it in binary states: like or not-like, accept or cancel. The functionality is simple — on or off — and gives the impression of meaningful interaction despite the very limited choices on an offer (like most interactive systems). Indeed this binary option might be considered to be more "interpassive" than interactive, like accepting the terms of conditions of a social media platform like Facebook without bothering to read the details, or "liking" something as a way of registering your engagement however superficial or fleeting. Permission for capture data is provided, and as such our friendships, thoughts, and experiences all become "datafied." Even our emotional states are monitored when it comes to the use of emoticons […].

[Datafication] — a contraction of data and commodification — refers to the ways in which all aspects of our life seem to be turned into data which is subsequently transferred into information which is then monetized.

At the moment, the most widely used web analytics service is provided by Google and contains tremendous amounts of data on website traffic and browsing behavior, including the number of unique visits, average time spent on sites, browser and operating system information, traffic sources and users' geographic locations, and so on. This data can then be further utilized to analyze customers' profiles and user behaviors.

Smart devices like our computers, phones, and other gadgets are commonly equipped with voice recognition — such as Siri, Google Assistant or Alexa — which turns audio input into commands for software, and feedback with more personalized experiences to assist in the execution of everyday tasks. You can find these voice assistants in just about everything now including, everyday objects like microwaves, and they become more and more conversational and "smart," one might say "intelligent," as machine learning develops. These "voice assistants," as they are known, carry out simple tasks very well, and become smarter, and at the same time capture voices for machine learning applications in general. Placing these tangible voice assistants in our homes allows the capturing of your choices and tastes when not facing a screen. In the internet of things, the device serves you, and you serve the device. Indeed we become "devices" that generate value for others.[4]

Fitness and well-being becomes datafied too, and with the setting of personal targets, also "gamified." As the welfare state is dismantled, personal well-being becomes more and more individualized and there is a growing trend for "self-tracking" apps to provide a spurious sense of autonomy. Movement, steps, heart rate, and even sleep patterns can be tracked and analyzed using wearable devices such as the Fitbit, or the Apple Watch. These practices of the "quantified self," sometimes referred to as "body hacking" or "self-surveillance," overlap with other trends that incorporate capture and acquisition into all aspects of daily life.

}

data(notes){

}

This is your code for additional digital resources.
Visit your personal hub to start coding!

sprout

loops(){

The core concept of a loop is that it enables you to execute a block of code many times. For example, if you have to draw one hundred lines that are placed vertically one after the other, you can of course write one hundred lines of code […].

A "for-loop" allows code to be executed repeatedly, and so provides an efficient way to draw the line one hundred times by setting up a conditional structure, counting the number of lines that have been drawn and counting the maximum number of lines. […]

To structure a for-loop, you need to ask yourself:

- What are the things/actions that you want to repeat in a sequence or pattern?
- More specifically, what is the conditional structure and when do you want to exit the loop?
- What do you want to do when this condition is, or is not, met?

Conditional structures are very useful as they allow you to set a different path by specifying conditions. Indeed, a conditional decision is not specific to programming. For example, in everyday life, you might say

"If I am hungry, I should eat some food, if I am thirsty, I should drink some water, otherwise I will just take a nap."

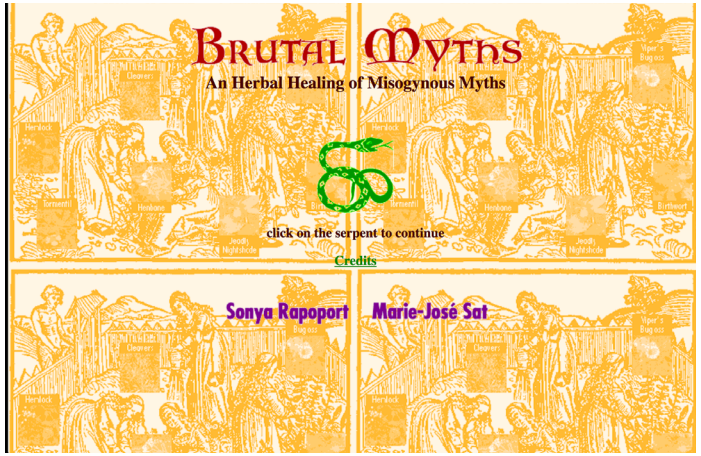
```
if (I am hungry) {  eat some food;} else if (thirsty) {
drink some water;} else{  take a nap;}
```

The above is an example of "pseudocode" to

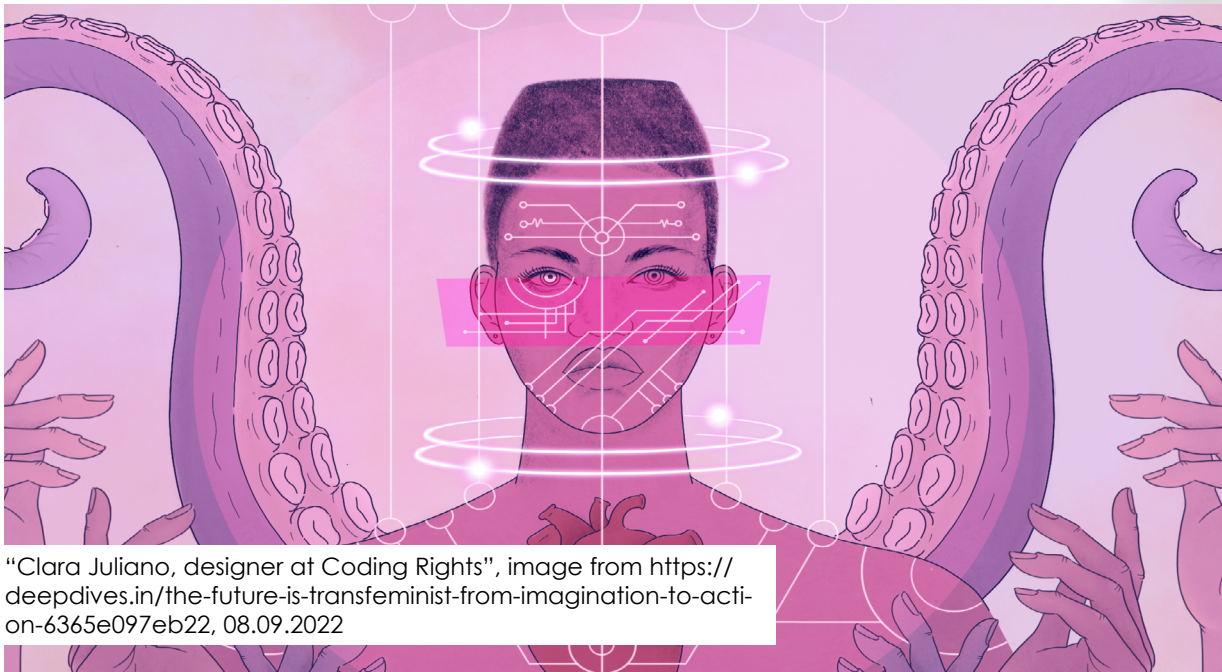
demonstrate what making an everyday decision might look like in programming. The keyword and syntax if is then followed by the condition and checks whether a certain condition holds. As such, the whole if statement is a "Boolean expression" — one of two possible values is possible, true or false, each of which leads to a different path and action. In computer science, the Boolean data type has two possible values intended to represent the two truth values of logic.

Loops offer alternative imaginaries, as is the case of the ancient image of a serpent eating its own tail. Ouroboros, from the Greek, expresses the endless cycle of birth and death, and therefore stands for the ability of processes to infinitely renew themselves. Alongside evocative references to autocannibalism and alchemy, loops are related to control and automation tasks, as well as repetitive procedures in everyday situations such as those heard in repeating sections of sound material in music. [5] In programming, a loop allows the repeated execution of a fragment of source code that continues until a given condition is met, such as true or false. Indeed a loop becomes an infinite (or endless) if a condition never becomes false.
The loop can be thought of as a repeating "if" statement and offers a good way of challenging conventional structures of linear time, and demonstrating how computers utilize time differently. Programming challenges many of our preconceptions about time including how it is organized, how the present is rendered using various time-specific parameters and conditions, as in the case of a throbber. We hope it is already clear that machine-time operates at a different register from human-time, further complicated by global network infrastructures, and notions of real-time computation.

}

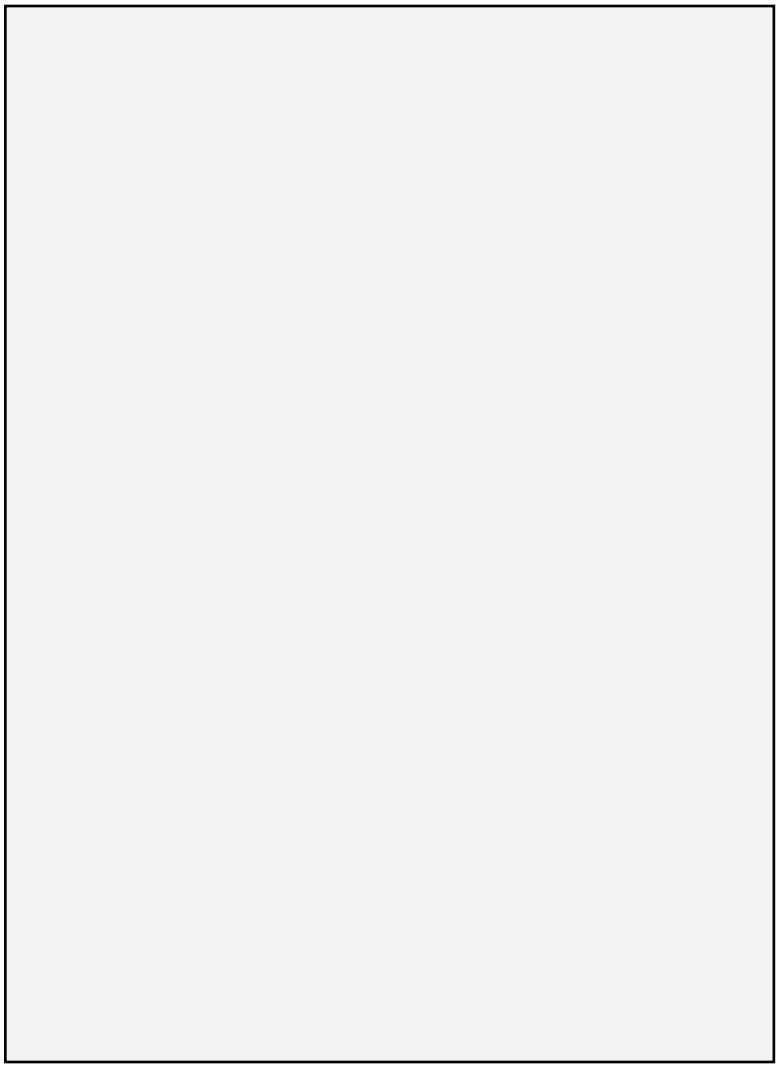


Screenshot, 2020, Firefox v76.0.1 on Mac OS 10.13.3; http://users.lmi.net/sonyarap/brutal/index.html featured in http://www.sonya-rapoport.org/portfolio/brutal-myths-1996/, 08.09.2022



"Clara Juliano, designer at Coding Rights", image from https://deepdives.in/the-future-is-transfeminist-from-imagination-to-action-6365e097eb22, 08.09.2022

loops(notes){



}

This is your code for additional digital resources.
Visit your personal hub to start coding!

thorn

generation(){

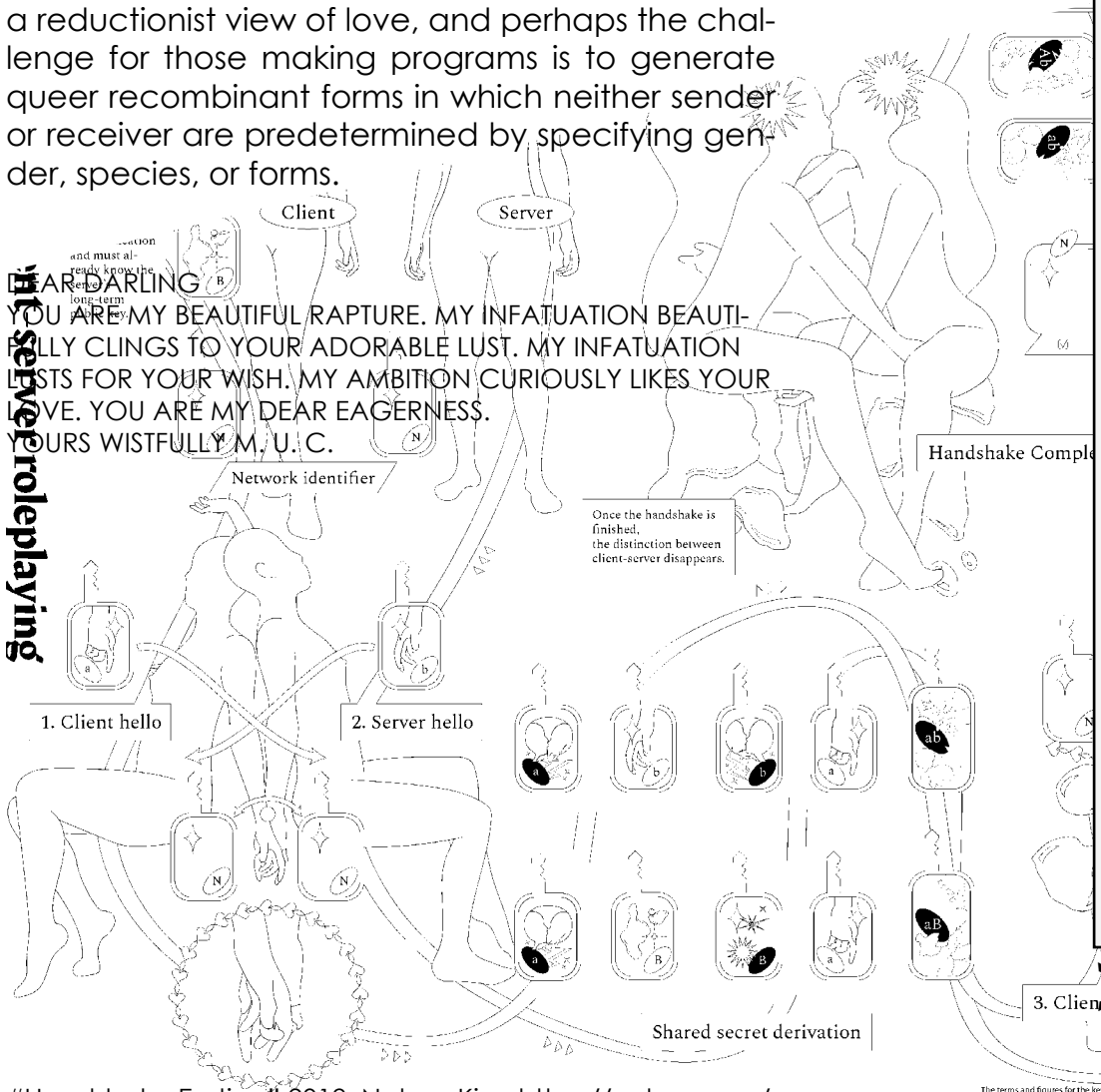
[The] discussion of more love and care in programming brings us to [...] the generative "love-letters" that appeared on the Manchester University Computer Department's noticeboard in 1953. These computer-generated declarations of love were produced by a program written by Christopher Strachey using the built-in random generator function of the M. U. C.(Manchester University Computer, the Ferranti Mark I), the earliest programmable computer. Regarded by some as the first example of digital art,[6] and by Jacob Gaboury as a critique of hetero-normative love, not least because Strachey like Turing was queer.[7] Moreover these letters are arguably more than a longing for same sex love, but human-machine love.

Artist David Link built a functional replica of both the hardware and the original program, following meticulous research into the functional aspects.[8] The main program is relatively simple, and uses loops and a random variable to follow the sentence structure: "You are my — Adjective — Substantive," and "My — [Adjective] — Substantive — [Adverb] — Verb — Your — [Adjective] — Substantive." Some words are fixed and some are optional, as indicated by the square brackets. The program selects from a list of options — adjectives, adverbs, and verbs — and loops are configured to avoid repetition. The software can generate over 318 billion variations. In terms of effect, the dialogue structure is important in setting up an exchange between "Me" (the program writer) and "You" (human reader), so you feel personally addressed. The resulting love letters provide a surpri-

}

sing tenderness of expression that runs contrary to what we consider the standard functional outcomes of computational procedures. This is far from a reductionist view of love, and perhaps the challenge for those making programs is to generate queer recombinant forms in which neither sender or receiver are predetermined by specifying gender, species, or forms.

DEAR DARLING
YOU ARE MY BEAUTIFUL RAPTURE. MY INFATUATION BEAUTIFULLY CLINGS TO YOUR ADORABLE LUST. MY INFATUATION LUSTS FOR YOUR WISH. MY AMBITION CURIOUSLY LIKES YOUR LOVE. YOU ARE MY DEAR EAGERNESS.
YOURS WISTFULLY M. U. C.



"Handshake Erotica" 2019, Nahee Kim, https://nahee.app/handshake_erotica.html, 08.09.2022

#serverroleplaying

}

This is your code for additional digital resources.
Visit your personal hub to start coding!

The terms and figures for the key exchange process are referenced from <Scuttlebutt Protocol Guide(https://ssbc.github.io/scuttlebutt-protocol-guide/)> and <Designing a Secret Handshake: Authenticated Key Exchange as a Capability

leaf

abstraction(){

In programming an object is a key concept, but it is also more generally understood as a thing with properties that can be identified in relation to the term subject. Put simply, and following philosophical conventions, a subject is an observer (we might say programmer) and an object is a thing outside of this, something observed (say a program). In this chapter we will learn to further manipulate objects and understand their complexity in line with people who think we need to put more emphasis on non-human things so we can better understand how objects exist and interact, both with other objects, but also with subjects.

Object abstraction in computing is about representation. Certain attributes and relations are abstracted from the real world, whilst simultaneously leaving details and contexts out. Let's imagine a person as an object (rather than a subject) and consider which properties and behaviors that person might have. We use the name "class" to give an overview of the object's properties and behaviors.

Properties: A person with the name Winnie, has black hair, wears glasses and their height is 164 cm. Their favorite color is black and their favorite food is tofu.

Behavior: A person can run from location A (home) to location B (university). (Pseudoklasse)

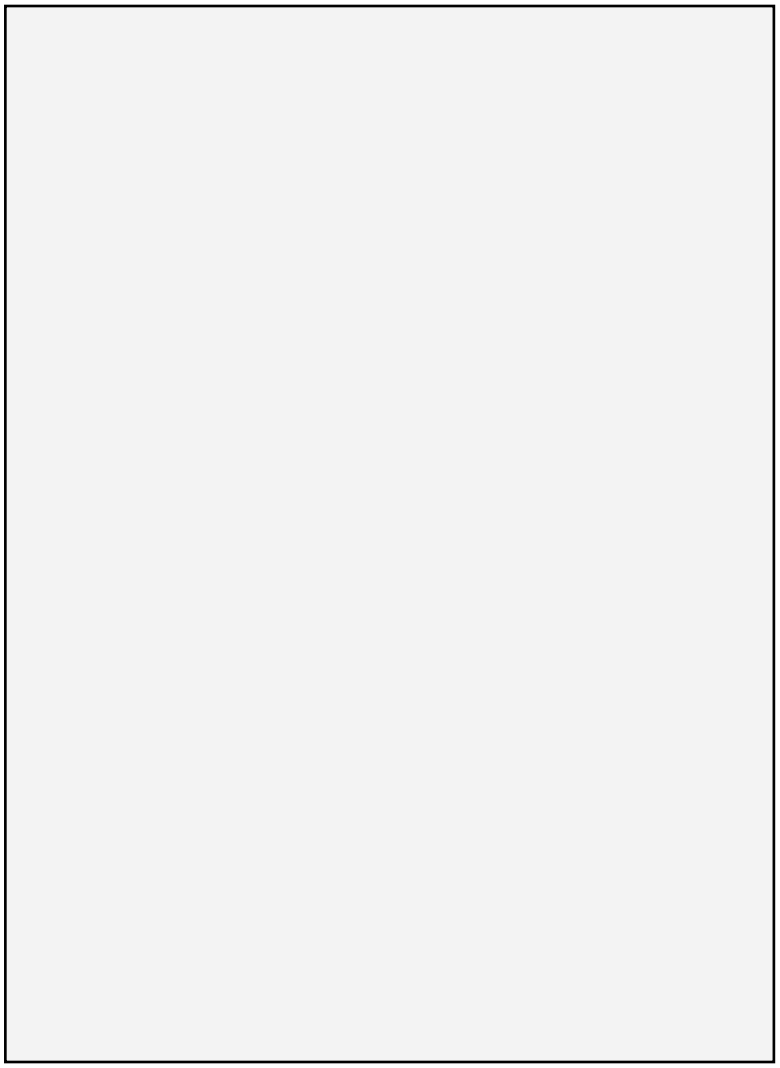
[...] object-oriented programming is highly organized and concrete even though objects are abstractions. It's also worth reiterating that OOP is designed to reflect the way the world is organized and imagined, at least from the computer programmers' perspective. It provides an understanding of the ways in which relatively independent objects operate through their relation to other objects.

[...] [Object-oriented] programming is highly organized and concrete even though objects are abstractions. It's also worth reiterating that OOP is designed to reflect the way the world is organized and imagined, at least from the computer programmers' perspective. It provides an understanding of the ways in which relatively independent objects operate through their relation to other objects.

}

Floppy disks of Cyber Rag, image from https://nymag.com/intelligencer/2018/04/claire-evanss-broad-band-excerpt.html, 08.09.2022

abstraction(notes){



}

This is your code for additional digital resources.
Visit your personal hub to start coding!

stem

queery(){

To query something is to ask a question about it, to check its validity, or accuracy. When querying a database, despite the apparent simple request for data that enables selectivity with regard to which and how much data is returned, we should clearly question this operation too. We need to query the query.

Search engines (like Google and Baidu) are a good example of applications that aggregate content and algorithmically return search results according to a keywords search. They promise to answer all our questions, but do not make the underlying processes (and ideology) visible that prioritize certain answers over others. In a query-driven society, search engines have become powerful mechanisms for truth-making and for our making sense of seemingly endless quantities of data, manifested as streams, and feeds — indicative of the oversaturation of information and the rise of the attention economy. […] The habit of searching, for instance, is transformed into data that is storable, traceable, and analyzable.

We have already explored some of the processes programs use to capture input data in [Chapter 'Data',] especially data that is connected to physical devices, and in this chapter we expand this exponentially to data hosted on online platforms. We scale up from the capture of data to the storage, and analysis of massive amounts of captured data — so-called "Big data" (or even "Big Dick Data" if we consider this to be a masculinist fantasy)[9] — which is in turn utilized for user-profiling,

targeted marketing, personalized recommendations, and various sorts of predictions and e-commerce, and so on. Subsequently it would seem that: *"We're not in control of our search practices — search engines are in control of us and we readily agree, though mostly unconsciously, to this domination."* [10]. But arguably it's not quite as deterministic as this, as these operations are part of larger socio-technical assemblages and infrastructures — including data, data structures, and human subjects — that are also constantly evolving and subject to external conditions.

"*Search happens in a highly commercial environment, and a variety of processes shape what can be found; these results are then normalized as believable and often presented as factual [and] become such a normative part of our experience with digital technology and computers that they socialize us into believing that these artifacts must therefore also provide access to credible, accurate information that is depoliticized and neutral.*" [11]

}

queery(notes){

}

This is your code for additional digital resources.
Visit your personal hub to start coding!

root

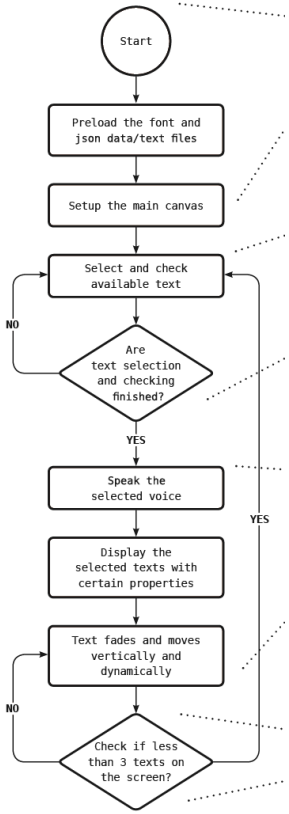
algorithm(){

"1. Begin reading this procedure, unless you have already begun to read it. Continue to follow the steps faithfully; […] 5. Is the subject of the chapter interesting you? If so, go to step 7; if not, go to step 6. 14. Are you tired? If not, go back to step 7; 15. Go to sleep. Then, wake up, and go back to step 7." [12]

The example serves to emphasize that we tend to follow instructions faithfully. However, we might also observe that algorithms are more than simply steps and procedural operations as there are wider cultural and political implications, not least in terms of whether we decide to interpret them on our own terms. In this sense, like cooking, algorithms express cultural differences, and matters of taste, even aesthetics. Extending the analogy to other cultural practices, Knuth quotes Ada Lovelace: *"The process of preparing computer programs for a digital computer is especially attractive, not only because it can be economically and scientifically rewarding, but also because it can be an aesthetic experience much like composing poetry or music."* [13]

Algorithms […] are there to transform, construct, and shape data, in order to then classify, rank, cluster, recommend, label, or even predict things. The concern is not how to build an efficient or optimized algorithm, but to understand these operative dimensions better.

In this chapter we will build on "diagramming," particularly the use of flowcharts to elaborate the practical and conceptual aspects of algorithmic procedures. Flowcharts, […] have been considered a fundamental explanatory tool since the early days of computer programming. One of their common uses is to illustrate computational operations and data processing for programming by "converting the numerical method into a series of steps." [14]. But flowcharts can also be considered to be representational diagrams which can also be used to communicate complex logic between programmers and others involved in software production. This is good practice of course, especially for beginners in a learning context, and is essential for communicating ideas in ways that can be easily understood by others. […] Moreover most software applications are not developed by a single programmer but are organized into tasks that are tackled collaboratively by programmers, as for instance when maintaining or debugging a program made by someone else. Collaborative workflows lends themselves to flowcharts.



Conventionally, each step in a flowchart is represented by a symbol and connecting lines that indicate the flow of logic towards a certain output. The symbols all have different meanings [...]:

OVAL: Indicates the start or end point of a program/system. (But this requires further refection on whether all programs have an end.)
RECTANGLE: Represents the steps in the process.
DIAMOND: Indicates the decision points with "yes" and "no" branches.
ARROW: Acts as a connector to show relationships and sequences, but sometimes an arrow may point back to a previous process, especially when repetition and loops are concerned.

}

algorithm(notes){

}

This is your code for additional digital resources.
Visit your personal hub to start coding!

fruit

learning(){

Machine learning is a term coined by Arthur Samuel in 1959 through his research at IBM on game development, with the ultimate goal to reduce or even eliminate the need for "detailed programming effort."[15]. The roots of how computers might begin to write their own programs lie in older discussions of artifical intelligence.

In machine learning, it is commonly understood that the style is learnt from training datasets through techniques to process and analyze large amounts of (natural language) data. As such, machine learning techniques such as "style transfer" rely on a process of generalization in order to identify patterns. However, this "pattern recognition" is clearly not a neutral process as it involves the identification of input data, and the "discrimination" of information.[16] It is clear that there is other kinds of discrimination in such processes [...]. Understood this way, pattern recognition is not only about smoothing tasks and making accurate predictions in terms of technical operations but also political operations as it creates "subjects and subjection, knowledge, authority" as well as classification and categorization.

It should be pointed out that although machine learning is part of AI, AI is a broader concept. AI, machine learning and deep learning are terms that are often used interchangeably but there are key distinctions to be made. To explain: "You can think of deep learning, machine learning and artificial intelligence as a set of Russian dolls nested within each other, beginning with the smallest and

working out. Deep learning is a subset of machine learning, and machine learning is a subset of AI, which is an umbrella term for any computer program that does something smart. In other words, all machine learning is AI, but not all AI is machine learning, and so forth."[17]

"If you consider a child's eyes as a pair of biological cameras, they take one picture about every two hundred milliseconds, the average time an eye movement is made. So by age three, a child would have hundreds of millions of pictures of the real world. That's a lot of training examples. So instead of focusing on solely better and better algorithms, my insight was to give the algorithms the kind of training data that a child was given by experiences, in both quantity and quality."[18]

If visual literacy is no longer simply an educational task for humans, but also for machines, then it becomes a question of human-machine literacy in its broadest sense. […] That machines can be said to "see" or "learn" is shorthand for calculative practices that only approximate likely outcomes by using various algorithms and models.

"But remember that I am controlling and using for my own purposes the means of reproduction needed for these programmes […] with this programme as with all programmes, you receive images and meanings which are arranged. I hope you will consider what I arrange but please remain skeptical of it."

What is learnt should not be separated from the means by which it is transmitted, nor the direction of travel from human to machine or from machine to human. More to the point, the production of meaning lies at the core of our discussion, as are concerns about what is being learnt, and to what extent this has been compromised or inflected by reductive ideas of how the world operates.

}

Everest Pipkin: Plotter Drawings 2017, https://everest-pipkin.com/#drawings/plotter.html, 08.09.2022

glossary(){

The following definitions are formulated under my own biases. you are free to re-consider their meaning.

PROGRAMMING - Formulation of concepts through syntax that is meant to communicate functionality to developers and computers. The respective programming languages work as a middle ground for thought and writing.

LITERACY - Here, Digital Literacy. The ability to estimate the potential and influence of (new) digital media and its evolution through lived experience and exploration.

VARIABLES - Control points within programs yielding values. Values can be assigned, read, and detached for the program to function.

FUNCTIONS - A named collections of operations within a program. Given an input in form of variables or values, a function produces an output that can be attached to further variables.

BIG DATA - The practice of collecting, processing, and storing data within the shortest amount of time possible.

loops - Repititions of operations during program runtime until a certain condition is fulfilled.

SYNTAX - Grammar for creating a program. In the nature of programming principles, complex computer operations are put into words that can be understood by humans.

OBJECTS - Complex building blocks in programs described by a respective class. Defined by their own functionality and data structure, they can act as simplifications of the real world.

QUERY - Search and organization of data from a larger data set. Information from large data sets can be deduced by searching data of certain properties.

ALGORITHMS - Conceptual ideas involving the input, process, and output of data. Their realization comes mostly through the use of programming.

MACHINE LEARNING - Iterative calculations to build an automated system for answering questions. By estimating control parameters, the machine is supposed to make precise decisions when confronted with new kinds of data.

}

1 Vee, Coding Literacy.

2 This point is largely derived from Kelty's Two Bits, which uses the phrase "running code" to describe the relationship between "argument-by-technology and argument-by-talk." See Christopher Kelty Two Bits: the Cultural Significance of Free Software (Durham: Duke University Press, 2008), 58. Clearly programmers are able to make arguments as people can in other rhetorical forms, see Kevin Brock, Rhetorical Code Studies: Discovering Arguments in and around Code (Ann Arbor, MN: University of Michigan Press, 2019).

3 Paraphrasing the final lines of Leslie's essay "The Other Atmosphere: Against Human Resources, Emoji, and Devices": "The workers become their own devices. They becomes devices of communicative capitalism […]."

4 Søren Pold, "Button," in Matthew Fuller ed., Software Studies (Cambridge, Mass.: MIT Press, 2008), 34. Users are seduced by the wording of the button not least, and Pold suggests that a button is developed with distinct functionality and signification (Ibid., 31).

5 The logic behind loops can be demonstrated by the following paradoxical word play: "The next sentence is true. The previous is false." Further examples of paradox, recursion, and strange loops can be found in Douglas R. Hofstadter's‘ Gödel, Escher, Bach: An Eternal Golden Braid (New York: Basic Books, 1999)

6 Jacob Gaboury, "A Queer History of Computing," Rhizome (April 9, 2013). We return to the issue of Turing's sexuality in Chapter 7, "Vocable Code".

7 Noah Wardrip-Fruin, "Christopher Strachey: The First Digital Artist?," Grand Text Auto, School of Engineering, University of California Santa Cruz (August 1, 2005).

8 David Link's LoveLetters_1.0: MUC=Resurrection was first exhibited in 2009, and was part of dOCUMENTA(13), Kassel, in 2012. Detailed description and documentation can be found at http://www.alpha60.de/art/love_letters/. Also see Geoff Cox, "Introduction" to David Link, Das Herz der Maschine, dOCUMENTA (13): 100 Notes - 100 Thoughts, 100 Notizen - 100 Gedanken # 037 (Berlin: Hatje Cantz, 2012).

9 Big data is referred to as "Big Dick Data" by Catherine D'Ignazio and Lauren Klein, to mock big data projects that are characterized by "masculinist, totalizing fantasies of world domination as enacted through data capture and analysis," see "The Numbers Don't Speak for Themselves," in Data Feminism (Cambridge, MA, MIT Press 2020), 151.

10 Big data is referred to as "Big Dick Data" by Catherine D'Ignazio and Lauren Klein, to mock big data projects that are characterized by "masculinist, totalizing fantasies of world domination as enacted through data capture and analysis," see "The Numbers Don't Speak for Themselves," in Data Feminism (Cambridge, MA, MIT Press 2020), 151.

11 Safiya Umoja Noble, Algorithms of Oppression: How Search Engines Reinforce Racism (New York: New York University Press, 2018), 24-25.

12 Knuth, The Art of Computer Programming, xv-xvi.

13 Knuth, The Art of Computer Programming, v.

14 Ferranti Limited, Ferranti Pegasus Computer, programming manual, Issue 1, List CS 50,September 1955

15 Machine learning is a term coined by Arthur Samuel in 1959 during his game development research at IBM which ultimately aimed to reduce or even eliminate the need for "detailed programming effort," using learning through generalization in order to achieve pattern recognition. See Arthur L. Samuel, "Some Studies in Machine learning Using the Game of Checkers," IBM Journal of research and development 3, no.3 (1959): 210-229

16 Clemens Apprich, "Introduction," in Clemens Apprich, Florian Cramer, Wendy Hui Kyon Chun, and Hito Steyerl, eds., Pattern Discrimination (Minnesota: Meson Press, 2018), x

17 See Pathmind's "A.I. Wiki: A Beginner's Guide to Important Topics in AI, Machine learning, and Deep Learning," https://pathmind.com/wiki/ai-vs-machine-learning-vs-deep-learning

18 The Fei Fei Li quote is taken from Nicolas Malevé's article, "'The cat sits on the bed': Pedagogies of vision in human and machine learning," Unthinking Photography (2016), https://unthinking.photography/articles/the-cat-sits-on-the-bed-pedagogies-of-vision-in-human-and-machine-learning.

https://assemblyre.glitch.me

Bauhaus-Universität Weimar

LYRE