

メディアアート・プログラミング2

東京藝術大学 芸術情報センター開設科目 後期金曜4限 第1週

2023.10.06 松浦知也 (matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com)



Google Classroom

ujriogo

履修人数的に抽選は無し、
CampusPlanでの登録を忘れずに



自己紹介

- 神奈川県茅ヶ崎市出身(1994)
- 東京藝術大学音楽環境創造科卒(2017)
- 九州大学大学院芸術工学府 修士、博士後期課程(2017~2022)
- 東京藝術大学芸術情報センター 特任助教
(2022~)
- YCAM(Intern,2015),teamlab(2015~2017),
School for Poetic Computation(2018)

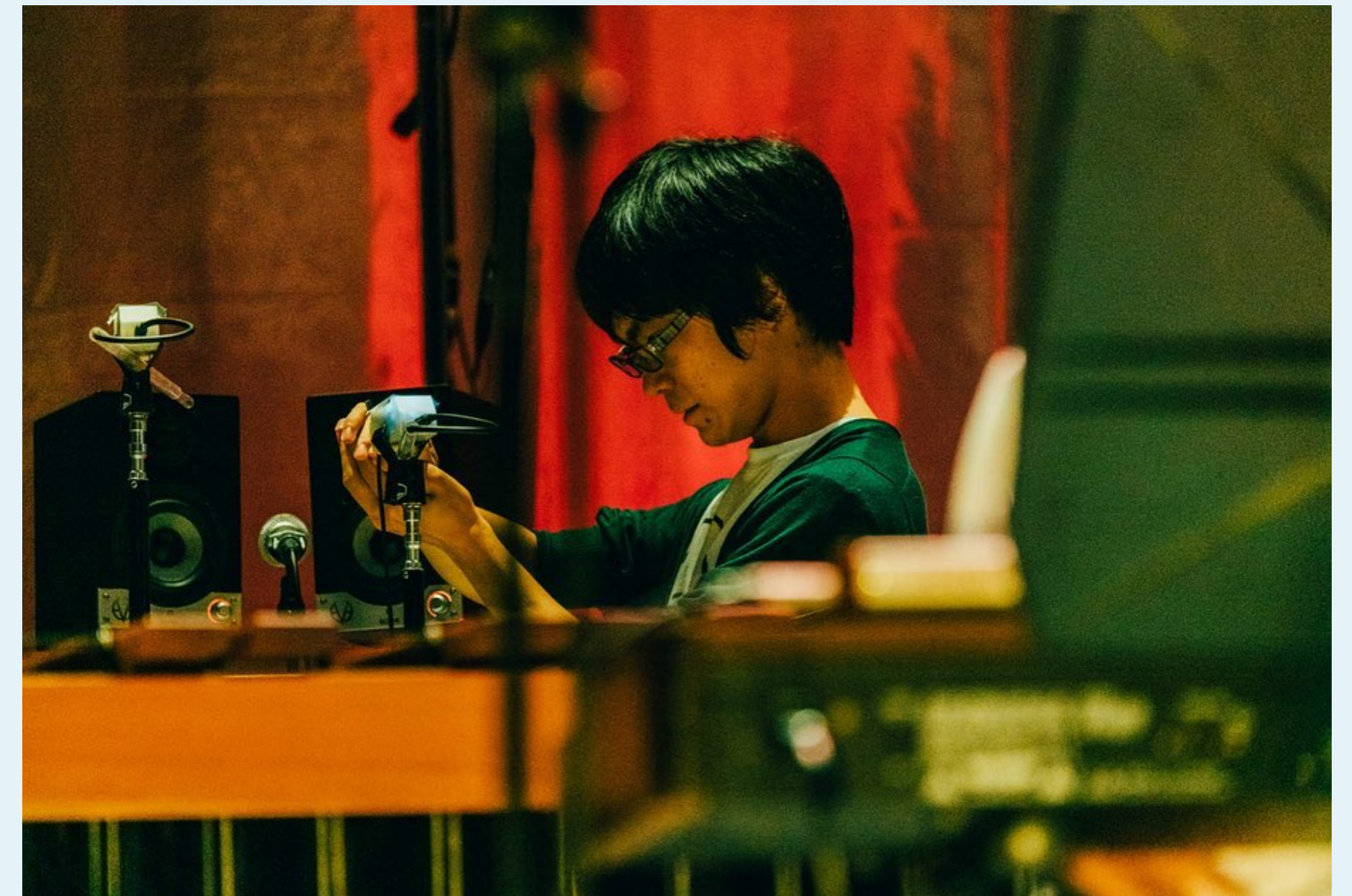


photo: Takehiro Goto

電子楽器を作って演奏したり、展示のための音プログラミングをしたり、音のためのプログラミング言語を作ったりしています。


<https://matsuuratomoya.com>

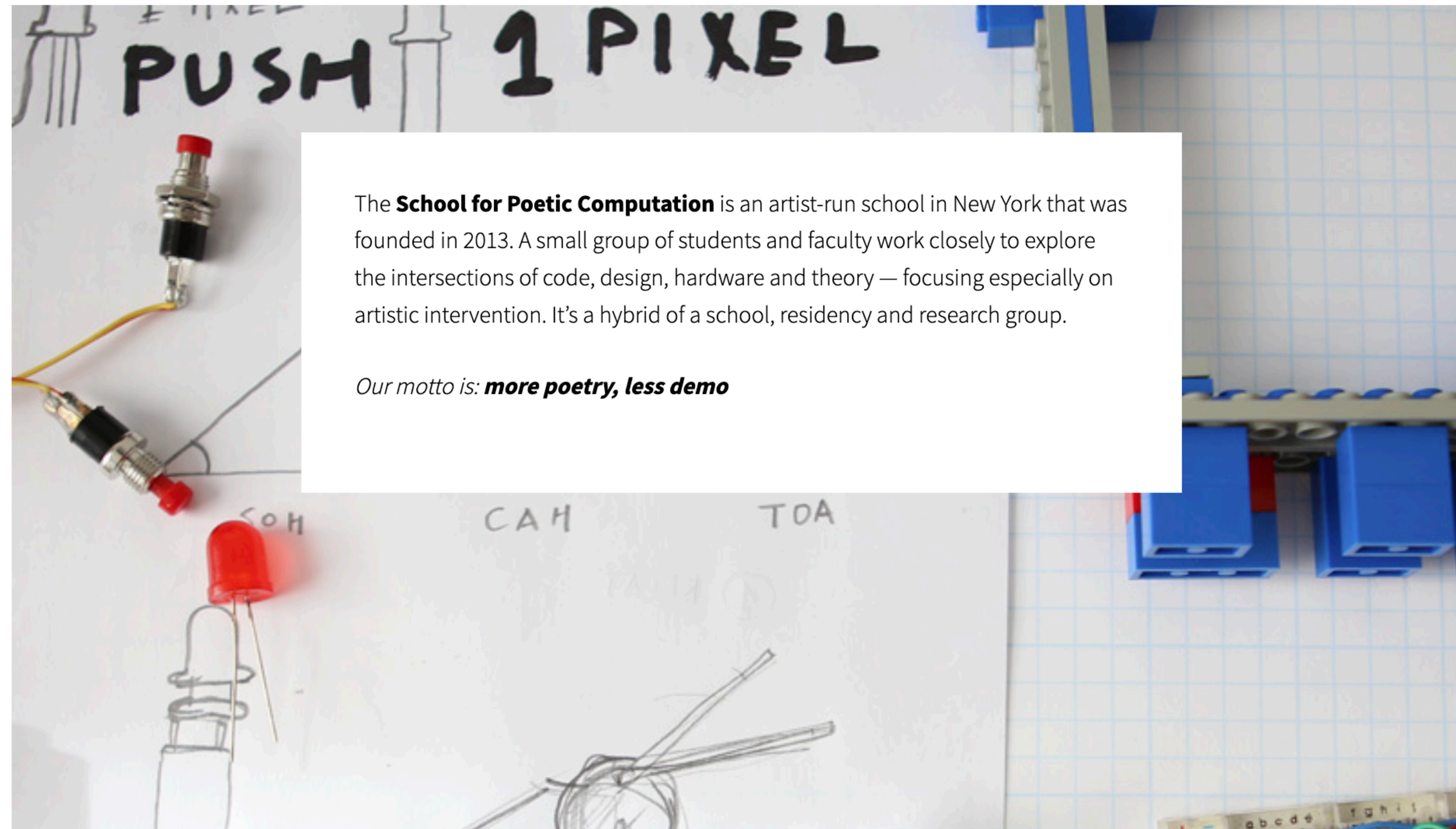
School for Poetic Computation

School for Poetic Computation

Participate:
Fall/Winter 2021-22
COCOON Summer
2021
Our Transition

Mission
People
Classes
Press
FAQ

Instagram 
Twitter 
Newsletter 
Blog
Finances



<https://medium.com/sfpc/sfpc-fall-2018-1%E9%80%B1%E7%9B%AE-6fc844be3d83>



SFPC Summer 2019 in Yamaguchi <https://vimeo.com/363716118>
<https://potari.jp/2019/12/3673/>

この授業で取り扱う内容

- プログラミングを通じて、テクノロジー批評の基礎を身につける
- 映像や音声など、コンピューターを手段として用いる作品よりも、プログラミングやコンピューターを使うこと自体を表現のテーマにしたものを扱う

詩

自己言及（再歸）

無駸

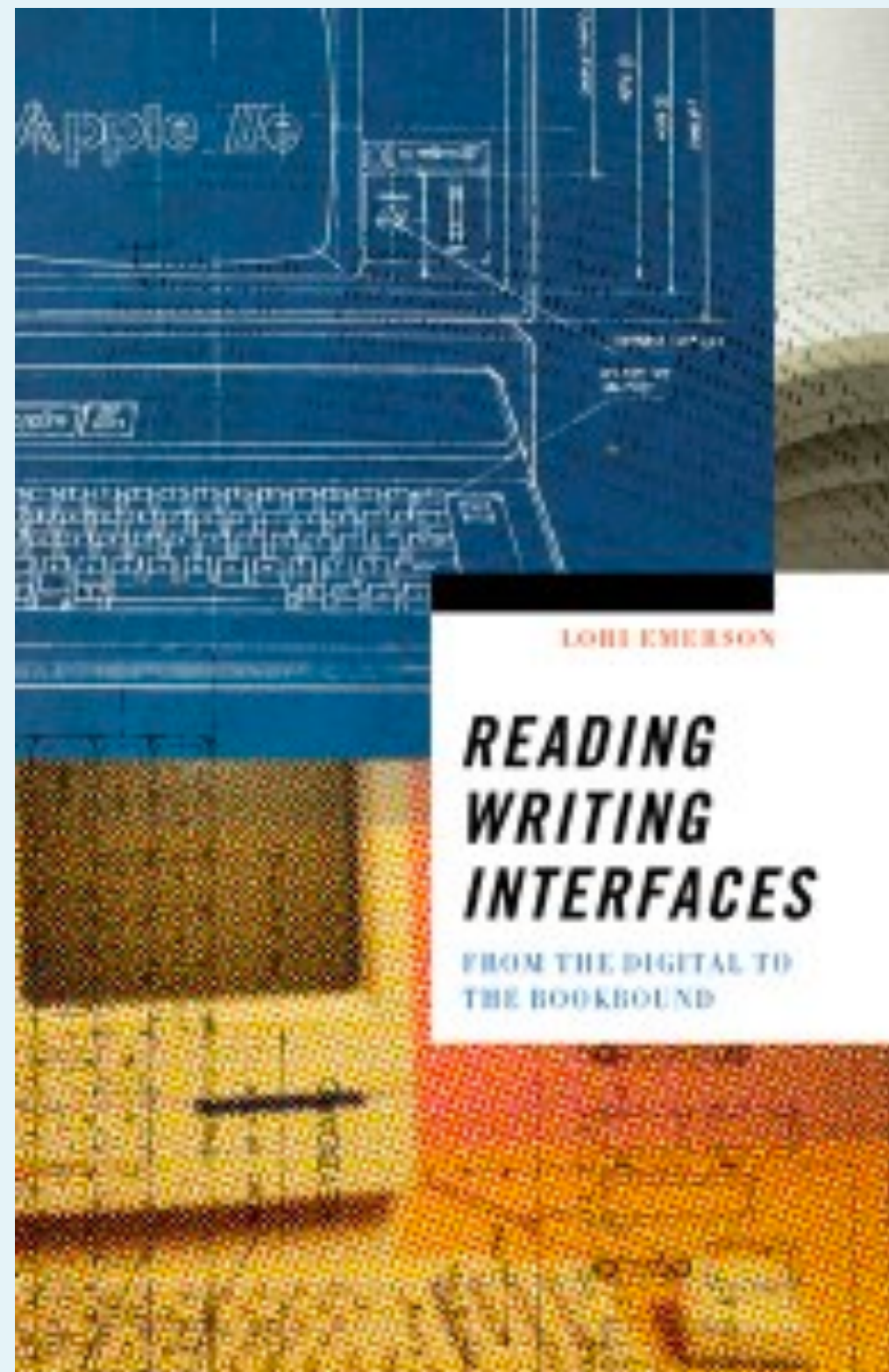
公共財

参考資料



「遙かなる他者のためのデザイン 久保田晃弘の思索と実装」(2017) 久保田 晃弘、BNN

參考資料



Reading Writing Interfaces: From the Digital to the Bookbound(2014)

Lori Emerson , University of Minnesota Press

<https://www.upress.umn.edu/book-division/books/reading-writing-interfaces>

参考文献

The screenshot shows the journal's interface. At the top left is the Duke University Press logo. A search bar is present with 'Critical AI' selected. The main header is a red bar with 'critical AI' in white, and navigation links for 'ISSUES', 'FOR AUTHORS', 'ALERTS', 'PURCHASE', and 'ABOUT'. The article title 'Scrapism: A Manifesto' is marked as 'FREE'. The author 'Sam Lavigne' and issue information 'Critical AI (2023) 1 (1-2)' are listed. A DOI link is provided. Below the title are options for 'Standard View', 'Share', and 'Tools'. The abstract text is displayed in a white box. On the right, there are icons for 'Contents', 'Data & Figures', 'References', and 'Related'. Below these is the issue information 'Volume 1, Issue 1-2' and the date 'October 1, 2023'. A large image of a hand holding a brush is shown at the bottom right.

DUKE UNIVERSITY PRESS

Search... Critical AI Advanced Search Register Sign In

critical AI

ISSUES FOR AUTHORS ALERTS PURCHASE ABOUT

RESEARCH ARTICLE | OCTOBER 01 2023

Scrapism: A Manifesto **FREE**

Sam Lavigne

Critical AI (2023) 1 (1-2)

<https://doi.org/10.1215/2834703X-10734046>

Standard View Share Tools

Abstract

Web scraping is a technique for automatically downloading and processing web content or converting online text and other media into structured data. This article describes the role that web scraping plays for web businesses and machine learning systems and the fundamental tension between the openness of the web and the interests of private corporations. It then goes on to sketch an outline for “scrapism,” the practice of using web scraping for artistic, critical, and political ends.

Issue Section: [Articles](#)

Contents Data & Figures References Related

Volume 1, Issue 1-2

October 1, 2023

Scrapism: A Manifesto, Sam Lavigne, critical AI vol.1 issue 1-2, 2023

1. [Scrapism](#)
2. [Intro to the Command Line](#)
3. [Intro to Python](#)
4. [Web Scraping Basics](#)
5. [Using Real Browsers](#)
6. [Scraping XHR](#)
7. [Downloading & Editing Media](#)
8. [Using Scrapy](#)
9. [Readings & Resources](#)

Scrapism

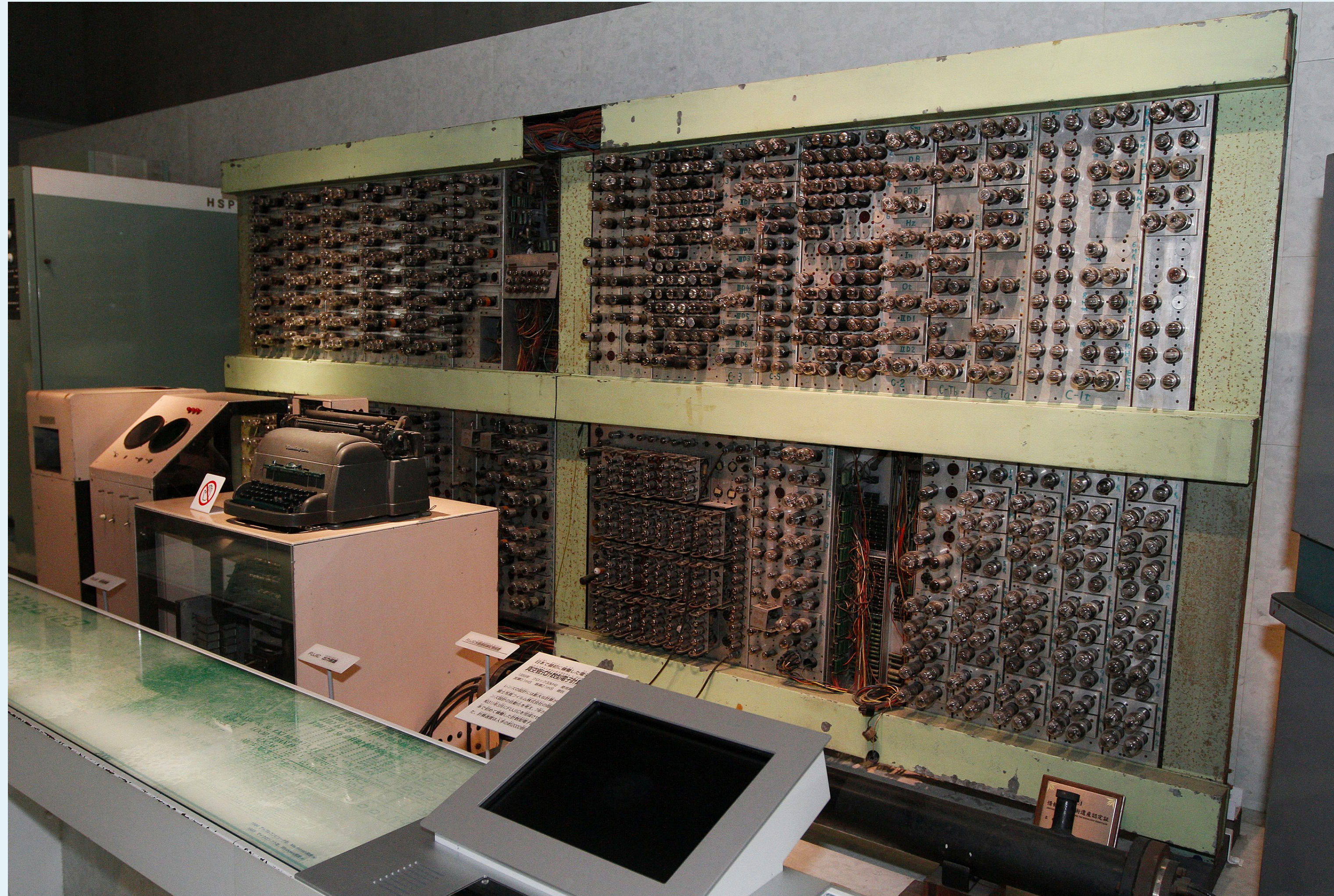
Web scraping describes techniques for automatically downloading and processing web content, or converting online text and other media into structured data that can then be used for various purposes. In short, the user writes a program to browse and analyze the web on their behalf, rather than doing so manually. This is a common practice in silicon valley, where open html pages are transformed into private property: Facebook began as a (horny) web scraping project, as did Google and all other search engines. Web scraping is also frequently used to acquire the massive datasets needed to train machine learning models, and has become an important research tool in fields such as journalism and sociology.

I define "scrapism" as the practice of web scraping for artistic, emotional, and critical ends. It combines aspects of data journalism, conceptual art, and hoarding, and offers a methodology to make sense of a world in which everything we do is mediated by internet companies. These companies surveill us, vacuum up every trace we leave behind, exploit our experiences and interject themselves into every possible moment. But in turn they also leave their own traces online, traces which when collected, filtered, and sorted can reveal (and possibly even alter) power relations. The premise of scrapism is that everything we need to know about power is online, hiding in plain sight.

This is a work-in-progress guide to web scraping as an artistic and critical practice, created by [Sam Lavigne](#). I will be updating it over the coming months! I'll also be doing occasional live demos either on Twitch or YouTube.

<https://scrapism.lav.io/>

今あえてコンピューターを
テキストで操作することを考える



FUJIC(日本で最初のコンピューター、上野の科学博物館で展示されています)

Morio - 投稿者自身による著作物, CC 表示-継承 3.0, <https://commons.wikimedia.org/w/index.php?curid=12663684>



FUJIC(日本で最初のコンピューター、上野の科学博物館で展示されています)

Morio - 投稿者自身による著作物, CC 表示-継承 3.0, <https://commons.wikimedia.org/w/index.php?curid=12663684>

テレタイプからターミナル



File:DEC VT100 terminal.jpg

Jason Scott, CC BY 2.0, via Wikimedia Commons,

https://commons.wikimedia.org/wiki/File:DEC_VT100_terminal.jpg

Xerox PARC 暫定版Dynabook (1972~1979)

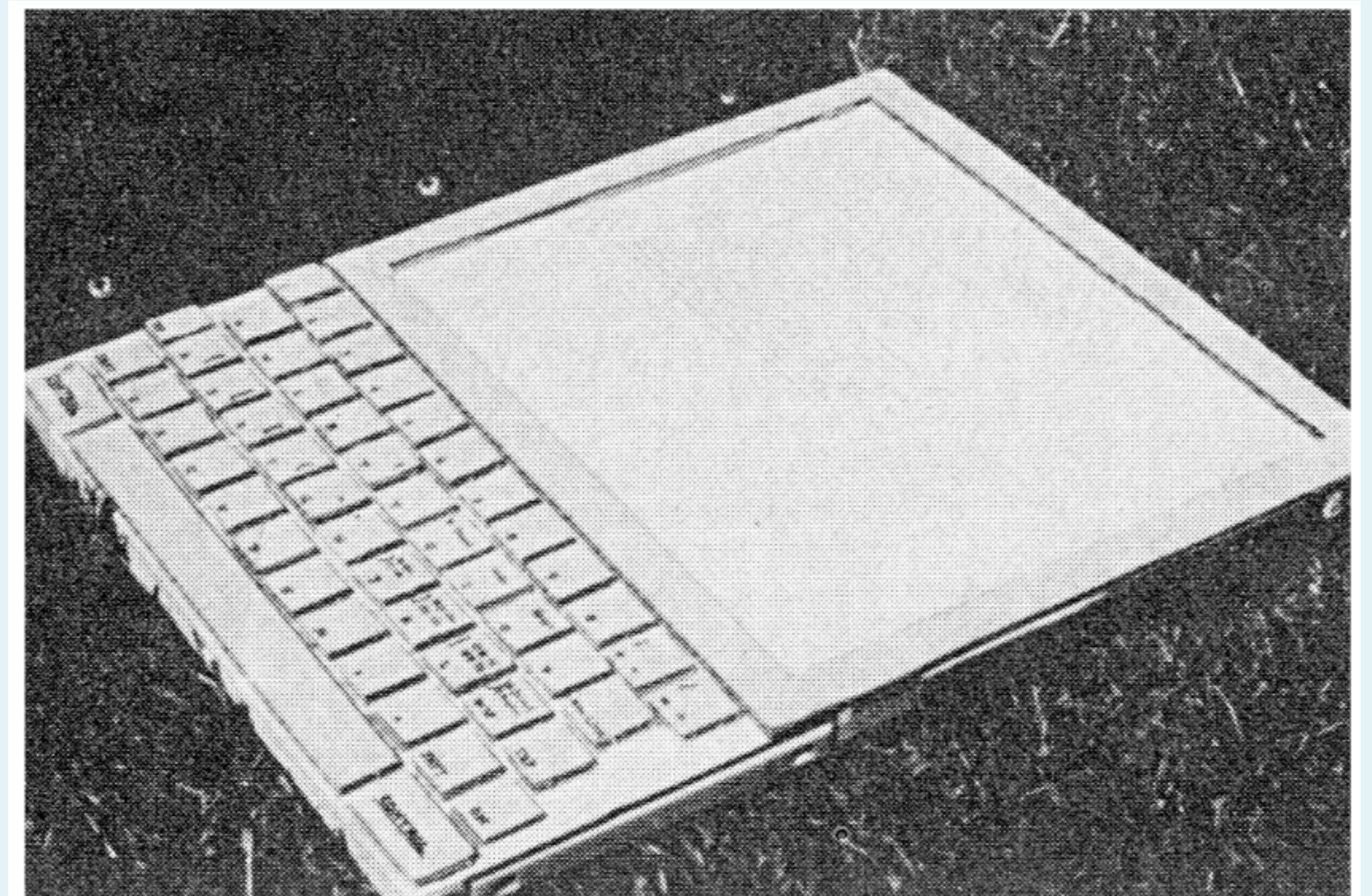


Figure 26.2. Mock-up of a future Dynabook.

Kay, Alan. 1979. "Programming Your Own Computer." In Science Year 1979. World Book Encyclopedia.

26. Personal Dynamic Media

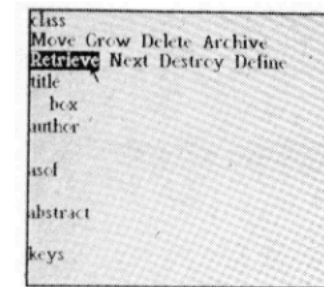


Figure 26.9. Retrieval in this filing tool is carried out by pointing to the command in the documents menu. The system will find every document with the title "box."

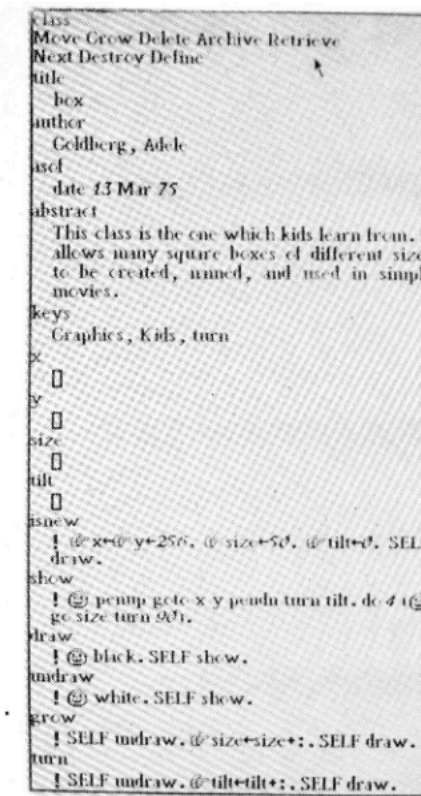


Figure 26.10. Here is a retrieved document that represents a description of a Smalltalk class definition.

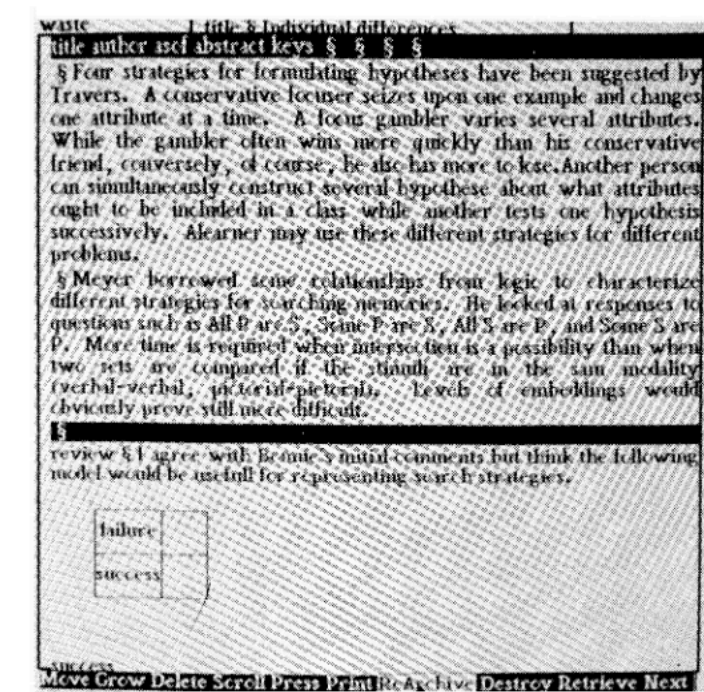


Figure 26.11. This is a document from an annotated bibliography for teachers. Details are suppressed but can be expanded by pointing to names in the black fields. Documents can also contain diagrams.

theNEWMEDIAREADER

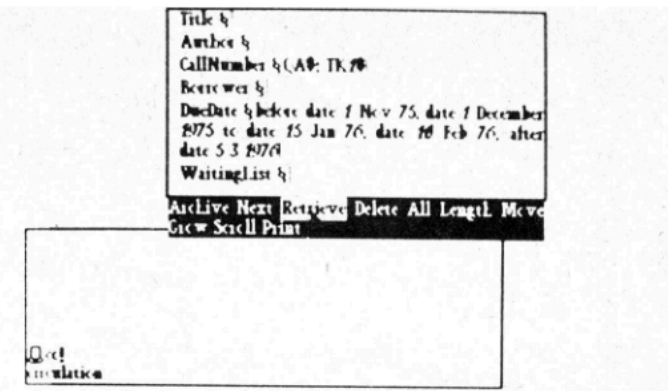


Figure 26.12. This retrieval request combines incomplete call numbers with date ranges. The example is taken from an experimental library system.

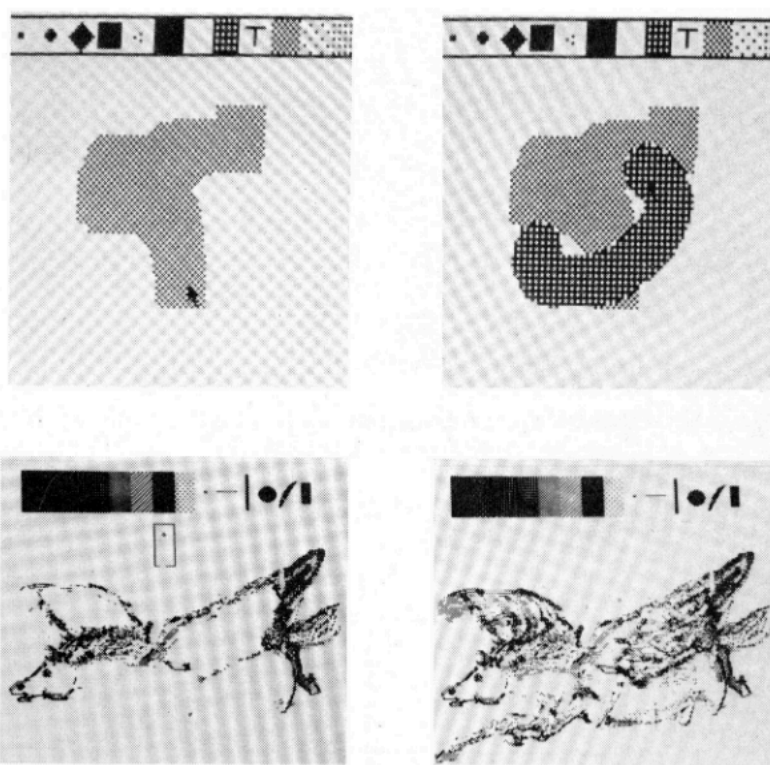


Figure 26.13. A sketch of Pegasus is shown being drawn with a Smalltalk drawing tool. The first two pictures in the sequence show halftone "paint" being scrubbed on.

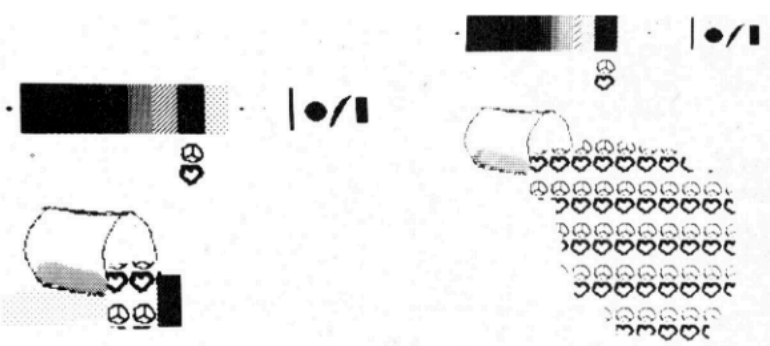
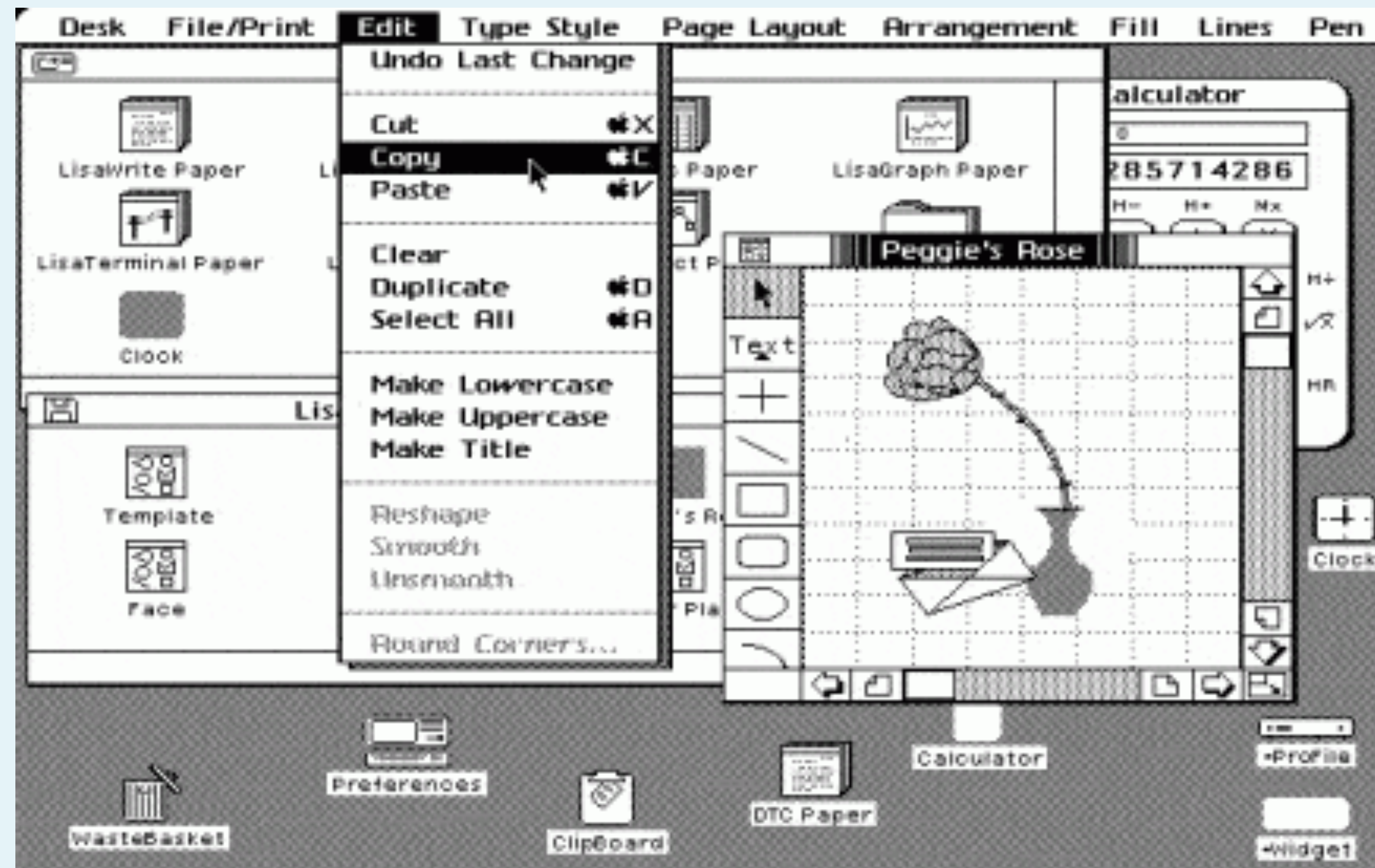


Figure 26.14. A sketch of a heart/peace symbol is created and used as a paint brush.

Apple Lisa (1983)

現実世界のメタファーへ



Apple Lisa by Christoph, Mac History, 2007

<https://www.mac-history.net/2007/10/12/apple-lisa/>



Write a script to automate sending daily email reports in Python, and walk me through how I would set it up.



Certainly! Automating the sending of daily email reports in Python can be accomplished using various libraries, but one of the most commonly used ones is `smtplib` for sending emails and `schedule` for scheduling the script to run daily. Additionally, you can use `pandas` for data manipulation and `email` for composing emails. Below is a step-by-step guide and a sample Python script to automate this process.



Step 1: Install Required Libraries

Make sure you have the necessary libraries installed. You can install them using pip:

```
pip install pandas
pip install schedule
```

Copy code

Step 2: Create Email Account

You'll need an email account to send emails from. You can use a Gmail account for this purpose.

Regenerate



Step 3: Generate App Password (Gmail)

Send a message



履修上の注意点

スケジュール1

- 1. (10/6) イントロダクション
- 2. (10/13) 10 PRINT
- 3. (10/20) テキスト
- 4. (10/27) グリッチ入門
- 5. (**10/31(火)**) 難解プログラミング言語1
- 6. (11/10) 難解プログラミング言語2
- 7. (11/17) プログラミング言語を作ってみる1

スケジュール2

- 8. (11/24) プログラミング言語を作ってみる2
- 9. (12/1) コンピューターに詩を書かせる (自然言語処理)
- 10. (12/8) インターネットと公共財1 (HTMLとWeb)
- 11. (12/15) インターネットと公共財2(Scrapism)
- 12. (12/22) インターネットと公共財3(ゲスト講義?)
- 13. (1/5) 課題制作1
- 14. (1/22) 課題制作2
- 15. (1/26) 講評・ふりかえりとまとめ

授業全体の課題：

コンピューターを無駄遣いしたプログラムを作成すること

課題

コンピューターを無駄遣いしたプログラムを作成すること

- 例えば：
- Or、授業内で出された小課題の延長
- 自由制作ももちろんOK
 - ただ、希望としてはなるべく自分の本業と関係のないものを作ること
- 具体例は授業内で少しずつ紹介していきます。
- 技術的な難易度よりも、なぜそれを作ったのか伝わるように

成績評価

- 成績は出席点を30%、授業中に数回提示される小課題点を30%、最終課題点を40%として評価する。
- 出席点は毎回授業の最後にGoogle Formでの感想、質問シートの提出をもってカウントする。必ず毎回その回でわからなかったことや気になっている内容について最低1つ質問してください！次の授業でフォローアップします。
- あらゆる課題について、無条件で、未提出は0点、期限を過ぎての提出は減点する。
- 課題を含めた授業関連のやりとりには全てClassroomを利用します

機材関連

- AMC演習室のコンピューターのデータはシャットダウンすると削除され、設定もリセットされます。
- データを保管しておきたい人は、ネットワーク上の“shared”の中のコードとデザインのフォルダに、自分の名前のフォルダを作成して保存してください。
- ただし、他の人からも見える&編集できてしまうので、USBメモリ等を持参するのをお勧めします

Homebrewのインストール (macOS向け)

Homebrewとは？

- macOS向け、コマンドラインツールのパッケージマネージャ
 - 有志で管理されてるApp Storeみたいなもの
 - 膨大な数のパッケージがコミュニティによってメンテナンスされている
- 基本的にはコマンドラインツールのみだが、GUIツールも主要なものはインストール可能



よく使う項目

AirDrop

最近の項目

アプリケーション

tomoya

デスクトップ

書類

ダウンロード

ムービー

ミュージック

ピクチャ

場所

iCloud Drive

tomoya の MacBook Air

Google Drive

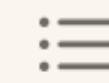
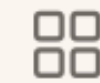
192.168.1.219

ネットワーク

タグ

レッド

< > ユーティリティ



- チェス.app
- テキストエディット.app
- ブック.app
- フリーボード.app
- プレビュー.app
- ボイスメモ.app
- ホーム.app
- ポッドキャスト.app
- マップ.app
- ミュージック.app
- メール.app
- メッセージ.app
- メモ.app
- ユーティリティ
- リマインダー.app
- 株価.app
- 計算機.app
- 時計.app
- 辞書.app
- 写真.app
- 探す.app
- 天気.app
- 連絡先.app

- AirMac ユーティリティ.app
- Audio MIDI 設定.app
- Bluetooth ファイル交換.app
- Boot Camp アシスタント.app
- ColorSync ユーティリティ.app
- Digital Color Meter.app
- Grapher.app
- VoiceOver ユーティリティ.app
- アクティビティモニタ.app
- キーチェーンアクセス.app
- コンソール.app
- システム情報.app
- スクリーンショット.app
- スクリプトエディタ.app
- ターミナル.app
- ディスクユーティリティ.app
- 移行アシスタント.app



tomoya — -zsh — 77x25

tomoya@tomoyanoMacBook-Air ~ % █



Homebrew

The Missing Package Manager for macOS (or Linux)

English ▾

Install Homebrew

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Paste that in a macOS Terminal or Linux shell prompt.

The script explains what it will do and then pauses before it does it. Read about other [installation options](#).

If you're on macOS, try our new [.pkg installer](#).

Download it from [Homebrew's latest GitHub release](#).

<https://brew.sh>



Homebrew

The Missing Package Manager for macOS (or Linux)

 ⌘ K

English ▾

Install Homebrew

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```



Paste that in a macOS Terminal or Linux shell prompt.

The script explains what it will do and then pauses before it does it. Read about other [installation options](#).

If you're on macOS, try our new [.pkg installer](#).

Download it from [Homebrew's latest GitHub release](#).

<https://brew.sh>

ここをクリックしてコマンドをコピー



tomoya@tomoyanoMacBook-Air ~ % `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

ペーストしてEnter

`/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

tomoya — bash -c #!/bin/bash\012\012# We don't need return codes for "\$(command)", only st...

```
[tomoya@tomoyanoMacBook-Air ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
==> Checking for `sudo` access (which may request your password)...
```

```
[Password:
```

パスワードを入力してEnter

```
==> This script will install:
```

```
/opt/homebrew/bin/brew
```

```
/opt/homebrew/share/doc/homebrew
```

```
/opt/homebrew/share/man/man1/brew.1
```

```
/opt/homebrew/share/zsh/site-functions/_brew
```

```
/opt/homebrew/etc/bash_completion.d/brew
```

```
/opt/homebrew
```

```
Press RETURN/ENTER to continue or any other key to abort:
```

Enterを入力

```
█
```



```
tomoya — -zsh — 77x25
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (4/4), done.
remote: Total 5 (delta 4), reused 4 (delta 4), pack-reused 1
Unpacking objects: 100% (5/5), 635 bytes | 79.00 KiB/s, done.
From https://github.com/Homebrew/brew
 * [new tag]          4.1.9      -> 4.1.9
HEAD is now at 7456789fb Merge pull request #16093 from Homebrew/spdx-update
Updated 1 tap (homebrew/test-bot).
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run)
.

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
  https://github.com/Homebrew/brew#donations

==> Next steps:
- Run brew help to get started
- Further documentation:
  https://docs.brew.sh

tomoya@tomoyanoMacBook-Air ~ %
```

こんな感じになればOK

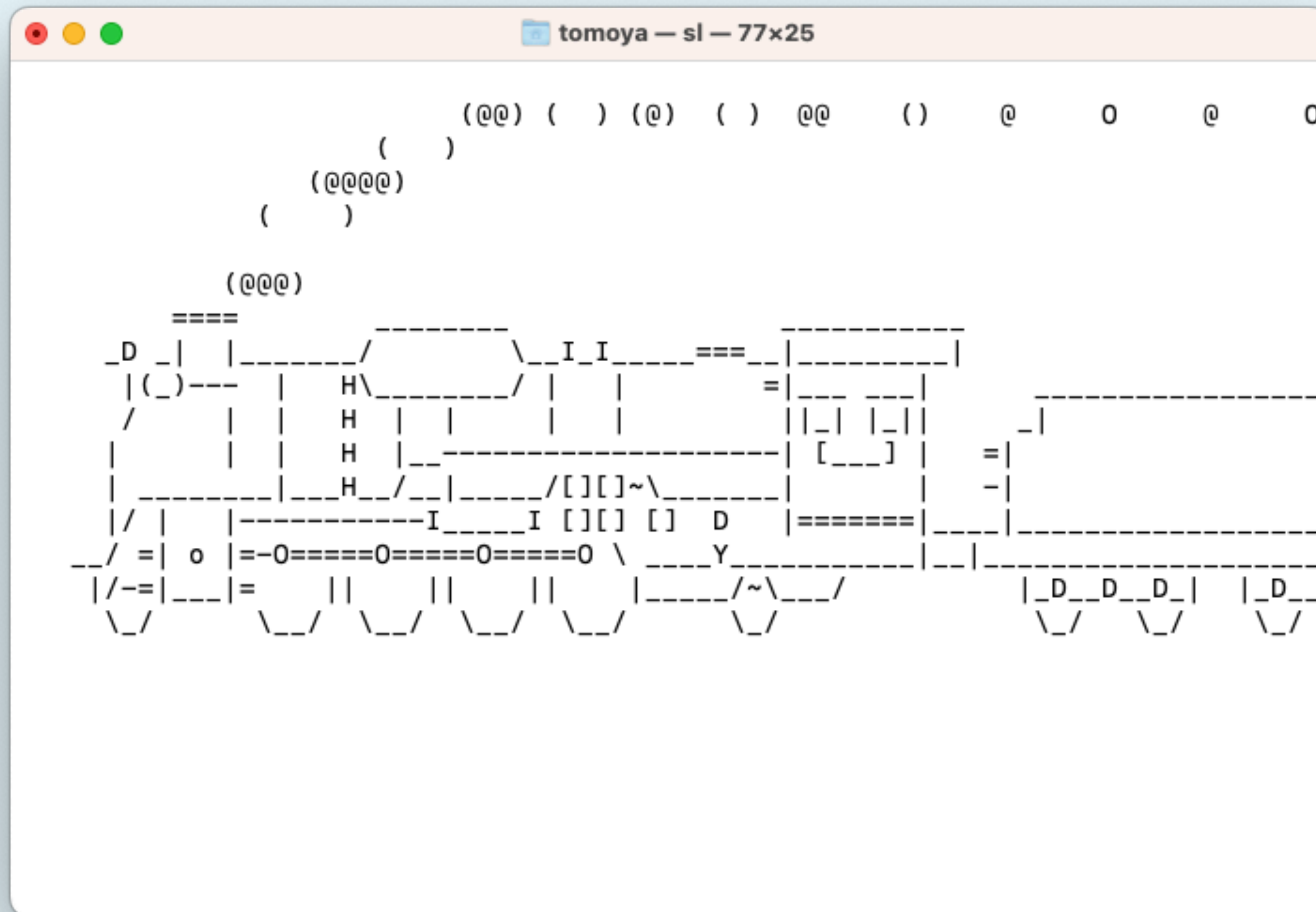
試しにインストールしてみよう

```
brew install sl
```

インストールできたら、

```
sl
```

と打ってEnter



※slコマンドはよく使うlsをタイプミスした時に出てくるジョークコマンド

Visual Studio Codeのインストール

A : 普通にインストール

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Version 1.83 is now available! Read about the new features and fixes from September.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

[Download Mac Universal](#)
Stable Build

[Web, Insiders edition, or other platforms](#)

By using VS Code, you agree to its [license](#) and [privacy statement](#).

EXTENSIONS: MARKETPLACE

- @sort:installs**
- Python** 2019.6.24221 54.9M ★ 4.5
Linting, Debugging (multi-threaded...
Microsoft [Install](#)
- GitLens — Git su...** 9.8.5 23.1M ★ 5
Supercharge the Git capabilities bui...
Eric Amodio [Install](#)
- C/C++** 0.24.0 23M ★ 3.5
C/C++ IntelliSense, debugging, and...
Microsoft [Install](#)
- ESLint** 1.9.0 21.9M ★ 4.5
Integrates ESLint JavaScript into V...
Dirk Baeumer [Install](#)
- Debugger for C...** 4.11.6 20.6M ★ 4
Debug your JavaScript code in the ...
Microsoft [Install](#)
- Language Sup...** 0.47.0 18.6M ★ 4.5
Java Linting, Intellisense, formattin...
Red Hat [Install](#)
- vscode-icons** 8.8.0 17.2M ★ 5
Icons for Visual Studio Code
VSCode Icons Team [Install](#)
- Vetur** 0.21.1 17M ★ 4.5
Vue tooling for VS Code
Pine Wu [Install](#)

```
src > components > JS blog-post.js > <function> > blogPost
1 import React from "react"
2 import Image from "gatsby-image"
3
4
5 export default ({ data }) => {
6   const blogPost = data.cms.blogPost
7   return (
8     <div>
9       {blogPost}
10      <div> {blogPost} </div>
11      <div> {blogPost} </div>
12      <img alt="Image" data-bbox="12 12 12 12" />
13    )
14  <h1>{blogPost} </h1>
15  <div>{blogPost} </div>
16  <div>{blogPost} </div>
17  </div>
18
19 }
20
21 export const query = graphql`
```

PROBLEMS TERMINAL ... 2: Task - develop + - - x

```
info i [wdm]: Compiling...
DONE Compiled successfully in 26ms 3:57:58 PM
info i [wdm]:
info i [wdm]: Compiled successfully.
```

master 0 1 0 0 1 Gatsby Develop (gatsby-graphql-app) Ln 6, Col 21 Spaces: 2 UTF-8 LF JavaScript

IntelliSense **Run and Debug** **Built-in Git** **Extensions**

<https://code.visualstudio.com/>

B : Homebrewでインストール

```
brew install --cask visual-studio-code
```