Aider와 함께하는 AI-driven Pair Programming

이재열

@kodingwarrior@hackers.pub

이재열 (@kodnigwarrior)

- 디스코드에서는 kokoko.kojima / SNS에서는 kodingwarrior라는 핸들을 사용
- 취미로 오픈소스를 뜯어보고 거기에 숨겨진 구현원리를 보며 아름다움을 느끼는 사람
- GUI 기반의 개발환경에서 100% 터미널 기반의 개발환경으로 갈아탄지 거의 3년
- **Aider 등과 같은 터미널 기반의 코딩 에이전트와 함께한지는 6개월째**

어떤 사람들을 위한 발표인가요?

- 사람은 그저 거들뿐! **기계가 주도적으로 코드 짜는 것을 경험**해보고 싶은 사람
- Claude Code, Gemini CLI, OpenAI Codex 등의 에이전트 외 **비용 절감 옵션**을 생각하는 사람
  - 토큰 소모량을 최소한으로 하면서 API 사용비용을 적게 내고 싶은 사람
- 특정 벤더에 의존하지 않는 코딩 에이전트를 찾고 있는 사람
  - OpenAI, Anthropic, Google, …
- 보안 상의 이유로 **Local LLM 기반으로 돌아가는 코딩 에이전트**를 쓰고 싶은 사람

여러분들이 이런 경험 있다고 가정합니다.

- ChatGPT, DeepSeek, Claude, Gemini 등 대화형 LLM 서비스 사용 경험
- GitHub Copilot 등을 사용해서 코드 자동완성이 되는걸 직접 혹은 간접적으로 경험
- OpenAI Codex, Claude Code, Gemini CLI 등의 LLM 기반 코딩 에이전트 사용 경험

물론 이 중에 포함되지 않더라도 상관은 없습니다.

Presentation Overview

01 AI 페어 프로그래머, Aider

02 Aider와 함께,
AI-driven Pair Programmig

03 앞으로 우리가 마주할 미래

사실은...

발표자료를 준비하는 동안에도 강산이 변했습니다.

* 3월 20일 : Aider를 본격적으로 워크플로우에 녹여내기 시작
* **5월 1일** : **Claude Code에서 Todo list 지원**으로 작업계획의 가독성 개선
* **5월 11일 : 파이콘 CFP 마감**
* **5월 16일 : OpenAI에서 OpenAI Codex 공식 발표**
* **5월 23일 : Claude Code 1.0 공식 발표 / Claude Code에 Claude 4 Sonnet/Opus 모델 도입 발표**
* 6월 13일 : Claude Code를 본격적으로 사용
* **6월 19일 : 파이콘 한국 2025 발표자로 선정**
* **6월 25일 : Gemini CLI 공식 발표**
* **7월 3일 : Cursor 1.2 (Queued messages)**
* **7월 13일 : AWS에서 Kiro 공식 발표 (Spec Driven Development)**
* **7월 24일 : Claude Code에 subagents 기능 도입**
* **7월 31일 : MCP, LSP 통합이 포함된 오픈소스 코딩 에이전트 Crush 출시 (by Charm Bracelet)**
* **8월 8일 : Cursor CLI 출시**

그리고 Agentic Coding의 발전은 현재진행형으로 계속 되는 중…

01

AI 페어프로그래머, Aider

용어 정리

- **AI-assisted programming**
  - Cursor, GitHub Copilot, Windsurf, Roo Code 등 개발 환경에서 제공되는
    대규모 언어 모델(LLM)의 기능을 활용하여 인간 개발자의 코딩 및 개발 작업을 보조하는 것
- **Vibe Coding** - 후술
- **AI-driven Pair Programming** - 후술

우리가 다룰 주제는

Vibe Coding도 아니고, **AI-driven Pair Programming**입니다.

*부연설명 : Vibe Coding

- **From “Andrej Karpathy” (ex- Open AI)**
  - *There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good*
- **요약** : LLM께서 의도한대로 잘 짜주시니 그냥 아예 흐름에 맡겨버리자
- 코드가 있다는 것 조차 인지하지 않고, LLM이 코드를 짜주는대로 개발을 이어가는 것

소프트웨어 엔지니어로서 소스코드의 세부사항을 조금이라도 인지한다면,
우리가 흔히 알고 있는 Vibe Coding은 Vibe Coding에 대한 엄밀한 정의에서 완전히 벗어남

AI-driven Pair Programming

- **Pair Programming** : 두 개발자가 한 화면을 보면서 함께 작업하며, 실시간으로 코드를 작성하고 검토하는 협업 개발 방식
  - **Navigator** - **Driver**가 작성하는 코드를 검토하고, 전체적인 방향과 전략을 제시하며 안내하는 역할
  - **Driver** - 키보드를 잡고 실제로 코드를 입력하며 현재 작업의 세부 구현에 집중하는 역할

정의 : **AI-driven Pair Programming**은 **AI가 Driver가 되어서** 페어프로그래밍을 하는 방식

Aider (AI pair programming in your terminal)

- https://aider.chat
- 터미널 환경에서 AI와 대화하며 코드 생성, 편집, 디버깅 등을 수행하는 도구
- 당연히… **Python 기반으로 만들어진 CLI 프로그램**임!
- Aider 코드 베이스의 80% 이상은 Aider로 만들어진 코드

Aider에서 제공하는 기본적인 기능들

- **다양한 LLM 모델 선택적 사용** (클라우드 및 로컬)
- 프로젝트 전체 코드베이스 분석 및 컨텍스트 이해 (tree-sitter 기반으로 코드베이스 매핑)
- 주요 프로그래밍 언어 대부분 지원 (100개 이상)
- 기존 IDE/편집기 환경과 통합 사용 가능
- 이미지, 웹 문서, 음성 등 다양한 방식의 컨텍스트 제공 및 입력
  - Voice to Code가 가능합니다.
- Git과 완벽하게 연동이 되어 있어, 지시한 작업이 끝난 뒤에 자동으로 커밋됨
  - auto-commit = false 옵션을 줄 수 있음

Aider LLM Leaderboard (LLM 모델 성능 지표)

Aider polyglot benchmark leaderboard (2025-07-06)

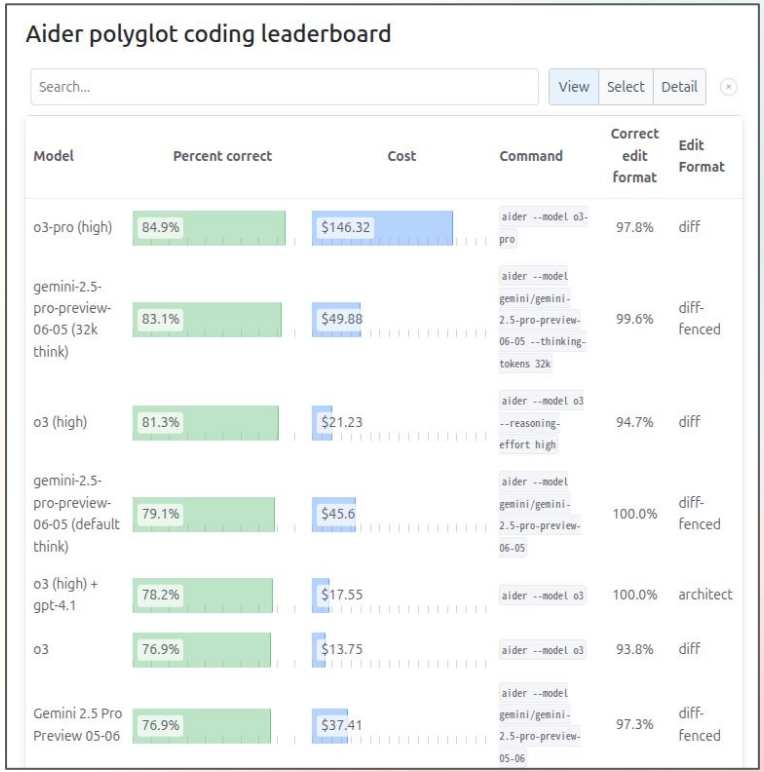
- Focuses on the most difficult 225 exercises out of the 697 that Exercism provides for those languages. The old benchmark simply included all 133 Python exercises, regardless of difficulty.

- Exercism에서 **225개 정도의 어려운 문제를 여러개의 다른 언어로 풀게 했을때 어떤 성능을 보였는가**를 기준으로 점수를 매김

Repository : https://github.com/Aider-AI/polyglot-benchmark/

* 개발의 복잡성을 반영하지는 않을 수 있으나,

어려운 문제를 어떤 모델이 잘 풀어낼 수 있는지는 나타낼 수 있음



Aider LLM Leaderboard (LLM 모델 성능 지표)

Aider polyglot benchmark leaderboard (2025-07-06)

- Focuses on the most difficult 225 exercises out of the 697 that Exercism provides for those languages. The main benchmark simply included all 133 Python exercises, regardless of difficulty.
- Exercism에서 225개의 가장 어려운 문제를 여러 개의 서로 다른 언어로 **풀게 했을때 어떤 반응을 보였는가를 기준으로 점수를 매김**

Repository : https://github.com/Aider-AI/polyglot-benchmark/

* 개발의 복잡성을 반영하지는 않을 수 있으나,
어려운 문제를 어떤 모델이 잘 풀어낼 수 있는지는 나타낼 수 있음

입맛대로 원하는대로
적당히 가성비 있고, 성능이 좋은 모델을
가져다 쓸 수가 있다.

번외: 자율적으로 행동하는 Agent (scripting API)

- Case Study : 연구하고 논문찍어내는 것을 자율수행
  - https://www.aitimes.com/news/articleView.html?idxno=162512
  - https://github.com/SakanaAI/AI-Scientist/
  - 추론 능력이 좋은 LLM, 구현 능력이 좋은 LLM 등을
    적절하게 조합하여 연구를 자율적으로 수행하고 협력하도록 설계

Showcase : Todo List 만들어보기

02

Aider와 함께 역할분담하기

사람은 그저 거들 뿐이다!

Aider 에이전트의 3가지 모드

* **/ask** : 코드베이스에 대해서 질문/답변을 주고 받는 모드. **코드를 직접 짜지 않는다**!
* **/code** : 프롬프트를 받은대로 코드를 짜준다.
* **/architect** : /ask와 /code의 하이브리드

    추론 능력이 뛰어난 모델(main model)에게 **질문을 하고, 제안을 받는다**.

    그리고 **에디터 모델**이 confirm 받은 대안으로 **코드를 짜준다**.

공식 권장사항은 ask/code를 반복하는 워크플로우

Mental Model : 2 Navigator / 1 Driver

AI Driver (Aider - Junior Developer) — /code

- 지칠 줄 모르는 코드 생성, 방대한 기술 지식 기반의 신속한 구현

AI Navigator (Senior Developer) — /architect, /ask

- 인간 Navigator가 Aider에게 작업을 지시하거나 조언을 구할 때 활용하는 가상의 전문가 페르소나.
- Aider(주니어)가 더 높은 수준의 결과물을 내도록 유도하는 장치.

Navigator (Human: Project Leader)

- 최종 목표 설정, 전체 아키텍처 설계, 작업 지시 및 분배.
- AI Driver(Aider)의 결과물 검토 및 피드백, 품질 관리.

STEP 1 : 프로젝트를 시작할때 요구사항 정리하기

Navigator로서의 첫 임무 : AI에게 명확한 목표와 임무를 전달

프롬프트 : *"/ask Todo List 앱을 만들어 볼거야. Todo List를 작성하기 위한 PRD를 작성해줘"*

프롬프트 : *"/ask Todo List 앱을 만들어 볼거야. Todo List 앱을 만들기 위해 필요한 체크리스트를 뽑아줘"*

프롬프트 : *"/ask Todo List 앱을 만들기 위한 체크리스트 기반으로, 시니어 풀스택 개발자로서 어떻게 작업을 이어갈 것인지 작업 계획을 짜줘"*

Navigator로서 시니어 역할인 AI Navigator가 올바르게 작업 계획을 짜는지 검토

/ask 모드로 여러번 핑퐁을 하면서 방향성이 잡히면,

/copy로 대화로 정리한 내용을 클립보드로 복사하여 **DESIGN.md / ARCHITECT.md** 등에 기록.

기록한 내용은 **/read-only** 커맨드로 메모리에 올린다

* Aider에 대한 잠깐의 토막지식

Aider는 **토큰 수 절약**을 위해 **필요한 소스코드/메타데이터**만 메모리에 올린다.

Aider는 프로젝트의 코드베이스를 전반적으로 이해하기 위해 tree-sitter로 파싱하여 repomap을 만들고 메타데이터로서 메모리에 포함시킨다

수정 작업을 허용하는 소스코드를 메모리에 추가할때는 **/add**

읽기 작업만 허용하는 소스코드를 메모리에 추가할때는 **/read-only**

ex. AGENTS.md, CONVENTION.md, README.md, TODO.md, DESIGN.md

STEP 2 : 정리된 요구사항 기반으로 Driver에게 전달할 프롬프트 생성

Navigator의 전략적 분할: 크고 복잡한 작업을 AI가 처리하기 쉬운 단위로 세분화

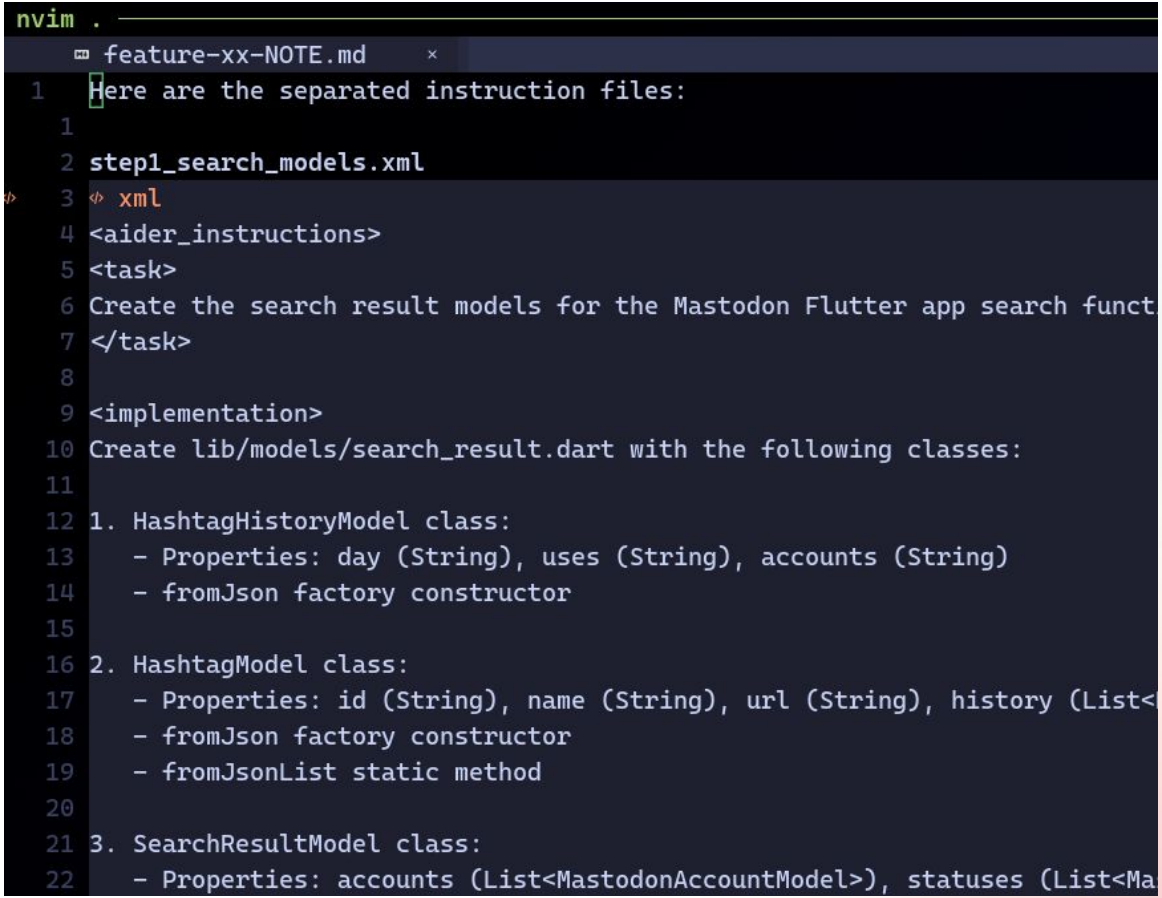
> *"/ask 기능에 대한 작업 계획을 기반으로 Aider에게 코드를 짜도록 시킬거야*
>
> *Aider가 어떻게 작업을 이어갈지 XML 기반으로 프롬프트를 짜서 전달해줘*
>
> *각각의 프롬프트는 Step by Step으로 전달할 수 있도록 여러개의 XML 파일로 전달해줘"*

응답으로 전달받은 XML 프롬프트들을 **/copy**로 복사하여, **/docs/feature-xx-NOTE.md** 에 기록

올바르지 않은 방향으로 간다면,

일부 직접 수정하거나 혹은 /ask 모드로 피드백 주고 받으면서 프롬프트를 다듬는다

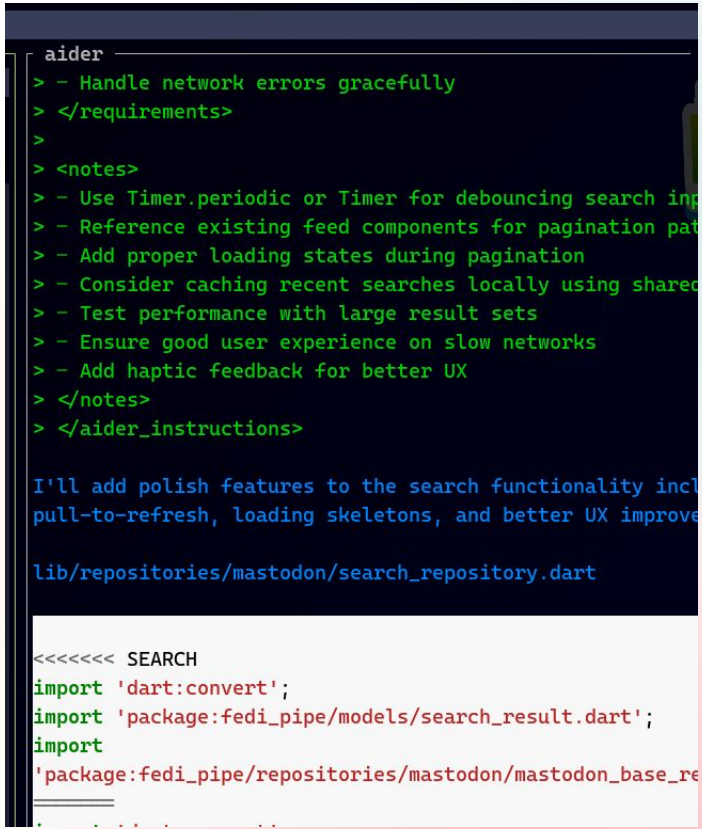
*** Aider는 메모리에 올라간 파일이 변경되더라도 거기에 담긴 내용은 매번 새로 읽습니다**


nvim .
feature-xx-NOTE.md
Here are the separated instruction files:
step1_search_models.xml
xml
<aider_instructions>
<task>
Create the search result models for the Mastodon Flutter app search funct
</task>
<implementation>
Create lib/models/search_result.dart with the following classes:
1. HashtagHistoryModel class:
- Properties: day (String), uses (String), accounts (String)
- fromJson factory constructor
2. HashtagModel class:
- Properties: id (String), name (String), url (String), history (List<
- fromJson factory constructor
- fromJsonList static method
3. SearchResultModel class:
- Properties: accounts (List<MastodonAccountModel>), statuses (List<Ma

STEP 3 : 전 단계에서 만든 프롬프트를 붙여넣기 하면서 경과 확인하기

인간 Navigator는

AI 드라이버가 올바른 길을 가지고 있는지 확인한다.



그 다음은…?

STEP 1으로 다시 돌아가서

1) 요구사항 정리 / 2) Driver를 위한 프롬프트 생성 / 3) Driver에게 지시

위의 단계들을 반복하면서 기능들을 개선한다.

맥락을 초기화하고 싶다면, **/clear** 명령을 실행

TIP 1) 시스템 프롬프트 정의하기

*** CONVENTION.md :** 컨벤션 준수 강제

*** README.md :** 프로젝트가 뭘하고자 하는 프로젝트인지 개요 설명

*** AGENTS.md :** AI 에이전트에 전달하고 싶은 시스템 프롬프트 전달

각각의 파일을 **/read-only** 커맨드로 포함시키면 성능을 좀 더 좋게 만들 수 있다.

혹은 ~/.aider.conf.yml 에서 **read-only 옵션**으로 시스템 프롬프트로 넣고 싶은 파일 나열할 수 있음

.aider.conf.yml

```yaml
## specify a read-only file (can be used multiple times)
#
# NOTE : If you running aider in a subdirectory with --subtree option, read-or
read:
 - docs/TODO.md
 - docs/DESIGN.md
 - docs/ARCHITECTURE.md
 - CONVENTIONS.md
 - ../docs/TODO.md
 - ../docs/DESIGN.md
 - ../docs/ARCHITECTURE.md
 - ../CONVENTIONS.md
 - ../../docs/TODO.md
 - ../../docs/DESIGN.md
 - ../../docs/ARCHITECTURE.md
 - ../../CONVENTIONS.md
 - ../../../docs/TODO.md
 - ../../../docs/DESIGN.md
 - ../../../docs/ARCHITECTURE.md
 - ../../../CONVENTIONS.md

## Use VI editing mode in the terminal (default: False)
vim: true
```

TIP 2) API 비용 절감을 위해 전체 맥락은 웹서비스를 이용

Aider를 비롯한 여러 AI 에이전트들은 **API 사용량에 따라 비용이 과금되는 방식**

비용을 아낄 수는 있다하더라도,
인풋으로 들어가는 토큰이 계속 누적이 될수록 과금되는 비용이 더 커질 수 있음

ChatGPT/Claude Desktop/Gemini 등 웹 서비스에서 **(월 $22 고정비용)**
Repomix output 파일을 업로드해서 컨텍스트로 넣으면 **비용을 절감**할 수 있음

TIP 3) 소스코드 전반적인 맥락을 던지고 싶을때는 repomix

Repomix는 **tree-sitter 기반으로 소스코드를 분석**해서
LLM 친화적인 텍스트로 변환해주는 CLI 프로그램

LLM에 **repomix output** 파일을 프롬프트로 넣어주면
소스코드를 전반적으로 이해한 상태에서 대답을 해줄 수 있는 가능성이 커짐

파이썬의 경우, 비슷한 구현체인 **gitingest**를 쓰는 쪽이 소스코드를 압축하기 용이함

근데, 굳이 이렇게까지 해야할까요?

물!론! Aider를 사용할때는 이렇게라도 하면 그나마 낫게 사용할 수는 있습니다

시중에서 사용되고 있는 상용 LLM 에이전트들은 아까전에 제안했던 방법들이 자동화되어서 실행되고 있을 정도로 계속 발전하고 있습니다

Aider 등을 비롯한 오픈소스 LLM 코딩 도구의 차별점은… **Zero 벤더 의존성**

요약하자면…

인간의 방향 지휘 / AI의 빠른 실행력이 시너지를 낸다

- **인간 Navigator** : **고차원적인 전략 수립과 방향성**에 집중, **최종 결과물의 품질과 비전에 대한 책임**을 가짐
- **AI Driver(Aider)** : 구체화된 지시를 이행하고, **빠르고 효율적으로 코드를 구현**하는 데 집중

인간은 프롬프트를 적게 넣더라도, AI 친화적인 소통이 가능하게 한다

- 인간이 가진 추상적인 아이디어나 복잡한 요구사항을 AI Driver가 이해하기 쉬운 형태로 변환.
- 페르소나 기반 질문은, 효과적인 프롬프트 생성에 유용한 가이드라인을 제공하여 인간의 프롬프트 작성 부담을 덜어줌. ex) “시니어 엔지니어라면 이 문제를 어떻게 접근할까?"

구현은 AI Driver에게, 전략은 인간에게.

짧은 반복 주기, 빠른 이슈 해결로 생산성 향상

하지만! 이것만큼은 주의합시다!!

흔하디 흔하지만 Hallucination의 위험성

코드가 빠르게 나오는 것도 중요하지만,

무엇보다도 중요한 것은 필요하다면 **변경사항을 검토할 수 있는 리터러시**

겉으로는 그럴싸해보이지만, 실제로 동작하지 않거나 심각한 오류를 가지고 있을 수 있습니다.

엔지니어로서,

소프트웨어의 품질을 책임지는 것은 LLM의 몫이 아니라 최종적으로는 우리의 몫입니다.

하지만! 여러분의 능력을 키우는 것도 중요!

- 당연한 얘기지만, LLM이 이루어낸 성과가 온전히 여러분의 성과는 아닙니다
- LLM이 여러분의 작업 속도에 날개를 달아주는 것은 맞지만,
  LLM에게 올바르게 날개를 달아주려면 **여러분의 올바른 지시가 필요**합니다
- 올바른 지시를 내릴려면, 그만큼 **배경지식에 대한 공부가 더욱 필요**합니다.

03

우리가 마주할 미래

개인적으로 느낀 AI-driven Pair Programming의 효능

- **시간 절약**
  - 디버깅을 하거나 모르는게 생겼을때 여기저기 찾아보는데 드는 시간을 줄였음
    - ex. Langchain 문서를 찾아보지 않고, 문서 요약기를 만듦
  - 인간의 생각/시행착오는 병목이 될 뿐.
  - **손이 빨라야 하는 작업은 LLM에게 맡김**으로써 절대적인 양의 시간을 절약할 수 있음
- **심리적인 저항감, 미루게 되는 습관을 일부 해소**
  - **5-6시간 이상 소요될 것 같은 작업**은 날 잡아서 하려고 하는게 보통이지만,

    1-2시간으로 대폭 줄게 되면 평일 자투리 시간에도 행동으로 옮길 수 있음
  - 심리적인 저항감이 생겨서 하고 싶지 않은 일이라도, 어떻게든 시작을 할 수 있음.
- **Unknown Unknown 영역의 궁금증을 빠른 속도로 해소**

개인적으로 생각하는 AI-assisted Programming

AI-driven Pair Programming은 어떻게 보면 **AI-assisted Programming의 부분집합**

- 정적 타이핑 언어 / 동적 타이핑 언어
- 정적 타이핑 언어 / 동적 타이핑 언어 / **AI-assisted Programming (new)**
  - 동적 타이핑 언어가 사람의 언어에 가깝게 코딩하면서 생산성을 높일 수 있었다면,
  - AI-assisted Programming은 사람의 언어를 그대로 전달하면서 초기 개발 생산성이 5~100배!
- AI-assisted Programming은 받아들일 수 밖에 없는 시대의 흐름

LLM이 주도해서 프로그래밍하는 기술이 더더욱 진보가 되고 있고, 현재진행형입니다.

요즘은 AI-agent가 주도적으로 코딩한다고 해서 **Agentic Coding**이라고도 표현합니다.

AI-assisted Programming의 시대에 인간이 가져야 할 역할

AI 드라이버의 능력은 네비게이터의 역량에 따라 크게 달라집니다.

- 어떤 모델을 적재적소에 사용하고 있는지 (추론 능력이 뛰어난 모델, 가성비가 좋은 모델, …)
- 컨텍스트를 올바르게 잘 제공해주고 있는지
  - 맥락 유지 및 지식 전달 (ex. TODO.md, DESIGN.md, …)
- 결과를 검토할 수 있는 Navigator의 리터러시
  - 올바르게 검토할 수 있도록 꾸준한 학습이 필요
  - AI에 지나치게 의존하지 않고, **주도적으로 사고할 수 있는 능력**
  - **요구사항에 부합하는지 철저하게 테스트** (논리적 오류, 잠재적 버그, 보안 취약점, …)
- 도구가 발전하는 시대에 꾸준히 따라잡고 학습할 수 있는 능력…

마치며...

발표자료를 준비하는 동안에도 강산이 변했습니다.

* 3월 20일 : Aider를 본격적으로 워크플로우에 녹여내기 시작
* **5월 1일** : **Claude Code에서 Todo list 지원**으로 작업계획의 가독성 개선
* **5월 11일 : 파이콘 CFP 마감**
* **5월 16일 : OpenAI에서 OpenAI Codex 공식 발표**
* **5월 23일 : Claude Code 1.0 공식 발표 / Claude Code에 Claude 4 Sonnet/Opus 모델 도입 발표**
* 6월 13일 : Claude Code를 본격적으로 사용
* **6월 19일 : 파이콘 한국 2025 발표자로 선정**
* **6월 25일 : Gemini CLI 공식 발표**
* **7월 3일 : Cursor 1.2 (Queued messages)**
* **7월 13일 : AWS에서 Kiro 공식 발표 (Spec Driven Development)**
* **7월 24일 : Claude Code에 subagents 기능 도입**
* **7월 31일 : MCP, LSP 통합이 포함된 오픈소스 코딩 에이전트 Crush 출시 (by Charm Bracelet)**
* **8월 8일 : Cursor CLI 출시**

그리고 Agentic Coding의 발전은 현재진행형으로 계속 되는 중…

하지만!!!!

코드를 빠르게 짜는 것은 빠르게 가설검증하고, 빠르게 실패하기 위한 수단

AI 도구가 계속해서 발전한다고 해서, 사람을 완전히 대체할 수는 없다.

우리가 풀고자 하는 문제는 사람이 해결해야 한다.

문제를 푸는데 실패한 책임은 기계가 지지 않는다. 결국엔 사람이 책임진다.

인간 자체가 강해져야 한다.

그리고 기계는 뜨개바늘에서 미싱머신으로 바뀌었을 뿐이다

읽으면 좋은 글들

- https://martinfowler.com/articles/exploring-gen-ai.html : 생성형 AI로 개발생산성을 높이는 것에 대한 여러가지 견해
- Django 창시자 Simon Willison의 LLM을 이용한 실험의 흔적
  - https://simonwillison.net/tags/vibe-coding/
  - https://simonwillison.net/tags/ai-assisted-programming/
- https://sourcegraph.com/blog/revenge-of-the-junior-developer
  - 우리의 일자리는 AI에 대체될 수 있을까? 소프트웨어 엔지니어는 미래에 어떻게 일하게 될까? 질문에 대한 답
  - 여러개의 Agent를 동시에 운용하는 미래는 이미 다가왔음
  - Geeknews 요약 : https://news.hada.io/topic?id=20068

감사합니다 :D

이재열
@kodingwarrior@hackers.pub