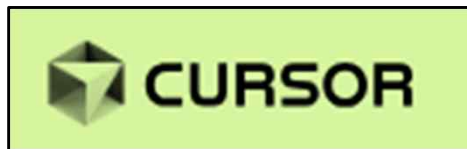


Cursor 기능 사용법 : Inline Edit (Cmd/Ctrl+K)



1.1 Cmd/Ctrl+K



Inline Edit

Cursor의 **Inline Edit** 기능은 에디터 내에서 코드 수정 과정을 보다 빠르고 직관적으로 할 수 있게 해주는 AI 보조 기능이다

1. 기본 개념

- 선택된 코드 블록을 AI가 이해하고 수정 제안을 직접 반영할 수 있게 한다.
- 즉, 주석을 쓰거나 커밋처럼 별도의 창을 열지 않고, 현재 코드 라인에서 바로 수정 가능하다.
- 기존 코드의 일부를 드래그하거나 커서로 지정한 후, "Inline Edit" 명령을 실행하면 된다.

1.1 Cmd/Ctrl+K

2. 동작 방식

1. **코드 선택**: 수정하고 싶은 코드 영역을 블록 선택.
2. **명령 실행**: `Ctrl+K` (기본 단축키) 또는 우클릭 메뉴에서 *Inline Edit* 선택.
3. **프롬프트 입력**: "이 부분을 async 함수로 바꿔줘" / "Python 3.12 문법에 맞게 변경" 등 요청 입력.
4. **즉시 수정**: AI가 제안한 코드가 기존 코드 위에 바로 반영된다.
 - 원본은 diff(차이) 형태로 표시되어 사용자가 수락하거나 되돌릴 수 있음.

단축키

- Windows/Linux : **Ctrl + K**
- macOS : **Cmd + K**

(⌘)

3. 주요 장점

- **빠른 반복:** 코드 창을 벗어나지 않고 즉시 수정 가능.
- **맥락 유지:** 선택된 범위의 전후 맥락을 함께 참고하므로 더 정확한 수정 가능.
- **Diff 확인:** 변경 전/후 비교가 자동 제공되어 실수 방지.
- **실시간 리팩토링:** 변수명 일괄 변경, 함수화, 오류 수정 등을 빠르게 수행.

4. 활용 예시

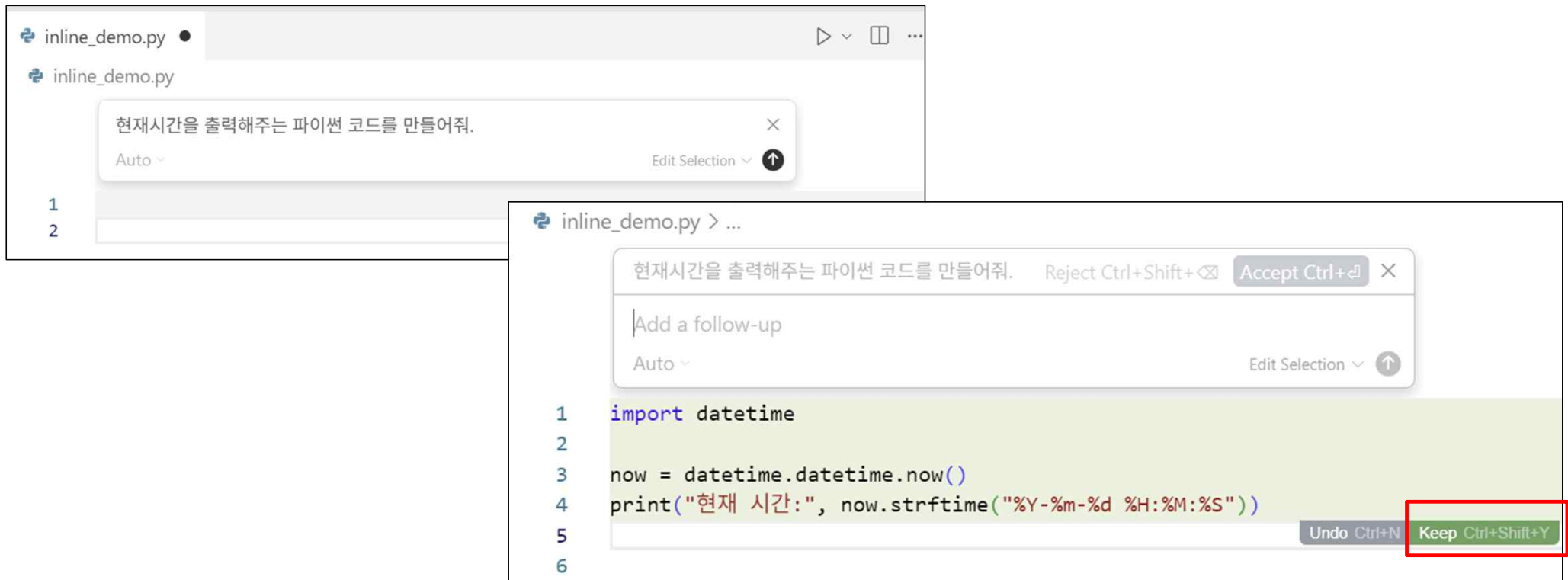
- Python 코드에서 for-loop를 list comprehension으로 바꾸기.
- JS 함수에 TypeScript 타입 추가하기.
- SQL 쿼리 최적화 요청하기.
- 함수에 주석 자동 추가하기.

1.1 Cmd/Ctrl+K

프로젝트에서 새 파일(inline_demo.py)을 만들고 Ctrl+K를 누른 다음 아래내용 입력.

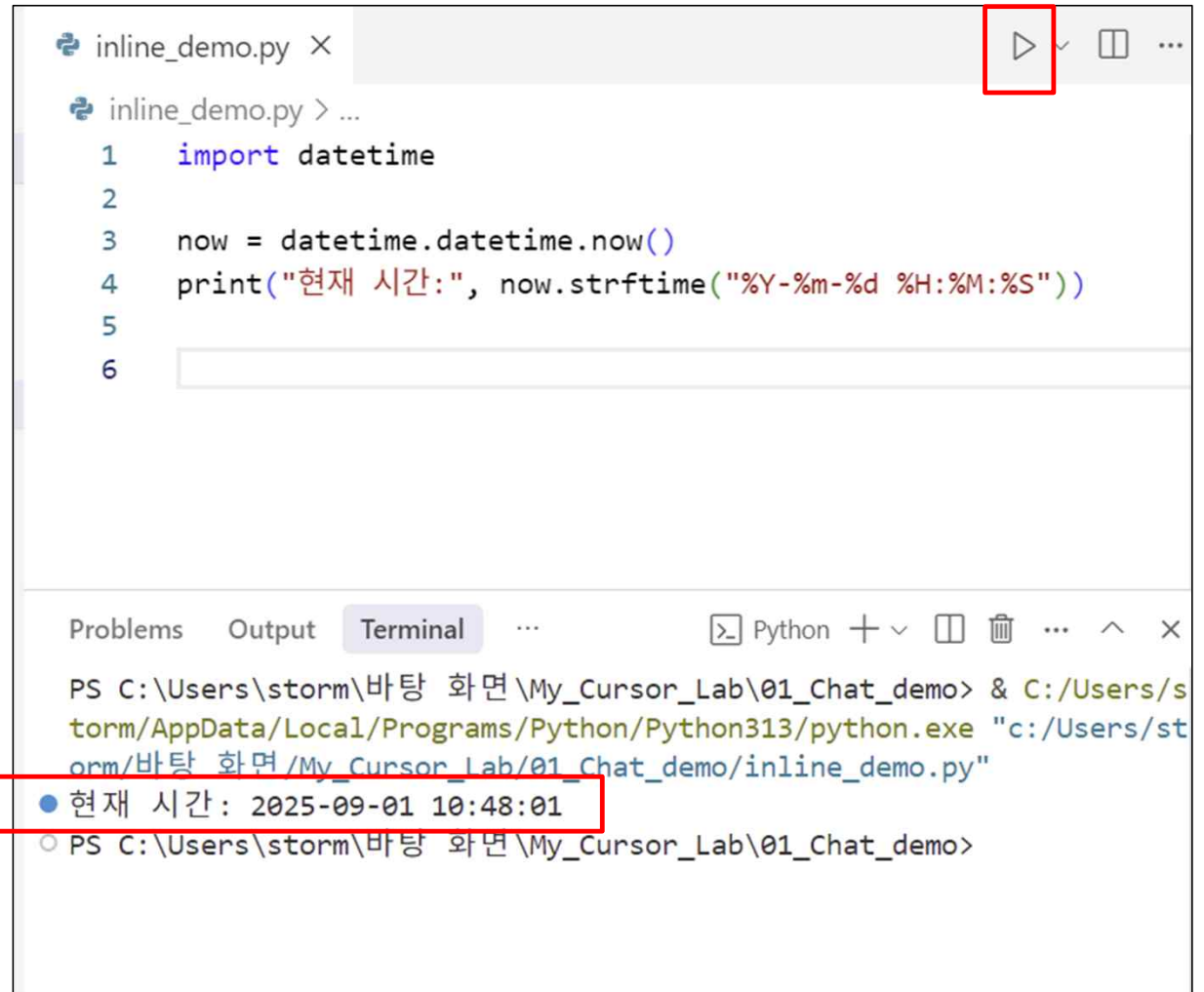
“현재 시간을 출력해주는 파이썬 코드를 만들어줘”

코드생성이 완료되면 [Keep]버튼(Ctrl-Shift+Y)을 눌러 생성된 코드를 적용한다



1.1 Cmd/Ctrl+K

우측 상단의 Run 버튼을 누르면
하단에 결과가 출력된다



The screenshot shows a code editor window with a file named `inline_demo.py`. The code in the editor is as follows:

```
1 import datetime
2
3 now = datetime.datetime.now()
4 print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"))
5
6
```

In the top right corner of the editor, a red box highlights the Run button (a play icon).

Below the editor is a terminal window. The terminal shows the command used to run the script:

```
PS C:\Users\storm\바탕 화면\My_Cursor_Lab\01_Chat_demo> & C:/Users/storm/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/storm/바탕 화면/My_Cursor_Lab/01_Chat_demo/inline_demo.py"
```

The output of the script is displayed in the terminal, with a red box highlighting the result:

```
● 현재 시간: 2025-09-01 10:48:01
```

Below the output, the prompt for the next command is visible:

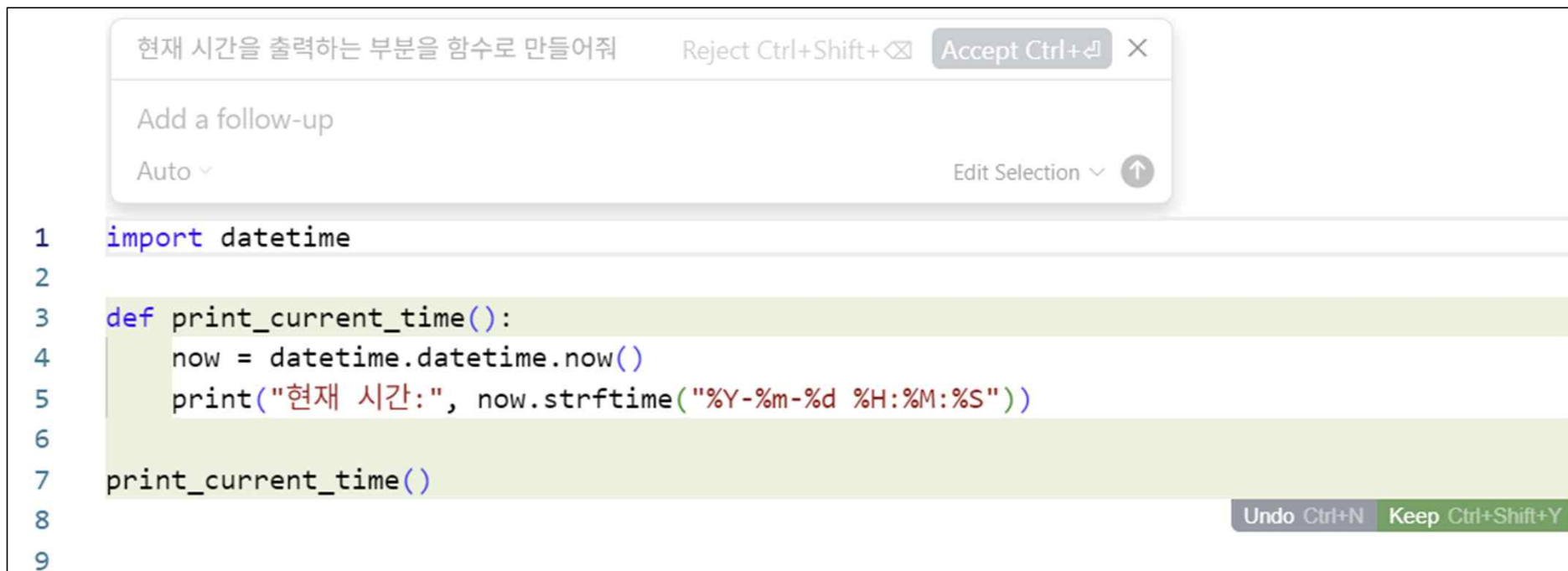
```
○ PS C:\Users\storm\바탕 화면\My_Cursor_Lab\01_Chat_demo>
```

1.1 Cmd/Ctrl+K

소스 코드 수정하기(Refactoring)

소스 코드 전체를 마우스로 선택하고 Ctrl+K를 누르고 아래 내용을 입력한다

“현재 시간을 출력하는 부분을 함수로 만들어줘”



```
1 import datetime
2
3 def print_current_time():
4     now = datetime.datetime.now()
5     print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"))
6
7 print_current_time()
8
9
```

현재 시간을 출력하는 부분을 함수로 만들어줘 Reject Ctrl+Shift+⌘ Accept Ctrl+⌘ X

Add a follow-up

Auto ▾ Edit Selection ▾ ⬆

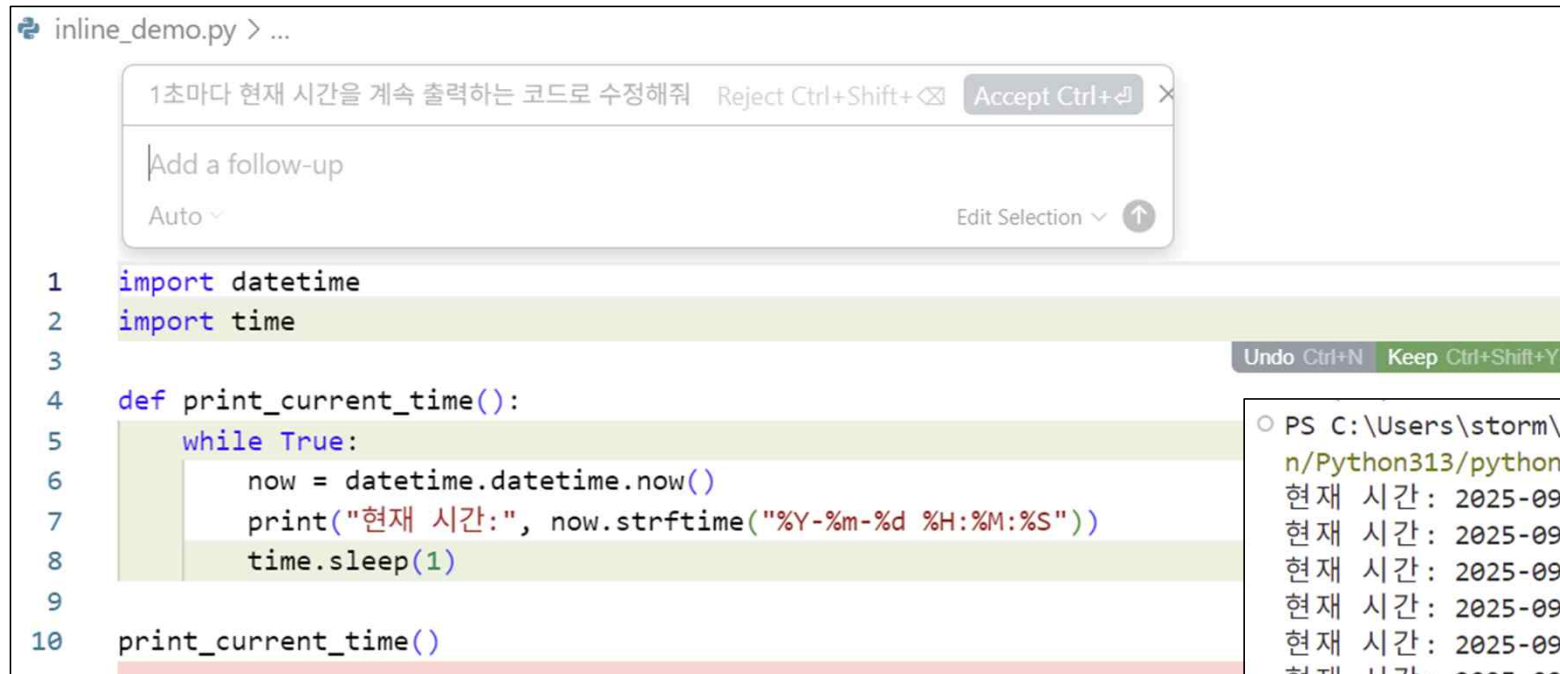
Undo Ctrl+N Keep Ctrl+Shift+Y

1.1 Cmd/Ctrl+K

소스 코드 수정하기(Refactoring)

소스 코드 전체를 마우스로 선택하고 Ctrl+K를 누르고 아래 내용을 입력한다

“1초마다 현재 시간을 계속 출력하는 코드로 수정해줘”



```
inline_demo.py > ...  
1초마다 현재 시간을 계속 출력하는 코드로 수정해줘 Reject Ctrl+Shift+⌘ Accept Ctrl+⌘  
Add a follow-up  
Auto Edit Selection  
1 import datetime  
2 import time  
3  
4 def print_current_time():  
5     while True:  
6         now = datetime.datetime.now()  
7         print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"))  
8         time.sleep(1)  
9  
10 print_current_time()
```

코드 실행 화면

```
PS C:\Users\storm\바탕 화면\My_Cursor_Lab\01_Chat  
n/Python313/python.exe "c:/Users/storm/바탕 화면/  
현재 시간: 2025-09-01 11:04:49  
현재 시간: 2025-09-01 11:04:50  
현재 시간: 2025-09-01 11:04:51  
현재 시간: 2025-09-01 11:04:52  
현재 시간: 2025-09-01 11:04:53  
현재 시간: 2025-09-01 11:04:54  
현재 시간: 2025-09-01 11:04:55  
현재 시간: 2025-09-01 11:04:56
```

1.1 Cmd/Ctrl+K

소스 코드 수정하기(Refactoring)

소스 코드 전체를 마우스로 선택하고 Ctrl+K를 누르고 아래 내용을 입력한다

“출력 결과에 한국어 요일도 함께 표시해줘”

출력 결과에 한국어 요일도 함께 표시해줘

Reject Ctrl+Shift+⌘ Accept Ctrl+⌘ X

Add a follow-up

Auto

Edit Selection

```
1 import datetime
2 import time
3
4 def print_current_time():
5     weekdays_kr = ['월', '화', '수', '목', '금', '토', '일']
6     while True:
7         now = datetime.datetime.now()
8         print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"))
9         weekday_kr = weekdays_kr[now.weekday()]
10        print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"), f"({weekday_kr}요일)")
11        time.sleep(1)
12 print_current_time()
```

코드 실행 화면

현재 시간:	2025-09-01 11:09:08	(월요일)
현재 시간:	2025-09-01 11:09:09	(월요일)
현재 시간:	2025-09-01 11:09:10	(월요일)
현재 시간:	2025-09-01 11:09:11	(월요일)
현재 시간:	2025-09-01 11:09:12	(월요일)
현재 시간:	2025-09-01 11:09:13	(월요일)

1.1 Cmd/Ctrl+K

소스 코드 수정하기(Refactoring)

소스 코드 전체를 마우스로 선택하고 Ctrl+K를 누르고 아래 내용을 입력한다

“이 코드를 GUI 버전으로 만들어줘.”

```
이 코드를 GUI 버전으로 만들어줘. Reject Ctrl+Shift+⌘ Accept Ctrl+⌘ X
Add a follow-up
1 import datetime
import time
2 import tkinter as tk
3
4 def print_current_time():
5 def update_time():
now = datetime.datetime.now()
6 weekdays_kr = ['월', '화', '수', '목', '금', '토', '일']
while True:
now = datetime.datetime.now()
weekday_kr = weekdays_kr[now.weekday()]
print("현재 시간:", now.strftime("%Y-%m-%d %H:%M:%S"), f"({weekday_kr}요일)")
time.sleep(1)
7 weekday_kr = weekdays_kr[now.weekday()]
8 time_str = now.strftime("%Y-%m-%d %H:%M:%S")
9 label.config(text=f"현재 시간: {time_str} ({weekday_kr}요일)")
10 root.after(1000, update_time)
11
12 root = tk.Tk()
13 root.title("현재 시간 표시")
14
15 label = tk.Label(root, font=("Arial", 20))
16 label.pack(padx=20, pady=20)
17
18 print_current_time()
19 update_time()
20 root.mainloop()
```

GUI 버전 실행 화면



실습 예제 시나리오

- 파일명: guess_num.py

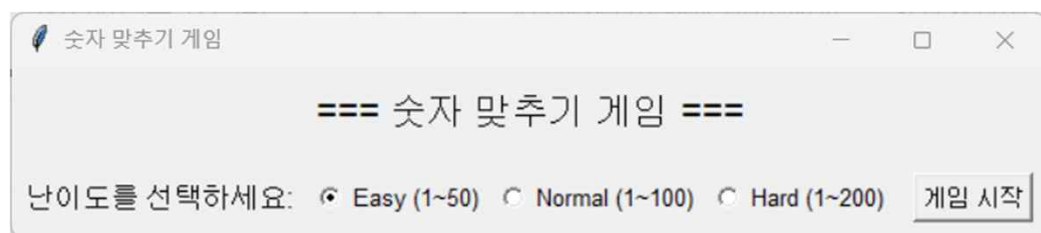
1. 숫자 맞추기 텍스트 게임을 파이썬으로 만들어줘
2. 정답을 맞출 때까지 계속 입력 받는 코드로 수정해줘
3. 몇 번 만에 맞췄는지 시도 횟수를 출력하도록 바꿔줘
4. 이 코드를 tkinter GUI 게임으로 바꿔줘
5. 숫자를 직접 입력하는 대신 버튼을 눌러서 선택하도록 바꿔줘
6. 맞추면 점수를 올리고, 틀리면 점수를 깎는 점수 시스템을 추가해줘
7. 게임 시작 전에 Easy, Normal, Hard 난이도를 선택할 수 있도록 수정해줘

- 만일 오류 발생시 수정 질의 예시

소스코드 오류야. 계속 “더 큰 숫자야”만 나와. 오류 수정해줘.

1.1 Cmd/Ctrl+K

코드 실행 화면



1.1 Cmd/Ctrl+K

Edit Selection 모드

선택한 소스만 수정할 때 사용

짧은 블록만 변경하고 싶을 때 사용

Edit Full File 모드

파일 전체를 수정할 때 사용

전체구조를 변경하고 싶을 때 사용

Quick Question 모드

선택한 소스 코드에 질문을 할 때 사용. “**선택한 코드 설명해줘.**” 와 같이 요청하거나

“**이 코드를 어떻게 개선하면 좋을까?**” 와 같이 제안 요청할 때 사용.

제안이 생성되면 “**적용해줘.**”와 같이 입력하면 제안 대로 수정 가능.



Inline Edit 모드 선택 메뉴

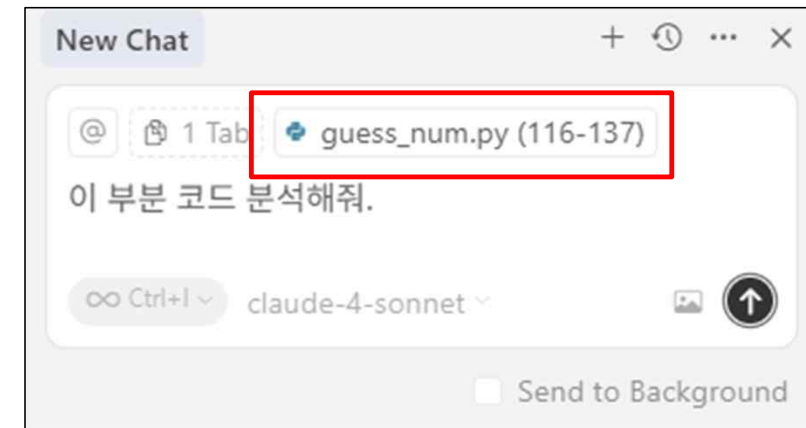
1.1 Cmd/Ctrl+K

Send to Chat

멀티 파일 편집이나 고급 기능이 필요하면 Ctrl+L로 선택한 코드를 Chat으로 보낸다.
멀티 파일 편집, 상세한 설명, 고급 AI 기능을 제공한다.

소스 코드를 선택한 상태에서 바로 Ctrl+L을 눌러 Chat 창으로 보낼 수 있다.

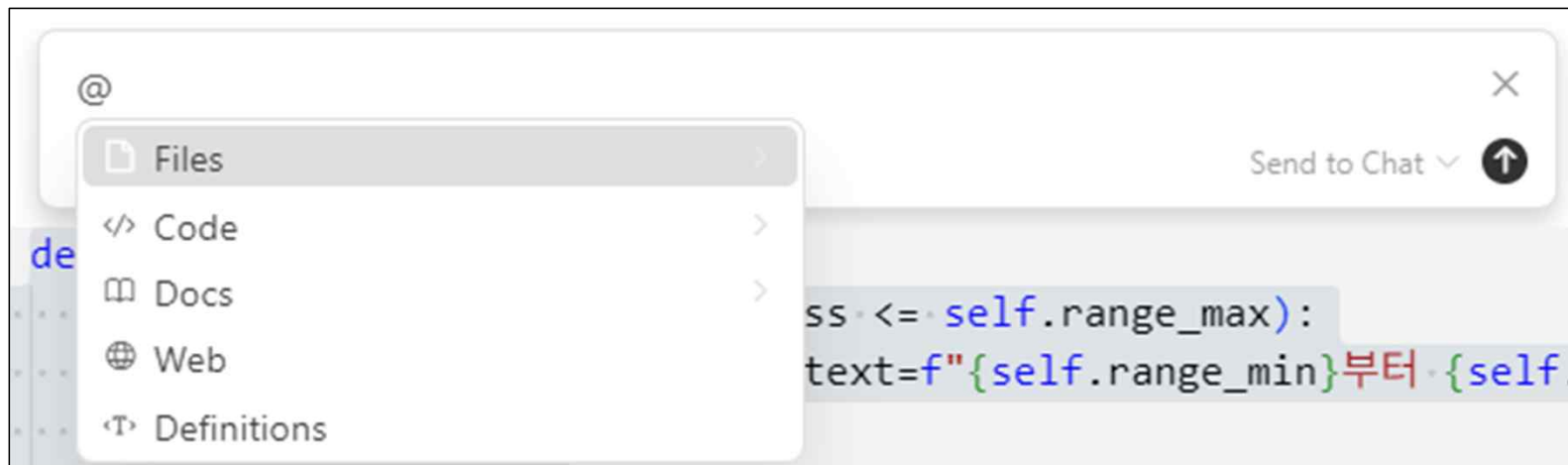
```
116 def check_guess(self, guess):
117     if not (self.range_min <= guess <= self.range_max):
118         self.label_result.config(text=f"{self.range_min}부터 {self.range_max} 사이의 숫자를 선택하세요.", fg="red")
119     return
120     self.attempts += 1
121     self.label_attempts.config(text=f"시도 횟수: {self.attempts}")
122     if guess > self.answer:
123         self.label_result.config(text="더 작은 숫자입니다.", fg="blue")
124         self.score -= 1 # 오답 시 점수 감소
125     elif guess < self.answer:
126         self.label_result.config(text="더 큰 숫자입니다.", fg="blue")
127         self.score -= 1 # 오답 시 점수 감소
128     else:
129         self.label_result.config(
130             text=f"축하합니다! {self.attempts}번 만에 정답을 맞추셨습니다.", fg="green"
131         )
132         self.score += 5 # 정답 시 점수 증가
133         # 모든 버튼 비활성화
134         for btn in self.number_buttons:
135             btn.config(state="disabled")
136         self.button_restart.config(state="normal")
137         self.label_score.config(text=f"점수: {self.score}")
```



1.1 Cmd/Ctrl+K

@ 기호

AI Pane의 Chat과 마찬가지로 @기호를 사용하여 **심볼 참조 기능**을 사용할 수 있다.
다음 단원의 Chat 기능에 자세히 설명되어 있음.



1.1 Cmd/Ctrl+K

Inline Edit 질의의 히스토리는 볼수 없나?

Cursor의 Inline Edit (Ctrl+K / Cmd+K) 기능은

👉 “특정 코드 블록을 선택 → 프롬프트 입력 → 수정된 결과를 바로 코드에 반영” 하는 구조입니다. 그래서 일반 대화(Chat) 모드처럼 질의 히스토리를 목록으로 남겨두지는 않는다.

- 볼 수 있는 것:
 - 변경된 코드 (diff/undo/redo 이력)
 - Git을 쓰고 있다면 커밋 이력
- 볼 수 없는 것:
 - Inline Edit에서 입력했던 프롬프트 텍스트 자체
 - Inline Edit의 프롬프트 히스토리 목록

✅ 만약 프롬프트 히스토리까지 남기고 싶다면

- 직접 Chat 탭에서 질의 → 코드 수정 후 붙여넣기
- 또는 Git commit 메시지에 “어떤 프롬프트로 수정했는지” 메모

이런 식으로 관리해야 합니다.

1.2 터미널 Cmd/Ctrl+K



1.2 터미널 Cmd/Ctrl+K

Terminal

Cursor 터미널에서 Ctrl+K를 눌러 아래쪽에 프롬프트 바를 열어 원하는 작업을 설명하면 실행가능한 명령을 만들어 준다.(소스 코드가 아닌 터미널 명령을 생성)

파이썬 소스 파일들만 목록을 보여줘!

Quick Question

Problems Output Debug Console **Terminal** Ports

PS C:\Users\storm\바탕 화면\My_Cursor_Lab\01_Chat_demo> ls *.py

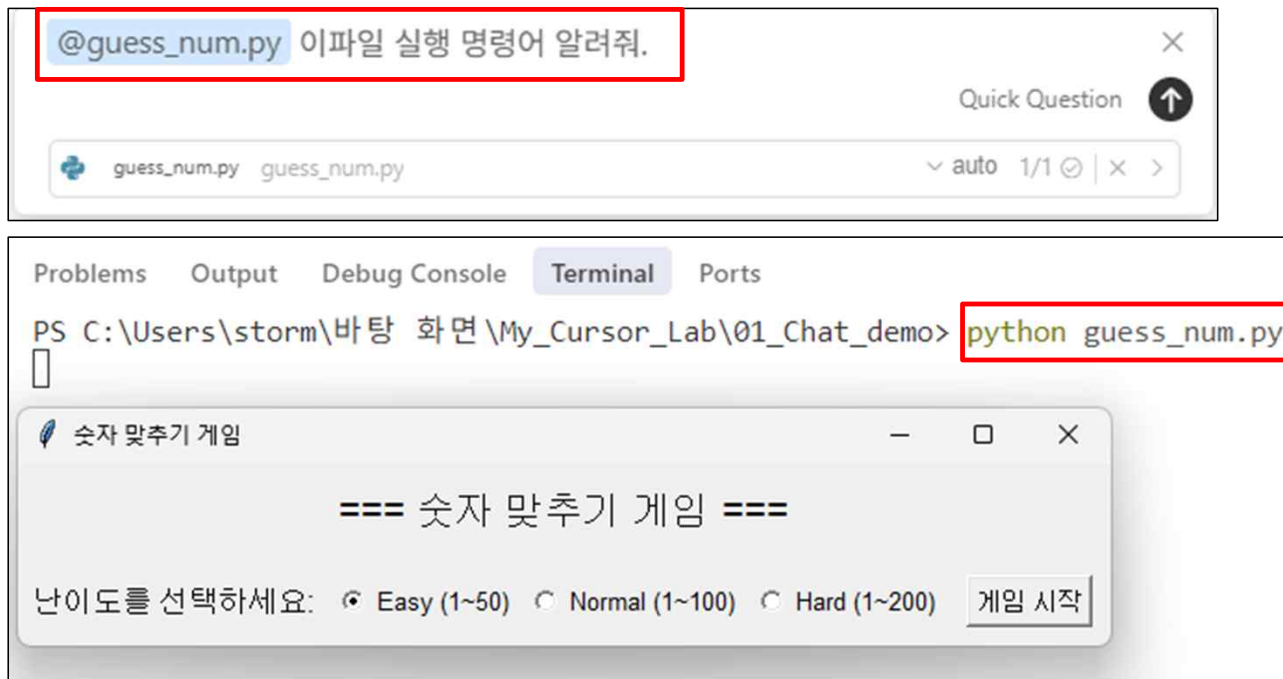
디렉터리 : C:\Users\storm\바탕 화면\My_Cursor_Lab\01_Chat_demo

Mode	LastWriteTime	Length	Name
-a----	2025-08-31 오전 12:59	684	Chat_example.py
-a----	2025-08-30 오후 12:16	3046	Flask_demo.py
-a----	2025-09-01 오후 1:10	6875	guess_num.py
-a----	2025-09-01 오전 11:14	533	inline_demo.py

1.2 터미널 Cmd/Ctrl+K

심볼 참조 질의하기

@을 입력 -> File 메뉴 선택 후 원하는 파이썬 파일을 선택하고 “이 파일 실행 명령어 알려줘.” 를 입력한다. 명령어가 생성되면 터미널에서 Enter를 쳐서 실행 시킨다



1.3 Tab



1.3 Tab

Tab

Cursor의 Tab은 코드 편집 시 인라인으로 코드를 자동 완성하는 기능이다

제안된 내용을 **[Tab 키]**로 수락하고 **[Esc 키]**로 거절한다

Tab로 할 수 있는 것들 :

여러 줄을 한 번에 수정하기

누락된 import 문 자동 추가

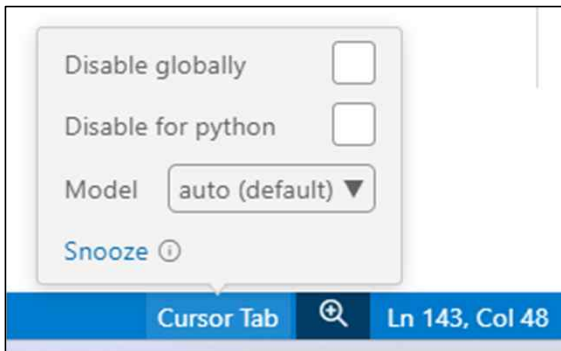
파일 내·파일 간 점프하며 연계 편집

최근 변경, 린터 오류(문법 오류), 누락된 편집을 기반으로 제안 받기

1.3 Tab

Tab

Cursor에서 소스 파일을 열어 놓고 화면 아래 상태바의 [Cursor Tab]에 마우스를 위치시켜 놓으면 설정 메뉴를 볼 수 있다



Disable globally : 모든 파일에서 Tab 끄기

Disable for {extensions} : 특정 언어 소스 코드 파일

(예: python 또는 javascript)에서 Tab 끄기

Snooze : 지정한 기간 동안 Tab을 일시적으로 끄기.

(1분 ~ 3시간까지 메뉴 선택)

Model : Cursor Tab에 사용할 AI 모델을 설정

auto (default) 는 자동으로 적절한 모델을 선택해주고

fast는 가장 빠른 모델을 선택하여 사용하게 해준다

Tab 실습 예제

[1] 함수를 만들면 함수명과 인자를 보고 다음 코드를 미리 보여준다

`def multiply(a,b):` 만 입력하고 하면 엔터를 치면 자동으로 다음 줄을 제안해준다
Tab 키로 제안을 수락하고 Esc키로 거절 할 수 있다

```
def multiply(a,b):  
    return a*b
```

```
def add(a,b):  
    return a+b
```



```
def add(a,b):  
    return a+b  
  
def subtract(a,b):  
    return a-b  
  
def divide(a,b):  
    return a/b
```

```
def greet(name):  
    ?
```

add 함수 까지만 수락했는데
나머지 관련 함수도 자동으로
제안해준다

[2] 주석문

코드 상단에 # 만 입력해도 주석문을 제안해준다

#계산기 함수 정의

[3] import 제안

time.sleep(1) 만 입력하고 소스 맨 위에 import 해야 할 모듈이 자동으로 제안 된다
Tab 키로 수락한다

```
Tab_demo.py > ...  
1  #계산기 함수 정의  
2  
3  import time CR
```

[4] 반복문

반복문을 자동으로 완성해준다

```
for i in range(5):
```



```
for i in range(5):  
    print(add(i,i+1))
```

[5] 파일 읽기

아래 주석문만 작성하면 다음 라인을 차례로 제안해 준다

example.txt 파일을 열어서 한 줄씩 출력

```
with open("example.txt", "r") as f:  
    for line in f:  
        print(line)
```

1.3 Tab

[6] 간단한 게임

아래 두 줄만 작성하고 Tab 키를 계속 치면 완성된다

랜덤 숫자 맞추기 게임

import random

```
# 랜덤 숫자 맞추기 게임
import random
random_number = random.randint(1,100)

while True:
    guess = int(input("1부터 100 사이의 숫자를 맞춰보세요: "))
    if guess == random_number:
        print("정답입니다!")
        break
    elif guess < random_number:
        print("더 큰 숫자입니다.")
    else:
        print("더 작은 숫자입니다.")
```

Thank You!!