.mni 데이터타입(MVP->Full)

1) 공통 원칙

- schema_version: 세분화(예: "1.0-mvp", "1.0").
- **하위 호환**: MVP ⊂ Full (MVP 파일은 Full 런타임이 그대로 실행 가능).
- **결정성**: 핵심 필드만으로 hash_key 산출 → 캐시/증분 렌더 기준.

---

2) MVP 스키마 (필수 최소집합)

2.1 필드 정의

- **필수**
  - schema_version (string; 예: "1.0-mvp")
  - problem (object)
    - id (string, unique)
    - statement (string)
    - metadata (object; 최소 subject, unit 권장)
  - proof_tape (array<string> 또는 array<object>{step, expr_in/out})
  - visual (object)
    - type = "ManimScene"
    - sections (array<{section_name, steps[]}>)
  - verification (object)
    - sympy (string|object; 최소 "검증식 또는 코드")
- **선택**
  - build (object; 렌더 옵션 일부)
  - notes (string; 제작 코멘트)

2.2 제약(Validation)

- visual.sections[].steps[] 의 action 은 MVP 액션 셋으로 제한
  (CreateAxes, PlotFunction, HighlightPoint, CreateTex, FadeIn, Indicate)
- verification.sympy는 **실행 가능한 최소 코드 문자열** 또는 {code, status} 형태.
- **외부 CAS(fallback), 다중 풀이경로(ToT/PoT), 배포/관측**은 **없음**.

2.3 MVP 구조


```
{
  "schema_version": "1.0",
  "problem": {
    "id": "QF001",
    "statement": "함수 y = x^2 - 4x + 3의 꼭짓점을 구하라",
    "metadata": { "subject": "수학", "unit": "이차함수", "difficulty": "중간", "time_estimate_min": 3 }
  },
  "structure": {
    "tot": {
      "nodes": [
        {"id":"n1","type":"goal","text":"꼭짓점 좌표"},
        {"id":"n2","type":"transform","text":"완전제곱식"},
        {"id":"n3","type":"result","text":"(2,-1)"}
      ],
      "edges": [{"from":"n1","to":"n2"},{"from":"n2","to":"n3"}]
    },
    "pot": {
      "lang": "python",
      "cells": [
        "import sympy as sp; x=sp.Symbol('x'); f=x**2-4*x+3",
        "vx=sp.solve(sp.diff(f,x),x)[0]; vy=f.subs(x,vx)"
      ]
    }
  },
  "proof_tape": [
    {"step":1, "rule":"complete_square", "expr_in":"x^2-4x+3", "expr_out":"(x-2)^2-1", "comment":"
표준형"},
    {"step":2, "rule":"vertex", "expr_in":"(x-2)^2-1", "expr_out":"(2,-1)"}
  ],
  "visual": {
    "type": "ManimScene",
    "sections": [
      {
        "section_name": "Graph",
        "steps": [
          { "action": "CreateAxes", "x_range": [-2,6], "y_range": [-2,10] },
          { "action": "PlotFunction", "function": "x**2 - 4*x + 3" },
          { "action": "HighlightPoint", "point": [2, -1], "color": "yellow" }
        ]
      }
    ]
  },
  "verification": {
```

```
    "sympy": { "code":"...", "status":"pass", "artifacts":["vx=2","vy=-1"] },
    "wolfram_alpha": { "query":"vertex of y=x^2-4x+3", "used": false }
  },
  "build": {
    "options": { "fps": 30, "resolution": "1400x800", "theme":"dark" },
    "hash_key": "QF001:9b7c...:v1",
    "created_at": "2025-09-04T07:15:00Z"
  },
  "publish": {
    "targets": [
      { "type": "web", "url": "https://example.com/qf001", "meta": {"visibility":"unlisted"} }
    ]
  },
  "metrics": { "verify_pass": true, "render_ms": 1250, "cache_hit": false }
}
```

---

3) Full 스키마 (확장·운영·재현성)

3.1 필드 정의

- **필수(=MVP 필수 + α)**
  - schema_version (예: "1.0")
  - problem (MVP 동일 + 난이도/시간 등 확장 메타 권장)
  - proof_tape (array<object> 권장: {step, rule, expr_in, expr_out, comment})
  - visual (MVP 동일 + 3D/GeoGebra/템플릿 참조 허용)
  - verification (object; 멀티 엔진)
    - sympy {code, status, artifacts[]}
    - wolfram_alpha {query, used, status} (optional)
- **선택 확장**
  - structure
    - tot (추론 트리: {nodes:[{id,type,text}], edges:[{from,to,label}]})
    - pot (계산 코드: {lang:"python", cells:["...", ...]})
  - assets (외부 리소스 목록: 폰트/이미지/오디오)
  - build {options:{fps,resolution,theme}, hash_key, created_at}
  - publish {targets:[{type,url,meta}]}

- metrics {verify_pass, render_ms, cache_hit, ...}
- i18n (다국어: {lang:"ko", statements:{}, subtitles:{}})
- provenance (작성/리뷰 이력, 저작권 표시)

3.2 제약(Validation)

- structure.pot.lang ∈ {"python","sympy"} (초기)
- visual 의 type ∈ {"ManimScene","GeoGebra","ThreeJS"}
- build.hash_key는 **결정적 해시** (문제ID + 정규화된 수식 + visual.sections + build.options).

3.3 Full 예시(요약형)


```
{
  "schema_version": "1.0",
  "problem": {
    "id": "QF001",
    "statement": "함수 y = x^2 - 4x + 3의 꼭짓점을 구하라",
    "metadata": { "subject": "수학", "unit": "이차함수", "difficulty": "중간", "time_estimate_min": 3 }
  },
  "structure": {
    "tot": {
      "nodes": [
        {"id":"n1","type":"goal","text":"꼭짓점 좌표"},
        {"id":"n2","type":"transform","text":"완전제곱식"},
        {"id":"n3","type":"result","text":"(2,-1)"}
      ],
      "edges": [{"from":"n1","to":"n2"},{"from":"n2","to":"n3"}]
    },
    "pot": {
      "lang": "python",
      "cells": [
        "import sympy as sp; x=sp.Symbol('x'); f=x**2-4*x+3",
        "vx=sp.solve(sp.diff(f,x),x)[0]; vy=f.subs(x,vx)"
      ]
    }
  },
  "proof_tape": [
```

```
    {"step":1, "rule":"complete_square", "expr_in":"x^2-4x+3", "expr_out":"(x-2)^2-1", "comment":"표준형"},
    {"step":2, "rule":"vertex", "expr_in":"(x-2)^2-1", "expr_out":"(2,-1)"}
  ],
  "visual": {
    "type": "ManimScene",
    "sections": [
      {
        "section_name": "Graph",
        "steps": [
          { "action": "CreateAxes", "x_range": [-2,6], "y_range": [-2,10] },
          { "action": "PlotFunction", "function": "x**2 - 4*x + 3" },
          { "action": "HighlightPoint", "point": [2, -1], "color": "yellow" }
        ]
      }
    ]
  },
  "verification": {
    "sympy": { "code":"...", "status":"pass", "artifacts":["vx=2","vy=-1"] },
    "wolfram_alpha": { "query":"vertex of y=x^2-4x+3", "used": false }
  },
  "build": {
    "options": { "fps": 30, "resolution": "1400x800", "theme":"dark" },
    "hash_key": "QF001:9b7c...:v1",
    "created_at": "2025-09-04T07:15:00Z"
  },
  "publish": {
    "targets": [
      { "type": "web", "url": "https://example.com/qf001", "meta": {"visibility":"unlisted"} }
    ]
  },
  "metrics": { "verify_pass": true, "render_ms": 1250, "cache_hit": false }
}
```

4) MVP ↔ Full 전환(진화 규칙)

- **그대로 승격**: MVP의 proof_tape:string[] → Full에서 {step, rule, ...} 객체 배열로 확장(기존 문자열은 comment로 흡수).
- **확장 추가**: structure.tot/pot, publish, metrics, i18n, provenance, assets는 **추가만으로** 호환.
- **해시 안정성**: build.hash_key는 **렌더 결과에 영향을 주는 필드만** 포함 (문제ID/정규화 수식/visual/build.options). 관측·배포·메모는 해시에서 제외.

5) 데이터 타입 요약(실무 체크리스트)

- problem.id: ^[A-Za-z0-9._-]{3,64}$
- proof_tape: MVP= string[] | Full={step:number, rule?:string, expr_in?:string, expr_out?:string, comment?:string}[]
- visual.sections[].steps[]: {action:string, ...payload} (액션별 payload 스키마 정의)
- verification.sympy: MVP=string | Full={code:string, status:"pass"|"fail", artifacts?:string[]}
- structure.pot.lang: "python"|"sympy"
- build.options: {fps:number, resolution:"{w}x{h}", theme?:string}
- metrics: 숫자/불리언 위주, 렌더·검증 지표

6) 구현 순서 (MVP→Full)

1. **MVP 실행 경로**: problem + proof_tape(string[]) + visual + verification.sympy(string) → Manim 렌더 파이프라인
1. **템플릿 확장**: visual 액션 추가 &공통 스타일 템플릿 도입
1. **검증 강화**: verification.sympy 객체화 + 외부 CAS fallback
1. **구조화 도입**: structure.tot/pot 추가, ProofTape와 연결
1. **운영 단계**: build.hash_key·publish·metrics·i18n·provenance 순차 도입