

바이브 코딩: 개념

#바이브코딩 #프롬프트엔지니어링 #프로그래밍활용

목차

- 01** 바이브 코딩이란
- 02** 다양한 바이브 코딩 서비스
- 03** 프롬프트 엔지니어링
- 04** 프로그래밍에서의 프롬프트 활용
- 05** 프로그래밍에서의 프롬프트 엔지니어링

바이브 코딩(Vibe Coding)

바이브 코딩이란?

LLM 모델에 자연어 프롬프트를 입력하여 코드를 생성 받아 문제를 해결하는 AI 기반 프로그래밍 방식



바이브 코딩과 기존 프로그래밍 비교

기능	기존 프로그래밍	바이브 코딩
코드 생성	한 줄씩 수동으로 코딩	자연어 프롬프트에서 AI가 생성
개발자 또는 사용자의 역할	설계자, 구현자, 디버거	프롬프트, 가이드, 테스터, 정제기
코딩 전문 지식 필수	비교적 높음	비교적 낮음
기본 입력	정확한 코드	자연어 프롬프트 및 피드백
개발 속도	일반적으로 느리고 체계적	더 간단한 프로토타입 제작 시 더 빠를 수 있음
오류 처리	코드 이해를 기반으로 한 수동 디버깅	대화형 피드백을 통한 개선
러닝 커브	높음	진입 장벽이 낮을 수 있음



바이브 코딩의 특징

❖ 대화형 코딩:

자연어로 요구사항을 설명하면 AI가 바로 코드로 변환

❖ 맥락 이해:

이전 대화, 프로젝트 구조, 변수명 등의 맥락을 이해하며 제안

❖ 빠른 반복:

효율 개선, 오류 해결 등 빠른 피드백 루프를 제공

❖ 자동 문서화:

함수 설명, 주석, 테스트 등 문서화까지 자동으로 생성

❖ 학습형 지원:

AI의 코드 제안을 통해 코딩 패턴과 문법을 자연스럽게 학습 가능



다양한 바이브 코딩 서비스



Github Copilot

코드 자동 완성형, VSCode 연동(플러그 기반), Github 학습 기반



The screenshot shows the VSCode interface with the GitHub Copilot extension active. The left panel displays a file named `runner-service.ts` containing the following code:

```
1 import { type Runner } from '@prisma/client';
2 import prisma from '$lib/data/prisma';
3
4 export interface RunnerService {
5   getById(id: number): Promise<Runner | null>;
6 }
7
8 const runnerService: RunnerService = {
9   getById: async (id: number) => {
10     return await prisma.runner.findUnique({
11       where: { id },
12     });
13   }
14 };
15
16 export default runnerService;
```

The right panel is titled "Edit with Copilot" and is set to "Agent mode". It includes instructions: "Start your editing session by defining a set of files that you want to work with. Then ask Copilot for the changes you want to make." Below this, it states: "Copilot is powered by AI, so mistakes are possible. Review output carefully before use." At the bottom, there are buttons for "Add Context..." and "service.md", and a message: "Edit files in your workspace in agent mode". The status bar at the bottom right shows "Agent" and "Claude 3.5 Sonnet".



Cursor AI

AI 기반 에디터, 실시간 코드 대화/제안 및 프로젝트 파일 맥락 인식



The screenshot shows the Cursor AI interface. On the left, there's a sidebar with project files: ML-RESEARCH-NOTEBOOK, notebooks, train_model.py, evaluation.py, experiments, config.yaml, run_experiment.py, and requirements.txt. The main area is a code editor with the file train_model.py open. The code is:

```
import torch
import torch.nn as nn
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
from tqdm import tqdm
import yaml
from pathlib import Path
import json

def get_dataloaders(batch_size=64):
    transform = transforms.Compose([transforms.ToTensor()])
    train = datasets.MNIST(root="data", train=True)
    test = datasets.MNIST(root="data", train=False)
    return DataLoader(train, batch_size=batch_size)
def load_config(config_path="experiments/config.yaml"):
    with open(config_path) as f:
        return yaml.safe_load(f)

def get_dataloaders(config):
    transform_list = [transforms.ToTensor()]
    if config['data'].get('normalize', True):
        transform_list.append(transforms.Normalize((0.1307,), (0.3081,)))
```

To the right of the code editor, there's a sidebar titled "PyTorch MNIST Experiments" with the following text:

Add mixed precision training, learning rate scheduling, and proper validation. Also create an experiment config system so I can easily run different hyperparameter settings.

Now let me update the evaluation module to save results and generate a detailed report:

evaluation.py +24-6

Let me create experiments easy to execute:

run_experiment.py +29

Done, configurable MNIST experiment framework with AMP and reports.

- Training: AMP, train/val split, cosine schedule, gradient clipping, checkpoints
- Experimentation: YAML config, saved history, confusion matrix + classification report, CLI runner

Plan, see cursor-agent

+ Add a follow-up

GPT-5

/ for commands . for files

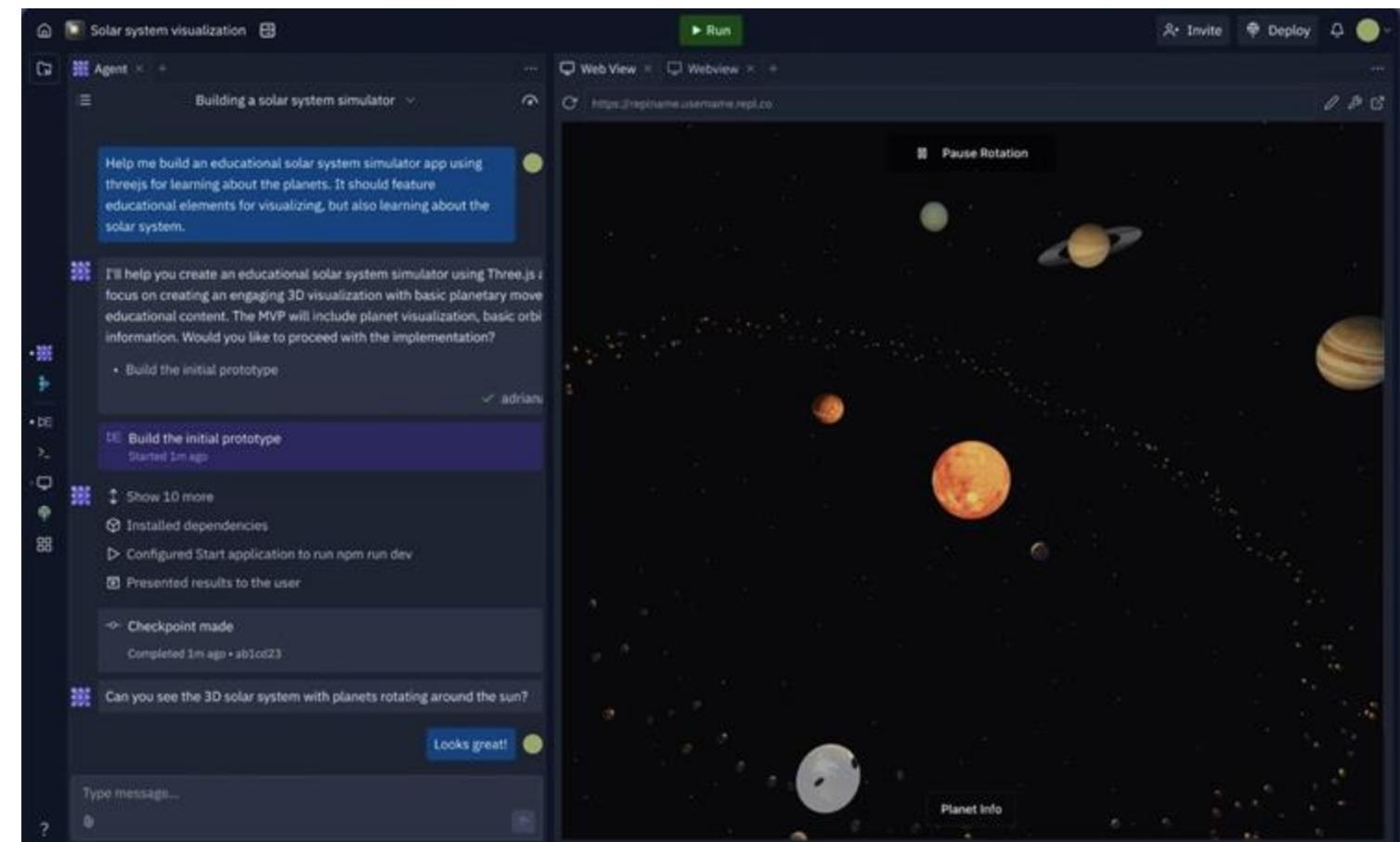


Replit

온라인 IDE에서 AI 대화형 코딩 지원(아이디어 -> 앱 제작)



replit

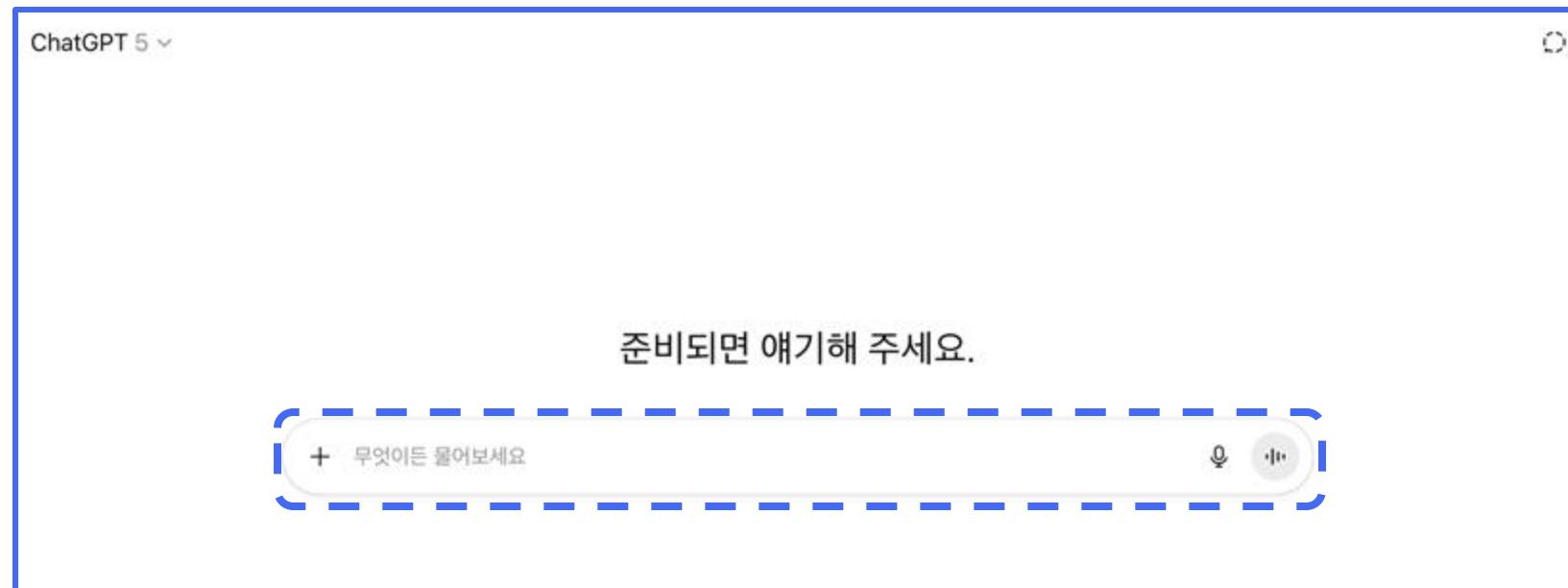




프롬프트 엔지니어링

프롬프트(Prompt)

인공지능에게 전달하는 질문이나 지시





프롬프트 엔지니어링의 필요성





프롬프트 엔지니어링의 필요성





프롬프트 엔지니어링의 필요성





프롬프트 엔지니어링의 필요성



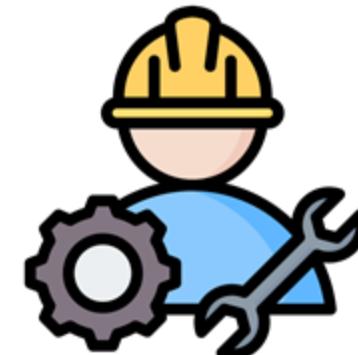
구체적인 요청 → 원하는 결과



프롬프트 엔지니어링(Prompt Engineering)

프롬프트와 엔지니어링의 합성어

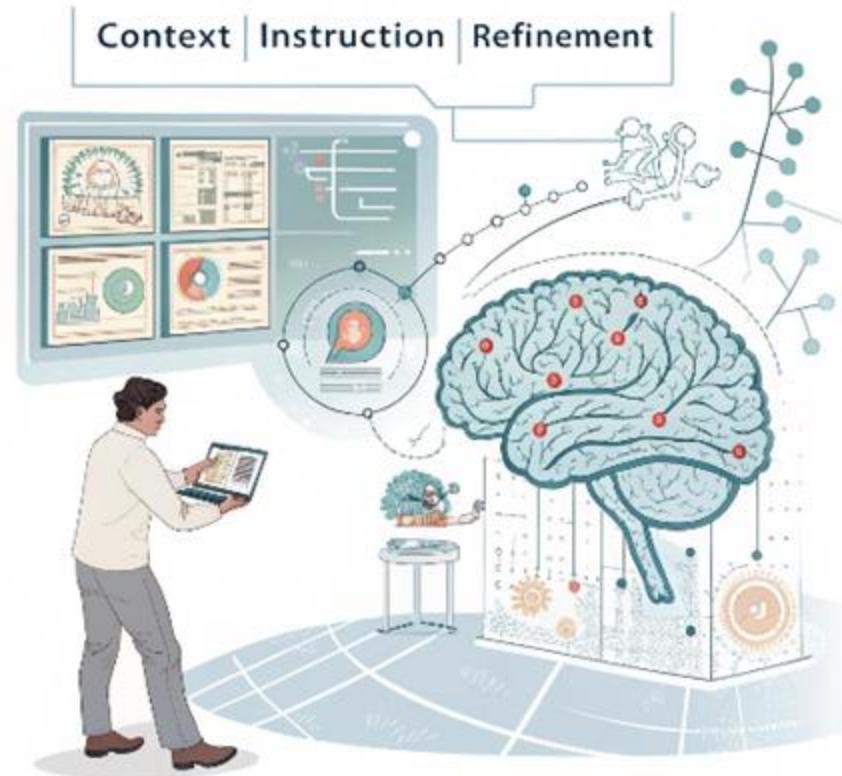
프롬프트 + 엔지니어링
(Prompt + Engineering)





프롬프트 엔지니어링(Prompt Engineering)

인공지능에게 원하는 결과물을 잘 뽑아내기 위해 프롬프트를 잘 입력하는 방법들



프롬프트에 포함된 예제, 명령, 단어 선택 등
작은 변화에 따라서 응답 결과가 완전히 달라질 수
있음

원하는 결과물을 얻기 위해서는 좋은 프롬프트를
설계할 필요가 있음

좋은 프롬프트 설계 = 프롬프트 엔지니어링



프롬프트 엔지니어링을 위한 핵심 개념

1. 작업(Task)

동사, 서술어를 목적이 분명하게 드러나도록 명료하게 작성

2. 맥락(Context)

작업과 연관있는 내용 작성(배경, 목적, 환경적 요인 등)

3. 인터렉션(Interaction)

단발성 또는 연속성 상호작용



프롬프트 엔지니어링 기법

핵심적인 기법

역할을 설정해서 질문하기

결과물을 명확히 작성하기

답변의 형식 지정하기

형식 구조화하기

프롬프트 엔지니어링 기법: 역할을 설정해서 질문하기

전문가 등 역할을 부여해, 그 역할처럼 행동하며 몰입감 있게 답변(Act as ~)

데이터 분석에 대해 알려줘

단순하게 요청

너는 20년차 데이터 분석가야.
데이터 분석에 대해 알려줘

역할을 부여하여, 요청



프롬프트 엔지니어링 기법: 역할을 설정해서 질문하기

전문가 등 역할을 부여해, 그 역할처럼 행동하며 몰입감 있게 답변(Act as ~)

데이터 분석에 대해서 알려줘

데이터 분석은 데이터를 통해 의미 있는 통찰(insight)을 얻고, 의사결정에 활용하는 과정이에요.

아주 쉽게 말하면, 데이터를 관찰하고 → 정리하고 → 해석해서 → 문제를 해결하는 일이라고 볼 수 있습니다.

아래는 단계별로 핵심을 정리한 개요입니다 🌟

너는 20년차 데이터 분석가야. 데이터 분석에 대해서 알려줘

💡 1. 데이터 분석은 “숫자를 다루는 기술”이 아니라 “의사결정의 언어”다

많은 사람들이 데이터 분석을 “엑셀 잘하는 사람”, “파이썬으로 그래프 그리는 사람”이라고 오해합니다.
하지만 실제로는 “**조직의 의사결정을 데이터로 설득하는 일**”이에요.

데이터 분석의 진짜 목적은 단순히 결과를 보여주는 게 아니라,

“무엇을 해야 할지를 명확히 말해주는 것”
입니다.

예를 들어

“이번 달 매출이 15% 하락했습니다.”
이건 단순한 리포트입니다.
하지만
“지난달 대비 신규 유입은 유지됐지만, 재구매율이 40%→25%로 급감했습니다. 이달 고객의 70%가 A 상품군 사용자였고, 이들의 주요 불만은 배송 지연이었습니다.”
이건 데이터 분석입니다.
즉, 데이터를 통해 행동(Action)을 이끌어내는 것이죠.

1. 데이터 분석의 개념

데이터 분석(Data Analysis)은 수집된 데이터를 탐색·정리·가공·해석하여 패턴, 추세, 관계 등을 찾아내는 과정입니다.

목표는 단순히 수치를 계산하는 것이 아니라, 문제를 해결하거나 의사결정에 도움을 주는 “이야기”를 만드는 것이에요.



프롬프트 엔지니어링 기법

핵심적인 기법

역할을 설정해서 질문하기

결과물을 명확히 작성하기

답변의 형식 지정하기

형식 구조화하기



프롬프트 엔지니어링 기법: 결과물을 명확히 작성하기

작업(Task)에 대해 구체적으로 지정(사용자의 의도, 목적, 상황을 구체적으로 제시)

리스트를 정렬하는 코드를 만들어줘

리스트를 정렬하는 방법은 여러 가지가 있지만, 가장 기본적인 예시는 파이썬의 `sort()` 메서드나 `sorted()` 함수를 사용하는 것입니다. 아래에 예시를 보여드릴게요 ↴

✓ 방법 1: `list.sort()` 사용 (원본 리스트가 변경됨)

python

코드 복사

```
numbers = [5, 2, 9, 1, 7]
numbers.sort()
print(numbers) # [1, 2, 5, 7, 9]
```

단순하게 작업 요청

✓ 방법 2: `sorted()` 함수 사용 (새로운 리스트 반환)

python

코드 복사

```
numbers = [5, 2, 9, 1, 7]
sorted_numbers = sorted(numbers)
```





프롬프트 엔지니어링 기법: 결과물을 명확히 작성하기

작업(Task)에 대해 구체적으로 지정(사용자의 의도, 목적, 상황을 구체적으로 제시)

사용자로부터 입력받은 5개의 정수를 오름차순으로 정렬해 출력하는 파이썬 코드를 작성해줘.
단, 입력값은 공백으로 구분되며, 결과는 한 줄로 출력되게 해줘.

아래는 요구사항에 맞는 파이썬 코드입니다 ✨



python

☞ 코드 복사

```
# 사용자로부터 입력받기 (공백으로 구분된 5개의 정수)
numbers = list(map(int, input("정수 5개를 공백으로 구분하여 입력하세요: ").split()))

# 오름차순 정렬
numbers.sort()

# 한 줄로 출력
print(*numbers)
```

구체적으로 명시





프롬프트 엔지니어링 기법

핵심적인 기법

역할을 설정해서 질문하기

결과물을 명확히 작성하기

답변의 형식 지정하기

형식 구조화하기



프롬프트 엔지니어링 기법: 답변의 형식 지정하기

표, 리스트 등 출력의 형식을 지정(구체적인 템플릿 또는 예시)

나이가 30이상이고, 성별이 남자인 데이터를 조회하는 파이썬 코드를 작성해줘

물론이죠! 👍 아래는 Pandas를 사용하여 “나이가 30 이상이고 성별이 남자(‘남’ 또는 ‘Male’)인 행만 조회”하는 예시 코드입니다.

✓ 예시 코드

python

```
import pandas as pd

# 예시 데이터프레임 생성
data = {
    '이름': ['홍길동', '이영희', '김철수', '박민수', '최지훈'],
    '나이': [28, 32, 45, 25, 37],
    '성별': ['남', '여', '남', '남', '남']
}

df = pd.DataFrame(data)
```

코드 복사

나이가 30이상이고, 성별이 남자인 데이터를 조회하는 SQL 쿼리를 작성해줘

물론입니다 😊

다음은 나이가 30 이상이고 성별이 ‘남’인 데이터만 조회하는 SQL 쿼리 예시입니다.

✓ 기본 SQL 예시

sql

```
SELECT *
FROM 사용자
WHERE 나이 >= 30
AND 성별 = '남';
```

코드 복사



프롬프트 엔지니어링 기법: 답변의 형식 지정하기

응용으로 추가 제한 조건 설정하기

파이썬으로 퀵 정렬 알고리즘에 대해 알려줘
초등학교 저학년이 이해할 수 있도록 쉽게 설명해줘

좋아요 😊

이해하기 쉽게, 초등학교 저학년 수준으로 **퀵 정렬(Quick Sort)**을 이야기해볼게요.

답변의 수준, 분량, 제한 등 추가

1 퀵 정렬이란?

"퀵 정렬은 정리 정돈을 아주 똑똑하게, 빠르게 하는 방법이에요!"

예를 들어, 친구들이 키 순서대로 줄 서야 한다고 해보자 😊😊😊😊

그런데 한 명이 가운데 서서 이렇게 말해요:

"나보다 키 작은 애들은 내 왼쪽!

나보다 키 큰 애들은 내 오른쪽!"

이렇게 사람들을 왼쪽(작은 쪽), **오른쪽(큰 쪽)**으로 나누는 거예요.

그다음에, 왼쪽 무리끼리 또 같은 방법으로 나누고,

오른쪽 무리도 또 같은 방법으로 나누는 걸 반복해요.





프롬프트 엔지니어링 기법

핵심적인 기법

역할을 설정해서 질문하기

결과물을 명확히 작성하기

답변의 형식 지정하기

형식 구조화하기



프롬프트 엔지니어링 기법: 형식 구조화하기

원하는 틀(Format)에 맞춰서 답변을 요청하는 것 (마크다운 또는 사용자 정의)

리스트의 평균을 구하는 파이썬 함수를 작성해줘.

다음 형식을 반드시 따라야 해:

1. 함수명: calculate_average
2. 매개변수: numbers (list of int or float)
3. 반환값: float
4. 주석: 함수 설명 포함
5. 사용 예시 포함

“데이터 전처리”를 주제로 강의 노트를 만들어줘.

형식은 다음과 같아야 해:

제목
정의
주요 단계
예시 코드
주의사항

텍스트 구조화 및 가독성 향상



프롬프트 엔지니어링 기법: 형식 구조화하기

원하는 틀(Format)에 맞춰서 답변을 요청하는 것 (마크다운 또는 사용자 정의)

```
def calculate_average(numbers):
    """
    리스트의 평균값을 계산하여 반환한다.

    :param numbers: 숫자로 이루어진 리스트
    :return: 평균값 (float)
    """

    if not numbers:
        return 0.0
    return sum(numbers) / len(numbers)

# 사용 예시
data = [10, 20, 30, 40]
result = calculate_average(data)
print(result) # 25.0
```

데이터 전처리

정의

데이터를 분석에 적합한 형태로 정리하는 과정이다.

주요 단계

1. 결측치 처리
2. 이상치 제거
3. 데이터 타입 변환

예시 코드

```
```python
df.fillna(0, inplace=True)
```



## 프롬프트 엔지니어링 기법 적용 예시

### 회의실 배정

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	264984	90930	62503	31.839%

### 문제

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의  $i$ 에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

### 입력

첫째 줄에 회의의 수  $N$  ( $1 \leq N \leq 100,000$ )이 주어진다. 둘째 줄부터  $N+1$  줄까지 각 회의의 정보가 주어지는데 이것은 공백을 사이에 두고 회의의 시작시간과 끝나는 시간이 주어진다. 시작 시간과 끝나는 시간은  $2^{31}-1$ 보다 작거나 같은 자연수 또는 0이다.

### 출력

첫째 줄에 최대 사용할 수 있는 회의의 최대 개수를 출력한다.



## 프롬프트 엔지니어링 기법 적용 예시

### [문제]

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의 I에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

### [입력]

첫째 줄에 회의의 수 N( $1 \leq N \leq 100,000$ )이 주어진다. 둘째 줄부터  $N+1$  줄까지 각 회의의 정보가 주어지는데 이것은 공백을 사이에 두고 회의의 시작시간과 끝나는 시간이 주어진다. 시작 시간과 끝나는 시간은 231-1보다 작거나 같은 자연수 또는 0이다.

### [출력]

첫째 줄에 최대 사용할 수 있는 회의의 최대 개수를 출력한다.

### [입력 예시]

```
11
14
3 5
0 6
5 7
3 8
5 9
6 10
8 11
8 12
2 13
12 14
```

### [예시 출력]

```
4

```

너는 20년차 파이썬 능군가야. 위 문제를 구현하는 파이썬 코드를 작성해줘

다음은 고전적인 그리디(활동 선택) 풀이입니다.

핵심 아이디어: 끝나는 시간이 빠른 회의부터 선택하면 항상 최적입니다. (끝시간↑, 등률이면 시작시간↑로 정렬)

python

☞ 코드 복사

```
import sys

def max_meetings(intervals):
 # 끝나는 시간, 시작 시간 기준으로 정렬
 intervals.sort(key=lambda x: (x[1], x[0]))
 count = 0
 current_end = -1

 for s, e in intervals:
 # 직전 회의가 끝난 시각(current_end)과 같거나 이후에 시작하면 선택
 if s >= current_end:
 count += 1
 current_end = e
 return count

def main():
 input = sys.stdin.readline
 n_line = input().strip()
 # 빈 입력 방어
 if not n_line:
 print(0); return
 n = int(n_line)

 intervals = []
 for _ in range(n):
 s, e = map(int, input().split())
 intervals.append((s, e))

 print(max_meetings(intervals))

if __name__ == "__main__":
 main()
```

## 고급 프롬프트 엔지니어링 전략

**Few-Shot Prompting**

**Chain of Thought (CoT)**



## 고급 프롬프트 엔지니어링 전략: In Context Learning

주어진 입력에서 패턴을 파악해 알맞은 응답을 생성





## 고급 프롬프트 엔지니어링 전략: Few-Shot 프롬프팅

In-Context Learning을 활용하여 예시를 통해 원하는 응답 유도

**Zero-Shot  
Prompting**

예시 없이 입력

**One-Shot  
Prompting**

예시 1개와 함께 입력

**Few-Shot  
Prompting**

예시 여러 개와 함께 입력  
(개수에 따라 K-Shot  
Prompting이라 불림)



## 고급 프롬프트 엔지니어링 전략: Few-Shot 프롬프팅

In-Context Learning을 활용하여 예시를 통해 원하는 응답 유도

예시1

```
예시 1
설명: 리스트의 합을 구하는 함수를 작성하세요.
출력:
def list_sum(lst):
 return sum(lst)
```

예시2

```
예시 2
설명: 문자열을 뒤집는 함수를 작성하세요.
출력:
def reverse_string(s):
 return s[::-1]
```

새로운 예제

```
새로운 입력
설명: 주어진 숫자의 제곱을 구하는 함수를 작성하세요.
출력:
```

예시들의 일관된 패턴에 맞춰서  
답변 생성



## 고급 프롬프트 엔지니어링 전략: Chain of Thought(CoT)

### 언어모델의 약점: 복잡한 추론의 한계

질문: 철수는 5개의 테니스 공을 갖고 있습니다. 그는 2 개의 테니스 공 세트를 샀습니다. 각각의 테니스 공 세트에는 3개의 테니스 공이 들어 있습니다. 철수는 지금 몇 개의 테니스 공을 갖고 있을까요?

답변: 답은 11개입니다.

질문: 카페에는 23개의 사과가 있습니다. 점심시간동안 20개의 사과를 사용했고 6개의 사과를 추가로 사왔습니다. 지금 몇 개의 사과가 있을까요?



답변: 답은 27개입니다.



## 고급 프롬프트 엔지니어링 전략: Chain of Thought(CoT)

정답만 요구하지 않고, 사고 과정을 논리적으로 설명하도록 유도

질문: 철수는 5개의 테니스 공을 갖고 있습니다. 그는 2개의 테니스 공 세트를 샀습니다. 각각의 테니스 공 세트에는 3개의 테니스 공이 들어 있습니다. 철수는 지금 몇 개의 테니스 공을 갖고 있을까요?

답변: 철수는 처음에 5개의 공을 갖고 있습니다. 3개의 테니스 공이 들어있는 세트 2개에는 총 6개의 테니스 공이 있습니다.  $5+6=11$ . 답은 11개입니다.

질문: 카페에는 23개의 사과가 있습니다. 점심시간동안 20개의 사과를 사용했고 6개의 사과를 추가로 사왔습니다. 지금 몇 개의 사과가 있을까요?



답변: 카페에는 원래 23개의 사과가 있었습니다. 20개를 점심을 만드는데 썼습니다. 그래서  $23-20=3$ 개가 남습니다. 6개의 사과를 더 샀으므로  $3+6=9$ . 답은 9개입니다.



## 고급 프롬프트 엔지니어링 전략: Chain of Thought(CoT)

정답만 요구하지 않고, 사고 과정을 논리적으로 설명하도록 유도

입력: 17

출력: 소수입니다.

일반 프롬프트

입력: 17이 소수인지 판단해보자.

단계별로 생각해보자.

1. 17은 1보다 크다.

2. 2부터 16까지 나누어 떨어지는 수가 있는지 확인하자.

3.  $17 \% 2 = 1$ ,  $17 \% 3 = 2$ , ...,  $17 \% 16 = 1 \rightarrow$  나누어 떨어지지 않는다.  
따라서 17은 소수이다.

출력: 소수입니다.

CoT 적용 프롬프트

Let's think step by step



# 프로그래밍에서의 프롬프트 활용

## 프로그래밍을 위한 프롬프트 활용

### 자연어를 활용하여 인공지능에게 원하는 코드 생성 명령

주어진 텍스트가 영어인지 구분하는 코드를 파이썬으로 작성해줘

좋아요 👍

영문 텍스트인지 판별하는 방법은 여러 가지가 있습니다. 여기서는 간단한 규칙 기반 방식부터 라이브러리 활용 방식까지 단계별 예시를 보여드릴게요.

✓ 1 기본적인 정규식 기반 판별

영문 알파벳(a~z, A~Z)이 주로 포함되어 있는지 확인하는 방식입니다.

python

코드 복사

```
import re

def is_english(text):
 # 알파벳과 공백, 일부 문장부호만 허용
 pattern = re.compile(r'^[A-Za-z0-9\s.,!?\'"()\-\-]+$')
 return bool(pattern.match(text))

예시
texts = [
 "Hello world!",
 "안녕하세요!",
 "こんにちは",
 "Привет!",
 "Hello"
]
```

↓

# 프로그래밍에서의 활용

## 프로그래밍을 위한 프롬프트 활용

### (1) 기능 구현 요청: 간단한 프롬프트

프롬프트

주어진 텍스트가 영어인지 구분하는 코드를  
파이썬으로 작성해줘

주어진 텍스트가 영어인지 구분하는 코드를 파이썬으로 작성해줘

좋아요 👍

영문 텍스트인지 판별하는 방법은 여러 가지가 있습니다. 여기서는 간단한 규칙 기반 방식부터 라이브러리 활용 방식까지 단계별 예시를 보여드릴게요.

✓ ⓘ 기본적인 정규식 기반 판별

영문 알파벳(a~z, A~Z)이 주로 포함되어 있는지 확인하는 방식입니다.

python

코드 복사

import re

def is\_english(text):

# 알파벳과 공백, 일부 문장부호만 허용

pattern = re.compile(r'^[A-Za-z0-9\s.,!?\\"(\")\-\]+\$')

return bool(pattern.match(text))

# 예시

texts = [

"Hello world!",



# 프로그래밍에서의 활용

## 프로그래밍을 위한 프롬프트 활용

### (1) 기능 구현 요청: 구체적인 프롬프트

프롬프트

단일 폴더에 “[카테고리]\_[숫자].jpg” 네이밍 컨벤션을 따르는 이미지 파일이 혼재되어 있어. 이를 각 카테고리 별로 폴더를 만든 후, 분류하는 코드를 파이썬으로 작성해줘

아래는 [카테고리]\_[숫자].jpg 형태의 파일 이름을 가진 이미지들을 카테고리별 폴더로 자동 분류하는 파이썬 코드입니다

👉

✓ 코드 예시

python

코드 복사

```
import os
import shutil

이미지들이 들어 있는 폴더 경로
base_dir = "images" # 예: "C:/Users/user/Desktop/images"

폴더 내의 모든 파일 확인
for filename in os.listdir(base_dir):
 if filename.endswith(".jpg"):
 # 파일명에서 카테고리 추출 (예: "cat_1.jpg" → "cat")
 category = filename.split("_")[0]

 # 카테고리 폴더 경로 설정
 category_folder = os.path.join(base_dir, category)
 ↓

 # 폴더가 없으면 생성
```

## 프로그래밍을 위한 프롬프트 활용

### (1) 기능 구현 요청: 구체적인 프롬프트

Prompt Engineering?

프롬프트

맥락 정보

단일 폴더에 “[카테고리]\_[숫자].jpg” 네이밍 컨벤션을 따르는 이미지 파일이 존재되어 있어.

Task 정보

이를 각 카테고리 별로 폴더를 만든 후, 분류

결과 형식

코드를 파이썬으로 작성해줘

## 프로그래밍을 위한 프롬프트 활용

### (2) 코드 언어 변환 요청

프롬프트

아래 주어진 C++ 코드를 파이썬으로 변환해줘

```
#include <iostream>
using namespace std;

int main() {
 cout << "Hello, World!" << endl;
 return 0;
}
```

물론입니다 😊  
아래는 해당 C++ 코드를 동일한 동작을 하는 파이썬 코드로 변환한 버전입니다 👍

Python 버전

python

코드 복사

```
print("Hello, World!")
```

설명

- #include 나 using namespace 같은 선언은 파이썬에서는 필요 없습니다.
- main() 함수도 기본적으로 생략 가능하며,
- 단순히 print() 함수로 문자열을 출력하면 됩니다.

실행 결과:

코드 복사

Hello, World!

↓

## 프로그래밍을 위한 프롬프트 활용

### (3) 코드 해석 요청

프롬프트

```
print('.join(chr((i*7)%26 + 65) for i in [sum(map(ord, w))%26 for w in input("단어들을 입력하세요: ").split()]))
```

이 코드가 무슨 코드인지 해석해줘

어렵다면 쉽게 설명해달라고 추가 요청하자!

한 단계씩 분석

① `input("단어들을 입력하세요: ")`

사용자로부터 여러 개의 단어를 입력받습니다.

예를 들어 `hello world ai`

nginx

python

이 문자열이 입력되면 `split()`을 통해 공백 기준으로 나뉩니다.

python

② `[sum(map(ord, w)) % 26 for w in ...]`

각 단어 `w`에 대해:

- `map(ord, w)` → 각 문자를 아스키 코드(정수)로 변환

예: `'a' -> 97, 'b' -> 98`

## 프로그래밍을 위한 프롬프트 활용

### (4) 코드 기능 개선 요청(Refactoring); 기능 확장, 성능 개선, 안정성, 편의성 개선

프롬프트

[파이썬 버블 정렬 코드]  
이 코드의 정렬 알고리즘을 더 효율적으로  
개선해줘

좋아요! 버블 정렬( $O(n^2)$ )보다 훨씬 효율적인 정렬로 바꿔볼게요.  
파이썬에서는 \*\*내장 정렬(Timsort)\*\*이 가장 빠르고 안정적입니다.

가장 권장: 내장 정렬 사용 (Timsort, 평균/최악  $O(n \log n)$ , 거의 정렬된 경우  $O(n)$ )

```
python

def fast_sort(arr):
 arr.sort() # 원본 리스트 정렬 (제자리, 안정적)
 return arr

예시
nums = [5, 3, 8, 4, 2]
print(fast_sort(nums)) # [2, 3, 4, 5, 8]
```

코드 복사

📘 학습용 대안 1: 병합 정렬 (Merge Sort, 안정적,  $O(n \log n)$ , 추가 메모리  $O(n)$ )

```
python

def merge_sort(a):
 if len(a) <= 1:
 return a
 mid = len(a) // 2
```

코드 복사

## 프로그래밍을 위한 프롬프트 활용

### (5) 코드 가독성 개선 요청(Reformatting)

로직은 동일하게 유지하면서, 변수명/함수명/들여쓰기/주석 등을 개선하여 코드 가독성 품질 향상

#### 프롬프트

```
def c(x):
 if x>0:
 if x%2==0:
 print("Even")
 else:
 print("Odd")
```

이 코드를 가독성 있게 리팩토링해줘.  
- 변수와 함수명은 의미 있게 변경  
- 주석을 한국어로 추가

가독성을 높이기 위해 변수명과 함수명을 의미 있게 변경하고,  
한국어 주석을 추가한 리팩토링 버전은 아래와 같습니다 🎉

python

코드 복사

```
def print_number_type(number: int) -> None:
 """
 입력된 숫자가 양수일 경우 짝수/홀수를 판별하여 출력합니다.
 """

 # 숫자가 양수인지 확인
 if number > 0:
 # 짝수인 경우
 if number % 2 == 0:
 print("짝수입니다.")
 # 홀수인 경우
 else:
 print("홀수입니다.")
 # 음수 또는 0인 경우
 else:
 print("0 이하의 숫자는 판별하지 않습니다.")
```

# 프로그래밍에서의 활용

## 프로그래밍을 위한 프롬프트 활용

### (6) 오류 메시지 분석/해결 요청

#### 프롬프트

# 문제: 서로 다른 타입을 더함

```
a = 10
b = "5"
c = a + b
```

위 코드에서 아래와 같은 오류가 발생했어. 원인이 무엇인지 알려주고, 오류를 수정해줘

```
line 4, in <module>
 c = a + b # TypeError: unsupported operand type(s)
for +: 'int' and 'str'
 ~~^~~
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

The screenshot shows a web-based tool for analyzing errors. It has a sidebar on the left with a blue button labeled '프롬프트'. The main area is divided into several sections:

- 오류 원인 분석**: Shows the original code: 

```
python
a = 10
b = "5"
c = a + b
```

 and the resulting error message: 

```
bash
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

.
- 원인**: Lists the causes of the error:
  - a는 정수(int),
  - b는 문자열(str)입니다.
  - 파이썬에서는 int + str 연산을 허용하지 않습니다.  
( + 연산자는 숫자끼리 더하거나, 문자열끼리 이어붙일 때만 가능)It also notes that the types differ because addition is only allowed between numbers or strings.
- 수정 방법**:
  - 방법 1: 문자열을 정수로 변환 (int() 사용)**: Shows the corrected code: 

```
python
a = 10
b = "5"
c = a + int(b)
print(c) # 출력: 15
```

 with a note: "숫자 덧셈으로 계산을 수행할 수 있습니다."
  - 방법 2: 정수를 문자열로 변환 (str() 사용)**: Shows another way to correct it: 

```
python
a = 10
b = "5"
c = str(a) + b
print(c) # 출력: "105"
```



# 프로그래밍에서의 프롬프트 엔지니어링

## 프롬프트 작성

### 1. 코드를 작성하는 환경의 맥락 정보 제공

- Jupyter Notebook 환경에서 코드를 작성 중이야.
- 나는 지금 Linux 환경이야.
- 지금 파이썬 버전을 3.10을 사용 중이야.

언어, 프레임워크, 버전,  
라이브러리, IDE, 운영체제 등

어떤 상황에서 코드를 작성?

## 프로그래밍을 위한 프롬프트 활용

### 2. 명확하고 구체적으로 프롬프트 작성(**Task 명료화**)

- 사용할 함수는 `calculate_bmi`로 설정해줘
- 반복문은 `for`문 말고 `while`문을 사용해줘
- `class`와 함수를 사용해서 최대한 간결한 코드를 작성해줘

입출력, 제약조건, 제한, 형식 등

무엇을 원하나?

## 프로그래밍을 위한 프롬프트 활용

### 3. 작은 단위로 프롬프트를 구성하여 원하는 결과물 도출하기

나는 파이썬을 활용해 아래 Step을 기반으로 회귀 분석을 할거야.

1. 회귀 분석을 진행하는 파이썬 코드를 작성하고 싶어. 우선 csv 파일 데이터를 불러오는 코드를 작성해줘.
2. 불러온 데이터는 5개의 변수를 포함하고 있고, 총 데이터는 300개야. 이 데이터에서 결측치를 제거하는 전처리 코드를 파이썬으로 작성해줘.
3. 결측치가 제거된 데이터에 이상치가 있는지 확인하기 위해 Box Plot을 그려보고 싶어. 그리는 파이썬 코드도 이어서 작성해줘.
4. 전처리 완료된 데이터를 사용해서 회귀 분석을 진행할건데, 회귀 분석 모델을 Pytorch를 사용해서 작성해줘.

세분화된 단계별 요청으로 점진적 결과 도출

## 프로그래밍을 위한 프롬프트 활용

### 4. 오류 상황 발생 시, 수정 요청

오류가 나는 상황은 비일비재!

때로는 AI를 통해 오류가 포함된 코드가 생성될 수 있기에,  
즉시 사용이 아닌 검토가 필요함.

수정 요청, 수정 방향 제시!

ex) 존재하지 않는 함수, 허용되지 않는 방식 구현

- \* 오류 메시지 전체 그대로 복사
- \* 어떤 부분, 어떤 동작 중 발생했는지 명시
- \* 원인 + 수정된 코드 형태로 요청

오류 발생 시, 해당 오류 메시지와 함께 수정 요청

## AI가 코드를 짜주는 시대!.. 하지만?

기반(base) 지식 및 용어를 알아야 더 정확한 원하는 답변을 얻을 수 있다.

무엇이 잘못됐는지  
알 수 있어야 한다

기능을 원하는 대로  
생성/수정 받기

기본 용어 및 개념을  
알아야 한다

어디서? 왜?

자유자재로 커스터마이징

정확한 표현 = 정확한 답변

ex) 이상한 값 (X), 결측치 (O)

# 감사합니다!