상태평가 함수

일반 바닥판 상태평가

```
def evaluate_slab_condition(crack_width=None, crack_ratio=None, leak_ratio=None,
                            surface_damage_ratio=None, rebar_corrosion_ratio=None,
                            other_damage=False):
    """
    일반 콘크리트 바닥판 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        crack_ratio: 균열률 (%)
        leak_ratio: 누수 및 백태 면적률 (%)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        other_damage: 기타 손상 여부 (기준에 명시되지 않은 손상이 있는 경우 True)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ------------------------------------------
    ✓ 표면손상의 정의:
    - 표면손상은 콘크리트에 국한된 손상으로,
      파손, 박락, 재료분리, 층분리 등을 의미함.
    - 철근노출, 철근부식, 균열, 백태 등은 별도 항목으로 평가함.
    ------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 균열폭 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 1.0:
            return 'e'
        elif crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    # 균열률 기준
    if crack_ratio is not None:
        has_any_damage = True
```

```
        if crack_ratio >= 20:
            return 'e'
        elif crack_ratio >= 10:
            grade = max(grade, 'd')
        elif crack_ratio >= 2:
            grade = max(grade, 'c')
        elif crack_ratio > 0:
            grade = max(grade, 'b')

    # 누수 및 백태 기준
    if leak_ratio is not None:
        has_any_damage = True
        if leak_ratio >= 10:
            grade = max(grade, 'c')
        elif leak_ratio > 0:
            grade = max(grade, 'b')

    # 표면손상 기준
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    # 철근 부식 기준
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    # 손상이 전혀 없는 경우
    if not has_any_damage:
        grade = 'a'

    return grade
```

프리스트레스 바닥판 상태평가

```
def evaluate_psc_slab_condition(crack_width=None, crack_ratio=None, leak_ratio=None,
                                surface_damage_ratio=None, rebar_corrosion_ratio=None,
                                other_damage=False):
    """
    프리스트레스 콘크리트 바닥판 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        crack_ratio: 균열률 (%)
        leak_ratio: 누수 및 백태 면적률 (%)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        other_damage: 기타 손상 여부 (기준에 명시되지 않은 손상이 있는 경우 True)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ------------------------------------------
    ✔ 표면손상의 정의:
    - 표면손상은 콘크리트에 국한된 손상으로,
      파손, 박락, 재료분리, 층분리 등을 의미함.
    - 철근노출, 철근부식, 균열, 백태 등은 별도 항목으로 평가함.
    ------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 균열폭 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 0.5:
            return 'e'
        elif crack_width >= 0.3:
            grade = max(grade, 'd')
        elif crack_width >= 0.2:
            grade = max(grade, 'c')
        else:
            grade = max(grade, 'b')

    # 균열률 기준
    if crack_ratio is not None:
        has_any_damage = True
        if crack_ratio >= 20:
            return 'e'
        elif crack_ratio >= 10:
```

```
            grade = max(grade, 'd')
        elif crack_ratio >= 2:
            grade = max(grade, 'c')
        else:
            grade = max(grade, 'b')

    # 백태/누수 기준
    if leak_ratio is not None:
        has_any_damage = True
        if leak_ratio >= 10:
            grade = max(grade, 'c')
        elif leak_ratio > 0:
            grade = max(grade, 'b')

    # 표면 손상 기준
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    # 철근 부식 기준
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    # 손상이 전혀 없는 경우
    if not has_any_damage:
        grade = 'a'

    return grade
```

철근콘크리트 거더 상태평가

```
def evaluate_rc_girder_condition(crack_width=None, surface_damage_ratio=None,
                                 rebar_corrosion_ratio=None,
                                 structural_issue=False, other_damage=False):
    """
    철근콘크리트 거더 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        structural_issue: 휨균열 과다, 과대처짐, 지점부 파손 등의 구조적 결함 여부 (True/False)
        other_damage: 명시되지 않은 기타 손상 여부 (True/False)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ------------------------------------------
    ✅ 표면손상의 정의:
    - 콘크리트 표면에 국한된 손상(파손, 박락, 재료분리, 층분리 등)
    - 균열, 철근노출, 백태 등은 별도 항목으로 평가함
    ------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 균열폭 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 1.0:
            return 'e'
        elif crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    # 표면손상 기준
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
```

```
            grade = max(grade, 'b')

    # 철근부식 기준
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')

    # 구조적 손상 여부
    if structural_issue:
        has_any_damage = True
        return 'e'

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    # 손상 전혀 없으면 a 유지
    if not has_any_damage:
        grade = 'a'

    return grade
```

프리스트레스 거더 상태평가

```
def evaluate_psc_girder_condition(crack_width=None, surface_damage_ratio=None,
                                  rebar_corrosion_ratio=None, tendon_corrosion_level=None,
                                  structural_issue=False, other_damage=False):
    """
    프리스트레스트 콘크리트 거더 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        tendon_corrosion_level: 강연선 부식 수준 (문자열: 'none', 'surface', 'section_loss', 'broken')
        structural_issue: 강연선 파단, 단부 파손, 정착부 손상 등의 구조적 손상 여부 (True/False)
        other_damage: 기타 손상 여부 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ---------------------------------------------
    ✅ 강연선 부식 수준 정의 (tendon_corrosion_level):
    - 'none': 노출 없음
    - 'surface': 표면부식
    - 'section_loss': 단면손실 동반한 부식
    - 'broken': 파단(단선)
    ---------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 균열 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 0.5:
            return 'e'
        elif crack_width >= 0.3:
            grade = max(grade, 'd')
        elif crack_width >= 0.2:
            grade = max(grade, 'c')
        else:
            grade = max(grade, 'b')

    # 표면 손상
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
```

```
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    # 철근 부식
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')

    # 강연선 부식 상태
    if tendon_corrosion_level is not None:
        has_any_damage = True
        if tendon_corrosion_level == 'broken':
            return 'e'
        elif tendon_corrosion_level == 'section_loss':
            grade = max(grade, 'd')
        elif tendon_corrosion_level == 'surface':
            grade = max(grade, 'c')
        elif tendon_corrosion_level == 'none':
            grade = max(grade, 'b')

    # 구조적 손상
    if structural_issue:
        has_any_damage = True
        return 'e'

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    if not has_any_damage:
        grade = 'a'

    return grade
```

강 바닥판, 강 거더, 강 교각, 강 주탑 상태평가

```
def evaluate_steel_component(main_rust_area=None, sub_rust_area=None,
                             section_loss_area=None, weld_defect_level=None,
                             bolt_damage=False, structural_issue=False,
                             other_damage=False):
    """
    강 바닥판 / 강 거더 / 강 교각(강 주탑) 상태평가 등급(a~e) 반환

    인자:
        main_rust_area: 주부재 부식 면적률 (%)
        sub_rust_area: 보조부재 부식 면적률 (%)
        section_loss_area: 단면손상 면적률 (%)
        weld_defect_level: 용접결함 수준 ('none', 'minor', 'severe')
        bolt_damage: 연결볼트 이완/탈락/파단 여부
        structural_issue: 좌굴, 변형, 파단 등 구조적 손상 여부
        other_damage: 기타 손상 여부 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'
    """
    grade = 'a'
    has_any_damage = False

    # 주부재 부식
    if main_rust_area is not None:
        has_any_damage = True
        if main_rust_area >= 10:
            return 'e'
        elif main_rust_area >= 2:
            grade = max(grade, 'd')
        elif main_rust_area > 0:
            grade = max(grade, 'c')

    # 보조부재 부식
    if sub_rust_area is not None:
        has_any_damage = True
        if sub_rust_area >= 10:
            grade = max(grade, 'd')
        elif sub_rust_area >= 2:
            grade = max(grade, 'c')
        elif sub_rust_area > 0:
            grade = max(grade, 'b')

    # 단면손실
    if section_loss_area is not None:
        has_any_damage = True
        if section_loss_area >= 10:
```

```
            return 'e'
        elif section_loss_area >= 2:
            grade = max(grade, 'd')

    # 용접결함
    if weld_defect_level is not None:
        has_any_damage = True
        if weld_defect_level == 'severe':
            grade = max(grade, 'd')
        elif weld_defect_level == 'minor':
            grade = max(grade, 'c')

    # 연결 볼트 손상
    if bolt_damage:
        has_any_damage = True
        grade = max(grade, 'd')

    # 구조적 손상
    if structural_issue:
        has_any_damage = True
        return 'e'

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    if not has_any_damage:
        grade = 'a'

    return grade
```

콘크리트 가로보 상태평가

```
def evaluate_concrete_crossbeam(crack_width=None, surface_damage_ratio=None,
                                rebar_corrosion_ratio=None, other_damage=False):
    """
    콘크리트 가로보 상태평가 등급(a~d)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        other_damage: 기타 손상 여부 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd'

    ------------------------------------------
    ✅ 표면손상의 정의:
    - 콘크리트에 국한된 손상으로, 파손, 박락, 재료분리, 층분리 등을 의미함
    - 균열, 백태, 철근노출 등은 별도 항목으로 평가함
    ------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 균열폭 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    # 표면손상 기준
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    # 철근부식 기준
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
```

```
    if rebar_corrosion_ratio >= 2:
        grade = max(grade, 'd')
    elif rebar_corrosion_ratio > 0:
        grade = max(grade, 'c')

# 기타 손상
if other_damage:
    has_any_damage = True
    grade = max(grade, 'b')

# 손상 없음
if not has_any_damage:
    grade = 'a'

return grade
```

강 가로보·세로보 상태평가

```
def evaluate_steel_crossbeam(main_rust_area=None, sub_rust_area=None,
                             section_loss_area=None, weld_defect_level=None,
                             structural_issue=False, other_damage=False):
    """
    강 가로보·세로보 상태평가 등급(a~d)을 반환하는 함수

    인자:
        main_rust_area: 주부재 부식 면적률 (%)
        sub_rust_area: 보조부재 부식 면적률 (%)
        section_loss_area: 부식에 의한 단면손상 면적률 (%)
        weld_defect_level: 용접결함 수준 ('none', 'minor', 'severe')
        structural_issue: 변형, 파단, 좌굴 등의 구조적 손상 여부
        other_damage: 기타 손상 여부 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd'
    """
    grade = 'a'
    has_any_damage = False

    # 주부재 부식
    if main_rust_area is not None:
        has_any_damage = True
        if main_rust_area >= 2:
            grade = max(grade, 'd')
        elif main_rust_area > 0:
            grade = max(grade, 'c')

    # 보조부재 부식
    if sub_rust_area is not None:
        has_any_damage = True
        if sub_rust_area >= 10:
            grade = max(grade, 'd')
        elif sub_rust_area >= 2:
            grade = max(grade, 'c')
        elif sub_rust_area > 0:
            grade = max(grade, 'b')

    # 단면손상
    if section_loss_area is not None:
        has_any_damage = True
        if section_loss_area >= 2:
            grade = max(grade, 'd')
        elif section_loss_area > 0:
            grade = max(grade, 'c')
```

```
# 용접결함
if weld_defect_level is not None:
    has_any_damage = True
    if weld_defect_level == 'severe':
        grade = max(grade, 'd')
    elif weld_defect_level == 'minor':
        grade = max(grade, 'c')

# 구조적 손상
if structural_issue:
    has_any_damage = True
    grade = max(grade, 'd')

# 기타 손상
if other_damage:
    has_any_damage = True
    grade = max(grade, 'b')

# 손상 없음
if not has_any_damage:
    grade = 'a'

return grade
```

케이블 상태평가

```
def evaluate_cable_component(corrosion_length_ratio=None, wire_break_ratio=None,
                             sheath_damage_ratio=None, anchorage_damage=False,
                             structural_failure=False, other_damage=False):
    """
    케이블 부재 상태평가 등급(a~e)을 반환하는 함수

    인자:
        corrosion_length_ratio: 점녹/부식 발생 길이 비율 (%)
        wire_break_ratio: 소선 단선 비율 (%)
        sheath_damage_ratio: 보호관 손상 길이 비율 (%)
        anchorage_damage: 정착구/행어밴드/새들 손상 여부 (True/False)
        structural_failure: 소선 단선(10% 이상), 행어밴드 파손 등 구조적 위험 (True/False)
        other_damage: 기타 손상 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'
    """
    grade = 'a'
    has_any_damage = False

    # 소선 단선율
    if wire_break_ratio is not None:
        has_any_damage = True
        if wire_break_ratio >= 10:
            return 'e'
        elif wire_break_ratio >= 2:
            grade = max(grade, 'd')
        elif wire_break_ratio > 0:
            grade = max(grade, 'c')

    # 점녹/부식 길이 비율
    if corrosion_length_ratio is not None:
        has_any_damage = True
        if corrosion_length_ratio >= 2:
            grade = max(grade, 'd')
        elif corrosion_length_ratio > 0.1:
            grade = max(grade, 'c')
        else:
            grade = max(grade, 'b')

    # 보호관 손상 길이 비율
    if sheath_damage_ratio is not None:
        has_any_damage = True
        if sheath_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif sheath_damage_ratio >= 2:
```

```
            grade = max(grade, 'c')
        elif sheath_damage_ratio > 0:
            grade = max(grade, 'b')

    # 정착구, 행어밴드 등 손상
    if anchorage_damage:
        has_any_damage = True
        grade = max(grade, 'd')

    # 구조적 파손 또는 단선
    if structural_failure:
        return 'e'

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    if not has_any_damage:
        grade = 'a'

    return grade
```

교대 상태평가

```
def evaluate_abutment_condition(crack_width=None, surface_damage_ratio=None,
                                rebar_corrosion_ratio=None, structural_issue=False,
                                severe_structural_risk=False, other_damage=False):
    """
    교대 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 교대 균열폭 (mm)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        structural_issue: 침하, 기울음, 날개벽 손상 등 구조적 손상 여부
        severe_structural_risk: 날개벽 전도 위험, 코핑 파손 등 안전성 저하 (True = 무조건 e)
        other_damage: 기타 손상 여부 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'
    """
    grade = 'a'
    has_any_damage = False

    # 균열폭 기준
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 1.0:
            return 'e'
        elif crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    # 표면손상 기준
    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    # 철근부식 기준
    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
```

```
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')

    # 구조적 손상 (침하, 기울음, 배면토 유출 등)
    if structural_issue:
        has_any_damage = True
        grade = max(grade, 'd')

    # 심각한 구조적 위험 (전도 위험, 코핑 파손 등)
    if severe_structural_risk:
        return 'e'

    # 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    if not has_any_damage:
        grade = 'a'

    return grade
```

교각 상태평가

```
def evaluate_pier_condition(crack_width=None, surface_damage_ratio=None,
                            rebar_corrosion_ratio=None, structural_issue=False,
                            severe_structural_risk=False, other_damage=False):
    """
    콘크리트 교각 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 균열폭 (mm)
        surface_damage_ratio: 표면 손상 면적률 (%)
        rebar_corrosion_ratio: 철근 부식 면적률 (%)
        structural_issue: 부등침하로 인한 기울음 등
        severe_structural_risk: 코핑부 파손, 거더 탈락 가능성 등 (True → e)
        other_damage: 기타 손상 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'
    """
    grade = 'a'
    has_any_damage = False

    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 1.0:
            return 'e'
        elif crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    if surface_damage_ratio is not None:
        has_any_damage = True
        if surface_damage_ratio >= 10:
            grade = max(grade, 'd')
        elif surface_damage_ratio >= 2:
            grade = max(grade, 'c')
        elif surface_damage_ratio > 0:
            grade = max(grade, 'b')

    if rebar_corrosion_ratio is not None:
        has_any_damage = True
        if rebar_corrosion_ratio >= 2:
            grade = max(grade, 'd')
        elif rebar_corrosion_ratio > 0:
            grade = max(grade, 'c')
```

```
if severe_structural_risk:
    return 'e'

if structural_issue:
    has_any_damage = True
    grade = max(grade, 'd')

if other_damage:
    has_any_damage = True
    grade = max(grade, 'b')

if not has_any_damage:
    grade = 'a'

return grade
```

기초 상태평가

```
def evaluate_foundation_condition(crack_width=None, section_loss=False,
                                  rebar_exposed=False, settlement_or_scour=False,
                                  severe_risk=False, other_damage=False):
    """
    기초 상태평가 등급(a~e)을 반환하는 함수

    인자:
        crack_width: 직접기초 상부 균열폭 (mm)
        section_loss: 단면손상 발생 여부 (True/False)
        rebar_exposed: 철근 노출 여부 (True/False)
        settlement_or_scour: 부등침하, 측방유동, 전반적 노출 등 (True/False)
        severe_risk: 단차, 상부구조 파손 등 구조적 위험 여부 (True → e)
        other_damage: 기타 손상 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'
    """
    grade = 'a'
    has_any_damage = False

    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width > 0:
            grade = max(grade, 'b')

    if section_loss:
        has_any_damage = True
        grade = max(grade, 'c')

    if rebar_exposed:
        has_any_damage = True
        grade = max(grade, 'd')

    if settlement_or_scour:
        has_any_damage = True
        grade = max(grade, 'd')

    if severe_risk:
        return 'e'

    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')
```

```
if not has_any_damage:
    grade = 'a'

return grade
```

교량받침 상태평가

```
def evaluate_bearing_condition(rubber_split=False, rubber_bulging=False,
                               shear_deformation='정상',
                               corrosion_area='없음',
                               crack_width=None,  # ← 균열 전용 항목
                               structural_failure=False,
                               other_damage=False):
    """
    교량받침 상태평가 등급(a~e)을 반환하는 함수

    인자:
        rubber_split: 고무재 갈라짐 여부 (True/False)
        rubber_bulging: 고무재 부풀음 여부 (True/False)
        shear_deformation: 전단변형 상태 ('정상', '0.7T 이상', '1.5T 이상')
        corrosion_area: 부식/부착불량 면적 수준 ('없음', '일부', '1/2 이상')
        crack_width: 받침 관련 균열폭 (mm)
                    - 받침 하부 콘크리트
                    - 몰탈층
                    - 기초받침부 등
                    - 모든 균열 표현에 해당
        structural_failure: 구조기능 상실 우려 여부 (True/False)
        other_damage: 기타 손상 여부 (True/False)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ------------------------------------------
    ✅ crack_width 항목 설명:
    - 아래와 같은 표현 모두 포함하여 평가
        · 받침 콘크리트 균열
        · 몰탈층 균열
        · 콘크리트 받침부 미세균열
        · 받침부 기초 미세균열 등
    - 단순 표현 차이와 무관하게 균열폭 기준으로 통합 평가
    ------------------------------------------
    """
    grade = 'a'
    has_any_damage = False

    # 1. 구조적 위험
    if structural_failure:
        return 'e'

    # 2. 부식/부착불량
    if corrosion_area == '1/2 이상':
        return 'e'
    elif corrosion_area == '일부':
```

```
        has_any_damage = True
        grade = max(grade, 'd')

    # 3. 전단변형량
    if shear_deformation == '1.5T 이상':
        has_any_damage = True
        grade = max(grade, 'd')
    elif shear_deformation == '0.7T 이상':
        has_any_damage = True
        grade = max(grade, 'c')

    # 4. 고무재 손상
    if rubber_split:
        has_any_damage = True
        if rubber_bulging:
            grade = max(grade, 'd')
        else:
            grade = max(grade, 'c')

    # 5. 균열 평가 (모든 종류 포함)
    if crack_width is not None:
        has_any_damage = True
        if crack_width >= 1.0:
            return 'e'
        elif crack_width >= 0.5:
            grade = max(grade, 'd')
        elif crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width >= 0.1:
            grade = max(grade, 'b')

    # 6. 기타 손상
    if other_damage:
        has_any_damage = True
        grade = max(grade, 'b')

    # 7. 손상 없음
    if not has_any_damage:
        grade = 'a'

    return grade
```

신축이음 상태평가

```
def evaluate_expansion_joint(aging_or_dirt=False,
                             function_degradation=False,
                             impact_or_noise=False,
                             structural_damage=False,
                             other_damage=False):
    """
    신축이음 상태평가 등급(a~d)을 반환하는 함수

    인자:
        aging_or_dirt: 고무재 노후화(경화, 미세균열), 토사퇴적, 이물질 끼임
        function_degradation: 유간 이상, 누수, 기울어짐 등 기능 저하
        impact_or_noise: 차량 통과 시 충격/이상음 발생 여부
        structural_damage: 본체 파손, 금속부재 탈락 등 구조적 손상
        other_damage: 기타 손상 (기준 외 항목 포함)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd'
    """
    grade = 'a'

    if structural_damage:
        return 'd'

    if impact_or_noise:
        grade = max(grade, 'd')

    if function_degradation:
        grade = max(grade, 'c')

    if aging_or_dirt:
        grade = max(grade, 'b')

    if other_damage:
        grade = max(grade, 'b')

    return grade
```

교면포장 상태평가

```
def evaluate_pavement_condition(damage_ratio=None,
                                puddle_present=False,
                                pavement_type='아스팔트'):
```

```
"""
교면포장 상태평가 등급(a~d)을 반환하는 함수

인자:
    damage_ratio: 포장불량률 (%)
        - 박리, 박락, 균열 등 손상 면적 비율
    puddle_present: 물고임 발생 여부 (True/False)
    pavement_type: 포장 종류 ('아스팔트' 또는 '콘크리트')

반환:
    상태평가 등급: 'a', 'b', 'c', 'd'

-----------------------------------------
✔ 포장불량 정의:
- 박리, 박락, 균열, 들뜸 등
- 시멘트 포장의 경우 파손, 충격으로 인한 파단도 포함
✔ 물고임(puddle_present=True):
- 발생 시 무조건 최소 b등급 이상
-----------------------------------------
"""
grade = 'a'

if pavement_type not in ['아스팔트', '콘크리트']:
    raise ValueError("pavement_type must be either '아스팔트' or '콘크리트'")

if damage_ratio is not None:
    if pavement_type == '아스팔트':
        if damage_ratio >= 10:
            grade = 'd'
        elif damage_ratio >= 5:
            grade = 'c'
        elif damage_ratio > 0:
            grade = 'b'
    elif pavement_type == '콘크리트':
        if damage_ratio >= 30:
            grade = 'd'
        elif damage_ratio >= 10:
            grade = 'c'
        elif damage_ratio > 0:
            grade = 'b'

if puddle_present:
    grade = max(grade, 'b')

return grade
```

배수시설 상태평가

```
def evaluate_drainage_facility(deposit_amount='none',
                               leakage=False,
                               corrosion_due_to_leakage=False,
                               outlet_risk=False,
                               damaged_or_aged=False):
    """
    배수시설 상태평가 등급(a~d)을 반환하는 함수

    인자:
        deposit_amount: 퇴적물 정도 ('none', 'some', 'many')
        leakage: 누수 여부
        corrosion_due_to_leakage: 누수로 인한 구조물 부식 여부
        outlet_risk: 배수관 유출구 위치로 인한 위험 여부
        damaged_or_aged: 파손 또는 노후화 상태 여부

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd'
    """
    if damaged_or_aged:
        return 'd'

    if leakage or corrosion_due_to_leakage or outlet_risk:
        return 'c'

    if deposit_amount == 'many':
        return 'c'
    elif deposit_amount == 'some':
        return 'b'
    else:
        return 'a'
```

난간 및 연석 상태평가

```
def evaluate_railing_or_curb(paint_damage_ratio=0,
                             local_looseness=False,
                             crack_width=None,
                             section_loss_ratio=0,
                             spalling_or_exposed_rebar_ratio=0,
                             rebar_corrosion_length_ratio=0,
```

```
                                    overturning_risk=False):
    """
    난간 및 연석 상태평가 등급(a~d)을 반환하는 함수

    인자:
        paint_damage_ratio: 도장 불량 비율 (%)
        local_looseness: 고정장치 국부 이완 여부
        crack_width: 균열폭 (mm)
        section_loss_ratio: 단면 손상/파손 비율 (%)
        spalling_or_exposed_rebar_ratio: 박리/철근노출 비율 (%)
        rebar_corrosion_length_ratio: 철근 부식 길이 비율 (%)
        overturning_risk: 전도 위험 여부

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd'
    """
    grade = 'a'

    if overturning_risk:
        return 'd'

    if section_loss_ratio >= 10 or spalling_or_exposed_rebar_ratio >= 10 or rebar_corrosion_length_ratio
>= 2:
        return 'd'

    if section_loss_ratio > 0 or spalling_or_exposed_rebar_ratio > 0 or rebar_corrosion_length_ratio > 0:
        grade = max(grade, 'c')

    if crack_width is not None:
        if crack_width >= 0.3:
            grade = max(grade, 'c')
        elif crack_width > 0:
            grade = max(grade, 'b')

    if paint_damage_ratio >= 10:
        grade = max(grade, 'c')
    elif paint_damage_ratio > 0:
        grade = max(grade, 'b')

    if local_looseness:
        grade = max(grade, 'b')

    return grade
```

탄산화 상태평가

```
def evaluate_carbonation(remaining_depth=None, rebar_corrosion_confirmed=False):
    """
    탄산화 상태평가 등급(a~e)을 반환하는 함수

    인자:
        remaining_depth: 철근까지 남은 탄산화 잔여 깊이 (mm)
        rebar_corrosion_confirmed: 철근부식 발생 여부 (True → e)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ⚠ 입력이 없을 경우 → 'a'
    """
    if remaining_depth is None:
        return 'a'

    if remaining_depth < 0:
        return 'e' if rebar_corrosion_confirmed else 'd'
    elif remaining_depth < 10:
        return 'c'
    elif remaining_depth < 30:
        return 'b'
    else:
        return 'a'
```

염화물 상태평가

```
def evaluate_chloride(total_chloride=None, rebar_corrosion_confirmed=False):
    """
    염화물 상태평가 등급(a~e)을 반환하는 함수

    인자:
        total_chloride: 전염화물 이온량 (kg/m³)
        rebar_corrosion_confirmed: 철근부식 발생 여부 (True → e)

    반환:
        상태평가 등급: 'a', 'b', 'c', 'd', 'e'

    ⚠ 입력이 없을 경우 → 'a'
    """
    if total_chloride is None:
        return 'a'

    if total_chloride >= 2.5:
        return 'e' if rebar_corrosion_confirmed else 'd'
    elif total_chloride >= 1.2:
        return 'c'
    elif total_chloride > 0.3:
        return 'b'
    else:
        return 'a'
```

구조형식에 따른 가중치 함수

```
# 구조형식별 가중치 사전 정의
STRUCTURE_WEIGHTS = {
    'PSC 박스거더교': {
        '바닥판': 20, '거더': 23, '교대/교각': 13, '기초': 7,
        '교량받침': 9, '신축이음': 9, '교면포장': 7, '배수시설': 3,
        '난간/연석': 2, '탄산화_상부': 2, '탄산화_하부': 2,
        '염화물_상부': 2, '염화물_하부': 1,
    },
    '강상형교': {
        '바닥판': 20, '강거더': 25, '교대/교각': 13, '기초': 7,
        '교량받침': 9, '신축이음': 9, '교면포장': 7, '배수시설': 3,
        '난간/연석': 2, '탄산화_상부': 1, '탄산화_하부': 1,
        '염화물_상부': 2, '염화물_하부': 2,
    },
    '라멘교': {
        '바닥판': 25, '교각': 20, '기초': 10,
        '신축이음': 10, '교면포장': 8, '난간/연석': 7,
    },
    '강박스거더교': {
        '바닥판': 20, '강거더': 25, '교대/교각': 13, '기초': 7,
        '교량받침': 9, '신축이음': 9, '교면포장': 7, '배수시설': 3,
        '난간/연석': 2, '탄산화_상부': 1, '탄산화_하부': 1,
        '염화물_상부': 2, '염화물_하부': 2,
    },
    'RC슬래브교': {
        '슬래브': 28, '교대/교각': 15, '기초': 7,
        '신축이음': 10, '교면포장': 10, '배수시설': 5,
        '난간/연석': 5, '탄산화_상부': 3, '탄산화_하부': 3,
        '염화물_상부': 2, '염화물_하부': 2,
    },
    '현수교': {
        '상판': 20, '주탑': 20, '케이블': 20, '보강형': 15, '기초': 5,
        '신축이음': 5, '교면포장': 5, '난간/연석': 5,
        '탄산화_상부': 2, '탄산화_하부': 2, '염화물_상부': 1, '염화물_하부': 1,
    },
    '사장교': {
        '상판': 22, '주탑': 20, '케이블': 18, '보강형': 15, '기초': 5,
        '신축이음': 5, '교면포장': 5, '난간/연석': 5,
        '탄산화_상부': 2, '탄산화_하부': 2, '염화물_상부': 1, '염화물_하부': 1,
    },
    '아치교': {
        '상부구조': 30, '활하중지지부': 25, '기초': 10,
        '신축이음': 10, '교면포장': 10, '난간/연석': 5,
        '탄산화_상부': 3, '탄산화_하부': 3, '염화물_상부': 2, '염화물_하부': 2,
    },
    # 필요시 구조형식 추가 가능
```

```
}

# 등급 → 점수 변환
def grade_to_defect_score(grade):
    score_map = {'a': 0.0, 'b': 1.0, 'c': 3.0, 'd': 7.0, 'e': 15.0}
    return score_map.get(grade.lower(), 0.0)

# 점수 → 종합 상태등급
def defect_score_to_grade(score):
    if score < 0.13:
        return 'A'
    elif score < 0.26:
        return 'B'
    elif score < 0.49:
        return 'C'
    elif score < 0.79:
        return 'D'
    else:
        return 'E'

# 구조형식별 환산결함도 계산
def calculate_structure_defect(component_grades, weight_table):
    total_weight = 0
    weighted_sum = 0
    for component, weight in weight_table.items():
        grade = component_grades.get(component, 'a')  # 입력 없으면 'a'
        score = grade_to_defect_score(grade)
        weighted_sum += score * weight
        total_weight += weight
    return round(weighted_sum / total_weight, 3) if total_weight else 0.0

# 구조형식 이름만 입력하면 자동으로 가중치 적용 + 전체 상태 평가까지 수행
def evaluate_bridge_total_condition(structure_data):
    result = {
        'structure_scores': [],
        'total_defect_score': 0.0,
        'total_grade': 'A'
    }

    total_length = sum(item['length'] for item in structure_data)
    weighted_score_sum = 0

    for item in structure_data:
        name = item['name']
        length = item['length']
        grades = item['component_grades']
        weight_table = STRUCTURE_WEIGHTS.get(name)
        if not weight_table:
```

```
            raise ValueError(f"지원하지 않는 구조형식: {name}")

        defect_score = calculate_structure_defect(grades, weight_table)
        length_ratio = length / total_length if total_length else 0
        weighted_score_sum += defect_score * length_ratio

        result['structure_scores'].append({
            'name': name,
            'length': length,
            'defect_score': round(defect_score, 3),
            'grade': defect_score_to_grade(defect_score),
            'length_ratio': round(length_ratio, 3)
        })

    result['total_defect_score'] = round(weighted_score_sum, 3)
    result['total_grade'] = defect_score_to_grade(weighted_score_sum)

    import pandas as pd
    import ace_tools as tools
    df = pd.DataFrame(result['structure_scores'])
    tools.display_dataframe_to_user(name="구조형식별 환산결함도 결과", dataframe=df)

    return result
```