인공지능플래닝
과제 1

과목명	인공지능플래닝
담당교수님	송동호 교수님
학 과	소프트웨어학과
학 번/ 이름	2019125038 양재원

1. 문제정의

본 과제의 목표는 200명의 학생을 6개의 반으로 나누는 것이다. 각 반은 33명(4개 반), 34명(2개 반)으로 구성되며, 단순 분할이 아닌 다양한 제약 조건을 충족해야 한다.

제약 조건은 다음과 같다:

1. **문제 아동은 같은 반에 배치하지 않음**
2. **비등교/따돌림 위험 학생은 챙겨주는 친구와 같은 반 배치**
3. **리더십 학생은 각 반에 최소 1명**
4. **피아노 연주 가능 학생을 균등하게 분배**
5. **성적(총점) 편차 최소화**
6. **비등교자 균등 분배**
7. **남녀 비율 균등**
8. **운동 능력 학생 균등 분배**
9. **전년도 학급이 너무 많이 겹치지 않도록 분산**
10. **클럽 활동 학생도 한 반에 몰리지 않도록 균등**

2. 사용 도구 및 환경:

개발 환경: Google Colab (Python 3.12)

라이브러리:

A. **OR-Tools (CP-SAT Solver)**
B. **Pandas, Numpy (데이터 처리)**

입력 데이터: 학급반편성 CSP 문제 입력파일.csv

출력 데이터: class_assignment.csv

3. 모델링 및 제약 조건

변수: x[i][c] = 학생 i가 반 c에 배정되면 1

기본 조건:

1. **각 학생은 반드시 한 반에만 배정**

2. 반 정원 (33명×4, 34명×2)

추가 조건:

나쁜관계 학생은 같은 반에 배정되지 않음, 좋은관계 학생은 반드시 같은 반 배정, 각 반에 리더십 학생 최소 1명, 성별, 피아노, 비등교, 운동, 전년도 학급, 클럽 분배를 균등하게 유지

목적: 반별 총점 편차 최소화

```
from ortools.sat.python import cp_model
```

```
import os
os.listdir()

from google.colab import files
uploaded = files.upload()

import pandas as pd
fname = next(iter(uploaded))
df = pd.read_csv(fname, encoding='utf-8-sig')
df.head()
```

파일 선택 학급반편성CSP...제 입력파일.csv
학급반편성CSP 문제 입력파일.csv(text/csv) - 8467 bytes, last modified: 2025. 9. 24. - 100% done
Saving 학급반편성CSP 문제 입력파일.csv to 학급반편성CSP 문제 입력파일 (1).csv

	id	name	sex	score	24년 학급	클럽	좋은관계	나쁜관계	Leadership	Piano	비등교	운동선호
0	202501	Andy	boy	90	a	노래	202502.0	202503.0	NaN	NaN	NaN	NaN
1	202502	Briada	girl	51	b	댄스	NaN	NaN	NaN	yes	NaN	NaN
2	202503	Charlie	boy	91	c	야구	NaN	NaN	yes	NaN	NaN	yes
3	202504	Kevin W	boy	60	f	미술	NaN	202677.0	NaN	NaN	NaN	NaN
4	202505	Matthew G	boy	96	f	댄스	NaN	NaN	NaN	NaN	yes	NaN

다음 단계: df 변수로 코드 생성 New interactive sheet

라이브러리와 데이터를 불러오고 cp모델을 생성해준다.

```
# Boolean 컬럼 처리
def to_bool(s):
    return str(s).strip().lower() == 'yes'

for col in ['Leadership', 'Piano', '비등교', '운동선호']:
    df[col] = df[col].apply(to_bool)

# 관계 id 처리
def id_or_none(v):
    try:
        if pd.isna(v): return None
        return int(v)
    except:
        return None

df['좋은관계'] = df['좋은관계'].apply(id_or_none)
df['나쁜관계'] = df['나쁜관계'].apply(id_or_none)

# 전년도/클럽 결측값 채우기
df['24년 학급'] = df['24년 학급'].fillna('미지정').astype(str)
df['클럽'] = df['클럽'].fillna('미지정').astype(str)

df.head()
```

	id	name	sex	score	24년 학급	클럽	좋은관계	나쁜관계	Leadership	Piano	비등교	운동선호
0	202501	Andy	boy	90	a	노래	202502.0	202503.0	False	False	False	False
1	202502	Briada	girl	51	b	댄스	NaN	NaN	False	True	False	False
2	202503	Charlie	boy	91	c	야구	NaN	NaN	True	False	False	True
3	202504	Kevin W	boy	60	f	미술	NaN	202677.0	False	False	False	False
4	202505	Matthew G	boy	96	f	댄스	NaN	NaN	False	False	True	False

제약식을 만들 때 타입이 명확해야 하고, 관계제약(같은 반/ 다른 반)에서 대상 학생을 인덱스로 찾기 쉽게 하기 위해 숫자화

```
n = len(df)                        # 학생 수 = 200
num_classes = 6
class_size = [33,33,33,33,34,34]

id_to_idx = {int(r.id): i for i, r in df.iterrows()}

# 균등 분배 범위 함수
def bounds(total, k=num_classes):
    return floor(total/k), ceil(total/k)
```

```
model = cp_model.CpModel()

# x[i][c] : 학생 i가 반 c에 배정되면 1
x = [[model.NewBoolVar(f'x_{i}_{c}') for c in range(num_classes)] for i in range(n)]

# (기본) 각 학생은 정확히 한 반
for i in range(n):
    model.Add(sum(x[i][c] for c in range(num_classes)) == 1)

# 각 반 정원 (33,33,33,33,34,34)
for c in range(num_classes):
    model.Add(sum(x[i][c] for i in range(n)) == class_size[c])
```

X [i] [c] 가 핵심 의사결정 변수이고, 두 개의 기본 제약(한 반만, 정원 맞추기)은 문제의 골격이다

```
# (1) 문제 아동 배치
for i, row in df.iterrows():
    bad_id = row['나쁜관계']
    if bad_id is not None and bad_id in id_to_idx:
        j = id_to_idx[bad_id]
        for c in range(num_classes):
            model.Add(x[i][c] + x[j][c] <= 1)

    if row['비등교']:
        good_id = row['좋은관계']
        if good_id is not None and good_id in id_to_idx:
            j = id_to_idx[good_id]
            for c in range(num_classes):
                model.Add(x[i][c] == x[j][c])

# (2) 리더십 최소 1명
leaders = df.index[df['Leadership']].tolist()
for c in range(num_classes):
    model.Add(sum(x[i][c] for i in leaders) >= 1)

# (3) 피아노 균등
pianos = df.index[df['Piano']].tolist()
lb, ub = bounds(len(pianos))
for c in range(num_classes):
    model.Add(sum(x[i][c] for i in pianos) >= lb)
    model.Add(sum(x[i][c] for i in pianos) <= ub)

# (4) 성적 균형
scores = df['score'].astype(int).tolist()
target = sum(scores) // num_classes
class_score = [model.NewIntVar(0, 100*40, f'score_{c}') for c in range(num_classes)]
for c in range(num_classes):
    model.Add(class_score[c] == sum(scores[i]*x[i][c] for i in range(n)))

dev = [model.NewIntVar(0, 100*40, f'dev_{c}') for c in range(num_classes)]
for c in range(num_classes):
    tmp = model.NewIntVar(-100*40, 100*40, f'diff_{c}')
    model.Add(tmp == class_score[c] - target)
    model.AddAbsEquality(dev[c], tmp)
dev_max = model.NewIntVar(0, 100*40, 'dev_max')
model.AddMaxEquality(dev_max, dev)

# (5) 비등교 균등
at_risk = df.index[df['비등교']].tolist()
lb, ub = bounds(len(at_risk))
for c in range(num_classes):
    model.Add(sum(x[i][c] for i in at_risk) >= lb)
    model.Add(sum(x[i][c] for i in at_risk) <= ub)
```

```python
    # (6) 남녀 균등
    boys = df.index[df['sex']=='boy'].tolist()
    girls= df.index[df['sex']=='girl'].tolist()
    lb_b, ub_b = bounds(len(boys))
    lb_g, ub_g = bounds(len(girls))
    for c in range(num_classes):
        model.Add(sum(x[i][c] for i in boys) >= lb_b)
        model.Add(sum(x[i][c] for i in boys) <= ub_b)
        model.Add(sum(x[i][c] for i in girls) >= lb_g)
        model.Add(sum(x[i][c] for i in girls) <= ub_g)

    # (7) 운동 균등
    sports = df.index[df['운동선호']].tolist()
    lb, ub = bounds(len(sports))
    for c in range(num_classes):
        model.Add(sum(x[i][c] for i in sports) >= lb)
        model.Add(sum(x[i][c] for i in sports) <= ub)

    # (8) 전년도 학급 분산
    for g, group_df in df.groupby('24년 학급'):
        idxs = group_df.index.tolist()
        lb, ub = bounds(len(idxs))
        for c in range(num_classes):
            model.Add(sum(x[i][c] for i in idxs) >= lb)
            model.Add(sum(x[i][c] for i in idxs) <= ub)

    # (9) 클럽 분산
    for g, group_df in df.groupby('클럽'):
        idxs = group_df.index.tolist()
        lb, ub = bounds(len(idxs))
        for c in range(num_classes):
            model.Add(sum(x[i][c] for i in idxs) >= lb)
            model.Add(sum(x[i][c] for i in idxs) <= ub)

    # 목적함수: 성적 편차 최소화
    model.Minimize(dev_max)
```

관계 제약 -> 학급 갈등 방지/안정성 핵심

리더/피아노 -> 반별 최소/균등 요구의 전형

성적 편차 최소화는 최종 공정성 지표

여러 특성을 고르게 섞는 것이 실제 학급 운영의 품질을 좌우

그룹별 분신을 전부 같은 형태로 통일해 모델이 깔끔하고 확장 가능

```python
assign = []
for i in range(n):
    c = max(range(num_classes), key=lambda k: solver.Value(x[i][k]))
    assign.append(c)

df_out = df.copy()
df_out['new_class'] = [a+1 for a in assign]

# 반별 요약
def cnt(mask, cls):
    idxs = df.index[mask].tolist()
    return sum(solver.Value(x[i][cls]) for i in idxs)

print("\n=== 반별 요약 ===")
for c in range(num_classes):
    male = cnt(df['sex']=='boy', c)
    female = cnt(df['sex']=='girl', c)
    lead = cnt(df['Leadership'], c)
    piano= cnt(df['Piano'], c)
    risk = cnt(df['비등교'], c)
    sport= cnt(df['운동선호'], c)
    sc = solver.Value(class_score[c])
    print(f'반 {c+1}: 인원 {class_size[c]}, 남 {male}, 여 {female}, 리더 {lead}, 피아노 {piano}, 위험 {risk}, 운동 {sport}

# CSV 저장
df_out.to_csv('class_assignment.csv', index=False)
print("\n저장 완료: class_assignment.csv")
```

```
=== 반별 요약 ===
반 1: 인원 33, 남 23, 여 10, 리더 1, 피아노 4, 위험 3, 운동 3, 총점 2632
반 2: 인원 33, 남 23, 여 10, 리더 5, 피아노 3, 위험 3, 운동 4, 총점 2630
반 3: 인원 33, 남 23, 여 10, 리더 5, 피아노 3, 위험 3, 운동 3, 총점 2631
반 4: 인원 33, 남 24, 여 9, 리더 3, 피아노 4, 위험 4, 운동 3, 총점 2631
반 5: 인원 34, 남 24, 여 10, 리더 3, 피아노 3, 위험 4, 운동 4, 총점 2632
반 6: 인원 34, 남 24, 여 10, 리더 3, 피아노 3, 위험 3, 운동 3, 총점 2630

저장 완료: class_assignment.csv
```

학생i마다 할당 된 c를 solver.value(x[i][c])로 찾고 기록, 반별로 보기 좋게 라벨해준 뒤 특정 조건의 학생 집합이 반에 몇명인지 카운트 해준다. 마지막으로 반별 요약 출력을 해준다.