🏋️ Exercise App 사용 가이드

> `plan.py` *알고리즘이 HTML 웹앱으로 어떻게 구현되었는지 설명하는 문서*

📁 파일 구조

파일	역할
`plan.py`	Python CLI 버전 - 알고리즘 원본 설계
`index.html`	웹앱 UI 구조
`app.js`	`plan.py` 알고리즘을 JavaScript로 포팅
`style.css`	스타일링

1️⃣ 데이터 관리

plan.py 원본

```
# 경로 설정
ROUTINES_FILE = BASE / "routines.json"  # 루틴 저장
RECORDS_FILE = BASE / "records.csv"      # 기록 저장

def load_routines():       # JSON 파일에서 루틴 로드
def load_records():        # CSV 파일에서 기록 로드
def save_record_row():     # CSV에 한 줄 추가
```

HTML 구현 (app.js)

```
const STORAGE_KEYS = {
    ROUTINES: 'exercise_routines',
    RECORDS: 'exercise_records'
};

function loadRoutines()      // localStorage에서 로드
function loadRecords()       // localStorage에서 로드
function saveRecord(record)  // localStorage에 저장 (업데이트 지원)
```

차이점:

- Python: 파일(JSON/CSV) 기반 저장
- HTML: 브라우저 `localStorage` 기반 저장 (서버 없이 동작)

2️⃣ 오늘 루틴 조회

plan.py 원본

```
def get_today_routine(routines, date=None):
    if date is None:
        date = datetime.now()
    weekday = date.strftime("%a")  # Mon, Tue, ...
    return {ex: v for ex, v in routines.items() if weekday in
v["days"]}
```

▶️ 오늘 요일에 해당하는 운동만 필터링

HTML 구현 (app.js)

```
function getTodayExercises(date) {
    const routines = loadRoutines();
    const weekday = getWeekday(date);  // ["Sun","Mon",...]
[date.getDay()]
    const result = {};
    for (const [key, info] of Object.entries(routines)) {
        if (info.days && info.days.includes(weekday)) result[key] =
info;
    }
    return result;
}
```

UI 반영:

- `#todayExerciseList` 에 운동 카드 렌더링
- 휴식일이면 "🎉 오늘은 휴식일!" 표시

3️⃣ 회복 필요 판단 알고리즘

plan.py 원본

```
def need_recovery(df: pd.DataFrame):
    reasons = []

    # 조건 A: 최근 14일 RPE 평균 ≥ 8
    if len(rpe_vals) >= 3 and rpe_vals.mean() >= 8:
        reasons.append("최근 RPE 평균이 높아 과부하 가능성이 있습니
다.")

    # 조건 B: 3일 이상 연속 운동
    if streak >= 3:
```

```
        reasons.append("3일 이상 연속 운동하여 회복이 필요할 수 있습니다.")

    # 조건 C: 최근 5회 중 3회 이상 미완료
    if (last5["done"] == "N").sum() >= 3:
        reasons.append("최근 수행률이 하락하여 회복이 필요할 수 있습니다.")

    return len(reasons) > 0, reasons
```

HTML 구현 (app.js)

```
function needRecovery() {
    const reasons = [];

    // 1. RPE 평균 체크 (동일)
    if (avgRpe >= 8) reasons.push("최근 RPE 평균이 높습니다...");

    // 2. 연속 운동일 체크 (4일로 상향)
    if (streak >= 4) reasons.push(`${streak}일 연속 운동 중!...`);

    return { need: reasons.length > 0, reasons };
}
```

UI 반영:

- 상단에 노란색 `#recoveryBanner` 경고 배너 표시
- 첫 번째 이유만 표시 (간결함 유지)

4️⃣ 추천 시스템 (Rule-based)

plan.py 원본

```python
def recommend_exercise(df: pd.DataFrame, routines: dict):
    recs = []

    # 상체/하체 균형 분석
    if upper < lower * 0.6:
        recs.append("상체 운동 비중이 낮습니다. 푸시업/풀업 추가 권장.")

    # 특정 운동 집중 경고
    if counts.max() > max(3, int(counts.mean() * 2)):
        recs.append(f"{most} 비중이 높음 → 유사 대체 운동 추가 권장.")

    # RPE 기반 저강도 권장
    if recent_rpe.mean() >= 8:
        recs.append("최근 RPE가 높습니다. 저강도 권장.")

    return recs
```

HTML 구현 (app.js)

```javascript
function getRecommendations() {
    const recs = [];

    // 1. 상체/하체 균형 (양방향 확장)
    if (upper < lower * 0.6) recs.push("💪 상체 운동 비중이 낮습니
다...");
    if (lower < upper * 0.6) recs.push("🦵 하체 운동 비중이 낮습니
다...");
    if (core < 3)            recs.push("🎯 코어 운동을 더 해보세
요...");
```

```
    // 2. RPE 양방향 (높/낮 모두 처리)
    if (avgRpe >= 8)  recs.push("😓 최근 강도가 높았습니다...");
    if (avgRpe <= 4)  recs.push("⚡ 강도를 조금 높여보는 건 어떨까
요?");

    // 3. 수행률 피드백 (추가)
    if (completionRate >= 0.8) recs.push("🔥 최근 수행률이 우수합니
다!");

    return recs;
}
```

UI 반영:

- 우측 `#recommendSection` 에 AI 추천 카드로 표시
- 이모지 추가로 시각적 구분

5️⃣ 스케줄 최적화

plan.py 원본

```
def optimize_schedule(df: pd.DataFrame, routines: dict):
    perf = df.groupby("weekday")["done"].apply(lambda x: (x ==
"Y").mean())

    # 수행률 낮은 요일 감지
    for d in low_days.index:
        suggestions.append(f"{day_names[d]}요일 수행률 낮음 → 강도 완
화 권장.")

    # 최적 시간대 분석
    best = int(time_perf.idxmax())
    suggestions.append(f"가장 효율적 시간대: {best}시")
```

```
    return suggestions
```

HTML 구현

- **직접 포팅되지 않음** (월간 보고서에서 일부 활용)
- 대신 `getWeeklyStats()` 로 주간 요일별 수행률 차트 제공

6️⃣ 시각화

plan.py 원본

```
def plot_weekday_heatmap(df, save_path):    # 요일별 수행률 히트맵
def plot_intensity_trend(df, save_path):    # 월별 강도 추이
def plot_stacked_volume(df, save_path):     # 월별 누적 운동량
def plot_monthly_adherence(df, save_path):  # 월간 수행률 추이
```

▶️ matplotlib으로 PNG 파일 생성

HTML 구현 (app.js + Chart.js)

```
function renderChart() {        // 이번 주 요일별 수행 횟수 (막대)
function renderReport() {       // 월간 주별 운동 횟수 (막대)
```

UI 반영:

- 메인: `#weeklyChart` - 주간 요일별 막대 차트
- 보고서: `#monthlyChart` - 월간 주별 막대 차트

7️⃣ 기본 루틴 데이터

plan.py 원본

```
sample = {
    "squat":   {"days": ["Mon","Wed","Fri"], "reps": 20,
"intensity": 3, "type": "lower"},
    "pushup":  {"days": ["Tue","Thu"],       "reps": 15,
"intensity": 2, "type": "upper"},
    "plank":   {"days": ["Mon","Thu"],       "reps": 60,
"intensity": 2, "type": "core", "unit": "sec"},
    "lunge":   {"days": ["Wed","Sat"],       "reps": 12,
"intensity": 3, "type": "lower"},
    "stretch": {"days": ["Sun"],             "reps": 10,
"intensity": 1, "type": "mobility"}
}
```

HTML 구현 (app.js)

```
const DEFAULT_ROUTINES = {
    squat:   { name: "스쿼트",   days: [...], reps: 20, intensity: 3,
type: "lower",    unit: "회" },
    pushup:  { name: "푸시업",   days: [...], reps: 15, intensity: 2,
type: "upper",    unit: "회" },
    plank:   { name: "플랭크",   days: [...], reps: 60, intensity: 2,
type: "core",     unit: "초" },
    lunge:   { name: "런지",     days: [...], reps: 12, intensity: 3,
type: "lower",    unit: "회" },
    stretch: { name: "스트레칭", days: [...], reps: 10, intensity: 1,
```

```
type: "mobility", unit: "분" }
};
```

변경점: `name` 필드 추가 → 한글 표시명 지원

🆕 plan.py에 없는 HTML 전용 기능

> *아래 기능들은 `plan.py` 에 구현되지 않았지만, 사용자 경험을 위해 웹앱에서 추가됨*

1. 날짜 선택 기능

```
<div class="date-picker">
    <button id="prevDate">◀</button>
    <input type="date" id="dateInput">
    <button id="nextDate">▶</button>
</div>
```

```
selectedDate.setDate(selectedDate.getDate() - 1);  // 이전날
selectedDate.setDate(selectedDate.getDate() + 1);  // 다음날
```

▶️ **plan.py는 오늘만 처리**, HTML은 **과거/미래 날짜 선택 가능**

---

2. 운동 기록 모달 (RPE 입력 UI)

```
<div class="modal" id="recordModal">
    <button id="btnDone">완료 ✓</button>
```

```
    <button id="btnNotDone">미완료</button>
    <input type="range" id="rpeSlider" min="1" max="10">
</div>
```

```
function saveCurrentRecord() {
    saveRecord({
        date: formatDateISO(selectedDate),
        exercise: selectedExercise,
        done: 'Y' or 'N',
        RPE: parseInt(rpeSlider.value),
        hour: new Date().getHours()
    });
}
```

▶️ **plan.py는 CLI 입력**, HTML은 **시각적 모달 UI**

---

3. 루틴 관리 (추가/삭제/요일 수정)

```
<div class="modal" id="manageModal">
    <button id="btnAddExercise">+ 새 운동 추가</button>
</div>
```

```
function saveNewExercise() {
    r[name.toLowerCase() + '_' + Date.now()] = {
        name, days, reps, unit, type, intensity
    };
```

```
    saveRoutines(r);
}
```

기능:

- ✅ 요일별 체크박스로 루틴 요일 수정
- ✅ 새 운동 추가 (이름/목표량/단위/부위/강도 설정)
- ✅ 운동 삭제 (× 버튼)

▶️ **plan.py는 JSON 파일 직접 수정 필요**, HTML은 **GUI로 편집**

---

4. 월간 보고서

```
<div class="modal" id="reportModal">
    <span id="reportMonthLabel">2024년 12월</span>
    <div id="reportSummary"></div>     <!-- 요약 카드 -->
    <canvas id="monthlyChart"></canvas> <!-- 주별 차트 -->
    <div id="reportDetails"></div>      <!-- 운동별 상세 -->
</div>
```

```
function getMonthlyStats(year, month) {
    return {
        totalDays,         // 운동한 날 수
        totalExercises,    // 총 운동 횟수
        avgRpe,            // 평균 RPE
        completionRate,    // 수행률 %
        exerciseStats,     // 운동별 완료 횟수
        weeklyData         // 주별 운동 횟수
    };
}
```

▶️ **plan.py는 4개의 개별 PNG 차트 생성**, HTML은 **통합 월간 리포트 모달**

5. 진행률 바

```
<div class="progress-card">
    <span id="progressText">0/0 완료</span>
    <div class="progress-bar">
        <div class="progress-fill" id="progressFill"></div>
    </div>
</div>
```

```
const done = dateRecords.filter(r => r.done === 'Y').length;
progressFill.style.width = `${(done / total) * 100}%`;
```

▶️ **plan.py에 없음** - 오늘의 진행 상황을 시각적으로 표시

---

6. 휴식일 표시

```
<div class="empty-state" id="emptyState">
    <span class="empty-icon">🎉</span>
    <p>오늘은 휴식일!</p>
</div>
```

▶️ **plan.py는 빈 리스트 출력**, HTML은 **친근한 휴식일 메시지**

---

7. 주간 수행률 차트

```
function getWeeklyStats() {
    const stats = [0, 0, 0, 0, 0, 0, 0];  // 월~일
    // 이번 주 완료 횟수 집계
    return stats;
}
```

▶️ **plan.py는 히트맵(2D)**, HTML은 **막대 차트(bar)**로 간소화

📊 기능 비교 요약

기능	plan.py	HTML 웹앱
오늘 루틴 조회	✅	✅
회복 필요 판단	✅	✅
추천 시스템	✅	✅ (확장)
스케줄 최적화	✅	❌
요일별 히트맵	✅	❌
월별 강도 추이	✅	❌
월별 운동량 차트	✅	⚠️ (주별로 변경)
날짜 선택	❌	✅
기록 입력 UI	❌ (CLI)	✅ (모달)
루틴 관리 GUI	❌ (JSON)	✅
월간 보고서	❌	✅
진행률 표시	❌	✅
휴식일 표시	❌	✅
주간 차트	❌	✅

🚀 사용 방법

1. **웹앱 실행**: `index.html` 을 브라우저에서 열기
2. **운동 기록**: 운동 카드 클릭 → 완료/미완료 선택 → RPE 입력 → 저장
3. **루틴 관리**: ⚙️ 버튼 → 요일 수정 또는 새 운동 추가
4. **보고서 확인**: 📊 버튼 → 월간 통계 확인
5. **과거 기록**: 날짜 선택기로 이전 날짜 확인/수정

이 문서는 `plan.py` 와 `app.js` 의 코드 비교를 기반으로 작성되었습니다.