SWAAT#00

Colophon

This is the first in what is hoped to a be a series of publications reflecting the experiences of teaching with free software in the Experimental Publishing Masters course (XPUB) of the Piet Zwart Institute in Rotterdam, the Netherlands. Specifically, the editors are two tutors of the "Prototyping" course, which aims to engage students with the materiality of software, and aims to see it as more than just a way of getting things done, but rather as a way of seeing and forming the world, a practice with a history and a sociality, and one increasingly with practitioners with diverse (disciplinary) backgrounds and experiences, with an interest in actively critiquing many assumptions associated with software's computer science origins.

July 2021

Editors: Manetta Berends & Michael Murtaugh

XPUB1: Kendal Beynon, Martin Foucaut, Camilo García A., Clara Gradel, Nami Kim, Euna Lee, Jacopo Lega, Federico Poni, Louisa Teichmann and Floor van Meeuwen

XPUB2: Damlanur Bilgin, Mark van den Heuvel, Avital Barkai, Max Lehmann, Ioana Tomici, Clara Noseda, Mika Motskobili, Anna Sandri, Tisa Neža Herlec

XPUB
Master Experimental Publishing, Piet Zwart Institute, Willem de Kooning Academy, Rotterdam
Course Director: Aymeric Mansoux
Course Administrator: Leslie Robbins
Course tutors: Clara Balaguer, Manetta Berends, Cristina Cochior, Michael Murtaugh, Steve Rushton, Amy Suo Wu, Marloes de Valk
System administrator: gnd

https://xpub.nl

Published in the context of:

-- The prototyping sessions of the XPUB1 year 2020/2021
https://pzwiki.wdka.nl/mediadesign/Prototyping
-- Special Issues: https://issue.xpub.nl/
-- Special Issue 13: Wor(l)ds for the future, guest editor: Nienke Scholts
https://pzwiki.wdka.nl/mediadesign/Category:WordsfortheFuture
-- Special Issue 14: I Don't Know Where We're Going, But, guest editor: Lídia Pereira, partner: Sébastien Tien (PNF)
https://pzwiki.wdka.nl/mediadesign/Category:Situationist_Times
-- Special Issue 15: Radio Implicancies, guest editor: Femke Snelting
https://pzwiki.wdka.nl/mediadesign/Category:Implicancies#Radio_Implicancies

Publishing pipeline:

```
curl https://pad.xpub.nl/p/prototyping-handout/
  export/txt > SWAAT-00.md && \
curl https://pad.xpub.nl/p/prototyping-
  handout.css/export/txt > SWAAT-00.css && \
pandoc -f markdown --pdf-engine weasyprint --toc
  --toc-depth 2 -c SWAAT-00.css SWAAT-00.md -o
  SWAAT-00.pdf && \
pandoc -f markdown -t html --standalone --toc --
  toc-depth 2 -c SWAAT-00.css SWAAT-00.md -o
  SWAAT-00.html
```

About Prototyping (from the course wiki):

Prototyping is about conducting research through an iterative process of making, communicating & testing, and reflection. Prototyping asks you to combine practical technical knowledge with your own research questions linked to the thematic projects, and encourages producing designs that "work" not only in terms of the technology, but on a communicative level to explore particular ideas.

Through prototyping, fundamental concepts of programming will be explored in the context of tools and methods familiar to those with a design background. Graphical interfaces will be contrasted with command-line interfaces as a means of going beyond traditional "iconic" and "user-friendly" ways of working - for example with Graphical User Interfaces and What You See Is What You Get tools - toward the procedural and text-based. Effort will be placed at finding ways to bridge traditional top-down design tools with a code-oriented approach. You do not need to have a particular level of technical experience. What is expected of you is an active engagement with and willingness to explore networked digital media technology. Students of all levels and previous experience will be encouraged to stretch their ways of working and knowledge to hopefully explore previously unknown or uncomfortable territory and broaden their palette of tools.

https://pzwiki.wdka.nl/mediadesign/Prototyping#2015/2016

```css
    figure{
        margin: 0
    0 1em 0;

    padding: 0;
    }
    figcaption{
        margin-
top: -1em;
    }
figcaption,
small{
    font-
size: 10px;
    line-height: 1.25 !
important;
}
sup{
    font-
size: 75%;
    line-
height: 0;
}
hr{

    border:
0;
    border-bottom: 1px
solid black;

    margin:
2em 0;

}
p{
    margin: 0
0 1em 0;
}
pre,
code{
    font-family:
monospace;
    font-size: var(--
font-size);
    line-height: var(--line-
height);
    /* word-wrap: break-word;
*/
}
pre.sourceCode.python{
    page-break-inside:
avoid;
}
ul {
    margin: 0
0 1em 0;

    padding:
0;
}
    ul li {
        list-style:
none;
        margin-
bottom: 0;
    }

    ul
    li::before{
        content:
"--";
        margin-left:
-1.25em;
        margin-right:
0.5em;
    }
    blockquote{
        font-style:
italic;
        margin:
1em 1em;
}

    section.footnotes{
        page-break-before:
always;

        margin:
1em 0;
}

    section.footnotes::before{
        content:
"Footnotes";
        font-size: var(--h1-
font-size);
        font-weight:
bold;

        padding: 1em 0;
    }
    section.footnotes
    hr{
        display:
none;
    }
    section.footnotes a.footnote-
back{
        padding-left:
0.5em;
        text-decoration:
none;
    }
section#backcover{
    page:
backcover;
}
    section#backcover
    pre{
        font-
size: 6px;
        line-
height: 1;
        white-space: pre-
wrap;
    }
```

Appendix: stylesheet.css

```css
:root{
    --font-size: 12px;
    --line-height: 16px;
    --text-color: black;
    --h1-font-size: 32px;
    --h1-line-height: 32px;
}

html,
body{
    color: var(--text-color);
    hyphens: auto;
    font-family: serif;
    font-size: var(--font-size);
    line-height: var(--line-height);

    margin: 0;

    padding: 0;
    hyphens: auto;
}
section{
    columns: 2 auto;
    column-fill: auto;
    column-gap: 8mm;
}

section#sec_appendix{
    column-gap: 3mm !important;
}

@page{
    size: 210mm 297mm;
    margin: 10mm 7mm 20mm 7mm;
```

```css
    @top-left{
        content: string(doctitle);
        font-family: monospace;
        font-size: 9pt;
    }
    @top-center{
        /* content: string(sectiontitle); */
        font-family: monospace;
        font-size: 9pt;
        text-align: left;
    }
    @bottom-left{
        content: counter(page);
        font-family: monospace;
        font-size: 9pt;
    }
}
@page:first{
    @top-center{

        content: "";
    }
}
@page backcover{
    background-image: url('https://hub.xpub.nl/sandbot/SWAAT/00/handout/Quilt_WFTF.jpg');
    background-size: auto 100%;
    background-repeat: no-repeat;

    @top-center{

        content: "";
    }
}
h1.title{
    string-set: doctitle content();
}
#TOC{
    page-break-after: always;
}
    #TOC ul{
```

```css
    margin: 0;

    padding: 0;
    }
    #TOC ul li ul{
        margin-left: 1.25em;
        margin-top: 0.5em;
    }
    #TOC li{
        list-style: none;
        margin-bottom: 0.5em;
    }
    #TOC li a::before{
        content: target-counter(attr(href), page);
        float: right;
    }

h1{
    page-break-before: always;
    font-size: var(--h1-font-size);
    line-height: var(--h1-line-height);
    string-set: sectiontitle content();
}
h2 {
    font-size: var(--font-size);
    line-height: var(--line-height);

    margin: 1em 0;
}

h2.timestamp,
    h3.timestamp{
        font-size: var(--font-size);
        line-height: var(--line-height);
        font-weight: normal;
        text-decoration: underline;
    }
img{
    max-width: 100%;
    margin: 0 auto 1em auto;
```
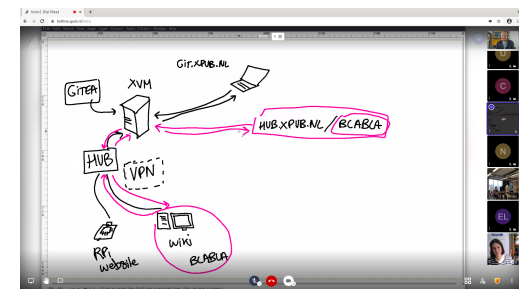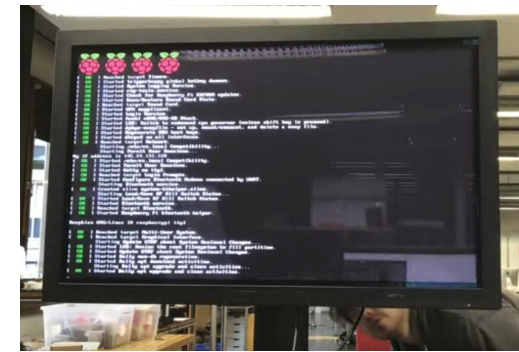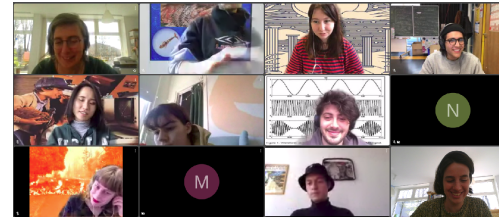

Screenshots

Software with an attitude

You're browsing a database with a program called WikiWikiWeb. And the program has an attitude. The program wants everyone to be an author. So, the program slants in favor of authors at some inconvenience to readers.

From Ward Cunningham's (the first) wiki software

https://pzwiki.wdka.nl/mediadesign/Category:Cookbook

So-called "code cookbooks" exist as long as the practice of programming. Compilations of snippets of code, known to work in a past context, serve as "cut and paste" starting points for future projects. The XPUB wiki contains traces of many such snippets, some of which have been tagged / linked with the wiki category Cookbook. For instance, this tasty recipe:

```
while true
  do
    curl -s http://www.bbc.co.uk/blogs/
thereporters/willgompertz/
2010/02/40_wild_birds_play_a_gibson_le.html | \
    lynx -dump -nolist -
stdin | \
    dadadodo -c 20
- | \
    espeak -s 120 -v mb-en1 --
stdin | \
    mbrola -e /usr/share/mbrola/voices/en1 -
- | \
    ices2 net-art-
radio.xml
done
```

At first glance, one might admire it as an example of a shell pipeline, chaining different tools together (**curl**, **lynx**, **dadadodo**, **espeak**, **mbrola**, **icecast**). At the same time it makes one aware of what is missing: context! While it's a nice proof of concept, it doesn't "just run", contingent resources such as the configuration file of an icecast server are missing, and you don't necessarily understand what it produces or why it was made.

Online tutorials often push a specific kind of idea of making-with-code and a specific way of technical learning in which a tool is shown to do one particular thing. Often, in presenting a solution, there is a tendency to compress: eliding the specificity of a tool or of the context of its use.

This is where this publication steps in, as a format that allows to be verbose and a medium that asks for distribution and sharing. In preparing this publication, a starting point is a shift of focus from the "standalone" to more "embedded" and holistic code practices. How can thinking be informed by making, how can it be one process? How can we merge editorial work with technical learning/experimenting/thinking?

The PERL Journal <https://www.foo.be/docs/tpj/>

Sometimes it's easy to forget that computer languages, like human languages, aren't merely ways to accomplish tasks, but tools for thinking about the world. It would be easier for us to concentrate on the "how" of Perl rather than the "why" or the "what now", but not as much fun. We want TPJ to be intricate and interesting, like Perl itself.

Jon Orwant writing in the first issue of The Perl Journal. 1996.

There is more than one way to do it. This phrase is often associated with the PerlLanguage. Unlike other languages (notably Scheme), Perl intentionally contains many simple expressions that are equivalent in result. Its inventor, LarryWall, is trained as a linguist. He has this crazy notion that sometimes phrasing a sentence differently might make its meaning clearer...

http://wiki.c2.com/?ThereIsMoreThanOneWayToDoIt

Part of the challenge of understanding algorithmic oppression is to understand that mathematical formulations to drive automated decisions are made by human beings. While we often think of terms such as "big data" and "algorithms" as benign, neutral, or objective, they are anything but. The people who make these decisions hold all types of values, many of which openly promote racism, sexism, and false notions of meritocracy, which is well documented in studies of Silicon Valley and other tech corridors.

Safiya Umoja Noble. *Algorithms of Oppression*, 2018, p. 2.

In "tech" circles, claims that the products of coding are "just tools" are still routinely made and often unchallenged; in contrast, we find inspiration in intersectional feminisms and decolonial computing, for their insistence on holistic approaches to technology where social engagement and moral responsibility are never detached. In contrast, these approaches offer a lens for deepening one's understanding and engagement with a view of technology that not only offers ways of "thinking about the world", but which acknowledges the role technology plays in "worlding" and its potential for perpetuating "epistemic violence".

notebook was used to generate different visuals or different aspects of this online publication and of course the next step when you start to open up basically this is this path there is no end to it so why not also connect it with the print publication so you start to break also the boundaries of media you start to have this sort of both this implosion and explosion of practices where you're just starting to create your own technical or instrumental environment and then use different type of media and format in a complementary way another thing i'm taking the time i don't want to take too much time we got a little bit delayed but i think that to sum up i think that the different type of work that come out when the students are graduating can be put into i would say like four categories one belongs to this explosion of media that i was just uh mentioning where suddenly you realize that this sort of artificial boundaries that comes from you know classical forms of engaging with art and design is also a parameter with which you can play with

---

https://tv.lumbung.space/videos/watch/7d3596c8-d4fe-4224-ac6b-7b6d69410b09

https://www.youtube.com/watch?v=Go-N_0AA8lk&t=3367s

https://screenwalks.com/
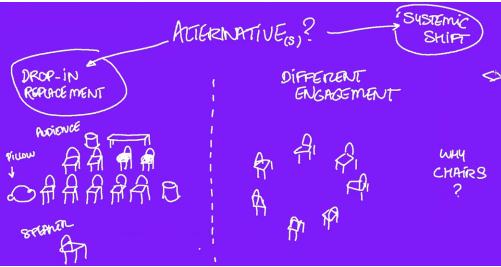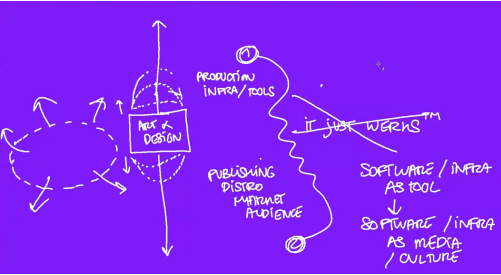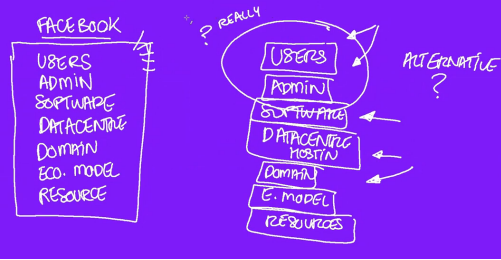
Artificial Boundaries

A transcribed snippet of *SCREEN WALK WITH AYMERIC MANSOUX AND ROEL ROSCAM ABBING 30 JUNE 2021 - 6 PM UK TIME / 7 PM CET*, hosted by Fotomuseum Winterthur and The Photographers' Gallery.

The transcript starts at 53m36s: Aymeric talks about the XPUB infrastructure and how these setups shape XPUB's learning environment.

```
youtube-dl --write-auto-sub --skip-download
https://www.youtube.com/watch?v=Go-N_0AA8lk
```

i'm just going to go through um some of the services that that that will run in the course so if you lose yourself in the different things that we are running uh you can go to xbob.nl and you will see a couple of things so these are some of the tip of the iceberg and see this is sort of the most visible thing that we that we run on and this is all self-hosted and self-managed by uh by by the staff and the students and um one of the most important one probably the one that is central through which everyone is introduced at the beginning when they start to study with us is the wiki so the wiki uh again i don't want to to make it very simple um if some of you are familiar with wikipedia maybe you might not know you may not know or you may not be aware that uh the software that runs wikipedia is actually called media wiki and it's something that you can install and run on the servers or the machine that you own so we basically have a similar environment as wikipedia to keep it simple and this environment is just used as a wiki for the course this is where we're going to to develop the curriculum where we're going to to make public the different educational resources but this is also a place for students to document their work and that's the first thing that they're going to encounter with this course to say okay like how can you use this wiki this self-hosted place then try to make it yours to document your practice uh in a very mixed media way so using scripture video documentation bits of code all sorts of different things that you're going to be exposed through during during the your study with us and this is a very very it's usually the first dip into these questions of infrastructure the capacity that oh wait i can actually run my own wiki oh wait i can actually use that i can actually publish right away and document my own practice and create some sort of history as a matter of fact uh i mean we're getting to the point where we have so many so much content o n this wiki that we are finally on finally finally but we're really reaching the point where it's getting difficult to find things or that we actually forget that we have already made a page for certain things and of course you have the holy story of the past students who went through the course who have been using it to document their work so you start to have um this creates

a very interesting institutional glue and culture around around the course another thing that we use a lot is the pad so ether pad is a project that maybe some of you are familiar with again something that you can sell first which is a very simple system in which you you can collectively write on a blank page essentially with text but you can also add some plugin to include images and visuals and you can see you can type and you will see the others logged on the same page typing at the same time as you so for collective session this is very very handy to the point that we have started to adopt something that we call the pad of the day and the part of the day is what it is is that for every day while there is a lesson or there is a workshop a pad with the time stamp is being created so we can use that as a sort of an accumulation of of notes that are being taken collect collectively collaboratively by both the staff and the students on a guest lecture on the workshop on the brainstorming about certain work and and then if necessary this is going to be copy edited and then move to wiki for more stable location at least that's the ideal situation another thing that we use also quite a lot is as we engage with with programming is our own code repository so essentially to make it simple that's a way to store uh and to enter public make public source code on its digital file so uh there are different let's say a repository that we use all the students have access to it you even have uh even alumni you keep on using it after graduating it's a it's a kind of a big resource for uh for for us to make public the more technical aspects of the of the course now things get of course start to get interesting when you start to connect this with other aspects so very briefly for instance this website here is essentially a collection of different git repository so whenever the students are working on a new publication and there is a documentation that needs to be developed or written this is not made through let's say like a float a bunch of floating files there is a git repository so repository that store all these files and the students can push changes to this repository and that's automatically published and again this is this is a way you know you can you can have it's not it's not different in a way of installing wordpress and having access to the back end of the of the system and then have people to fill things on the cms or something like that but here the difference is that you have actually you are making your own backend you are deciding okay i'm going to combine these different tools together to make it my own publishing workflow and so when you start to dive into this then things can get quite become a bit of a crazier ride so recently we've been experimenting with with jupiter which is a computational notebook that allows to essentially write and execute code uh in in the browser and i knew that also as a sort of publishing and teaching environment so everyone as as a publishing notebook and that allows to run different for instance generative experiments like on this course that was done by one of our tutorial bands um so this starts to also feed into the practice of the students so when a new project like this one is being is being developed by by the students it's never just a sort of isolated single page or something that is let's say let's say yeah a designed a top-down design it's always uniform in sort of bottom-up bottom-up way by the different tools and techniques that i've seen in the in the course so for instance this jupiter

Tool diversity

> *While a familiar gesture is one that fits perfectly well in a generally accepted model, an awkward gesture is a movement that is not completely synchronic. It's not a countermovement, nor a break from the norm; it doesn't exist outside of the pattern, nor completely in it. Like a moiré effect reveals the presence of a grid, awkward behaviour can lead to a state of increased awareness; a form of productive insecurity that presents us with openings that help understand the complex interaction between skills, tools and medium.*

Femke Snelting. *Awkward gestures: designing with Free Software.* 2008. https://freeze.sh/_/2008/awkward/#

2020-12-10 11:18

Is it an idea that we do a day together on the 11th of January? We could take that day for the "software diversity day" we spoke about? We could read the Awkward Gestures text from Femke, do a sort of "very useful tools" round (thinking out loud now here), and perhaps some sort of exercise to make space for awkwardness? Or perhaps read another text together around "diversity", "multiplicity", "software richness" … ?

2021-01-04 13:01

So, yes, I agree it's a good idea to make an intro day where we can make the approach of diverse software tools apparent, and indeed give space for the topic of awkwardness. Given the new lockdown situation, I'm thinking that it will be all the more important to make use of the "sandbot" as a platform and as a way work together (at a distance). On a practical level, it's maybe a good thing that we didn't do the transfer yet, as I still have easy access to the machine here in Brussels.

2021-01-04 13:24

Agreed, good idea to continue using the Sandbot. There is just a big questionmark around the collaborative aspect. The not-being-able-to-see-eachothers-work is a bummer … It's one of the things i will dive into this afternoon.

One more question, but perhaps it's easier to talk over Zulip: shall we still organise a dedicated "software diversity day", or, do we weave it into the trimester? I think it would be nice to emphasize this super clearly and spend a day on it, but we would need to think this through!

2021-01-05 12:00

I am thinking more of weaving the idea into the trimester as it's such a central issue. I think it's important to "perform" the diversity rather than somehow making a singular day.

Over the break, the thinking about the question of software diversity has meshed with some of the reading I've been doing (and hope also to incorporate into the trimester), namely the idea of "platforms" – the way that (in this case) software isn't written in a vacuum, but responds to the particularities/capabilities of, for instance a hardware platform or an API, and that ultimately these also connect to communities of practice (and use).

In this way, I think it's also a good bridge between the exploration of game platforms and the continuation of exploration of diverse ways of publishing.

2021-05-19

It has been important to put a lot of different things on the table (feeling of overload is maybe okay for the first year). At the same time, and this is something that was missing this year I feel, it's important to ask the group where they come from, which tools they work with in their practice and what kind of things they want to learn.

Learning to work with new tools is a challenge! Tools are so much entangled with habits, workflows, aesthetics, etc.

Jacopo and Floor asked me this question in the first trimester: "Why are we making plain text layouts?" When speaking about this "why", we talked about experimental design practices – which is where these particular ideas come from – but also about what kind of space these tools create for rethinking workflows and layout making. It started to make sense within the context of a design practice that wants to question how tools-shape-practice and practice-shapes-tools.
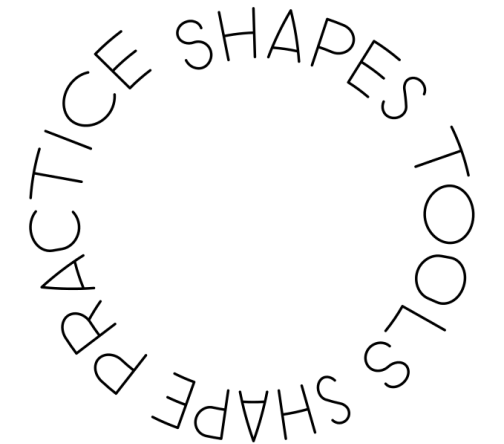
Tools shape practice shape tools …, a work method and motto of Open Source Publishing (OSP) http://osp.kitchen/about

How can this continuously be acknowledged in the prototyping sessions? Can tool diversity operate as a methodology? How do the tools we work with speak back to the different kind of practices in the room: design practices, artistic research, software studies, publishing, …?

Toolbox

An expanding collection of tools and resources that crossed paths with the Prototyping sessions this year.

Beautiful Soup

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

"Beautiful Soup, so rich and green,
Waiting in a hot tureen! Who for such dainties would
not stoop? Soup of the evening, beautiful Soup! Soup
of the evening, beautiful Soup!

Once upon a time, a **python** programmer trying to work with web pages was faced with tools that looked down their **XML**-straightened noses at pages from the "wilds" of the web and would simply report "Invalid document" and summarily refuse to continue. Then came the **Beautiful Soup** library, inspired by the universe of Alice in Wonderland, where mathematical precision and fantastic absurdity comfortably co-exist. **Beautiful Soup** was a breakthrough library that aimed to read *any* web page and provided a plentitude of ways to traverse and manipulate them programmatically. The project had a direct impact on the subsequent development of the **HTML** standard and the "flexibility" of contemporary libraries like **html5lib**.

In other words, a **python** library for parsing webpages, which is basically a way to navigate through a webpage using code. You can for example, ask for all the links by searching for all the <a>'s, or for all the images by looking for all the <img> elements.

Bitsy

A web-based "fantasy console" (see **TIC80**) and "tiny game" development environment based in **javascript**.

http://ledoux.io/bitsy/editor.html
https://github.com/le-doux/bitsy
https://github.com/seleb/bitsy-hacks
https://hub.xpub.nl/sandbot/PrototypingTimes/sketches/Camilo%20/S1.html

*Had a really interesting session with Camilo to unpack his "**bitsy** hack" and unpack the onion layers of the **bitsy** ecosystem: where "hacks" are scripts that get processed with **rollup** to then be triggered in the **bitsy** interface as commands embedded in a game "dialog".*

Though initially kind of maddening, the convoluted nature of many modern **javascript** frameworks is actually something to celebrate …what's powerful (and complicated) is that these different layers co-exist and rely on each other, and are being developed by diverse communities & users in parallel.

It's a reminder of how **Javascript** is actually quite a difficult language to master as one needs to be able to work as an archaeologist, pulling the different styles, asumptions, and dependencies of the layers apart and understanding at what level (and in what way) it's possible/desirable to make interventions.

An early bitsy sketch from Camilo

Decolonial computing

*Practitioners and researchers adopting a **decolonial computing** perspective are required, at a minimum, to do the following: Firstly, consider their **geo-political and body-political orientation** when designing, building, researching or theorizing about computing phenomena; and secondly, embrace the 'decolonial option' as an ethics, attempting to think through what it might mean to design and build computing systems with and for those situated at the peripheries of the world system, informed by the epistemologies located at such sites, with a view to undermining the asymmetry of local-global power relationships and effecting the 'decentering' of Eurocentric / West-centric universals.*

*I want to suggest that it is necessary to reconsider the rise of Big Data/datafication, with respect to both its rhetoric (or mythology) and its reality (or 'material' power), in terms of how this phenomenon contributes to maintaining, expanding and refining (or adapting) the racial political economy of global white supremacy under what is purported to be an increasingly **techno-scientific postmodern/postcolonial condition**. In this connection, and building on the arguments of others who refer to the construction of a 'Big Data Divide', I suggest there is a need to think about how datafication contributes to 'iteratively' re-inscribing the 'Digital Divide', itself a legacy of earlier colonial 'divides'.*

Syed Mustafa Ali, Towards a decolonial computing. 2014.

https://pad.constantvzw.org/p/implicanties

weaving with words and
code
weaving with words and code
                    weaving with words and
code
eaving with words and
code
weaving with words and c
de
weaving with words and code
                    weaving with words
and
code
        weaving with words and
code
weaving wi
h words and
code
weaving with words and code

weaving with words and code
                    weaving with
words and
code
            weaving with words and
code
    weaving with words and
code
weaving w
th words and
code
weaving with words and
code
weaving with words and code

weaving with words and code

weaving with words and
code
                    weaving with
words and
code
                    weaving with words
and
code
            weaving with words and
code
                    weaving with words and
code
        weaving with words and
code
    weaving with words and
code
  weaving with words and
code
weaving with words and
code
eaving with words and
code
eaving with words and
code
weaving with words and
code

                    weaving with words and
code
            weaving with words and
code
                    weaving with words and
code
                    weaving with words and
code
                    weaving with words and
code
                    weaving with words and
code
                        weaving with words
and
code
                            weaving with
words and
code

weaving with words and
code
weaving with words and code

weaving with words and code

weaving with words and
code
weaving w
th words and
code
        weaving with words and
code
                    weaving with words and
code
                            weaving with
words and
code

weaving with words and code

weaving with words and code

weaving wi
h words and
code
            weaving with words and
code
                        weaving with words
and
code

weaving with words and code

weaving with words and c
de
eaving with words and
code
                    weaving with words and
code

weaving with words and
code
weaving with words and
code   w

eaving with words and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code weaving with
words and code weaving with words and code weaving
with words and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code weaving with
words and code weaving with words and code weaving
with words and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code weaving with
words and code weaving with words and code weaving
with words and code weaving with words and code weaving
with words and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code weaving with
words and code weaving with words and code weaving
with words and code weaving with words and code
weaving with words and code weaving with words and
code weaving with words and code weaving with words
and code weaving with words and code weaving with
words and code

```
# ... and then cut it up into lines of 100
  characters, to show the pattern nicely,
tmp_line =
''

for character in
  line:
    if
  len(tmp_line) <
  99:
        tmp_line +=
  character

    else:

  print(tmp_line)
        tmp_line =
''

        count =
  0
```

weaving with words and codeweaving with words
and code weaving with words and code weaving
with wo
ds and code   weaving with words and code
weaving with words and code   weaving with

words and
code   weaving with words and code
weaving with words and code   weaving with
words a
d code   weaving with words and
code   weaving with words and
code   weaving w
th words and code   weaving with words
and code   weaving with words and
code
           weaving with words and
code   weaving with words and
code   weav
ng with words and code   weaving
with words and code   weaving
with wo
ds and code   weaving with words
and code   weaving with words
and
code   weaving with words and
code   weaving with words and
co
e   weaving with words and
code   weaving with words
and c
de   weaving with words
and code   weaving with
words
nd code   weaving with
words and code
weaving wi
h words and code
weaving with words and
code
  weaving with words and
code   weaving with
words and code
                   weaving with words and
code   weaving
with words a
d code   weaving
with words and
code
  weaving with words and
code   weaving
with words and code
                        weaving with words
and code
weavi
g with words and
code
weaving with words and code
                        weaving with words
and code
weav
ng with words and
code
weaving with words and code
                            weaving with
words and
code
    weaving with words and
code

Distribusi

https://pypi.org/project/distribusi/

A content management system for the web that produces static index pages based on folders in the **file system**. It is inspired by the automatic index functions featured in **web servers**. It works by traversing the file system and directory hierarchy to automatically list all the files in the directory and providing them with **HTML** classes and tags for easy styling.

-- generating webpages using the file system as content management system (CMS)
-- working with generated lay-outs
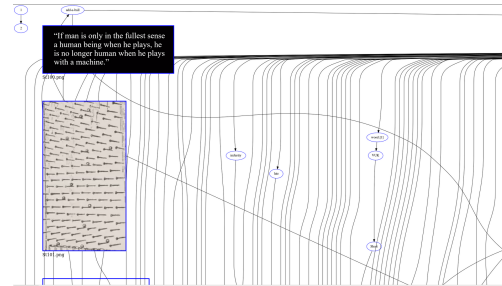
Lecture zine by Camilo, https://hub.xpub.nl/sandbot/Lecture_Zine/

Flask

https://flask.palletsprojects.com/

Inspired by the success of the **python Django** framework, **Flask** aimed to deconstruct the 12 course meal that is **Django** into a minimalist à la carte framework that makes minimal assumptions about what ingredients you want to use to make a web application.

-- making systems (web applications), not just interfaces
-- combine writing code in Python, using query strings and URL variables (`?query=hello&type=text`) to pass data on from the webpage to the Flask server, designing interfaces with Jinja templates, HTML and CSS
-- running multiple Flask servers (one per student) on the webserver of the Sandbot

Standart Dictionary by Euna, made with Flask https://hub.xpub.nl/sandbot/~eunalee/flask

Jinja

https://jinja.palletsprojects.com/

A **python** library for working with templates. Inspired by the integrated template tool in the **Django** framework, itself inspired by the mix of **HTML** and code popularized by **PHP**. **Jinja** templates are basically a format of hybrid documents mixing a "pseudo **python** source" and "destination" language (often **HTML**), where the "source" components are performed (with loops and variables executed in a context of some kind of structured input data) and "destination" parts that simply "pass through", the result being a new transformed / synthesized document in the destination language.

-- using variables in HTML templates
-- generated lay-outs
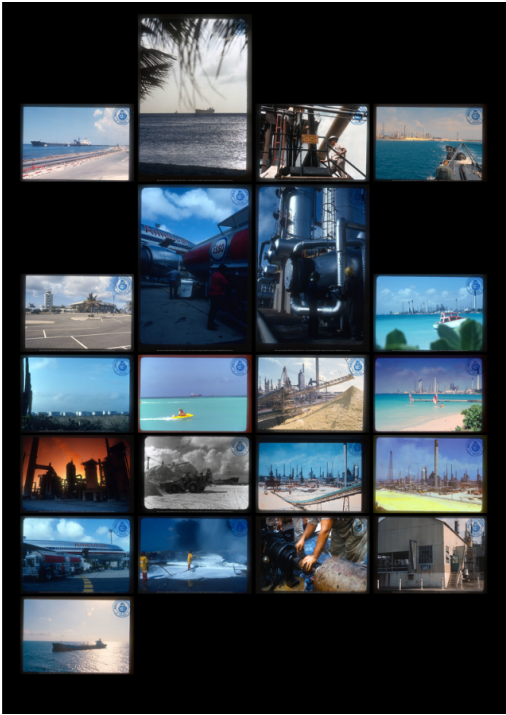
Esso Antilles by Pongie, combining automated search engine API requests with a Python script and Jinja templates to generate this PDF/zine https://hub.xpub.nl/sandbot/~pongie/Esso-Antilles.pdf

Jupyter notebooks

https://jupyter.org/

Jupyter notebooks (aka **jupyter lab**) is a web-based CMS for writing, testing, and publishing hybrid code + text documents. With humble roots in the "interactive" version of python (**ipython**), the **Jupyter** project has expanded massively in the last years buoyed by an explosion of use and development from the so-called "data science" community, as the tool provides access for non-programmers to make use specialised tools such as statistical analysis and visualisation, and to make results publishable in a format that's both legible (by virtue of including narrative text and graphics) and re-producable (by virtue of including the "source" code). It's in many ways the embodiment of Knuth's holy grail of "Literate programming", or (perhaps more humbly) the contemporary equivalent of the spreadsheet. The recent "lab" interface adds drag and drop file management and an integrated "shell" making the interface a kind of trojan horse to teaching a **GNU/Linux** operating system.

In the spring XPUB was visited by artist David Benqué who shared his "critical algorithmic practice" using jypter notebooks to investigate social systems through diagramming.

https://davidbenque.com/

See also: Social shell, A FEMINIST NET/WORK, sandboxes, Jupyter Pi

Markdown

https://daringfireball.net/projects/markdown/

Markdown's origins are in the blogging community, born from a desire for "writerly" ways to publish **HTML** (e.g. using easy to type keyboard symbols and visual conventions rather than the stricter angle-bracket syntax of **HTML/XML**), and was inspired by **email** formatting and **ASCII art**. Although there are various atempts to standardize **markdown**, the format is popular due to its "creole" nature, many extensions exist to **markdown** reflecting usage in different publishing contexts (e.g. additions such as "code fencing" popularized by the github platform have become "standard extensions" available outside of that proprietary context).

MediaWiki

https://www.mediawiki.org/
https://www.mediawiki.org/wiki/API:Main_page
https://www.mediawiki.org/wiki/API:Etiquette

Everybody knows about wikipedia; less people know that the software that wikipedia is based on is called **mediawiki**, and this software can be self-hosted to make your own (personal / institutional) wikis. It's behind the XPUB wiki and has been used as part of many special issue and graduation projects. Also, every **mediawiki** *instance* by default offers its contents via a well-defined **API** that unlike proprietary web services requires no legal agreements (in exchange for a revokable "API key" as in twitter, google, et al) but rather simply asks users to follow a community-defined API Etiquette. With a large and active developer community behind it, projects using **mediawiki** benefit from a sustaining energy that keeps the project not only alive, but also entering into new territories (see for instance the **semantic mediawiki** and **wikidata** project).

-- Ward Cunningham, "how ideas move around within communities": https://en.wikipedia.org/wiki/File:Ward_Cunningham,_Inventor_of_the_Wiki.webm
-- wiki design principles: https://web.archive.org/web/20080329114249/http://c2.com/cgi/wiki?WikiDesignPrinciples
-- wiki social norms: https://web.archive.org/web/20080329071215/http://c2.com/cgi/wiki?WikiSocialNorms.
-- further reading: http://www.workingwithmediawiki.com/book/

NLTK

https://www.nltk.org/
https://www.nltk.org/book/

Another "swiss army knife", the Natural Language Toolkit (**NLTK**) is, as the name suggests, itself an assemblage of many tools, to work with "natural" (ie spoken / written) language. Providing operations such as breaking texts into lists of words or sentences, tagging grammatical

```python
# or another
  one
width =
  20
height =
  20
line =
''


for y in
  range(height):

    # here i collect all the characters for one
line
    for x in
  range(width):

        # check if the line "number" is odd or
even
        # to do this, you can use the "%", called
the "modulo"
        if y %
2 == 0:
            line +=
'???'
            line +=
' '

        else:
            line +=
' '
            line +=
'!!!'

    # print &
reset
    print(line)
    line =
''
```

```
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
???   ???   ???   ???   ???   ???   ???   ???   ???
   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!!   !!
```

```python
# Let's also make a pattern that exceeds the
  width of a single line
# Here, we will first make one long
  line ...
sentence = 'weaving with words and
  code'
line =
''

for x in
  range(100):
    line +=
  sentence
    line += '
' * x

print(line)
```

```
weaving with words and codeweaving with words
and code weaving with words and code  weaving
with words and code   weaving with words and
code    weaving with words and code     weaving
with words and code      weaving with words and
code       weaving with words and code
weaving with words and code        weaving with
words and code         weaving with words and
code          weaving with words and
code           weaving with words and
code            weaving with words and
code            weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words and
code             weaving with words
and code              weaving with
words and code              weaving
with words and code
weaving with words and
code
words and code
weaving with words and
code
weaving with words and
code
weaving with
words and code
weaving with words and
code
weaving with
words and code
with words and
code
weaving with words and
code
with words and
code
weaving with words and
code
weaving
with words and
code
weaving
with words and
code
weaving
with words and
code
weaving
with words and
code
with words and
code
weaving with words and
code
```

```
012345678910111213141516171819202122232425262728 29
012345678910111213141516171819202122232425262728 29
012345678910111213141516171819202122232425262728 29
012345678910111213141516171819202122232425262728 29
012345678910111213141516171819202122232425262728 29

# More patterns, more ways of drawing in this
  canvas ...
width =
    60
height =
    20
line =
    ''

for y in
  range(height):
      # here i collect all the characters for one
  line
      line += '-' *
  width
      line += '+' *
  width

      # print &
  reset

  print(line)
      line =
      ''
```

```
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++
----------------------------------------------------
```

---

parts of speech, or producing syntax trees: **NLTK** reflects computer sciences' decades-long relationship to computational linguistics and as such a rich site of embedded assumptions worthy of active critique. The NLTK book is a useful source with lots of examples of how you can use NLTK in **python**.

Pandoc

https://pandoc.org/
https://pandoc.org/MANUAL.html#pandocs-markdown

Pandoc is a command-line tool described as a "swiss army knife" for converting between different "markup" formats. In practice, we use pandoc primarily for its flexible ways of converting **markdown** into both **HTML** and paginated **PDF** layout. **Pandoc** connects to the longer tradition of markup and layout tools around **TeX** (e.g. **LaTeX** and **ConTeXt**), and in its most recent versions supports **CSS**-based layout engines as well (see: **WeasyPrint**) when producing paginated output. **Pandoc** supports many variations of **markdown** by using a number of optional extensions that can switched on and off when invoking the command.

Prescriptive technologies

> Technology is not the sum of the artefacts, of wheels and gears, of rails and electronic transmitters. For me, technology is a system. It entails far more than the individual material components. Technology involves organisation, procedure, symbols, new words, equation, and most of all it includes the mindset.

Ursula Franklin, The Real World of Technology, Part 1. 1989.
https://www.cbc.ca/radio/ideas/the-1989-cbc-massey-lectures-the-real-world-of-technology-1.2946845

> [… T]here are control-related technologies, those developments that do not primarily address the process of work with the aim of making it easier, but try to increase control over the operation. Think of a *word processor*. A freestanding *word processor* is indeed work-related technology. But link those *word processors* into a *work station* — that is, into a system — and the technology becomes control related. Now workers can be timed, assignments can be broken up, and the interaction between the operators can be monitored. Most modern technological changes involve control and thus new control-related applications have increased much faster than work-related ones.

> […] The distinction we need to make is between holistic technologies and prescriptive technologies. Again, we are considering technology as practice, but now we are looking at what is actually happening on the level of work. The categories of holistic and prescriptive technologies involve distinctly different specializations and divisions of labour, and consequently they have very different social and political implications. Let me emphasize that we are not asking what is being done, but how it is being done.

https://archive.org/details/the-real-world-of-technology

What are examples of such prescriptive technologies?

Publishing tools or platforms that operate in a single fixed way shape us as a (so-called) "user" and take away a large portion of agency over formats, structures and other mechanisms that shape a publication. We will explore different ways to work with/on/through technology in a more *holistic* way. To do this, we will work on re-thinking, re-learning and re-making (re)publishing *systems*.

-- How can we get comfortable with defining our own systems?
-- How can we feed our "technical imaginations" an "layout imaginations"?
-- How can we be intimate with the materials and tools we work with?
-- How can we "thicken" a publication making process, by including layers, structures and logics?

https://pzwiki.wdka.nl/mediadesign/Prototyping/2020-2021/T1#Prototyping_sessions

ReportLab

https://www.reportlab.com/opensource/
https://www.reportlab.com/docs/reportlab-userguide.pdf

A **python** library and tools for doing layout and producing **PDF** output. A (small) UK-based company that uses the open source model of maintaining a free software engine that is then extended by proprietary tools, services, and support. The core API, described in the "user guide," provides multiple ways of thinking about the page, from a low-level "canvas" model, to a higher-level layout engine called platypus based on "flowables". Offers a more hands-on DIY alternative to using **HTML/CSS** (see **WeasyPrint**), though projects/tools exist to do the same with **ReportLab** (e.g. **python3-xhtml2pdf**).

Response-ability

> There are no solutions; there is only the ongoing practice of being open and alive to each meeting, each intra-action, so that we might use our ability to *respond*, our **responsibility**, to help awaken, to breathe life into ever new possibilities for living justly.

Karen Barad, Meeting the Universe Halfway. 2007.
https://hub.xpub.nl/bootleglibrary/book/621

The ability to respond. Responsibility doesn't exist when you don't have the ability to respond, creating a situation that cannot be responded to. In that way, response-ability operates as a relational type of ethics, in which the act of responding back to a situation is key for creating a healthy environment.

https://pad.constantvzw.org/p/implicanties

Technotexts

Technotexts: When a literary work interrogates the inscription technology that produces it, it mobilizes reflexive loops between its imaginative world and the material apparatus embodying that creation as a physical presence.

Katherine N. Hayles, Writing Machines. 2002.
https://monoskop.org/images/b/bf/
Hayles_N_Katherine_Writing_Machines.pdf

TIC80

http://tic80.com

TIC80 is a free software *fantasy console*: an intersection of communities of practice bridging "retro gaming", game modding and emulation. Fantasy consoles embrace the restrictions of historical platforms to preserve a continuity of practices (and concomitant aesthetics) in contemporary game design. **TIC80** games can be published in a web-playable format, a mixture of **javascript** and Web Assembly (or **WASM**). The project was inspired by other projects, including the popular (but not libre) **PICO8**. Fantasy consoles like the **TIC80** include integrated mini tools providing an environment where one can write code, draw graphics and run and test the game.

As I began to set about the task of creating whole games, I didn't know what kind of tools existed for making graphics, and it didn't occur to me to look. It became a kind of ritual at the start of each project to improve or rewrite a sprite editor, mapper, sound editor, or any other tools I needed to make each game.

[…] Working with these [file] formats in conjunction with complementary tools gave me a taste for creating something that I vaguely understood as platforms or mediums; in the same way that games made for retro computers had a particular look and feel, the underlying platform could be treated as a separate design problem that would drive the identity of games made with them.

Joseph White (aka zep). August 2015. *A Brief History of PICO-8*, Pico Zine #1

An early sketch from Nami shows the integrated console
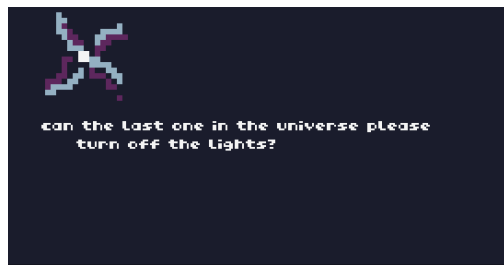
tic80-lou.png by Louisa

federicoparanoicotic.png by Federico

WeasyPrint

https://weasyprint.org/

A **python** library designed to take modern **HTML** + **CSS** (including paged media rules) and perform the layout, like a **web browser** but producing a paginated output in the form (typically) of a **PDF** file. The project has matured over the last years, as **CSS** paged media has, and serves as an alternative to other "web to print" techniques and tools based on browser engines (such as the command line tool **wkhtmltopdf** based on **WebKit**).

```
                y = 6
                y = 7
                y = 8
                y = 9
        x = 2
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 3
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 4
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 5
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 6
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 7
                y = 0
                y = 1
                y = 2
```

```
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 8
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9
        x = 9
                y = 0
                y = 1
                y = 2
                y = 3
                y = 4
                y = 5
                y = 6
                y = 7
                y = 8
                y = 9

# A loop in a loop + collecting characters in a
    line...
width =
    30
height =
    20
line =
    ''

for y in
    range(height):
    for x in
    range(width):
        line +=
    str(x)

    print(line)
    line = '' # try to comment this line out, to
    see the difference

0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
0123456789101112131415161718192021222324252627282 9
```

sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs

mini-exercise: try to replicate the other typewriter study yourself now!

```
# Let's continue a bit with this "canvas-mode" of
   working,
# and let's bring the random function to the
   table again.


# In order to slowly build a canvas of
   characters,
# we will use a variable (called 'line' in this
   case) to temporary save our line ...


import
   random
characters = ['▒','▓','▚'
   ,'▐','▌','▙']
width =
   100
height =
   30
line =
   ' '

for y in
  range(height):
    for x in
  range(width):
        line +=
  random.choice(characters)

  print(line)
    line =
```

```
# A loop in a
   loop?
width =
   10
height =
   10
for y in
  range(height):
    print('x
  =', x)
    for x in
  range(width):
        print('        y
  =', y)

x = 0
        y = 0
        y = 1
        y = 2
        y = 3
        y = 4
        y = 5
        y = 6
        y = 7
        y = 8
        y = 9
x = 1
        y = 0
        y = 1
        y = 2
        y = 3
        y = 4
        y = 5
```

Research logs

Exercise: "Active Essay" / writing about / documenting your research. An example of an "active essay" (now broken essay, 1996): http://www.playfulinvention.com/emergence/index.html.

What role could documentation and publishing-on-the-go play in a research trajectory?

-- *Hidden Worlds of Digital Voices* by Kendal, https://hub.xpub.nl/sandbot/~kendalb/SI15/notes.html
-- *PTOLEMY* by Floor, https://hub.xpub.nl/sandbot/~floorvanmeeuwen/15/sonicpi/
-- *vocoder* by Euna, https://hub.xpub.nl/sandbot/~eunalee/triphonic/vocoder.html
-- *childbirth* by Nami, https://hub.xpub.nl/sandbot/~namikim/indexinputs/childbirth.html
-- *The relation between Sound and Silence* by Pongie, https://hub.xpub.nl/sandbot/~pongie/md-annotations/page01.html
-- *An Instrument based on Web responsiveness* by Martin, https://hub.xpub.nl/sandbot/~foucaut/SI15/martin.html
-- *According to Belgians Royal Museum of Arts and History's Web Interface Carmentis, which allows impressively detailed research possibilities, the entire museum collection was sub-labeled by 598 different cultures.* by Pongie, https://hub.xpub.nl/sandbot/~pongie/pronunciation/phonetic.html
-- *Fuck wordpress' hegemony* by Poni, https://hub.xpub.nl/sandbot/~poni/SI15/
-- *♪ mixing anxiety ♪* by Clara, https://hub.xpub.nl/sandbot/~clara/
-- *sometimes when I talk* by Camilo, https://hub.xpub.nl/sandbot/~camilo/

Social shell, A FEMINIST NET/ WORK, sandboxes, and Jupyter Pis
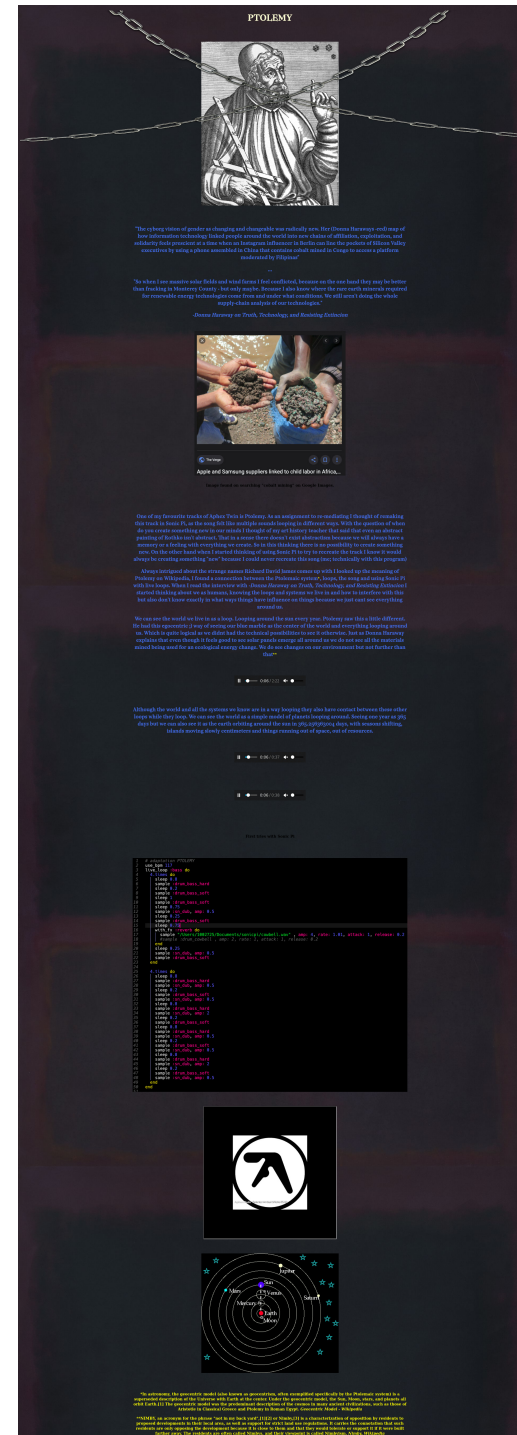
Working with a shared server became an important way to stay together in a year so shaped by lock-downs and remote learning. The shared environment of the academy was accessible for students, but not for teachers for the bigger part of the year. Having access to a shared server, which was physically in the room as it has been hosted from the XPUB studio, gave us the possibility to still inhabit a shared space.

What did this shared server enable? Which socialities did it shape? And how has the setup of the server been shaped by the social dynamics in which it operates?

The shared server of XPUB1 brings back a memory from the summer of 2014, when the summer school Relearn was organised by Constant & Open Source Publishing (OSP) in Brussels. One of the so called "tracks" of that year was shaped around the idea of the **social shell**, a term that Anne Laforet and Michael Murtaugh proposed, embracing the shell (a.k.a. terminal or command line) as a social environment:

Relearn is a collective learning experiment with as many teachers as it has participants. It is motivated by the possibility to displace parameters of/for research, studying and learning. Relearn appeared in 2013, 2014, 2015 (Brussels), 2016 (Calafou), 2017 (Rotterdam), 2019 (Rotterdam, Brussels, Paris), 2021 (Rotterdam). http:// relearn.be/

A Social Shell & Mesh Cookbooks proposes to deploy a network of small server nodes (via Raspberry Pi, Olimex, or possibly installed on individual's laptops) during the Summer School session. The software provides a web-based commandline/shell environment (a social shell) in which commandline programming is enhanced by "dynamic" manual pages provided by a live "cookbook" or collection of stored "recipes"/ scripts (the mesh cookbook). The cookbooks are distributed in that each node runs independently (and is specific to the group using it) but whose contents may be "merged" (itself a social operation) with others to form a distributed (code) documentation system for the Summer School event.

http://activearchives.org/wiki/A_Social_Shell_%26_Mesh_Cookbooks

Each room in this edition of Relearn was installed with a Raspberry Pi which generated a local hotspot. All the pi's together formed a local "internet" that was used to communicate across rooms. This feminist net/work announced itself in the format of **A FEMINIST NET/ WORK HOW-TO**:

Setup local server(s) & make our own network infrastructure, rather than "fixing the problems of internet access" by paying for increased bandwidth,

Prefer "read/write" networks to those that "just work",

Prefer pocket servers to those in the clouds,

Embrace a diversity of network topologies and different scales (machine to machine, local nodes, institutional infrastructure) and consider the implications of working with each, rather than a Web 2.0 model where resources must be uploaded onto "the 24/7 Internet (tm)" in order to share them,

Invite users to look critically at the implications of any infrastructural decisions, rather than imagining utopic and/or "killer" solutions,

Make that which is normally hidden and invisible (tending to surveillance), explicit and shared (as a gesture of collective authorship), for instance: instead of caching web resources (silently), we imagine services to archive pages and share them locally as networked cookbooks, rather than logging IRC conversations on a server or database accessible only by administrators, we imagine (local) logs available for reading / editing by participants and published conditionally.

The provided network is a starting point; during summer school the network topology can be tweaked, changed, extended. To this end, participants are encouraged to bring network devices (pirate box / openwrt routers / pi's / olimex / other obscure objects with ethernet ports). – http://relearn.be/2014/

In parallel Aymeric Mansoux was working with the **sandbox** as a figure to think about community embedded code practices at the time he was writing his PhD at Goldsmiths. Published in 2017 under the title *Sandbox Culture, A Study of the Application of Free and Open Source Software Licensing Ideas to Art and Cultural Production*. Here the **sandbox** is used as a term "to describe free cultural mechanisms, where software and legal code become a dual liberating and constraining constituent device for different communities to experience varying ideologies and practices" (p. xxvii). The sandbox as a shared system that brings together different people with different habits and different ways of working. A few lines further, the sandbox is connected to a specific interest in subcultures:

I also hope that the notion of cultural sandboxing can contribute a new way to approach and discuss post-subcultural dynamics, that cannot be easily analysed with existing static subcultural models in the context of groups that mixes operating systems with social systems. (p. xxvii)

Sandbox Culture, A Study of the Application of Free and Open Source Software Licensing Ideas to Art and Cultural Production,

---

```
40
50
60
70
80
90

# Now... write a loop in  a
  loop
for x in
  range(1,6):
    for y in
  range(1,6):

  print('x'*x,
  'y'*y)

x y
x yy
x yyy
x yyyy
x yyyyy
xx y
xx yy
xx yyy
xx yyyy
xx yyyyy
xxx y
xxx yy
xxx yyy
xxx yyyy
xxx yyyyy
xxxx y
xxxx yy
xxxx yyy
xxxx yyyy
xxxx yyyyy
xxxxx y
xxxxx yy
xxxxx yyy
xxxxx yyyy
xxxxx yyyyy
```

ASCII canvas

A not-too-big next step is to start generating patterns.

These varied experiments in articulation are to be understood not as an end in themselves but merely as a help to us in gaining new terms in the vocabulary of tactile language. (On Weaving, Anni Albers (1965))

Anni Albers - Typewriter Studies

```
# We could replicate Anni's typewriter study
  now:
width =
  100
height =
  20
for y in
  range(height):
    print('sS'
  * 50)
    print('_' *
  width)

sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
_____
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
_____
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
_____
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
_____
sSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSsSs
_____
```

words with and code words
words words and weaving with
and and and and code
words code weaving with weaving
code code code with weaving
and weaving weaving and with
weaving with words with and
and words and weaving and
words weaving code code weaving
code words words with code
weaving with words and code
weaving and and words weaving
and weaving weaving and words
with code weaving weaving and
with words words code with
code with weaving and weaving
weaving weaving weaving with with
weaving and code code with
words weaving and and weaving
weaving words weaving weaving weaving
with and with weaving and
and weaving with code code
code words words and code
weaving weaving weaving words and
and words words with and
and code code code weaving
weaving weaving words weaving with
words and code with code
weaving words and code code
and words words words words
weaving and code code weaving
with with words weaving and
words code weaving weaving code
code words words weaving weaving
words words code code words
words weaving with and and
with and and with and
with code code with words
with code code with words
words words and words code
code weaving words code code
weaving words and with with
with with weaving words and
weaving weaving weaving words code
and and weaving weaving weaving
code words code weaving weaving
with code and and code
with weaving weaving code words
and code weaving words with
and words words weaving weaving
with code with with code
weaving weaving code words words
and and and and words
words with weaving code weaving
words code with with code
code with with code code
and weaving and words code
code code with words and
code code and and words
and with words weaving weaving
with weaving with with and
words and words and with
weaving code words code words

with weaving with words words
weaving words and with and
code words and words and
and weaving weaving code code
with words and code words
with code weaving code code
code words code weaving and
words words code code weaving
weaving code and with and
with weaving with weaving words
with weaving and words and
and weaving words code and
with weaving with and code
weaving words with with words
words with weaving words and
weaving and words words words
with words weaving and and
weaving code weaving and words
weaving words weaving with code
with words and weaving weaving
code words with with code
weaving words words and weaving
and words words with and
and with code weaving code
code weaving words and weaving
with weaving words and and
code weaving and weaving and
with weaving code with weaving
and weaving code code with
and code words weaving words
and words weaving with code
words words weaving words with
code with with code and
words with code and code
code and and words with
with with words weaving and
words with weaving words and

```python
# You can use range also differently, for example
  to loop in between two numbers ...
for number in
  range(3,10):

  print(number)


3
4
5
6
7
8
9

# ... or with bigger
  steps:
for number in
  range(0,100,10):

  print(number)


0
10
20
30
```

Aymeric Mansoux (2017)
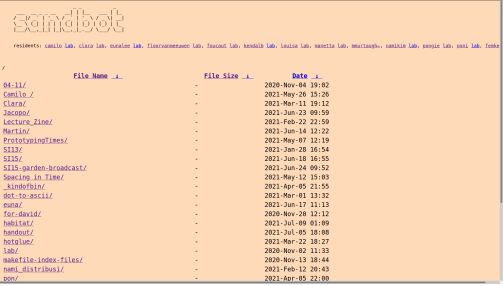https://www.bleu255.com/~aymeric/dump/aymeric_mansoux-sandbox_culture_phd_thesis-2017.pdf

Jupyter Pi

Following a model established by a "sandbox" server, a Raspberry Pi setup and initially hosted by tutor André Castro, and made public via a tinc-based VPN, a similar server system was established, the "sandbot", for use the the incoming first year students. In both cases, the idea was to provide the experience of sharing a digital server, each student having "super user" access and thus sharing the role of administrator. While we had experiences with self-contained server installations centered on the etherpad software (see: etherbox), the idea was to better support the use of python programming in the course by using the Jupyter Notebook / Jupyter Lab interface (see the Toolbox section).

Etherbox: https://networksofonesown.constantvzw.org/etherbox/

Initially we tried to use the TLJH (aka the littlest jupyter hub, see: https://tljh.jupyter.org/) software – but found it not "little" enough; the first install was extremely unstable due to insufficient resources with our dozen or so users. We replaced it with a simpler configuration using "normal" UNIX user accounts each running separate instances of the jupyter server (one for each user, running as that user, thus re-using the inherant UNIX system of users and permissions).

Quickly we realized that the jupyter lab provided a number of important features that went beyond its python roots: an integrated shell, making a full command line interface available via a web browser, and extensive file handling capabilities (new files can be uploaded with drag and drop, files and folders can be renamed and moved, an integrated file editor interface exists with syntax coloring for many kinds of languages like HTML and CSS). As a result, the "lab" interface became a kind of trojan horse and a means of students comfortably "discovering" many aspects of a linux server, and providing a means to collaboratively publish with a variety of means and workflows.

https://hub.xpub.nl/sandbot/

Jupyter Pi: https://git.xpub.nl/XPUB/jupyterpi

```
c.NotebookApp.base_url = '/sandbot/
  ~{{username}}/__lab__/'
c.NotebookApp.port =
  {{port}}
c.NotebookApp.trust_xheaders =
  True
c.NotebookApp.port_retries =
  0
c.NotebookApp.password =
  '{{password_sha1}}'
c.NotebookApp.allow_remote_access =
  True
```

Template for the jupyter lab configuration for each user, each student is assigned a unique port for their server

```
{% for user in
  users %}
location ^~ /~{{user.username}}/
  __lab__/ {
    proxy_pass http://localhost:
{{user.port}}/;
    include /etc/nginx/includes/
lab.conf;
}
{%-
  endfor
  %}
```

Jinja loop that generates the nginx *reverse proxy* configuration providing each user with a convenient URL for their lab interface

```
cat "/etc/systemd/system/
  jupyterlab@.service"
[Unit]
Description=Jupyter Lab Server
  (%i)
After=nginx.service

[Service]
User=%i
Group=%i
Type=simple
WorkingDirectory=/home/
  %i/
ExecStart=/usr/local/bin/jupyter-lab --config=/
  home/%i/.jupyter/jupyter_notebook_config.py
StandardOutput=null
Restart=always
RestartSec=10

[Install]
WantedBy=multi-
  user.target
```

A systemd "template" to manange starting/stopping each jupyer notebook instance

What is missing?

Just a note to remember that it is a pity that it is difficult to see each others notebooks on the jupyterpi. One thing that was added is a "shared" system link to the /var/www/, as a space where you can open each others notebooks. This both worked and didn't work, as not everyone was making use of

this space and when multiple people work in the same notebook at the same time, it creates a conflict.

ISSUE #8: Filed by Manetta on our self-hosted git(ea) titled *Togetherness on the jupyterpi*, https://git.xpub.nl/XPUB/jupyterpi/issues/8

Following Ursula Franklin's notion of prescriptive technology, we see how each feature of *collaboration* software proposes enforces a concomitant model of *conflict.*

With jupyter lab, the integration of the shell powerfully allows an unbounded multitude of external tools and methods to be used. For instance **git** can be used to manage versions, and to pull and push code to and from remote servers. Tools like **tmux** can provide realtime shared experiences via a shared shells. However, in the notebooks themselves there is a clear assumption of a single user writing and executing the code in a notebook in isolation. Attempts to open the same notebook on the server leads to "permissions errors" due to the tool being used in an unanticipated way. Sharing notebooks in practice is typically done afterwards by publishing with the assumption that each reader will performs the code for themselves in their isolated notebooks. While this fits many situations (certainly the model of academic publishing), other possibilities, like the "happy accidents" that can occur when sharing an etherpad or with "pair programming" are made difficult. How might *shared notebooks* support others forms of sociality?

Additionally when tools, or "boxes" make things easy, one always needs to consider what is made difficult in the process.

How do you balance providing easy access, with making sure people still learn the necessary skills to maintain access in the future for themselves, for instance when self-hosting? How do decisions get made the eventual archiving and preservation of digital work? Rather than solving these problems, shared "sandbox" servers can usefully create situations where difficult questions can be addressed first hand in a collective way.

patterns-generating.ipynb

https://git.xpub.nl/XPUB/S13-Words-for-the-Future-notebooks/src/branch/master/pdf-ascii-quilting/patterns-generating.ipynb

Generating patterns

Generating patterns, weaving & textile design.

The structure of a fabric or its weave — that is, the fastening of its elements of threads to each other — is as much a determining factor in its function as is the choice of the raw material. In fact, the interrelation of the two, the subtle play between them in supporting, impeding, or modiying each other's characteristics, is the essence of weaving. (p. 38)

Anni Albers - On Weaving (1965), https://monoskop.org/images/7/71/Albers_Anni_On_Weaving_1974.pdf

Red-Geen Slit Tapestry, Gunta Stölzl (1927/28)

Re-turning (to): Variables, Lists & Loops

```python
words = ['weaving', 'with', 'words',
 'and', 'code']
```

```python
# First a simple loop through the
  list
for word in
  words:

    print(word)
```

```
weaving
with
words
and
code
```

```python
# Then, a loop in which we start to play with
  random again
import
  random
for word in
  words:
    print(random.choice(words),
  random.choice(words), random.choice(words),
  random.choice(words), random.choice(words))
```

```
code weaving weaving code words
with with and code and
code code weaving and with
and weaving words code and
and with words words weaving
```

```python
# How to work with more iterations of the
  loop?
# For example
  100?
# You can use
  "range"
range?
```

```
Init signature: range(self, /, *args, **kwargs)
Docstring:
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of
integers from start (inclusive)
to stop (exclusive) by step.  range(i, j)
produces i, i+1, i+2, ..., j-1.
start defaults to 0, and stop is omitted!
range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list
of 4 elements.
When step is given, it specifies the increment
(or decrement).
Type:           type
Subclasses:
```

```python
# Make a loop that starts at 0 and ends at 99
  (100 iterations)
import
  random
for number in
  range(100):
    print(random.choice(words),
  random.choice(words), random.choice(words),
  random.choice(words), random.choice(words))
```