Pythonを用いた

時間割作成プログラム

脇町高等学校

佐藤千夏

中井怜奈

枝澤明誠

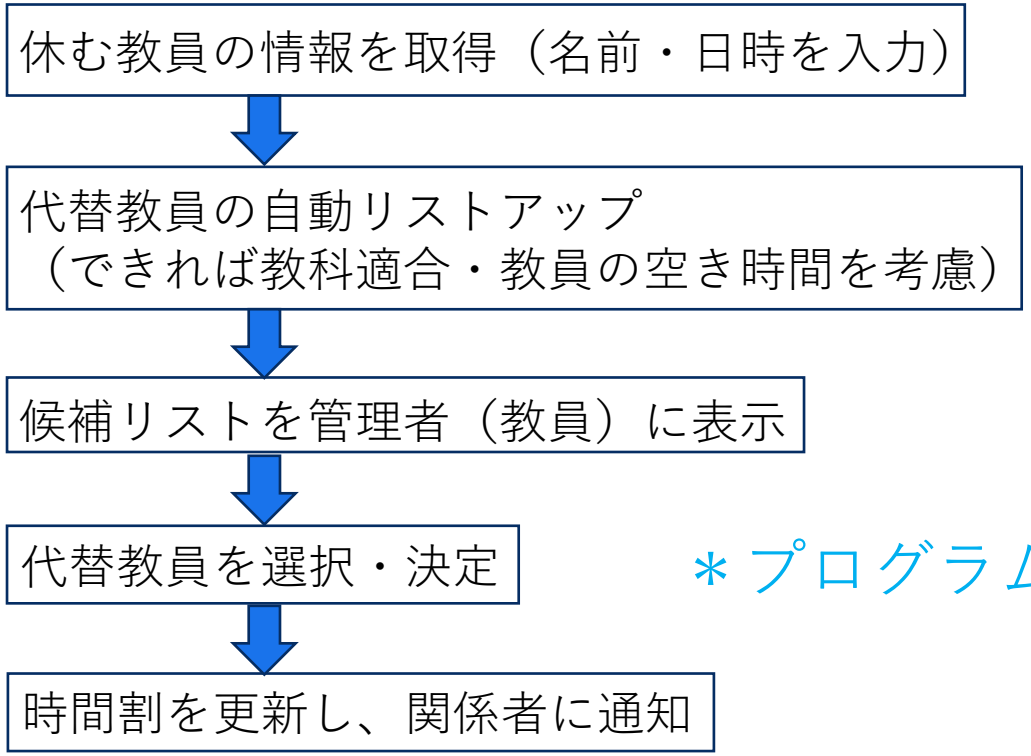
課題設定の理由

担任からの相談

「教員が欠席したときに時間割変更をするのだが、

いつも手作業で時間がかかる」

仮説



＊プログラムコードはChatGPTが作成

実装したい機能

①通常時に使う時間割の作成　→私たちが一週間に受ける単位数分の授業をランダムに並び替える
⇒「random」を用いて、授業の重複が起こらないように設定する

```
--- 21HR ---
月曜日: 論国２，体育２，英コミュ２，文学国語1，情報１，数Ⅱ１，芸術１
火曜日: 芸術２，論表３，情報１，数B/C１，古典１，公共２，情報２
水曜日: 地/化１，古典２，数B/C２，数Ⅱ１，保健１，公共２，論表３
木曜日: 数B/C３，論表３，情報２，芸術２，数B/C１，地/化２，英コミュ１
金曜日: 保健１，英コミュ３，古典２，英コミュ４，数Ⅱ３，英コミュ１，文学国語1
```

```
--- 22HR ---
月曜日: 社会１，論表３，数Ⅱ１，地/化２，保健１，文学国語1，体育１
火曜日: 論表２，Swing，英コミュ４，英コミュ１，数B/C２，古典１，地/化１
水曜日: 芸術１，文学国語1，文学国語2，体育２，論表３，体育１，情報２
木曜日: 情報１，体育１，社会２，社会１，Swing，数B/C１，古典１
金曜日: 論国１，数Ⅱ２，論表１，社会１，芸術１，体育２，論表２
```

```
--- 23HR ---
月曜日: 情報２，保健１，化基４，数B/C１，数Ⅱ２，論表３，論表２
火曜日: Swing，数B/C３，公共２，数B/C１，HR，化基４，英コミュ１
水曜日: 化基２，物生１，物生３，論表２，化基４，数Ⅱ２，英コミュ１
木曜日: 情報２，論表３，論表２，公共２，化基１，社会２，体育１
金曜日: 公共１，体育１，論国１，論表１，社会２，情報２，数B/C２
```

```
--- 24HR ---
月曜日: 数Ⅱ１，数B/C１，論表３，社会２，英コミュ１，論国１，英コミュ２
火曜日: 社会１，物生２，体育２，数Ⅱ１，物生３，英コミュ１，化基３
水曜日: 数Ⅱ２，英コミュ３，化基２，論国２，論表３，数B/C１，化基３
木曜日: Swing，英コミュ４，論表３，物生１，数B/C１，社会１，情報１
金曜日: 英コミュ２，物生３，英コミュ３，情報２，情報１，保健１，社会２
```

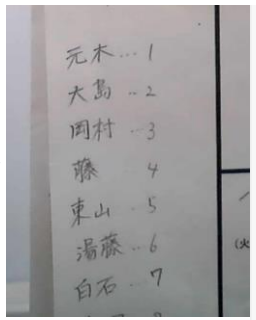
```
--- 25HR ---
月曜日: 社会１，芸/化２，地/化化２，情探２，数B/C２，数Ⅱ１，英コミュ３
火曜日: 数Ⅱ１，社会１，社会２，文/物生３，公共２，公共１，英コミュ１
水曜日: 情報１，地/化化１，英コミュ２，論国１，情探１，文/物生３，公共１
木曜日: 数Ⅱ３，古典１，地/化化２，数Ⅱ２，保健１，公共２，体育２
金曜日: 芸/化１，英コミュ１，芸/化２，論表２，体育２，数B/C２，公共１
```

②教員専用の時間割の作成　→個人が担当する教科と日時のみ表示できるようにする
⇒教員に番号（教員番号）を設定し、教科と対応させる

以前：教員の名前と対応→情報量が多い

```python
import random

# 各クラスの科目と担当教員
subjects_teachers = {
    '21HR': {
        '21HR論国１': ['元木先生'], '21HR論国２': ['元木先生'],
        '21HR文学国語1': ['元木先生'], '21HR文学国語2': ['元木先生'], '21HR文学国語3': ['元木先生
        '21HR古典１': ['大島先生'], '21HR古典２': ['大島先生'],
        '21HR社会１': ['岡村先生', '藤先生', '東山先生', '湯藤先生'], '21HR社会２': ['岡村先生',
        '21HR公共１': ['白石先生'], '21HR公共２': ['白石先生'],
        '21HR数Ⅱ１': ['辻岡先生'], '21HR数Ⅱ２': ['辻岡先生'], '21HR数Ⅱ３': ['辻岡先生'],
        '21HR数B/C１': ['奥村先生'], '21HR数B/C２': ['奥村先生'], '21HR数B/C３': ['奥村先生'],
```



現在：教員番号と対応→情報量少ない

```python
class_subjects = {
    '21HR': [
        '論国１:1', '論国２:1', '文学国語1:1', '文学国語２:1', '文学国語３:1',
        '古典１:2', '古典２:2', '社会１:3&4&5&6', '社会２:3&4&5&6', '公共１:7', '公共２:7',
        '数Ⅱ１:8', '数Ⅱ２:8', '数Ⅱ３:8', '数B/C１:9', '数B/C２:9', '数B/C３:9',
        '地/化１:10&11', '地/化２:10&11', '体育１:12', '体育２:12', '保健１:13',
        '英コミュ１:15', '英コミュ２:15', '英コミュ３:15', '英コミュ４:15',
        '論表１:16', '論表２:16', '論表３:16', '情報１:17', '情報２:17',
        'Swing:19', 'HR:7', '芸術１:19&20&21', '芸術２:19&20&21'
    ],
```

実装したい機能

例）教員専用の時間割　　教員番号1・2

```
--- 教員番号: 1 ---
月曜日: 21HR - 論国２（1限）, 21HR - 文学国語1（7限）, 25HR - 文/物生１（2限）, 25HR - 文/物生２（5限）
火曜日: 22HR - 論国１（1限）, 22HR - 文学国語２（3限）, 25HR - 文/物生３（6限）
水曜日: 21HR - 文学国語２（1限）, 21HR - 論国１（3限）, 22HR - 文学国語３（1限）
木曜日: 21HR - 文学国語３（3限）, 22HR - 文学国語1（6限）
金曜日: 22HR - 論国２（5限）

--- 教員番号: 2 ---
月曜日: 21HR - 古典１（2限）, 23HR - 論国１（4限）
火曜日: 23HR - 古典２（1限）, 24HR - Swing（1限）, 24HR - 論国２（5限）
水曜日: 23HR - 論国２（2限）, 24HR - 論国１（1限）, 24HR - 古典１（6限）
木曜日: 21HR - 古典２（6限）, 22HR - 古典２（7限）, 25HR - 古典１（3限）
金曜日: 22HR - 古典１（7限）, 23HR - 古典１（1限）, 24HR - 古典２（7限）, 25HR - 古典２（1限）, 25HR - 論国２（3限）, 25HR - 論国１（5限）
```

③代わりの先生の候補をあげる　→　条件を設けることで候補を絞る
⇒「同じ曜日・時間に授業を行う教員は候補から除外する」などの条件を設ける

例）「同じ曜日・時間に授業を行う教員は候補から除外する」のコード

```python
# 教師がその時間に授業を持っていないか確認
if teacher not in teacher_schedule:
    teacher_schedule[teacher] = {}

if day not in teacher_schedule[teacher]:
    teacher_schedule[teacher][day] = set()

# その教師が既に同じ曜日・時間に授業を持っていたらエラー
if period in teacher_schedule[teacher][day]:
    raise Exception(f"Teacher {teacher} already has a class on {day} during period {period}")
```

④③で絞られた候補の中から、代わりの教員を決定する　→　人が行う

例）教員番号1が水曜日に一日中欠席する
→22HR２・６限、25HR2限の代替候補の表示

```
削除する教員番号を入力してください: 1
対象の曜日を入力してください（例: 月曜日）: 水曜日
水曜日 2限（22HR 論国１）の代替候補: 22, 26, 8, 9
水曜日 6限（22HR 文学国語1）の代替候補: 16, 2, 7, 8
水曜日 2限（25HR 文/物生２）の代替候補: 8, 9
```

⑤①～④で変更した内容を全ての教員が見られるようにし、共有できるようにする

プログラムコード

```python
import random
from concurrent.futures import ThreadPoolExecutor, as_completed

days_of_week = ['月曜日', '火曜日', '水曜日', '木曜日', '金曜日']
periods_per_day = 7  # 1日の授業コマ数
total_periods = len(days_of_week) * periods_per_day  # 1週間の合計コマ数

# 各クラスの科目データ
class_subjects = {
    '21HR': [
        '論国１:1', '論国２:1', '文学国語1:1', '文学国語２:1', '文学国語３:1',
        '古典１:2', '古典２:2', '社会１:3&4&5&6', '社会２:3&4&5&6', '公共１:7', '公共２:7',
        '数Ⅱ１:8', '数Ⅱ２:8', '数Ⅱ３:8', '数B/C１:9', '数B/C２:9', '数B/C３:9',
        '地/化１:10&11', '地/化２:10&11', '体育１:12', '体育２:12', '保健１:13',
        '英コミュ１:15', '英コミュ２:15', '英コミュ３:15', '英コミュ４:15',
        '論表１:16', '論表２:16', '論表３:16', '情報１:17', '情報２:17',
        'Swing:19', 'HR:7', '芸術１:19&20&21', '芸術２:19&20&21'
    ],
    '22HR': [
        '論国１:1', '論国２:1', '文学国語1:1', '文学国語２:1', '文学国語３:1',
        '古典１:2', '古典２:2', '社会１:3&4&5&6', '社会２:3&4&5&6', '公共１:7', '公共２:7',
        '数Ⅱ１:8', '数Ⅱ２:8', '数Ⅱ３:8', '数B/C１:9', '数B/C２:9', '数B/C３:9',
        '地/化１:10&11', '地/化２:10&11', '体育１:12', '体育２:12', '保健１:13',
        '英コミュ１:15', '英コミュ２:15', '英コミュ３:15', '英コミュ４:15',
        '論表１:16', '論表２:16', '論表３:16', '情報１:17', '情報２:17',
        'Swing:19', 'HR:7', '芸術１:19&20&21', '芸術２:19&20&21'
    ],
}
```

import random
時間割作成時の重複をなくす

class_subjects
一週間に受ける授業のコマ数を設定する
（教科と教員番号を対応させる）

```python
def create_timetable(subjects):
    """科目をすべて使用し、「空き」がない時間割を作成"""
    timetable = {day: [] for day in days_of_week}
    subject_pool = subjects[:]
    random.shuffle(subject_pool)

    while len(subject_pool) < total_periods:
        subject_pool += random.sample(subjects, min(len(subjects), total_periods - len(subject_pool)))

    index = 0
    for day in days_of_week:
        for _ in range(periods_per_day):
            timetable[day].append(subject_pool[index])
            index += 1

    return timetable

def generate_timetables():
    """全クラスの時間割を生成"""
    all_timetables = {}

    with ThreadPoolExecutor() as executor:
        future_to_hr = {
            executor.submit(create_timetable, subjects): hr
            for hr, subjects in class_subjects.items()
        }

        for future in as_completed(future_to_hr):
            hr = future_to_hr[future]
            all_timetables[hr] = future.result()

    return all_timetables
```

def create_timetable(subjects)
class_subjectsで設定した教科を必ず一度だけ用いて、
かつrandomに並べる

def generate_timetable()
as_completed(future_to_hr)を使い、各クラスの時間割作成が完了した順に処理を進める
ここではif文はないが、非同期処理が完了したものから結果を取得している

プログラムコード

```python
def get_teacher_schedule(class_timetables):
    """各教員が担当する授業を曜日・時間ごとに記録"""
    teacher_schedule = {day: {period: set() for period in range(periods_per_day)} for day in days_of_week}

    for hr, timetable in class_timetables.items():
        for day, subjects in timetable.items():
            for period, subject in enumerate(subjects):
                if ':' in subject:
                    _, teachers = subject.split(':')
                    teacher_list = teachers.split('&')
                    for teacher in teacher_list:
                        teacher_schedule[day][period].add(teacher)

    return teacher_schedule
def remove_teacher_from_timetable_specific_day(class_timetables, teacher_id, target_day, teacher_schedule):
    """特定の曜日にある教員が担当する授業を「空き」に変更し、代替教員候補を表示"""
    modified_timetables = {}

    for hr, timetable in class_timetables.items():
        modified_timetables[hr] = {}

        for day, subjects in timetable.items():
            modified_subjects = []

            for period, subject in enumerate(subjects):
                if ':' in subject:
                    name, teachers = subject.split(':')
                    teacher_list = teachers.split('&')

                    if day == target_day and teacher_id in teacher_list:
                        # 教員を削除
                        teacher_list.remove(teacher_id)

                        # 「空き」を設定（他の教員が残っている場合はそのまま）
                        if not teacher_list:
                            teacher_list.append("空き")

                        # 代替教員候補を探す
                        available_teachers = teacher_schedule[day][period] - set(teacher_list)
                        available_teachers.discard(teacher_id)  # 削除対象の教員を除外

                        # 代替候補を表示
                        replacement_candidates = ', '.join(sorted(available_teachers)) if available_teachers else 'なし'
                        print(f"{day} {period+1}限 ({hr} {name}) の代替候補: {replacement_candidates}")

                    modified_subjects.append(f"{name}:{'&'.join(teacher_list)}")

            modified_timetables[hr][day] = modified_subjects

    return modified_timetables
def display_timetables(timetables):
    """時間割を整形して表示"""
    for hr, timetable in timetables.items():
        print(f"¥n{hr} の時間割:")
        for day, subjects in timetable.items():
            print(f"{day}: {', '.join(subjects)}")

# 1回だけ時間割を生成し、結果を保存
final_timetables = generate_timetables()
display_timetables(final_timetables)

# 教員の担当状況を取得
teacher_schedule = get_teacher_schedule(final_timetables)

# 教員番号と曜日を指定して授業を削除
teacher_id_to_remove = input("¥n削除する教員番号を入力してください: ")
target_day = input("対象の曜日を入力してください（例: 月曜日）: ")

if target_day not in days_of_week:
    print("入力された曜日が無効です。正しい曜日を入力してください。")
else:
    updated_timetables = remove_teacher_from_timetable_specific_day(final_timetables, teacher_id_to_remove, target_day, teacher_schedule)
    print("¥n=== 更新後の時間割 ===")
    display_timetables(updated_timetables)
```

If':'in subject:

subjectに：が含まれる場合のみ、科目名と教員番号を分割
split(':')で科目名と教員を分け、split('＆')で複数の教員が担当する場合にも対応　（例　地/化：10＆11）
この分岐がないと、教員が記載されていない科目でエラーが出る可能性がある

If day==target_day and teacher_id in teacher_list

指定された曜日（target_day)であり、かつ担当する教員（teacher_id）がいる場合のみ削除を実行
この分岐がないと、無関係な曜日や教員の授業まで変更されてしまう

If not teacher_list：

教員を削除した後、その科目を担当する教員が一人もいなくなった場合「空き」として登録する
これがないと、教員がいなくなった科目がエラーになってしまう可能性がある

If target_day not in days_of_week:

入力された曜日(target_day)がdays_of_weekに含まれていなければ、エラーメッセージを表示
これがないと、誤った曜日を入力した場合に処理が壊れる可能性がある

実験結果

元の時間割

21HR の時間割:
月曜日: 芸術１:19&20&21, 保健１:13, 英コミュ２:15, 文学国語1:1, 情報１:17, Swing:19, 論国１:1
火曜日: 英コミュ４:15, 文学国語２:1, 論表１:16, 数Ⅱ２:8, 情報２:17, 論国２:1, 英コミュ３:15
水曜日: 論表３:16, 数B/C３:9, 公共２:7, 数Ⅱ３:8, 数B/C２:9, HR:7, 芸術２:19&20&21
木曜日: 数Ⅱ１:8, 古典２:2, 論表２:16, 地/化１:10&11, 文学国語３:1, 地/化２:10&11, 社会２:3&4&5&6
金曜日: 古典１:2, 社会１:3&4&5&6, 体育２:12, 英コミュ１:15, 体育１:12, 数B/C１:9, 公共１:7

22HR の時間割:
月曜日: 古典１:2, 情報１:17, 体育２:12, 社会１:3&4&5&6, 数B/C１:9, 英コミュ１:15, 数B/C２:9
火曜日: 数Ⅱ２:8, 論表３:16, 論国２:1, 文学国語３:1, 英コミュ３:15, 論表１:16, 地/化１:10&11
水曜日: 英コミュ２:15, 論国１:1, Swing:19, 芸術２:19&20&21, 芸術１:19&20&21, 文学国語1:1, HR:7
木曜日: 古典２:2, 公共２:7, 保健１:13, 社会２:3&4&5&6, 体育１:12, 地/化２:10&11, 公共１:7
金曜日: 文学国語２:1, 数B/C３:9, 英コミュ４:15, 数Ⅱ１:8, 情報２:17, 数Ⅱ３:8, 論表２:16

23HR の時間割:
月曜日: 物生２:22&23, 保健１:13, 英コミュ３:16, 化基３:11, HR:9, 英コミュ１:16, 化基４:11
火曜日: 英コミュ２:16, 論国１:2, 数B/C４:8, 古典２:2, 情報１:17, 体育１:12, 公共２:5
水曜日: 論表１:15, 数B/C２:9, 社会２:4&6, 数B/C３:9, 化基２:11, 数Ⅱ１:8, 物生３:22&23
木曜日: 物生１:22&23, 社会１:4&6, 体育２:12, 論国２:2, 英コミュ４:16, 数Ⅱ２:8, 論表２:15
金曜日: 情報２:17, 数B/C１:9, 公共１:5, 論表３:15, Swing:16, 古典１:2, 化基１:11

24HR の時間割:
月曜日: 古典２:2, 物生２:23&24, 物生１:23&24, 情報１:17&18, 古典１:2, 論表３:15, 英コミュ３:16
火曜日: 情報２:17&18, 数B/C４:8, 社会１:3&5&6&7, 英コミュ２:16, 物生３:23&24, 保健１:13, 公共２:5
水曜日: Swing:2, 数Ⅱ１:8, 化基２:25, 数B/C３:8, 公共１:5, 英コミュ１:16, 社会２:3&5&6&7
木曜日: 数Ⅱ２:8, 数B/C１:8, 論表１:15, 化基４:25, 論国１:2, 論表２:15, 体育１:12&13&14
金曜日: 化基３:25, 論国２:2, HR:17, 数B/C２:8, 英コミュ４:16, 体育２:12&13&14, 化基１:25

25HR の時間割:
月曜日: 英コミュ２:15, 公共２:7, 論表１:16, 地/化１:10&11&27, 論国１:2, 数Ⅱ３:8, HR:15
火曜日: 社会２:6&5&3&7, 英コミュ３:15, 数B/C１:9, 論表２:16, 数B/C２:9, 保健１:13, 芸術２:19&20&21&27
水曜日: 情報２:17, 文/物生２:1&22&26, 英コミュ４:15, 数Ⅱ１:8, 社会１:6&5&3&7, 古典２:2, 体育１:12&13&14
木曜日: 数Ⅱ２:8, 文/物生３:1&22&26, 情探１:24, 情探２:24, 情報１:17, 体育２:12&13&14, 英コミュ１:15
金曜日: 論国２:2, 古典１:2, 地/化２:10&11&27, 文/物生１:1&22&26, 数B/C３:9, 公共１:7, 芸術１:19&20&21&27

削除する教員番号を入力してください: 1
対象の曜日を入力してください（例: 月曜日）: 水曜日
水曜日 2限 (22HR 論国１) の代替候補: 22, 26, 8, 9
水曜日 6限 (22HR 文学国語1) の代替候補: 16, 2, 7, 8
水曜日 2限 (25HR 文/物生２) の代替候補: 8, 9

=== 更新後の時間割 ===

21HR の時間割:
月曜日: 芸術１:19&20&21, 保健１:13, 英コミュ２:15, 文学国語1:1, 情報１:17, Swing:19, 論国１:1
火曜日: 英コミュ４:15, 文学国語２:1, 論表１:16, 数Ⅱ２:8, 情報２:17, 論国２:1, 英コミュ３:15
水曜日: 論表３:16, 数B/C３:9, 公共２:7, 数Ⅱ３:8, 数B/C２:9, HR:7, 芸術２:19&20&21
木曜日: 数Ⅱ１:8, 古典２:2, 論表２:16, 地/化１:10&11, 文学国語３:1, 地/化２:10&11, 社会２:3&4&5&6
金曜日: 古典１:2, 社会１:3&4&5&6, 体育２:12, 英コミュ１:15, 体育１:12, 数B/C１:9, 公共１:7

22HR の時間割:
月曜日: 古典１:2, 情報１:17, 体育２:12, 社会１:3&4&5&6, 数B/C１:9, 英コミュ１:15, 数B/C２:9
火曜日: 数Ⅱ２:8, 論表３:16, 論国２:1, 文学国語３:1, 英コミュ３:15, 論表１:16, 地/化１:10&11
水曜日: 英コミュ２:15, 論国１:空き, Swing:19, 芸術２:19&20&21, 芸術１:19&20&21, 文学国語1:空き, HR:7
木曜日: 古典２:2, 公共２:7, 保健１:13, 社会２:3&4&5&6, 体育１:12, 地/化２:10&11, 公共１:7
金曜日: 文学国語２:1, 数B/C３:9, 英コミュ４:15, 数Ⅱ１:8, 情報２:17, 数Ⅱ３:8, 論表２:16

23HR の時間割:
月曜日: 物生２:22&23, 保健１:13, 英コミュ３:16, 化基３:11, HR:9, 英コミュ１:16, 化基４:11
火曜日: 英コミュ２:16, 論国１:2, 数B/C４:8, 古典２:2, 情報１:17, 体育１:12, 公共２:5
水曜日: 論表１:15, 数B/C２:9, 社会２:4&6, 数B/C３:9, 化基２:11, 数Ⅱ１:8, 物生３:22&23
木曜日: 物生１:22&23, 社会１:4&6, 体育２:12, 論国２:2, 英コミュ４:16, 数Ⅱ２:8, 論表２:15
金曜日: 情報２:17, 数B/C１:9, 公共１:5, 論表３:15, Swing:16, 古典１:2, 化基１:11

24HR の時間割:
月曜日: 古典２:2, 物生２:23&24, 物生１:23&24, 情報１:17&18, 古典１:2, 論表３:15, 英コミュ３:16
火曜日: 情報２:17&18, 数B/C４:8, 社会１:3&5&6&7, 英コミュ２:16, 物生３:23&24, 保健１:13, 公共２:5
水曜日: Swing:2, 数Ⅱ１:8, 化基２:25, 数B/C３:8, 公共１:5, 英コミュ１:16, 社会２:3&5&6&7
木曜日: 数Ⅱ２:8, 数B/C１:8, 論表１:15, 化基４:25, 論国１:2, 論表２:15, 体育１:12&13&14
金曜日: 化基３:25, 論国２:2, HR:17, 数B/C２:8, 英コミュ４:16, 体育２:12&13&14, 化基１:25

25HR の時間割:
月曜日: 英コミュ２:15, 公共２:7, 論表１:16, 地/化１:10&11&27, 論国１:2, 数Ⅱ３:8, HR:15
火曜日: 社会２:6&5&3&7, 英コミュ３:15, 数B/C１:9, 論表２:16, 数B/C２:9, 保健１:13, 芸術２:19&20&21&27
水曜日: 情報２:17, 文/物生２:22&26, 英コミュ４:15, 数Ⅱ１:8, 社会１:6&5&3&7, 古典２:2, 体育１:12&13&14
木曜日: 数Ⅱ２:8, 文/物生３:1&22&26, 情探１:24, 情探２:24, 情報１:17, 体育２:12&13&14, 英コミュ１:15
金曜日: 論国２:2, 古典１:2, 地/化２:10&11&27, 文/物生１:1&22&26, 数B/C３:9, 公共１:7, 芸術１:19&20&21&27

エラーの種類

＊多い順に①→②→③　　⇒…改善方法を示す矢印

①論理エラー（Logic Error）の例

```
25HR の時間割:
月曜日: 英コミュ１:15, 情探１:24, 英コミュ２:15, 保健１:13, 地/化２:10&11&27, 英コミュ４:15, 論表２:16
火曜日: 文/物生１:1&22&26, 英コミュ３:15, 数B/C２:9, 公共１:7, 社会１:6&5&3&7, 論表１:16, 数B/C３:9
水曜日: 文/物生２:1&22&26, 論国１:2, 芸術１:19&20&21&27, HR:15, 数Ⅱ１:8, 公共２:7, 数Ⅱ２:8
木曜日: 数Ⅱ３:8, 情探２:24, 体育２:12&13&14, 情報２:17, 芸術２:19&20&21&27, 社会２:6&5&3&7, 文/物生３:1&22&26
金曜日: 地/化１:10&11&27, 数B/C１:9, 論国２:2, 情報１:17, 古典１:2, 体育１:12&13&14, 古典２:2

削除する教員番号を入力してください: 1
```

⇒

```
25HR の時間割:
月曜日: 英コミュ１:15, 情探１:24, 英コミュ２:15, 保健１:13, 地/化２:10&11&27, 英コミュ４:15, 論表２:16
火曜日: 文/物生１:空き&空き&空き, 英コミュ３:15, 数B/C２:9, 公共１:7, 社会１:6&5&3&7, 論表１:16, 数B/C３:9
水曜日: 文/物生２:空き&空き&空き, 論国１:2, 芸術１:19&20&21&27, HR:15, 数Ⅱ１:8, 公共２:7, 数Ⅱ２:8
木曜日: 数Ⅱ３:8, 情探２:24, 体育２:12&13&14, 情報２:17, 芸術２:19&20&21&27, 社会２:6&5&3&7, 文/物生３:空き&空き&空き
金曜日: 地/化１:10&11&27, 数B/C１:9, 論国２:2, 情報１:17, 古典１:2, 体育１:12&13&14, 古典２:2
```

⇒プログラムコードに問題箇所がないか探す

②構文エラー（Syntax Error)の例

```
  File "<ipython-input-20-04e4b80fed39>", line 81
    print(f"{day}: {', '.join([f'{subject}({", ".join(teacher)})' for subject, teacher in schedule])}")
                                                                  ^
SyntaxError: f-string: unterminated string
```

✖
```
f'{subject}({", ".join(teacher)})'
```

○
```
print(f"{day}: {', '.join([f'{subject}({", ".join(teacher)})' for subject, teacher in schedule])}")
```

もしくは

```
print(f'{day}: {", ".join([f"{subject}({", ".join(teacher)})" for subject, teacher in schedule])}')
```

この中にある”,”.join(teacher)の””（ダブルクオート）が、
外側の“”（f文字列）のダブルクオートと混ざってしまっている
→どこで文字列が終わるのかわからない
⇒区別できるように、’（シングルクオート）を用いる
（内側と外側どちらを変えてもいい）

エラーの種類

③実行時エラー（Runtime Error）の例

```
--- 21HR ---
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-d3424c5aa62e> in <cell line: 106>()
    106 for hr_class, class_schedule in schedule.items():
    107     print(f"--- {hr_class} ---")
--> 108     for (day, period), class_info in sorted(class_schedule.items()):
    109         print(f"{day} {period}時間目: {class_info}")

TypeError: '<' not supported between instances of 'tuple' and 'str'
```

✖
```
sorted(class_schedule.items())
```

○
```
class_schedule = {
    ("Monday", 1): "数学",
    ("Monday", 2): "英語",
    ("Tuesday", 1): "理科"  # ← 文字列ではなくタプルに修正！
}
```

もしくは

```
sorted(class_schedule.items(), key=lambda x: x[0] if isinstance(x[0], tuple) else (x[0], 0))
```

class_scheduleのキー（（day,period)の部分）が全て同じ型になっていない
→Pythonがそれらを比較できない
⇒キーのデータ型を統一する
　または、ソート時にキーを統一する

補足説明
class_schedule.items()という辞書の要素（キーと値のペア）をsorted()で並び替えようとしている
→Pythonはsorted()を使うとき、デフォルトで「キー同士を＜で比較して順番を決める」という動作をする
⇒しかし、比較するキーにタプル(("Monday",1))と文字列("Monday")の二種類が混ざっていると、
　Pythonは「タプルと文字列を比較できない…」となり、TypeErrorを発生させる

違う例）

```
class_schedule = {
    ("Monday", 1): "数学",
    ("Monday", 2): "英語",
    ("Tuesday", 1): "理科"
}  # ← すべてキーがタプル
```

```
class_schedule = {
    ("Monday", 1): "数学",
    ("Monday", 2): "英語",
    "Tuesday": "理科"  # ← ここだけキーが文字列（タプルじゃない！）
}
```

`("Monday", 1) < ("Tuesday", 2)` → OK（タプル同士の比較）

`("Monday", 1) < "Tuesday"` → NG（タプルと文字列の比較はできない）

今後の計画

短期目標

求めている結果を出す条件にできなかった
（例　時間割作成時、合同教科への考慮　　合同教科…2クラス合同で同じ時間帯に行う教科）
→今まで設定した条件の組み合わせを変える

長期目標

・他学年（1・3年）も設定に加える
・関係者全員がいつでもアクセスできる状態にする

参考文献

・萩谷昌己ほか10名　高校情報Ⅰ Python　実教出版株式会社　2024年
・森巧尚　Python 1年生 体験してわかる！会話でまなべる！プログラミングのしくみ　翔泳社　2017年
・クジラ飛行机　生成AI・ChatGPTでPythonプログラミングアウトプットを10倍にする！
　ソシム株式会社　2023年
・ChatGPT　https://openai.com/ja-JP/chatgpt/overview/