TOML v1.0.0. 쿡북

> “ 값 타입 , 구조 표현 , 실전 파일 패턴 기반 TOML 쿡북

읽은 결과와 오류 메시지는 Python 3.14.2 `tomllib` 실측. 날짜·시각은 Python 객체 표기.

- 1. 값 옆에 이유 적기
- 2. 여러 줄 텍스트
- 3. 경로·정규식·SQL — 백슬래시와 따옴표
- 4. 큰 수·권한·마스크·색상
- 5. 날짜·시각 고르기
- 6. 점·공백·유니코드가 든 키
- 7. 계층 — 헤더·dotted 키·인라인 테이블
- 8. 목록
- 9. 객체 목록
- 10. 루트 키와 섹션 순서
- 11. 섹션 나눠 쓰기 — 되는 것과 안 되는 것
- 12. 값 없음·선택 항목
- 13. JSON → TOML
- 14. 파일 한 장
- 참고

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

1. 값 옆에 이유 적기

`#`부터 줄 끝까지 주석. 전용 줄, 값 뒤 모두 가능. 파싱 결과에 남지 않는다.

```
# 프로젝트 설정 — 이 파일은 저장소에 커밋한다
title = "docs"        # 사이트 제목. 브라우저 탭에 나온다

# ---- 빌드 ------------------------------------------------
[build]
minify = true         # 배포 빌드에서만 켠다
```

```
{"title": "docs", "build": {"minify": true}}
```

- 정렬 공백·들여쓰기는 문법이 아니다. 자유롭게 맞춘다.
- 주석에 값을 되풀이하지 않는다(`# 10%`). 값을 고칠 때 같이 안 고쳐진다.
- 주석 처리로 키를 끄지 않는다. 파서에게는 없던 키와 같다. `enabled = false`로 끈다(레시피 12).

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

2. 여러 줄 텍스트

`"""..."""`. 여는 구분자 뒤 첫 줄바꿈은 버려지고, 닫는 구분자 앞 줄바꿈은 남는다. 줄 끝 `\`는 다음 비공백까지의 공백·줄바꿈을 지운다. 들여쓰기는 값에 그대로 들어간다.

```
banner = """
백업이 완료되었습니다.
보관 위치: /var/backups
"""
one_line = """\
    여러 줄로 나눠 적었지만 \
    실제 값은 한 줄이다.\
"""
indented = """
  들여쓴 줄
    더 들여쓴 줄
"""
```

```
{"banner": "백업이 완료되었습니다.\n보관 위치: /var/backups\n",
 "one_line": "여러 줄로 나눠 적었지만 실제 값은 한 줄이다.",
 "indented": "  들여쓴 줄\n    더 들여쓴 줄\n"}
```

- 마지막 줄바꿈이 싫으면 `"""`를 마지막 글자 바로 뒤에 붙인다.
- `"""` 안에서도 `\`는 이스케이프다. 백슬래시가 있는 텍스트는 `'''`(레시피 3).

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

3. 경로·정규식·SQL — 백슬래시와 따옴표

리터럴 문자열 `'...'` `'''...'''`은 이스케이프를 해석하지 않는다.

값 안에 있는 것	표기
백슬래시 (경로·정규식)	`'...'`
작은따옴표 (SQL)	`'''...'''` — 연속 두 개까지 안에 둘 수 있다
작은따옴표 + 백슬래시, 한 줄	`"..."` + 백슬래시만 `\\`
탭·줄바꿈을 글자로	`"..."` + `\t` `\n`

```
path  = 'C:\Users\name\file.txt'
regex = '\d{4}-\d{2}-\d{2}'
sql = '''
SELECT * FROM t WHERE name = 'a' AND note LIKE '%x%'
'''
both = "작은따옴표 ' 와 큰따옴표 \" 와 백슬래시 \\ 가 다 들어간 값"
quote_in_lit = '''it''s fine'''
```

```
{"path": "C:\\Users\\name\\file.txt", "regex": "\\d{4}-\\d{2}-\\d{2}",
 "sql": "SELECT * FROM t WHERE name = 'a' AND note LIKE '%x%'\n",
 "both": "작은따옴표 ' 와 큰따옴표 \" 와 백슬래시 \\ 가 다 들어간 값",
 "quote_in_lit": "it''s fine"}
```

(JSON 출력의 `\\`는 값 안에서 백슬래시 한 개.)

기본 문자열 이스케이프는 `\b \t \n \f \r \" \\ \uXXXX \UXXXXXXXX` 아홉 개. `"caf\u00E9 \U0001F600"` → `café 😀`.

```
# × 한 줄 리터럴 안의 작은따옴표 — 거기서 닫힌다
x = 'can't'
```

```
Expected newline or end of document after a statement (at line 1, column 10)
```

```
# × 기본 문자열의 Windows 경로 — \U 가 유니코드 이스케이프
path = "C:\Users\name"
```

```
Invalid hex value (at line 1, column 13)
```

```
path = "C:\temp\new"      # 오류 없음. \t 는 탭, \n 은 줄바꿈이 된다
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
{"path": "C:\temp\new"}
```

세 번째는 파싱이 성공하므로 가장 늦게 발견된다. 경로는 항상 `'...'`.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

4. 큰 수 · 권한 · 마스크 · 색상

자릿수 구분 `_`, 진법 접두 `0x` `0o` `0b`. 읽으면 전부 정수. 파서는 진법을 기억하지 않는다.

```
bytes  = 10_485_760      # 10 MiB
mode   = 0o755           # 파일 권한
mask   = 0b1111_0000     # 상위 4비트
color  = 0xFF_88_00      # RGB
ratio  = 0.25
avogad = 6.022e23
limit  = inf
undef  = nan
```

```
{"bytes": 10485760, "mode": 493, "mask": 240, "color": 16746496,
 "ratio": 0.25, "avogad": 6.022e+23, "limit": Infinity, "undef": NaN}
```

```
# × 리딩 제로
port = 007
# × 소수점 한쪽이 비어 있음
r = .5
# × 진법 접두에 부호
m = -0xFF
```

```
port: Expected newline or end of document after a statement (at line 1, column 9)
r:    Invalid value (at line 1, column 5)
m:    Expected newline or end of document after a statement (at line 1, column 7)
```

- 정수 범위는 64비트(-2^{63} ~ $2^{63}-1$). `9223372036854775808`을 `tomllib`은 읽지만(Python 정수가 임의 정밀도) 스펙 밖이다. 64비트 언어에서는 오류 또는 잘림. 그 크기는 문자열로.
- 실수는 binary64. 금액은 정수(최소 단위) 또는 문자열.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

5. 날짜·시각 고르기

따옴표 없이 쓴다. 기준: 어디서 읽어도 같은 순간이면 오프셋, 그 자리의 벽시계면 Local.

값의 뜻	종류	예
같은 순간 (배포 시각, 만료 시점)	Offset Date-Time	`2026-10-01T00:00:00+09:00`, `...Z`
반복되는 벽시계 시각 (점검 시작)	Local Time	`03:30:00`
하루 단위 (기한)	Local Date	`2026-10-01`
벽시계 날짜+시각, 시간대는 다른 키	Local Date-Time	`2026-10-01T09:00:00` + `timezone = "Asia/Seoul"`

```
released   = 2026-10-01T00:00:00+09:00
built      = 2026-10-01T00:00:00Z
local_dt   = 2026-10-01T09:00:00
launch_day = 2026-10-01
daily_at   = 03:30:00
space_sep  = 2026-10-01 09:00:00        # T 대신 공백
```

키	읽은 값
`released`	`datetime(2026, 10, 1, 0, 0, tzinfo=+09:00)`
`built`	`datetime(2026, 10, 1, 0, 0, tzinfo=UTC)`
`local_dt`, `space_sep`	`datetime(2026, 10, 1, 9, 0)` — tzinfo 없음
`launch_day`	`date(2026, 10, 1)`
`daily_at`	`time(3, 30)`

- `"2026-10-01"`은 문자열. `{"d": "2026-10-01"}`로 읽히고 날짜 연산이 안 된다.
- Local Date-Time은 시간대 해석이 읽는 쪽 몫이라 언어마다 갈린다(6편 레시피 3). 절대 시점은 오프셋을 붙인다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

6. 점·공백·유니코드가 든 키

bare 키는 `A-Z a-z 0-9 _ -`. 그 밖은 따옴표. 키는 항상 문자열이고 대소문자를 구분한다.

```
name = "bare"
ko-KR = "하이픈은 bare 키에 허용"
"ko KR" = "공백은 따옴표"
"api.example.com" = "점은 따옴표 없으면 경로"
'1.2' = "버전 문자열 키"
1234 = "숫자만 있는 키도 문자열 키"
"" = "빈 키도 허용되지만 쓰지 않는다"
```

```
{"name": "bare", "ko-KR": "하이픈은 bare 키에 허용", "ko KR": "공백은 따옴표",
 "api.example.com": "점은 따옴표 없으면 경로", "1.2": "버전 문자열 키",
 "1234": "숫자만 있는 키도 문자열 키", "": "빈 키도 허용되지만 쓰지 않는다"}
```

```
1.2 = "x"               # 오류 없음. 키 "1.2"가 아니라 경로
```

```
{"1": {"2": "x"}}
```

호스트명·버전·IP·패키지명은 따옴표 필수. 오류가 아니라 트리 모양이 바뀌므로 늦게 발견된다.

```
# × 같은 키 두 번
name = 1
name = 2
```

```
Cannot overwrite a value (at end of document)
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

7. 계층 — 헤더·dotted 키·인라인 테이블

세 표기의 결과는 같다.

```
# A. 테이블 헤더
[server]
host = "localhost"
port = 8080

[server.tls]
enabled = true
cert = "/etc/ssl/server.pem"
```

```
# B. dotted 키
server.host = "localhost"
server.port = 8080
server.tls.enabled = true
server.tls.cert = "/etc/ssl/server.pem"
```

```
# C. 인라인 테이블
server = { host = "localhost", port = 8080, tls = { enabled = true, cert = "/etc/ssl/server.pem" } }
```

```
{"server": {"host": "localhost", "port": 8080, "tls": {"enabled": true, "cert":
"/etc/ssl/server.pem"}}}
```

상황	표기
키가 많다, 값마다 주석	A 헤더
키 한두 개, 얕은 중첩	B dotted 키
키 서너 개, 한 줄에	C 인라인
3단 이상 중첩	A 헤더

- 인라인 테이블은 v1.0.0에서 한 줄, 후행 콤마 불가, 정의 후 키 추가 불가(레시피 11). 커질 것은 처음부터 헤더.
- 헤더 아래의 dotted 키는 그 헤더의 하위. `[server]` 아래 `tls.enabled = true`는 `server.tls.enabled`.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

8. 목록

배열 `[ ]`. v1.0.0에서도 여러 줄·후행 콤마·원소 옆 주석 가능.

```
origins = [
  "https://app.example.com",
  "https://admin.example.com",   # 관리 콘솔
]
ports   = [8080, 8081, 8082]
matrix  = [[1, 0], [0, 1]]
mixed   = [1, "two", 3.0, true, 2026-10-01]
empty   = []
nested_objs = [
  { name = "a", weight = 1 },
  { name = "b", weight = 3 },
]
```

```
{"origins": ["https://app.example.com", "https://admin.example.com"],
 "ports": [8080, 8081, 8082], "matrix": [[1, 0], [0, 1]],
 "mixed": [1, "two", 3.0, true, datetime.date(2026, 10, 1)],
 "empty": [], "nested_objs": [{"name": "a", "weight": 1}, {"name": "b", "weight": 3}]}
```

- 원소마다 줄바꿈 + 후행 콤마: 추가·삭제 diff가 한 줄.
- 타입 혼합은 허용되지만 설정에서는 한 배열 한 타입.
- 배열은 통째로 한 값. 나중에 덧붙이는 문법이 없고, 파일 겹쳐 읽기에서도 병합이 아니라 교체(6편 레시피 4). "기본 + 추가"는 키를 둘로.
- 인라인 테이블 배열(`nested_objs`)은 원소가 두세 키일 때. 커지면 레시피 9.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

9. 객체 목록

같은 꼴 객체의 반복. 헤더마다 원소 하나. 뒤따르는 `[name.sub]`은 직전 원소에 붙는다.

```
[[menu]]
name = "홈"
url  = "/"

[[menu]]
name = "문서"
url  = "/docs"

[menu.badge]          # 직전 원소(문서)에 붙는다
text = "new"

[[menu]]
name = "블로그"
url  = "/blog"
```

```
{"menu": [{"name": "홈", "url": "/"},
          {"name": "문서", "url": "/docs", "badge": {"text": "new"}},
          {"name": "블로그", "url": "/blog"}]}
```

부착 대상은 이름이 아니라 위치. `[menu.badge]`를 세 번째 `[[menu]]` 뒤로 옮기면 블로그에 붙는다.

상황	표기
원소마다 키 서너 개 이상, 원소별 주석	`[[menu]]`
원소가 한두 키, 전체가 한눈에	`menu = [{ ... }, { ... }]`
원소에 하위 테이블	`[[menu]]` + `[menu.sub]`

한 목록은 한 표기로. 대괄호 하나와 둘은 섞이지 않는다.

```
# × 테이블을 배열로 이어 쓰기
[server]
host = "a"
[[server]]
host = "b"
```

```
Cannot overwrite a value (at line 3, column 9)
```

```
# × 배열을 테이블로 이어 쓰기
[[a]]
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
x = 1
[a]
y = 2
```

```
Cannot declare ('a',) twice (at line 3, column 3)
```

```
# × 인라인 배열에 [[ ]]로 덧붙이기
products = [{ name = "a" }]
[[products]]
name = "b"
```

```
Cannot mutate immutable namespace ('products',) (at line 2, column 11)
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

10. 루트 키와 섹션 순서

헤더 없이 시작하는 키는 루트. 첫 헤더 뒤의 키는 전부 그 헤더 아래. 루트로 돌아오는 문법은 없다.

```
[server]
port = 8080
app_name = "orders"    # server 아래로 들어간다
```

```
{"server": {"port": 8080, "app_name": "orders"}}
```

```
app_name = "orders"
version  = "1.4.0"

[server]
port = 8080
```

```
{"app_name": "orders", "version": "1.4.0", "server": {"port": 8080}}
```

- 파일 전체의 메타 키(이름·버전·설명)는 첫 헤더 앞에.
- 섹션 순서는 파서에 의미 없음. 예외는 `[[ ]]` 부착(레시피 9).
- 헤더는 매번 루트 기준 절대 경로. `[a.b]` 뒤에 `[c]`로 얕아져도 된다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

11. 섹션 나눠 쓰기 — 되는 것과 안 되는 것

기준: 그 테이블이 이미 명시적으로 정의되었는가.

먼저 쓴 것	나중에
`[a.b]` (상위 `a`는 암묵 생성)	`[a]` 가능
`[a]`	`[a]` 불가, `[a.c]` 가능
`[a]` 안의 `b.c = 1`	`[a.b]` 불가
`a = { ... }`	`a.x = 1`, `[a]`, `[a.x]` 불가
`a = [{ ... }]`	`[[a]]` 불가 (레시피 9)

```
[a.b]
x = 1

[a]
y = 2
```

```
{"a": {"b": {"x": 1}, "y": 2}}
```

```
# × 같은 헤더 두 번
[a]
x = 1

[a]
y = 2
```

```
Cannot declare ('a',) twice (at line 4, column 3)
```

```
# × dotted 키로 만든 테이블을 헤더로 다시 열기
[a]
b.c = 1

[a.b]
d = 2
```

```
Cannot declare ('a', 'b') twice (at line 4, column 5)
```

```
# × 인라인 테이블에 밖에서 키 추가
point = { x = 1 }
point.y = 2
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
Cannot mutate immutable namespace ('point',) (at line 2, column 12)
```

허용되는 경우라도 같은 테이블의 키는 한 곳에 모은다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

12. 값 없음·선택 항목

null이 없다. "설정 안 함"은 키를 뺀다. "비움"은 `[]` `""`. "끔"은 `false`.

상태	TOML	읽는 쪽
설정하지 않음 (기본값)	키 없음	부재 → 기본값
명시적으로 비움	`[]`, `""`	빈 값
기능 끔	`enabled = false`	불리언
끄되 기한을 남김	`enabled = false` + `disabled_until = 2026-10-15`	날짜 비교

```
[proxy]
# url 키를 생략하면 프록시를 쓰지 않는다
timeout_sec = 5

[report]
recipients = []          # 빈 배열: 받는 사람 없음
subject = ""             # 빈 문자열: 제목 없음
```

```
{"proxy": {"timeout_sec": 5}, "report": {"recipients": [], "subject": ""}}
```

```
# × null 리터럴
x = null
```

```
Invalid value (at line 1, column 5)
```

"키 없음"과 "빈 값"의 뜻을 파일 주석과 읽는 코드가 같게 둔다.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

13. JSON → TOML

JSON	TOML	레시피
스칼라	그대로	3, 4
`null`	키 삭제	12
날짜처럼 생긴 문자열	따옴표 유지. 날짜 타입은 읽는 코드가 기대할 때만	5
작은 객체	인라인 테이블	7
큰 객체	`[a.b]` 헤더	7
원시 값 배열	배열	8
객체 배열	`[[a]]`	9
최상위 스칼라	첫 헤더 앞	10
점·공백 든 키	따옴표 키	6

```
{
  "name": "widget",
  "tags": ["a", "b"],
  "price": 9.5,
  "in_stock": true,
  "discount": null,
  "released": "2026-10-01",
  "dims": { "w": 10, "h": 20 },
  "variants": [
    { "sku": "W-1" },
    { "sku": "W-2", "color": "red" }
  ]
}
```

```
name     = "widget"
tags     = ["a", "b"]
price    = 9.5
in_stock = true
# discount: null은 TOML에 없다. 키를 뺀다
released = "2026-10-01"     # 원본이 문자열이면 따옴표 유지
dims     = { w = 10, h = 20 }

[[variants]]
sku = "W-1"

[[variants]]
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
sku   = "W-2"
color = "red"
```

두 파일의 파싱 결과는 null 키를 뺀 원본 기준 동일(`True`). `released`는 양쪽 다 문자열.

변환 도구 출력에는 주석이 없다. 옮긴 뒤 레시피 1.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

14. 파일 한 장

레시피 조합 예. 로컬 백업 도구 설정. 주석의 번호는 레시피.

```
# backup.toml — 로컬 백업 도구 설정                          (1)
name    = "laptop-backup"                                  # 루트 키는 첫 헤더 앞 (10)
version = "2"                       # 도구가 문자열로 비교. 숫자로 승격하지 않음 (13)

[schedule]
daily_at     = 02:30:00             # 반복 시각 = Local Time (5)
weekly_on    = "sun"
paused_until = 2026-10-15           # 기한 = Local Date. 키를 지우면 재개 (5, 12)

[retention]
daily   = 7
weekly  = 4
monthly = 12
min_free_bytes = 5_368_709_120      # 5 GiB (4)

[storage]
root = '/Volumes/Backup'            # 경로 = 리터럴 (3)
mode = 0o700                        # 권한 = 8진 (4)

[storage.remote]                    # 키가 늘 수 있는 하위 객체 = 헤더 (7)
bucket = "laptop-backup"
region = "ap-northeast-2"

[[targets]]                         # 반복 객체 (9)
name    = "documents"
path    = '~/Documents'
exclude = ['\.DS_Store$', '/node_modules/', '\.tmp$']   # 정규식 = 리터럴 (3)

[[targets]]
name     = "photos"
path     = '~/Pictures'
compress = false                    # 끔 = false (12)

[targets.limits]                    # 직전 원소(photos)에만 (9)
max_file_bytes = 4_294_967_296

[notify]
on_failure = { channel = "mail", to = ["me@example.com"] }   # 작은 객체 = 인라인 (7, 8)
on_success = { channel = "none" }
```

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

```
targets[1] = {"name": "photos", "path": "~/Pictures", "compress": false,
              "limits": {"max_file_bytes": 4294967296}}
storage.mode = 448 (0o700)
schedule.daily_at = datetime.time(2, 30)
```

섹션 순서는 메타 → 동작 → 저장 위치 → 대상 → 알림. 파서에 의미 있는 순서는 `[[targets]]`와 `[targets.limits]`의 상대 위치뿐.

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.

참고

- TOML v1.0.0 스펙 · v1.1.0 — 1.1 변경점은 1편 §6
- Python tomllib — 실측 파서

Copyright ⓒ TOMATO SYSTEM CO., LTD. All rights reserved.