

M01-04 · GIBALJA VISUAL MANUAL

오류 메시지를 읽고 질문하기

한 줄 목표: 오류 type·message·file·line·traceback·재현 절차를 읽고 가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.

1. 왜 지금 오류 메시지를 읽고 질문하기인가

M01-04 · 01

배우는 이유: 오류 메시지를 읽고 질문하기

- 1 오류 화면을 닫고 기억으로 설명함
- 2 전체 traceback을 읽지 않음
- 3 여러 설정을 동시에 바꿈
- 4 검색한 명령을 바로 실행함
- 5 수정 후 실패 case만 확인함

그림 1. 배우는 이유를 하나의 핵심 메시지로 정리합니다.

오류를 숨기거나 마지막 문장만 검색하면 원인과 조건을 잃습니다. 오류는 실패 통보가 아니라 어디서 무엇을 확인할지 알려 주는 구조화된 evidence입니다.

지금 상태	문제	학습 뒤 변화
아이디어·도구 중심	오류 화면을 닫고 기억으로 설명함	오류 해결 기록표
느낌으로 완료	재현 evidence 없음	다른 사람이 확인 가능한 기준
AI 결과 의존	사람 판단 경계 없음	원문·실행·승인 분리

2. 학습 전 확인과 완료 계약

M01-04 · 02

학습 계약: 오류 메시지를 읽고 질문하기

- 1
오늘 목표 · 오류 메시지를 읽고 질문하기
- 2
산출물 · 오류 해결 기록표
- 3
완료 · 실습 90분 + 셀프 테스트 + 자기 설명

그림 2. 학습 계약을 하나의 핵심 메시지로 정리합니다.

항목	계약
대상	IT 비전공 성인 학습자
선수지식	M01-03 개발 도구와 실행 환경 점검하기
예상시간	그림 학습 30분 + 실습 45분 + 복습 15분
산출물	오류 해결 기록표
완료	실습 영수증 + 셀프 테스트 + 자기 설명

실습에는 실제 개인정보·비밀번호·API key를 넣지 않습니다.

3. 전체 구조 한눈에 보기

M01-04 · 03

전체 개념 지도: 오류 메시지를 읽고 질문하기

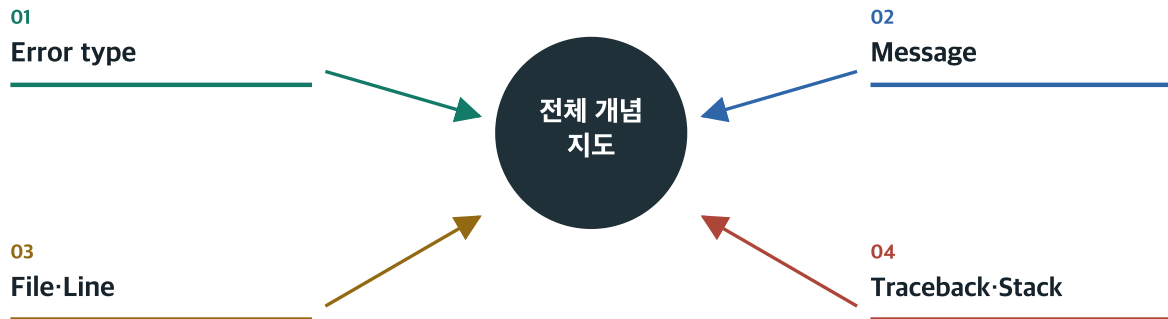


그림 3. 전체 개념 지도를 하나의 핵심 메시지로 정리합니다.

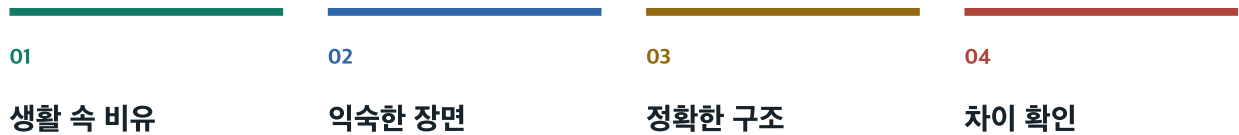
그림의 화살표를 먼저 따라가고, 각 단계에서 어떤 evidence가 남는지 찾습니다.

핵심 요소	학습 질문
Error type	어떤 오류 type인가
Message	내 코드의 첫 frame은 어디인가
File·Line	입력·환경·상태 중 무엇이 달랐나
Traceback·Stack	가장 작은 검증 실험은 무엇인가
Reproduction	수정이 다른 흐름을 깨뜨리지 않았나
Hypothesis·Regression	어떤 오류 type인가

4. 실생활 비유에서 정확한 구조로

M01-04 · 04

실생활 비유: 오류 메시지를 읽고 질문하기



자기 말로 설명

그림 4. 실생활 비유를 하나의 핵심 메시지로 정리합니다.

오류 message는 병원 진료 기록과 비슷합니다. 증상 한 줄만이 아니라 발생 시간·환경·이전 행동·검사 결과·변화를 함께 봐야 원인 후보를 좁힐 수 있습니다. 비유는 시작점일 뿐입니다. 정확한 구조는 역할·입력·처리·상태·실패·책임으로 다시 분리합니다.

5. 핵심 요소 여섯 가지

M01-04 · 05

핵심 요소: 오류 메시지를 읽고 질문하기

DO

① Error type

② Message

CHECK

③ File·Line

④ Traceback·Stack

⑤ Reproduction



그림 5. 핵심 요소를 하나의 핵심 메시지로 정리합니다.

#	핵심 요소	확인 행동
1	Error type	합성 오류를 실행
2	Message	type·message·file·line 표시
3	File·Line	최소 재현 작성
4	Traceback·Stack	가설 두 개 비교
5	Reproduction	수정·정상·회귀 evidence 기록
6	Hypothesis·Regression	합성 오류를 실행

6. 입력에서 산출물까지

M01-04 · 06

입력과 출력: 오류 메시지를 읽고 질문하기

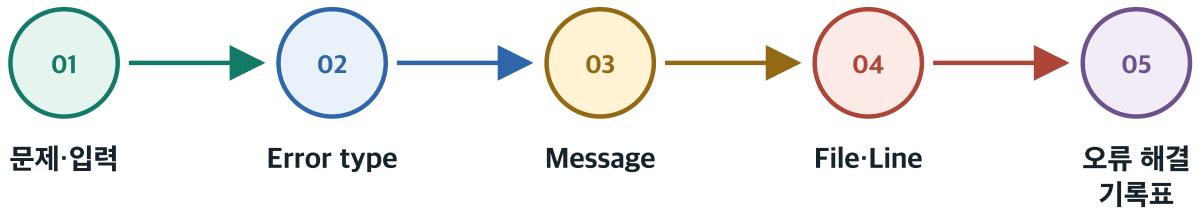


그림 6. 입력과 출력을 하나의 핵심 메시지로 정리합니다.

구간	입력	처리	출력
시작	사용자·문제·현재 상태	원문 오류를 그대로 보존한다	원문 error
중간	가정·범위·도구	최소 재현 절차를 만든다	재현 step
완료	검증 결과·한계	수정 뒤 정상·회귀를 검증한다	오류 해결 기록표

30초 확인: 어떤 오류 type인가? 답을 말한 뒤 다음 장으로 이동하세요.

7. 누가 무엇을 책임하는가

M01-04 · 07

사람과 역할: 오류 메시지를 읽고 질문하기

1 학습자

2 사용자

3 기획 책임

4 개발·AI

5 운영·검수

그림 7. 사람과 역할을 하나의 핵심 메시지로 정리합니다.

역할	책임	하면 안 되는 일
학습자	관찰·실행·기록	모르는 결과를 성공으로 표시
기발자	문제·범위·완료 기준	기술·법률 승인을 대신함
개발자·AI	구현·설명·검사 보조	최종 결정 자동 확정
Owner	승인·위험·운영 책임	evidence 없는 승인
사용자	실제 과업과 피드백	합성 persona로 대체

8. 헛갈리는 경계 분리하기

M01-04 · 08

경계와 책임: 오류 메시지를 읽고 질문하기

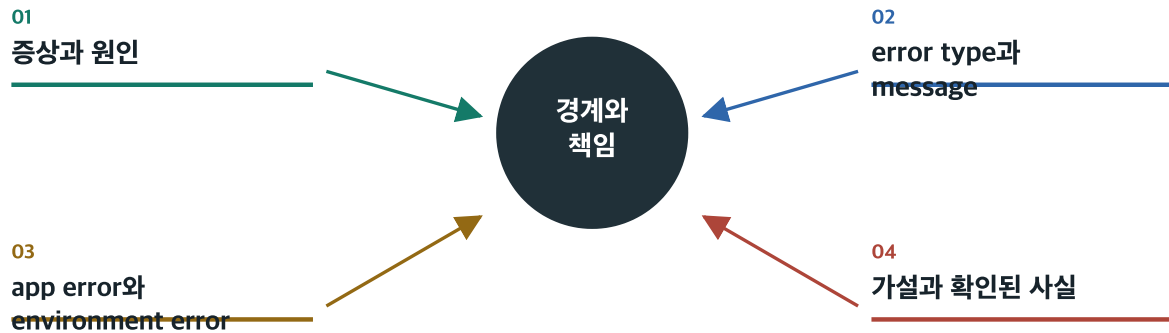


그림 8. 경계와 책임을 하나의 핵심 메시지로 정리합니다.

구분할 경계	왜 분리하나
증상과 원인	같이 보이지만 책임·검증·실패 영향이 다르기 때문
error type과 message	같이 보이지만 책임·검증·실패 영향이 다르기 때문
app error와 environment error	같이 보이지만 책임·검증·실패 영향이 다르기 때문
가설과 확인된 사실	같이 보이지만 책임·검증·실패 영향이 다르기 때문
임시 우회와 근본 수정	같이 보이지만 책임·검증·실패 영향이 다르기 때문

9. 정상 workflow 다섯 단계

M01-04 · 09

정상 workflow: 오류 메시지를 읽고 질문하기

- 1 원문 오류를 그대로 보존한다
- 2 마지막 줄과 첫 관련 frame을 읽는다
- 3 최소 재현 절차를 만든다
- 4 가설 하나씩 시험한다
- 5 수정 뒤 정상·회귀를 검증한다

그림 9. 정상 workflow을 하나의 핵심 메시지로 정리합니다.

순서	행동	남길 evidence
1	원문 오류를 그대로 보존한다	원문 error
2	마지막 줄과 첫 관련 frame을 읽는다	환경·입력
3	최소 재현 절차를 만든다	재현 step
4	가설 하나씩 시험한다	가설·실험
5	수정 뒤 정상·회귀를 검증한다	fix·regression 결과

10. 개발자가 결정할 다섯 질문

M01-04 · 10

판단 기준: 오류 메시지를 읽고 질문하기

DO

- 어떤 오류 type인가
- 내 코드의 첫 frame은 어디인가



CHECK

- 입력·환경·상태 중 무엇이 달랐나
- 가장 작은 검증 실험은 무엇인가
- 수정이 다른 흐름을 깨뜨리지 않았나

그림 10. 판단 기준을 하나의 핵심 메시지로 정리합니다.

질문	선택 evidence
어떤 오류 type인가	원문 error
내 코드의 첫 frame은 어디인가	환경·입력
입력·환경·상태 중 무엇이 달랐나	재현 step
가장 작은 검증 실험은 무엇인가	가설·실험
수정이 다른 흐름을 깨뜨리지 않았나	fix·regression 결과

11. 좋은 예: 작은 evidence가 이어지는 경우

M01-04 · 11

좋은 예: 오류 메시지를 읽고 질문하기

- 1 합성 오류를 실행
- 2 type·message·file·line 표시
- 3 최소 재현 작성
- 4 가설 두 개 비교
- 5 수정·정상·회귀 evidence 기록

그림 11. 좋은 예를 하나의 핵심 메시지로 정리합니다.

좋은 예는 완벽한 문서가 아니라 질문·행동·결과·한계가 이어지는 작은 기록입니다.

행동	좋은 기록
합성 오류를 실행	원문 error
type·message·file·line 표시	환경·입력
최소 재현 작성	재현 step
가설 두 개 비교	가설·실험
수정·정상·회귀 evidence 기록	fix·regression 결과

12. 나쁜 예: 그럴듯하지만 재현되지 않는 경우

M01-04 · 12

나쁜 예: 오류 메시지를 읽고 질문하기

- 1 오류 화면을 닫고 기억으로 설명함
- 2 전체 traceback을 읽지 않음
- 3 여러 설정을 동시에 바꿈
- 4 검색한 명령을 바로 실행함
- 5 수정 후 실패 case만 확인함

그림 12. 나쁜 예들 하나의 핵심 메시지로 정리합니다.

나쁜 예	왜 위험한가	바꿀 행동
오류 화면을 닫고 기억으로 설명함	원인·범위·책임을 잃음	합성 오류를 실행
전체 traceback을 읽지 않음	원인·범위·책임을 잃음	type·message·file·line 표시
여러 설정을 동시에 바꿈	원인·범위·책임을 잃음	최소 재현 작성
검색한 명령을 바로 실행함	원인·범위·책임을 잃음	가설 두 개 비교
수정 후 실패 case만 확인함	원인·범위·책임을 잃음	수정·정상·회귀 evidence 기록

30초 확인: 내 코드의 첫 frame은 어디인가? 답을 말한 뒤 다음 장으로 이동하세요.

13. 오류·오해·실패 위치 지도

M01-04 · 13

실패 위치 지도: 오류 메시지를 읽고 질문하기

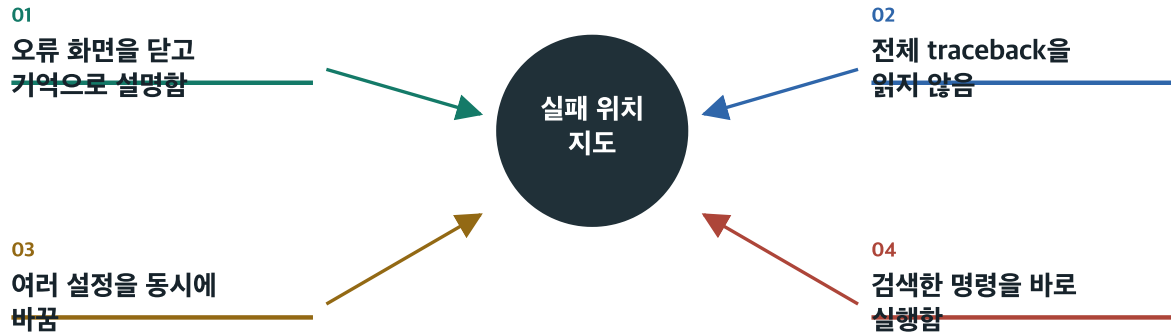


그림 13. 실패 위치 지도를 하나의 핵심 메시지로 정리합니다.

실패 신호	먼저 확인	복구
오류 화면을 닫고 기억으로 설명함	증상과 원인	합성 오류를 실행
전체 traceback을 읽지 않음	error type과 message	type·message·file·line 표시
여러 설정을 동시에 바꿈	app error와 environment error	최소 재현 작성
검색한 명령을 바로 실행함	가설과 확인된 사실	가설 두 개 비교
수정 후 실패 case만 확인함	임시 우회와 근본 수정	수정·정상·회귀 evidence 기록

14. 보안·개인정보·변경 안전

M01-04 · 14

안전·개인정보: 오류 메시지를 읽고 질문하기

01

최소 권한

02

실제 정보 금지

03

변경 전 확인

04

로그·
evidence

복구 가능성

그림 14. 안전·개인정보를 하나의 핵심 메시지로 정리합니다.

안전 질문	최소 통제
실제 정보인가	합성 값·redaction
변경 command인가	목적·대상·backup
권한이 필요한가	최소 권한·사람 승인
외부로 전송되는가	destination·보존·비용 확인
실패 뒤 돌아갈 수 있는가	diff·history·restore

15. 도구와 기록에서 무엇을 볼 것인가

M01-04 · 15

도구에서 찾기: 오류 메시지를 읽고 질문하기

DO

- 1 `python3 broken_example.py`
- 2 `echo $?`



CHECK

- 3 `git diff -- broken_example.py`
- 4 `python3 -m unittest -v`

그림 15. 도구에서 찾기를 하나의 핵심 메시지로 정리합니다.

도구 화면에서는 큰 성공 문구보다 현재 위치·대상·version·output·exit 상태를 먼저 찾습니다.

```
python3 broken_example.py
echo $?
git diff -- broken_example.py
python3 -m unittest -v
```

16. 실습 1: 관찰하고 표시하기

M01-04 · 16

실습 1 · 관찰: 오류 메시지를 읽고 질문하기

- 1 합성 오류를 실행
- 2 type·message·file·line 표시
- 3 최소 재현 작성
- 4 가설 두 개 비교
- 5 수정·정상·회귀 evidence 기록

그림 16. 실습 1 · 관찰을 하나의 핵심 메시지로 정리합니다.

실습 순서	행동	완료 표시
1	합성 오류를 실행	☐ 관찰 · ☐ 기록 · ☐ 확인
2	type·message·file·line 표시	☐ 관찰 · ☐ 기록 · ☐ 확인
3	최소 재현 작성	☐ 관찰 · ☐ 기록 · ☐ 확인
4	가설 두 개 비교	☐ 관찰 · ☐ 기록 · ☐ 확인
5	수정·정상·회귀 evidence 기록	☐ 관찰 · ☐ 기록 · ☐ 확인

17. 실습 2: 내 사례 작성하기

M01-04 · 17

실습 2 · 작성: 오류 메시지를 읽고 질문하기

- 1 합성 오류를 실행
- 2 `type·message·file·line` 표시
- 3 최소 재현 작성
- 4 가설 두 개 비교
- 5 수정·정상·회귀 evidence 기록

그림 17. 실습 2 · 작성을 하나의 핵심 메시지로 정리합니다.

`02_Labs/G01_Development_Environment/L01-04_error-triage-desk.html` 을 열고 02 수행 화면에서 내 사례를 작성합니다. 정답처럼 보이는 문장보다 선택 이유와 남은 질문을 적습니다.

18. Evidence package 만들기

M01-04 · 18

Evidence 남기기: 오류 메시지를 읽고 질문하기



그림 18. Evidence 남기기를 하나의 핵심 메시지로 정리합니다.

Evidence	필수 내용	검수 질문
원문 error	합성 오류를 실행	다른 사람이 같은 판단을 재현하는가
환경·입력	type·message·file·line 표시	다른 사람이 같은 판단을 재현하는가
재현 step	최소 재현 작성	다른 사람이 같은 판단을 재현하는가
가설·실험	가설 두 개 비교	다른 사람이 같은 판단을 재현하는가
fix·regression 결과	수정·정상·회귀 evidence 기록	다른 사람이 같은 판단을 재현하는가

30초 확인: 입력·환경·상태 중 무엇이 달랐나? 답을 말한 뒤 다음 장으로 이동하세요.

19. 개발자·외주사에게 확인할 질문

M01-04 · 19

개발자에게 물을 질문: 오류 메시지를 읽고 질문하기

- 1 오류가 처음 발생한 정확한 조건은 무엇인가
- 2 전체 traceback과 log가 있는가
- 3 최근 변경은 무엇인가
- 4 같은 입력에서 반복되는가
- 5 수정 뒤 어떤 test를 통과했는가

그림 19. 개발자에게 물을 질문을 하나의 핵심 메시지로 정리합니다.

확인 질문	좋은 답의 증거
오류가 처음 발생한 정확한 조건은 무엇인가	원문 error
전체 traceback과 log가 있는가	환경·입력
최근 변경은 무엇인가	재현 step
같은 입력에서 반복되는가	가설·실험
수정 뒤 어떤 test를 통과했는가	fix·regression 결과

20. AI에 안전하게 작업 요청하기

M01-04 · 20

AI에 작업 요청하기: 오류 메시지를 읽고 질문하기

DO

1 목적

2 맥락

CHECK

3 입력

4 완료 기준

5 금지·한계



그림 20. AI에 작업 요청하기를 하나의 핵심 메시지로 정리합니다.

AI 요청은 다음 구조로 씁니다.

목적: 오류 해결 기록표 초안을 만든다.

맥락: 오류 `type·message·file·line·traceback·재현` 절차를 읽고 가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.

입력: 아래 합성 사례와 확인된 사실만 사용한다.

완료 기준: 표의 모든 `field`, 정상·실패·한계, TBR를 포함한다.

금지: 실제 정보 추정, 확인하지 않은 성공 단정, 위험한 `command` 자동 실행.

출력 뒤: 누락·가정·검증 방법을 별도 목록으로 적는다.

21. AI 결과를 사람이 검수하기

M01-04 · 21

AI 결과 검수: 오류 메시지를 읽고 질문하기



그림 21. AI 결과 검수를 하나의 핵심 메시지로 정리합니다.

검수	질문	실패 시
원문	입력에 실제 존재하는가	삭제·수정
범위	이번 manual의 경계 안인가	TBR로 이동
정상	expected를 재현했는가	실행 evidence
실패	거부·오류·복구가 있는가	case 추가
승인	사람 owner가 이유를 남겼는가	확정 금지

22. 실습 화면에서 전체 지도 찾기

YEONCORE · GIBALJA LAB

오류 읽기·질문 데스크

M01-04 · v0.1.0

합성 사례 · 로컬 저장 · 외부 전송 없음

01 이해

02 수행

03 복습

전체 구조 이해

학습용 · SELF REVIEW

오류 type·message·file·line·traceback·재현 절차를 읽고 가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.

CORE 01

Error type

증상과 원인을 함께 확인합니다.

CORE 02

Message

error type과 message를 함께 확인합니다.

CORE 03

File·Line

app error와 environment error를 함께 확인합니다.

CORE 04

Traceback·Stack

가설과 확인된 사실을 함께 확인합니다.

CORE 05

Reproduction

임시 우회와 근본 수정을 함께 확인합니다.

CORE 06

Hypothesis·Regression

증상과 원인을 함께 확인합니다.

경계: 이 화면의 완료는 학습 산출물 작성 준비이며 실제 서비스·보안·운영 승인이 아닙니다.

실습 화면 1. 핵심 요소와 경계를 desktop에서 찾습니다.

23. 실습 화면에서 단계별 수행하기

YEONCORE · GIBALJA LAB

오류 읽기·질문 데스크

M01-04 · v0.1.0

합성 사례 · 로컬 저장 · 외부 전송 없음

01 이해

02 수행

03 복습

단계별 수행

학습용 · SELF REVIEW

오류 type·message·file·line·traceback·재현 절차를 읽고 가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.

1

합성 오류를 실행

2

type·message·file·line 표시

3

최소 재현 작성

4

가설 두 개 비교

5

수정·정상·회귀 evidence 기록

내 사례와 evidence

관찰한 사실, 선택 이유, 남은 질문을 적으세요.

학습 영수증 만들기

초기화

실습 화면 2. 다섯 단계를 체크하고 내 사례 evidence를 작성합니다.

24. 모바일에서 복습 영수증 읽기

YEONCORE · GIBALJA LAB

오류 읽기·질문 데스크

M01-04 · v0.1.0

합성 사례 · 로컬 저장 · 외부 전송 없음

01 이해02 수행03 복습

학습 영수증·복습

오류 type-message-file-line-traceback·재현 절차를 읽고
가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.

학습용 · SELF REVIEW

오류 해결 기록표 · 학습 영수증

Manual

M01-04 · 오류 메시지를 읽고 질문하기

완료 단계

5 / 5

남긴 기록

오류 원문과 첫 관련 frame을 보존하고 최소 재현·가설·수
정 뒤 회귀 결과를 기록했다.

Evidence

원문 error · 환경·입력 · 재현 step · 가설·실험 ·
fix-regression 결과

다음

M02-01 · 웹 서비스는 어떻게 움직이는가

셀프 질문: 다른 사람이 이 영수증과 오류 해결 기록표만 보
고 같은 판단을 재현할 수 있나요?

실습 화면 3. 390×844에서 완료 단계·기록·다음 매뉴얼을 복습합니다.

30초 확인: 가장 작은 검증 실험은 무엇인가? 답을 말한 뒤 다음 장으로 이동하세요.

25. 90분 실습 워크북

시간	행동	산출물
0~10분	전체 그림·경계 읽기	한 문장 설명
10~25분	핵심 요소·정상 흐름	표시된 concept map
25~45분	실습 화면 01·02	5단계 수행
45~60분	실패·안전·질문	오류·TBR
60~75분	템플릿 완성	오류 해결 기록표
75~90분	셀프 테스트·영수증	오답 복습

26. 오류 해결 기록표 템플릿

재사용 템플릿: [03_Templates/T01-04_error-triage-record.md](#) 빈칸을 한 번에 채우지 말고 관찰한 사실 → 선택 이유 → 미정 사항 순서로 작성합니다.

27. 기초 통합 용어집 300

통합 기초 용어집 [04_Glossary/GLOSSARY_foundation_orientation_environment.md](#) 의 10, 12~15 묶음을 먼저 봅니다. 용어를 외우는 대신 내 실습 화면과 산출물에서 실제 예를 찾습니다.

28. 공식 자료와 적용 경계

아래 링크는 현재 원리와 도구 동작을 확인하는 1차 자료입니다. 링크 사용은 적합성·인증·운영 승인을 뜻하지 않습니다. 확인 기준일은 2026-07-17입니다.

공식 자료	확인할 원리	적용 경계
Python Errors and Exceptions	syntax error·exception·traceback·handling의 정확한 구조	학습용 요약이며 실제 환경·사용자 검증 추가
Python traceback	stack trace를 출력·검색·구조화하는 방법	학습용 요약이며 실제 환경·사용자 검증 추가
Git diff	최근 변경과 오류 발생 전후 차이를 확인하는 방법	학습용 요약이며 실제 환경·사용자 검증 추가
Git status	수정·미추적 파일과 현재 작업 상태 확인	학습용 요약이며 실제 환경·사용자 검증 추가

29. 셀프 테스트 30

1. 이 매뉴얼의 최종 산출물은 무엇인가요?

정답 보기

정답: 오류 해결 기록표입니다.

2. 이 주제를 배우는 가장 직접적인 이유는 무엇인가요?

정답 보기

정답: 오류를 숨기거나 마지막 문장만 검색하면 원인과 조건을 잃습니다. 오류는 실패 통보가 아니라 어디서 무엇을 확인할지 알려 주는 구조화된 evidence입니다.

3. 전체 지도에서 구분할 여섯 핵심 요소는 무엇인가요?

정답 보기

정답: Error type · Message · File·Line · Traceback·Stack · Reproduction · Hypothesis·Regression입니다.

4. 정상 workflow의 첫 행동은 무엇인가요?

정답 보기

정답: 원문 오류를 그대로 보존한다입니다.

5. 정상 workflow의 마지막 행동은 무엇인가요?

정답 보기

정답: 수정 뒤 정상·회귀를 검증한다입니다.

6. 가장 먼저 답할 의사결정 질문은 무엇인가요?

정답 보기

정답: 어떤 오류 type인가입니다.

7. 완료 전에 확인할 마지막 결정은 무엇인가요?

정답 보기

정답: 수정이 다른 흐름을 깨뜨리지 않았나입니다.

8. 대표적인 실패 한 가지는 무엇인가요?

정답 보기

정답: 오류 화면을 닫고 기억으로 설명함입니다.

9. 또 다른 실패 신호는 무엇인가요?

정답 보기

정답: 전체 traceback을 읽지 않음입니다.

10. 왜 정상 경로만 기록하면 부족한가요?

정답 보기

정답: 여러 설정을 동시에 바꿈 같은 실패와 대안을 놓치기 때문입니다.

11. 첫 번째 책임 경계는 무엇인가요?

정답 보기

정답: 증상과 원인을 구분하는 것입니다.

12. 학습용 결과와 실제 승인을 왜 분리하나요?

정답 보기

정답: 학습 evidence가 실제 사용자·데이터·보안·운영 책임까지 자동으로 승인하지 않기 때문입니다.

13. 실습의 첫 단계는 무엇인가요?

정답 보기

정답: 합성 오류를 실행입니다.

14. 실습의 마지막 단계는 무엇인가요?

정답 보기

정답: 수정·정상·회귀 evidence 기록입니다.

15. 첫 번째 evidence는 무엇인가요?

정답 보기

정답: 원문 error입니다.

16. 마지막 evidence는 무엇인가요?

정답 보기

정답: fix-regression 결과입니다.

17. 개발자에게 가장 먼저 물을 질문은 무엇인가요?

정답 보기

정답: 오류가 처음 발생한 정확한 조건은 무엇인가입니다.

18. AI 요청에 반드시 포함할 다섯 요소는 무엇인가요?

정답 보기

정답: 목적·맥락·입력·완료 기준·금지/한계입니다.

19. AI 결과를 바로 확정하면 안 되는 이유는 무엇인가요?

정답 보기

정답: AI는 누락·과잉 단정·잘못된 경로·위험한 command를 만들 수 있으므로 원문과 실행 evidence로 사람이 검수해야 합니다.

20. 좋은 검수는 정상 결과만 보나요?

정답 보기

정답: 아닙니다. 정상·실패·권한·경계·되돌리기를 함께 확인합니다.

21. 실습 기록에 command나 행동 전 목적을 쓰는 이유는 무엇인가요?

정답 보기

정답: 결과가 예상과 다를 때 무엇을 시험했는지 되짚고 위험한 행동을 줄이기 위해서입니다.

22. 색상만으로 PASS·FAIL을 구분하면 안 되는 이유는 무엇인가요?

정답 보기

정답: 인쇄·저대비·색각 차이에서도 상태를 알 수 있도록 문자·아이콘·선으로 함께 표시해야 하기 때문입니다.

23. 실제 개인정보나 secret을 실습에 넣어도 되나요?

정답 보기

정답: 아닙니다. 합성 값과 placeholder를 사용하고 secret은 환경변수 등 분리된 경로로 다룹니다.

24. 한 번에 여러 원인을 바꾸면 왜 안 되나요?

정답 보기

정답: 어떤 변경이 결과를 바꿨는지 알 수 없으므로 가설 하나씩 시험해야 합니다.

25. 완료 기준은 느낌으로 적어도 되나요?

정답 보기

정답: 아닙니다. 다른 사람이 관찰·실행·비교할 수 있는 evidence로 적어야 합니다.

26. 실패는 항상 학습 실패인가요?

정답 보기

정답: 아닙니다. 예상한 실패를 안전하게 재현하고 원인·대안·복구를 설명하면 중요한 학습 evidence입니다.

27. 공식 출처를 남기는 이유는 무엇인가요?

정답 보기
정답: 변동 가능한 도구·단계·명령의 현재 의미를 다시 확인하고 교육 설명의 적용 경계를 밝히기 위해서입니다.

28. 모바일 화면에서도 확인할 핵심은 무엇인가요?

정답 보기
정답: 제목·입력·행동·상태·완료 evidence가 가로 넘침 없이 같은 순서로 읽히는지 확인합니다.

29. 이 매뉴얼을 완료했다는 가장 좋은 증거는 무엇인가요?

정답 보기
정답: 오류 해결 기록표와 실습 영수증을 다른 사람이 보고 같은 판단을 재현하는 것입니다.

30. 다음 매뉴얼로 넘어가기 전 한 문장으로 무엇을 설명해야 하나요?

정답 보기
정답: 오류 메시지를 읽고 질문하기의 핵심 구조와 한계를 자기 사례로 설명해야 합니다.

30. 한 장 요약과 다음 매뉴얼 M02-01

기억할 것	한 문장
목적	오류 type·message·file·line·traceback·재현 절차를 읽고 가설·실험·회귀 검증이 있는 오류 해결 기록을 만듭니다.
핵심	원문 오류를 그대로 보존한다 → 마지막 줄과 첫 관련 frame을 읽는다 → 최소 재현 절차를 만든다 → 가설 하나씩 시험한다 → 수정 뒤 정상·회귀를 검증한다
산출물	오류 해결 기록표
이전	M01-03 개발 도구와 실행 환경 점검하기
다음	M02-01 · 웹 서비스는 어떻게 움직이는가

완료 확인: 오류 해결 기록표와 실습 영수증을 보지 않고 핵심 흐름·실패·경계를 설명한 뒤 다음 매뉴얼로 이동합니다.