IMEN574/IDSC574 Programing for Data Science

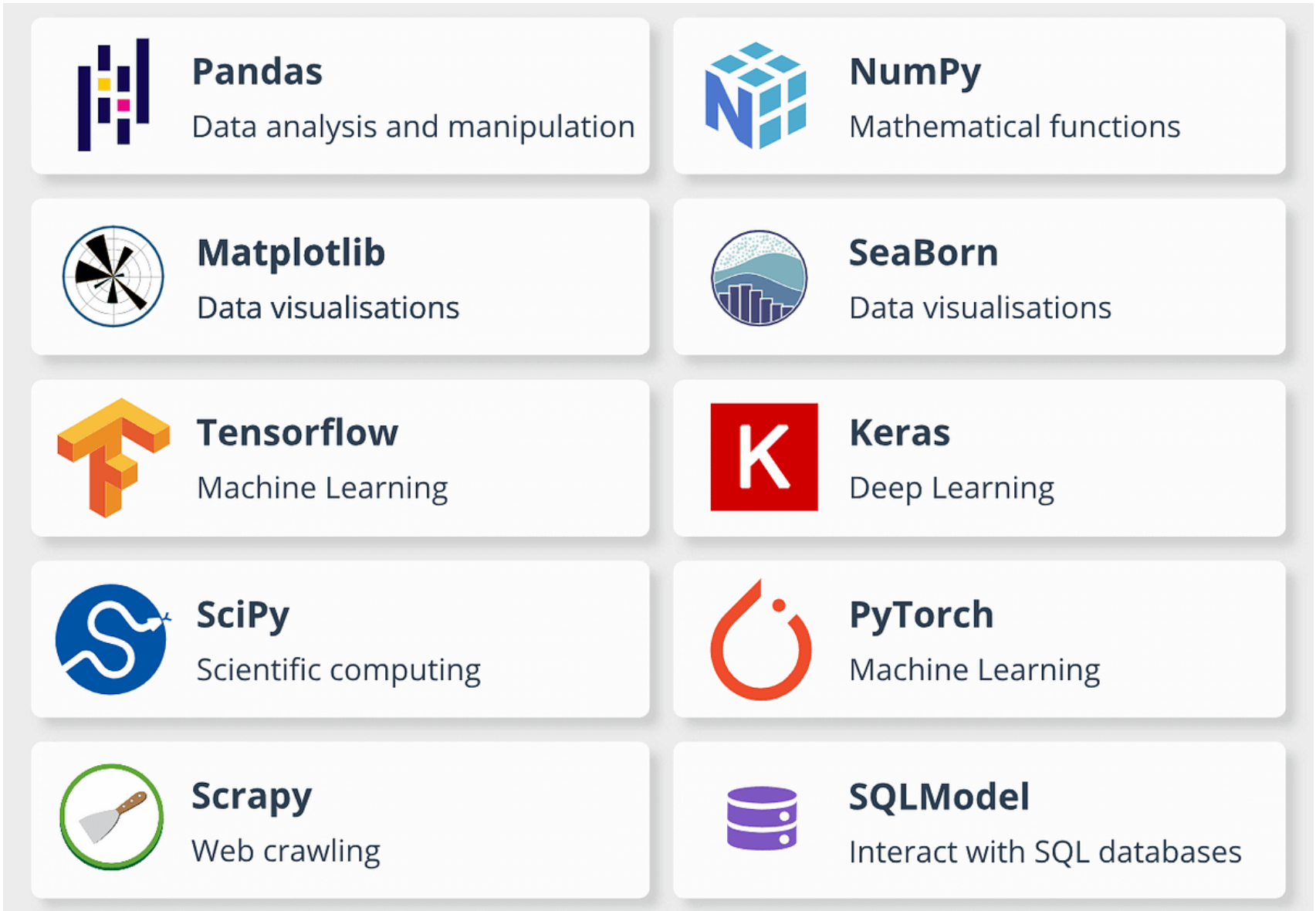
1. **Overview & Logistics**

Spring 2026

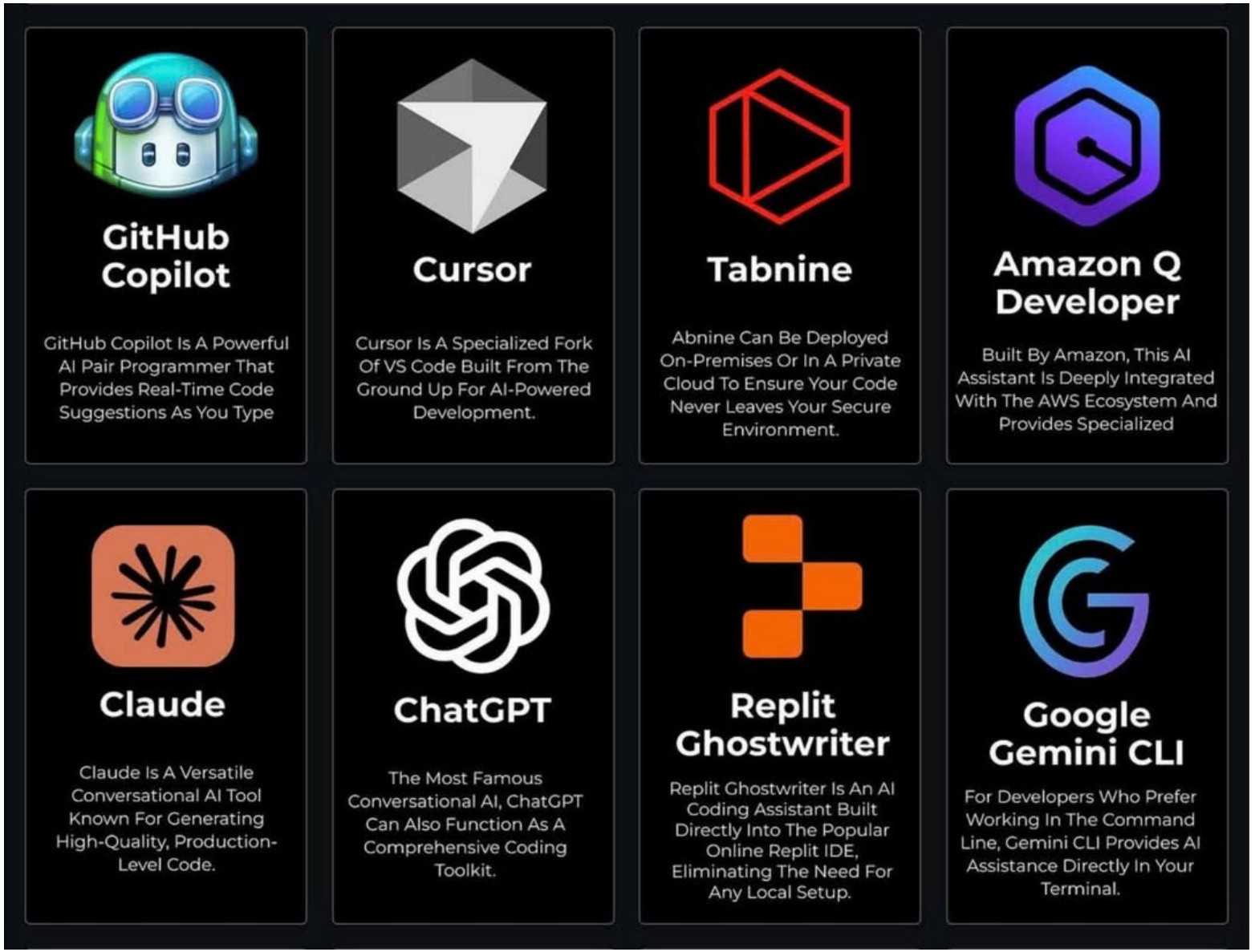
Instructor: **Sangwoo Mo**

Programing for Data Science

Programing for(→) Data Science


Pandas
Data analysis and manipulation
NumPy
Mathematical functions
Matplotlib
Data visualisations
SeaBorn
Data visualisations
Tensorflow
Machine Learning
Keras
Deep Learning
SciPy
Scientific computing
PyTorch
Machine Learning
Scrapy
Web crawling
SQLModel
Interact with SQL databases

Programing by(←) Data Science


GitHub Copilot
GitHub Copilot Is A Powerful AI Pair Programmer That Provides Real-Time Code Suggestions As You Type
Cursor
Cursor Is A Specialized Fork Of VS Code Built From The Ground Up For AI-Powered Development.
Tabnine
Abnine Can Be Deployed On-Premises Or In A Private Cloud To Ensure Your Code Never Leaves Your Secure Environment.
Amazon Q Developer
Built By Amazon, This AI Assistant Is Deeply Integrated With The AWS Ecosystem And Provides Specialized
Claude
Claude Is A Versatile Conversational AI Tool Known For Generating High-Quality, Production-Level Code.
ChatGPT
The Most Famous Conversational AI, ChatGPT Can Also Function As A Comprehensive Coding Toolkit.
Replit Ghostwriter
Replit Ghostwriter Is An AI Coding Assistant Built Directly Into The Popular Online Replit IDE, Eliminating The Need For Any Local Setup.
Google Gemini CLI
For Developers Who Prefer Working In The Command Line, Gemini CLI Provides AI Assistance Directly In Your Terminal.

Programing ⇔ Data Science

What is Programming?

What is Programming?

Software 1.0

computer code

computer



became programmable in **~1940s**

Andrej Karpathy, “Software in the era of AI”

What is Programming?

Software 1.0

computer code

↓ programs

computer

Software 2.0

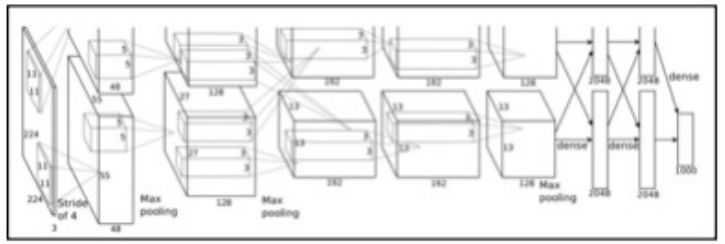
weights

↓ programs

neural net



became programmable in **~1940s**



fixed function neural net
e.g. AlexNet: for image recognition (**~2012**)

Andrej Karpathy, “Software in the era of AI”

What is Programming?

Software 1.0

computer code

programs

computer



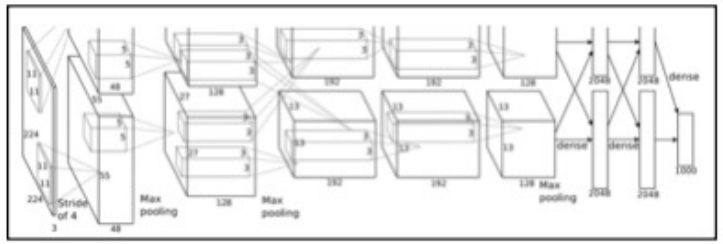
became programmable in **~1940s**

Software 2.0

weights

programs

neural net



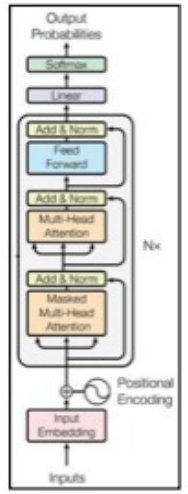
fixed function neural net
e.g. AlexNet: for image recognition (**~2012**)

Software 3.0

prompts

programs

LLM



~2019

LLM = programmable neural net!

Andrej Karpathy, “Software in the era of AI”

Example: Sentiment Classification

Software 1.0

python Copy

```python
def simple_sentiment(review: str) -> str:
    """Return 'positive' or 'negative' based on a tiny keyword lexicon."""
    positive = {
        "good", "great", "excellent", "amazing", "wonderful", "fantastic",
        "awesome", "loved", "love", "like", "enjoyed", "superb", "delightful"
    }
    negative = {
        "bad", "terrible", "awful", "poor", "boring", "hate", "hated",
        "dislike", "worst", "dull", "disappointing", "mediocre"
    }

    score = 0
    for word in review.lower().split():
        w = word.strip(".,!?;:")          # crude token clean-up
        if w in positive:
            score += 1
        elif w in negative:
            score -= 1

    return "positive" if score >= 0 else "negative"
```

Andrej Karpathy, “Software in the era of AI”

Example: Sentiment Classification

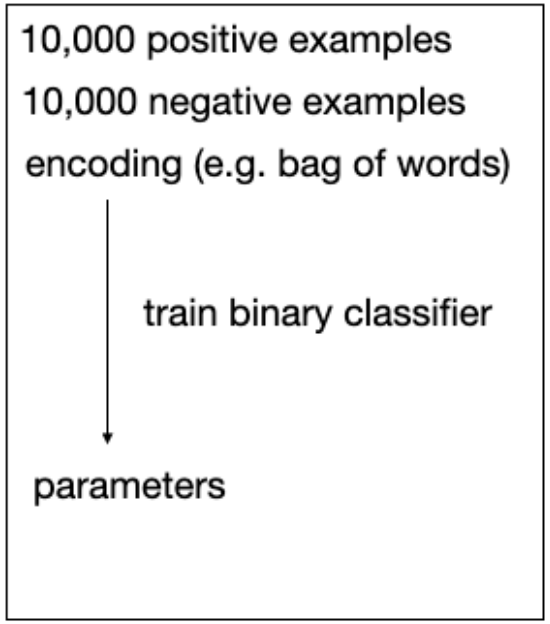
Software 1.0

```python
def simple_sentiment(review: str) -> str:
    """Return 'positive' or 'negative' based on a tiny keyword lexicon."""
    positive = {
        "good", "great", "excellent", "amazing", "wonderful", "fantastic",
        "awesome", "loved", "love", "like", "enjoyed", "superb", "delightful"
    }
    negative = {
        "bad", "terrible", "awful", "poor", "boring", "hate", "hated",
        "dislike", "worst", "dull", "disappointing", "mediocre"
    }

    score = 0
    for word in review.lower().split():
        w = word.strip(".,!?;:")          # crude token clean-up
        if w in positive:
            score += 1
        elif w in negative:
            score -= 1

    return "positive" if score >= 0 else "negative"
```

Software 2.0



Andrej Karpathy, “Software in the era of AI”

Example: Sentiment Classification

Software 1.0

```python
def simple_sentiment(review: str) -> str:
    """Return 'positive' or 'negative' based on a tiny keyword lexicon."""
    positive = {
        "good", "great", "excellent", "amazing", "wonderful", "fantastic",
        "awesome", "loved", "love", "like", "enjoyed", "superb", "delightful"
    }
    negative = {
        "bad", "terrible", "awful", "poor", "boring", "hate", "hated",
        "dislike", "worst", "dull", "disappointing", "mediocre"
    }

    score = 0
    for word in review.lower().split():
        w = word.strip(".,!?;:")          # crude token clean-up
        if w in positive:
            score += 1
        elif w in negative:
            score -= 1

    return "positive" if score >= 0 else "negative"
```

Software 2.0

10,000 positive examples

10,000 negative examples

encoding (e.g. bag of words)

train binary classifier

parameters

Software 3.0

You are a sentiment classifier. For every review that appears between the tags
<REVIEW> ... </REVIEW>, respond with **exactly one word**, either POSITIVE or NEGATIVE (all-caps, no punctuation, no extra text).

Example 1
<REVIEW>I absolutely loved this film—the characters were engaging and the ending was perfect.</REVIEW>
POSITIVE

Example 2
<REVIEW>The plot was incoherent and the acting felt forced; I regret watching it.</REVIEW>
NEGATIVE

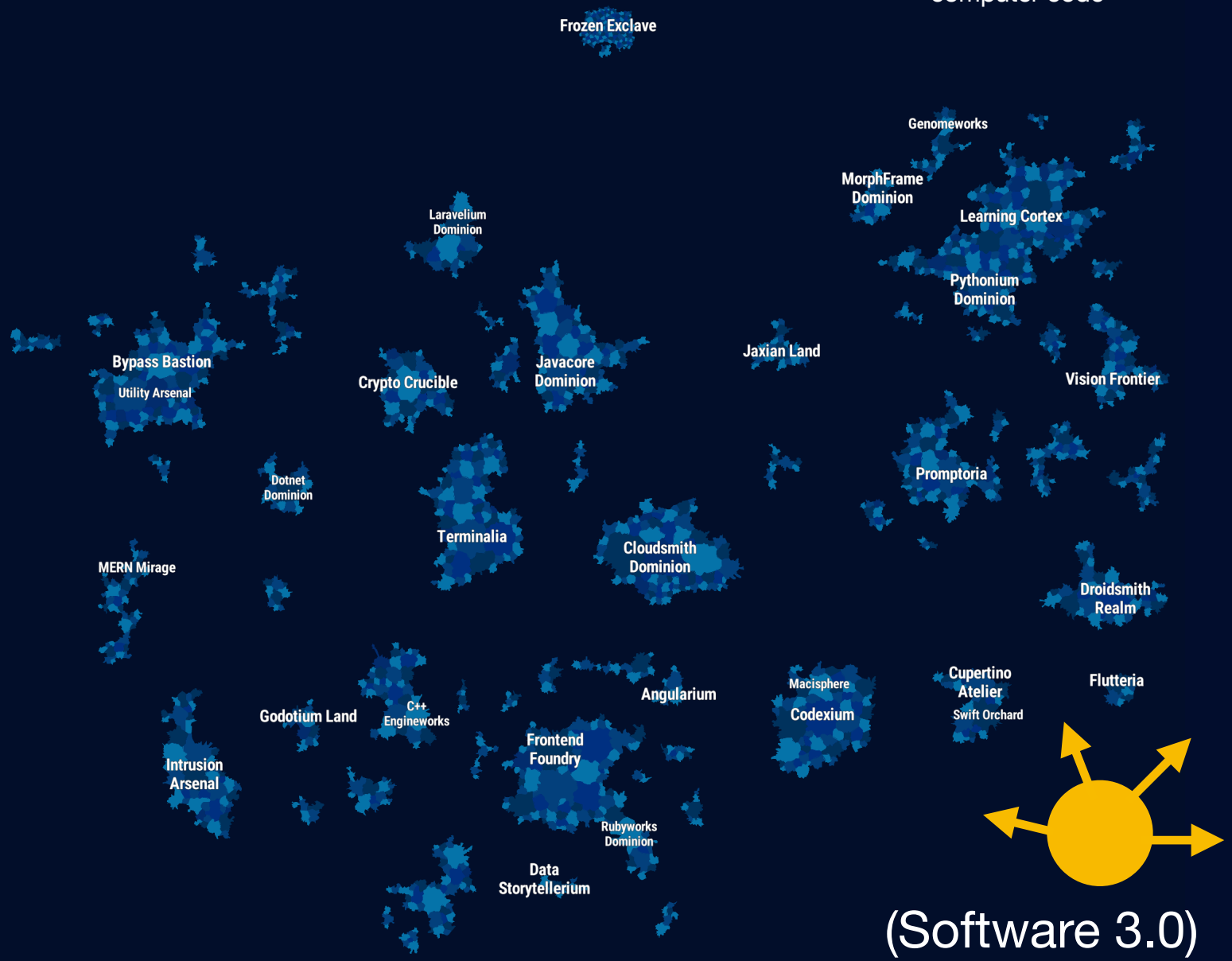
Example 3
<REVIEW>An energetic soundtrack and solid visuals almost save it, but the story drags and the jokes fall flat.</REVIEW>
NEGATIVE

Now classify the next review.

Andrej Karpathy, “Software in the era of AI”

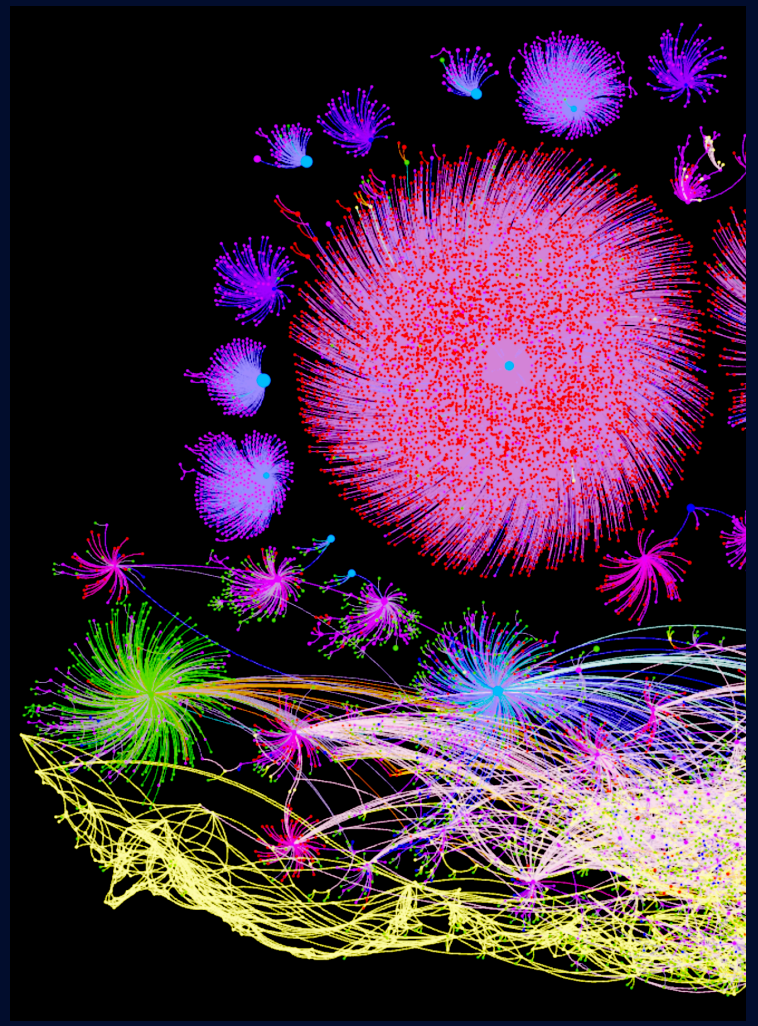
"Map of GitHub" (Software 1.0)

computer code



HuggingFace Model Atlas (Software 2.0)

neural network weights

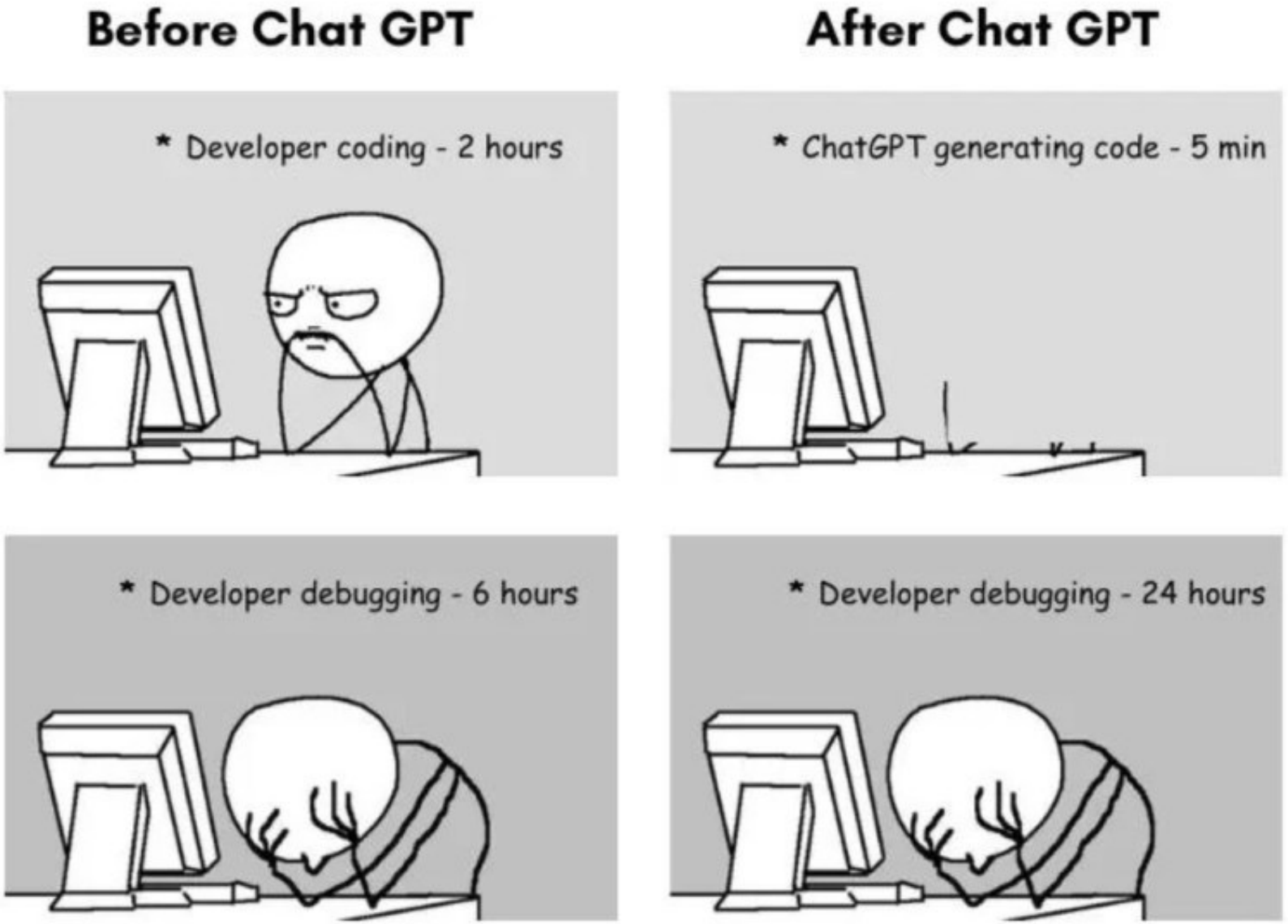


(Software 3.0)

LLM prompts, in English

Do we still need to learn Software 1.0?

Do we still need to learn Software 1.0?



Human Coding = Vibe Debugging

	VIBE CODING	REAL CODING
Method	Guesswork, AI prompts, or copying code	Logic, planning, understanding
Speed	Quick to start, slow/little long-term use	Slow to start, faster and reliable long-term use
Understanding	Often unclear	Clear and deliberate
Usefulness	Good for prototyping or experimenting	Best for complex and maintainable software
Risk	Higher risk: bugs, security issues, little scalability	Lower risk: more stable, easier to debug and build on

Lecture Overview

Outline	Software 1.0	Software 2.0	Software 3.0

Lecture Overview

Outline	Software 1.0	Software 2.0	Software 3.0

1	02/24	1	Overview & Logistics
2	02/26	2	Chatting with AI
3	03/03	3	Coding with AI
4	03/05	4	LLM Prompting
5	03/10	5	Coding Agents
6	03/12	6	Modern IDE

Lecture Overview

Outline	Software 1.0	Software 2.0	Software 3.0

1	02/24	1	Overview & Logistics
2	02/26	2	Chatting with AI
3	03/03	3	Coding with AI
4	03/05	4	LLM Prompting
5	03/10	5	Coding Agents
6	03/12	6	Modern IDE
7	03/17	7	Python
8	03/19	8	Numpy & Pandas
9	03/24	9	Regression
10	03/26	10	Classification
11	03/31	11	Clustering
12	04/02	12	Dimensionality Reduction
13	04/07	13	Model Selection
14	04/09	14	Interpretable ML
15	04/14		Study Day
16	04/16		Midterm Exam

Lecture Overview

Outline	Software 1.0	Software 2.0	Software 3.0

1	02/24	1	Overview & Logistics
2	02/26	2	Chatting with AI
3	03/03	3	Coding with AI
4	03/05	4	LLM Prompting
5	03/10	5	Coding Agents
6	03/12	6	Modern IDE
7	03/17	7	Python
8	03/19	8	Numpy & Pandas
9	03/24	9	Regression
10	03/26	10	Classification
11	03/31	11	Clustering
12	04/02	12	Dimensionality Reduction
13	04/07	13	Model Selection
14	04/09	14	Interpretable ML
15	04/14		Study Day
16	04/16		Midterm Exam
17	04/21	15	Deep Learning
18	04/23	16	Pytorch & JAX
19	04/28	17	Vision Models I
20	04/30	18	Vision Models II
21	05/05		Children's Day
22	05/07		Spring Festival
23	05/12	19	Language Models I
24	05/14	20	Language Models II
25	05/19	21	Multimodal Models I
26	05/21	22	Multimodal Models II
27	05/26	23	AI for Data Science
28	05/28	24	Guest Lecture (05/29, Yeong-Hwan Jeon)
29	06/02	25	Project Presentation
30	06/04	26	Project Presentation
31	06/09		No Class
32	06/11		No Class

Term Project

1. Choose a **Kaggle competition** of your preference

2. Solve it with **your team** (team size ≤ 4)

3. Share your experience in the **final presentation**

Term Project – Evaluation Criteria

- **Problem:** Why is this problem significant?
- **Method:** Is the solution original or trivial?
- **Results:** Final performance & Kaggle ranking
- **Teamwork:** Each member’s individual contribution
- **Presentation:** Quality of delivery and storytelling

Grading Policy

- **Attendance:** 0% (but **F** if absence > 1/3)
- **Mid-term exam:** 40%
- **Final project:** 60%