# プログラミング基礎II 第3回レポート課題
# 例外処理・ファイル処理

レポート提出期限：2026年2月10日
提出先：学務情報システム

## 課題1：テキストファイルの処理・例外処理

下の入出力例に示すように，テキストファイルに保存されている学生の在籍番号，名前，指導教員のデータを読み込んで，名前を昇順にソートした結果を別のテキストファイルに保存するプログラムを作成せよ．なお，入出力例で学生のデータを記録したテキストファイル‘student_list.txt’（ファイルはHPからダウンロードすること）を用い，出力ファイル‘output.txt’と生成すること．また，以下の条件を満たすこと．

1. 入力ファイルは各行がカンマ（,）で区切られた形式である（例：在籍番号, 氏名, 指導教員）．ファイルの読み込み・書き込み失敗時の対処として適切な例外処理を実装すること．
2. 学生の情報は以下のようなStudent型のクラスを作成してプログラム内で扱うこと．

```
class Student:
    def __init__(self, id='', name='', supervisor=''):
      self._id         = id          # 在籍番号
      self._name       = name        # 氏名
      self._supervisor = supervisor  # 指導教員

    @property  # Getter
    def name(self):
        return self._name

    @property  # Getter
    def id(self):
        return self._id

    @property  # Getter
    def supervisor(self):
        return self._supervisor

    @property  # Getter
    def makelist(self):
        # ID:12文字, 名前:20文字, 教員:20文字 分のスペースを確保
        fmt = "{:<12}  {:<20}  {:<20}\n"
        formatted_line = (
            fmt.format(self._id, self._name, self._supervisor))
        return [formatted_line]

    def __str__(self):
        str_  = (self._name + ' [ID：{}, Supervisor：{}]'.
                 format(self._id, self._supervisor))
        return str_
        return str_
```

3. テキストファイル student_list.txt の内容を Student 型のリストに格納する．このリストを引数としてデータを名前の昇順にソートする以下のような関数を作成し，これを用いること．名前の文字列における大小関係の比較には，大小比較演算子（>）を用いること．

   ```
   def sort_name (Student_list)
   ```

4. ソートするアルゴリズムとして，在籍番号の 3 桁の数を $x$ として 3 で割ったあまり $x$%3 に該当する番号のものを用いること．
   ・0：挿入ソート（insertion sort)
   ・1：交換ソート（bubble sort）
   ・2：選択ソート (selection sort)
   （アルゴリズムは，第 2 回レポートを参考にすること.）

5. 提出物

   - 以下の実行例が再現されるようにプログラムを作成しソースコード Student.py（py ファイル）と出力結果 output.txt（txt ファイル）
     ※ ファイルは複数になってもよい．複数ファイルは tar でまとめてもよい.
   - プログラムの内容・動作の解説及び実行結果の画面キャプチャを含む文書（PDF ファイル：3rd_rep_kadai1.pdf）

入出力例

```
ソートするファイル名： student_list.txt ↵
保存するファイル名： output.txt ↵

学生リスト
Yamada Mirai [ID：FCELE001, Supervisor：Watanabe Kenichi]
Yamamoto Kanon [ID：FCELE002, Supervisor：Kobayashi Sakura]
Matsumoto Mao [ID：FCELE003, Supervisor：Suzuki Masato]
Ikeda Kenta [ID：FCELE004, Supervisor：Tanaka Shota]
：
中略
：
Ito Shiori [ID：FCELE028, Supervisor：Watanabe Kenichi]
Ishikawa Ryota [ID：FCELE029, Supervisor：Tanaka Shota]
Yoshida Shiori [ID：FCELE030, Supervisor：Nakamura Ryuhei]

保存完了: output.txt

学生リスト（ソート後）
--- output.txt の内容 ---
Student ID    Student Name          Supervisor
FCELE017      Abe Kota              Kobayashi Sakura
FCELE020      Abe Mao               Tanaka Shota
FCELE013      Fujita Nao            Kinoshita Kenji
：
中略
：
FCELE009      Yoshida Ayaka         Nakamura Ryuhei
FCELE023      Yoshida Shiori        Sato Misaki
FCELE030      Yoshida Shiori        Nakamura Ryuhei
```

## 課題 2：これまでの応用

テキストファイルに保存されている学生データ（在籍番号，氏名，指導教員）を読み込み，各学生に対して「評価委員」を 2 名選出し，結果を別のテキストファイルに保存するプログラムを作成せよ．プログラムの作成要領は下記の通りである．

1. 入力データおよび教員データの定義：
   入力データの読み込みには，課題 1 で作成した `Student` クラスを使用すること．
   すなわち，プログラム内（`assignment.py`）でモジュール `Student` をインポートし，
   `'student_list.txt'` からデータを読み込んで `Student` オブジェクトのリストを作成し，以後の処理を行うこと．

   また，プログラム内で表 1 の教員リストと専門分野を辞書またはリストとして定義して使用すること．

表 1: 教員リスト

| Name | Field |
|---|---|
| Yamada Kenta, Sato Misaki, Suzuki Masato | A |
| Takahashi Yoko, Tanaka Shota, Ito Mayu | B |
| Watanabe Kenichi, Nakamura Ryuhei | C |
| Kinoshita Kenji, Kobayashi Sakura | D |

ソースコード 1: student_list.txt

```
Student ID,Student Name,Supervisor
FCELE001,Yamada Mirai,Watanabe Kenichi
FCELE002,Yamamoto Kanon,Kobayashi Sakura
FCELE003,Matsumoto Mao,Suzuki Masato
FCELE004,Ikeda Kenta,Tanaka Shota
FCELE005,Hashimoto Mirai,Kinoshita Kenji
FCELE006,Nakajima Takuya,Yamada Kenta
FCELE007,Yamada Hina,Ito Mayu
FCELE008,Ishii Daisuke,Takahashi Yoko
FCELE009,Yoshida Ayaka,Nakamura Ryuhei
FCELE010,Okada Tomoya,Sato Misaki
FCELE011,Suzuki Emi,Suzuki Masato
FCELE012,Watanabe Yui,Ito Mayu
FCELE013,Fujita Nao,Kinoshita Kenji
FCELE014,Kato Emi,Watanabe Kenichi
FCELE015,Shimizu Daisuke,Yamada Kenta
FCELE016,Shimizu Daisuke,Takahashi Yoko
FCELE017,Abe Kota,Kobayashi Sakura
FCELE018,Shimizu Kenta,Nakamura Ryuhei
FCELE019,Ishii Daisuke,Sato Misaki
FCELE020,Abe Mao,Tanaka Shota
FCELE021,Yamashita Kenta,Yamada Kenta
FCELE022,Fujita Yusuke,Kinoshita Kenji
FCELE023,Yoshida Shiori,Sato Misaki
FCELE024,Watanabe Naoto,Suzuki Masato
FCELE025,Saito Emi,Ito Mayu
FCELE026,Tanaka Takumi,Kobayashi Sakura
FCELE027,Ishii Kanon,Takahashi Yoko
FCELE028,Ito Shiori,Watanabe Kenichi
FCELE029,Ishikawa Ryota,Tanaka Shota
FCELE030,Yoshida Shiori,Nakamura Ryuhei
```

2. 制約条件と目標

プログラムは以下の制約を満たし，目標を達成するように作成すること．

(a) 必須制約

- 指導教員本人は評価委員になれない．
- 指導教員と同じ専門分野（A, B, C, D 等）の教員は評価委員になれない．
- 1 人の教員が担当できる最大件数は 7 件以下とする．

(b) 最適化目標

- 目標 1（最優先）：担当件数の平準化
  特定の教員に負担が偏らないよう，全員の担当件数を可能な限り均等にする．
  本課題では学生数 30 名に対し，各 2 名の評価委員を割り当てるため，総割り当て数は $30 \times 2 = 60$ 件となる．教員数は 10 名であるため，理想的な平均担当数は $60 \div 10 = 6$ 件となる．したがって，全員が 6 件となる状態を目指しつつ，やむを得ない場合でも制約（7 件以下）の範囲内に収めること．
- 目標 2：評価委員の分野分散
  可能な限り，2 名の評価委員の専門分野が重複しないようにする．

ヒント：負荷平準化のための評価関数

担当件数を均等にする（分散を最小化する）ため，評価委員のペア候補 $\{i, j\}$ を選定する際の「コスト」計算には工夫が必要である．単に担当件数の合計値を見るだけでは，不均衡なペア（例：0 件と 10 件）と均等なペア（例：5 件と 5 件）の区別がつかない．そこで，教員 $k$ の現在の担当件数を $n_k$ としたとき，そのペアに割り当てた場合の「担当割り当て後の件数」の二乗和をコスト $J$ として定義すればよい．

$$J(i, j) = (n_i + 1)^2 + (n_j + 1)^2 \tag{1}$$

解説：二乗関数 $y = x^2$ は値が大きくなるほど増加率が高くなる性質を持つ．

- 不均衡な場合 (0 件と 10 件)：$1^2 + 11^2 = 122$　（コスト大 → 選ばれにくい）
- 均等な場合 (5 件と 5 件)：$6^2 + 6^2 = 72$　（コスト小 → 選ばれやすい）

このように二乗和を最小化することで，全体の担当数を自然と平準化させることができる．

3. ファイル出力：
出力ファイル ‘assignment_result.txt’ を作成し,「在籍番号，学生氏名，指導教員，専門分野，評価委員 1，専門分野，評価委員 2，専門分野」を各列が見やすくなるように整形して各列の内容が縦に揃って見える形式として保存すること（ソースコード 2 の例を参照）．また，各列を学生番号（在籍番号）の昇順に並べ替えてから出力すること．

4. ファイル入出力と例外処理：
入力ファイルの読み込み失敗・出力ファイルの書き込み失敗に対して適切な例外処理を実装すること（ファイルが存在しない場合，文字コードが不正な場合など）．

例：`with open(input_file, 'r', encoding='utf-8') as f:`

5. 画面出力：
   下記の画面出力例のように，実行時に「割り振りを開始します...」「割り振り完了。」等を表示し， さらに全教員の担当件数一覧を表示すること．また，プログラム実行時に，目標 2（分野分散）を達成できず，やむを得ず「同じ専門分野の評価委員」をペアに選定した件数をカウントし，画面に表示すること．
   （表示例：同じ専門の評価委員の割り当て数（合計）：3）

入出力例

```
30 名のデータを読み込みました。割り振りを開始します...
割り振り完了。
同じ専門の評価委員の割り当て数（合計）：0
完了。結果は 'assignment_result.txt' に保存されました。

--- 担当件数確認 (目標: 5〜7件) ---
Kobayashi Sakura: 5件
Sato Misaki: 6件
Suzuki Masato: 6件
Takahashi Yoko: 6件
Tanaka Shota: 6件
Ito Mayu: 6件
Watanabe Kenichi: 6件
Nakamura Ryuhei: 6件
Kinoshita Kenji: 6件
Yamada Kenta: 7件
```

ソースコード 2: assignment_result.txt

| Student ID | Name | Supervisor | Field | Evaluator1 | Field | Evaluator2 | Field |
|---|---|---|---|---|---|---|---|
| FCELE001 | Yamada Mirai | Watanabe Kenichi | C | Tanaka Shota | B | Kobayashi Sakura | D |
| FCELE002 | Yamamoto Kanon | Kobayashi Sakura | D | Yamada Kenta | A | Nakamura Ryuhei | C |
| FCELE003 | Matsumoto Mao | Suzuki Masato | A | Takahashi Yoko | B | Watanabe Kenichi | C |
| FCELE004 | Ikeda Kenta | Tanaka Shota | B | Yamada Kenta | A | Kobayashi Sakura | D |
| FCELE005 | Hashimoto Mirai | Kinoshita Kenji | D | Suzuki Masato | A | Ito Mayu | B |
| FCELE006 | Nakajima Takuya | Yamada Kenta | A | Watanabe Kenichi | C | Kinoshita Kenji | D |
| FCELE007 | Yamada Hina | Ito Mayu | B | Suzuki Masato | A | Nakamura Ryuhei | C |
| FCELE008 | Ishii Daisuke | Takahashi Yoko | B | Sato Misaki | A | Watanabe Kenichi | C |
| FCELE009 | Yoshida Ayaka | Nakamura Ryuhei | C | Yamada Kenta | A | Takahashi Yoko | B |
| FCELE010 | Okada Tomoya | Sato Misaki | A | Ito Mayu | B | Watanabe Kenichi | C |
| FCELE011 | Suzuki Emi | Suzuki Masato | A | Ito Mayu | B | Watanabe Kenichi | C |
| FCELE012 | Watanabe Yui | Ito Mayu | B | Nakamura Ryuhei | C | Kobayashi Sakura | D |
| FCELE013 | Fujita Nao | Kinoshita Kenji | D | Yamada Kenta | A | Tanaka Shota | B |
| FCELE014 | Kato Emi | Watanabe Kenichi | C | Takahashi Yoko | B | Kinoshita Kenji | D |
| FCELE015 | Shimizu Daisuke | Yamada Kenta | A | Takahashi Yoko | B | Kinoshita Kenji | D |
| FCELE016 | Shimizu Daisuke | Takahashi Yoko | B | Yamada Kenta | A | Kinoshita Kenji | D |
| FCELE017 | Abe Kota | Kobayashi Sakura | D | Sato Misaki | A | Takahashi Yoko | B |
| FCELE018 | Shimizu Kenta | Nakamura Ryuhei | C | Sato Misaki | A | Takahashi Yoko | B |
| FCELE019 | Ishii Daisuke | Sato Misaki | A | Ito Mayu | B | Kinoshita Kenji | D |
| FCELE020 | Abe Mao | Tanaka Shota | B | Sato Misaki | A | Kobayashi Sakura | D |
| FCELE021 | Yamashita Kenta | Yamada Kenta | A | Takahashi Yoko | B | Kobayashi Sakura | D |
| FCELE022 | Fujita Yusuke | Kinoshita Kenji | D | Sato Misaki | A | Ito Mayu | B |
| FCELE023 | Yoshida Shiori | Sato Misaki | A | Tanaka Shota | B | Nakamura Ryuhei | C |
| FCELE024 | Watanabe Naoto | Suzuki Masato | A | Tanaka Shota | B | Nakamura Ryuhei | C |
| FCELE025 | Saito Emi | Ito Mayu | B | Suzuki Masato | A | Nakamura Ryuhei | C |
| FCELE026 | Tanaka Takumi | Kobayashi Sakura | D | Yamada Kenta | A | Tanaka Shota | B |
| FCELE027 | Ishii Kanon | Takahashi Yoko | B | Suzuki Masato | A | Watanabe Kenichi | C |
| FCELE028 | Ito Shiori | Watanabe Kenichi | C | Ito Mayu | B | Kobayashi Sakura | D |
| FCELE029 | Ishikawa Ryota | Tanaka Shota | B | Suzuki Masato | A | Kinoshita Kenji | D |
| FCELE030 | Yoshida Shiori | Nakamura Ryuhei | C | Sato Misaki | A | Tanaka Shota | B |

6. 提出物

   - 以下の実行例が再現されるようにプログラムを作成しソースコード `assignment.py`（py ファイル）と出力結果 `assignment_result.txt`（txt ファイル）
     ※ ファイルは複数になってもよい．複数ファイルは tar でまとめてもよい．

- プログラムの内容・動作の解説及び実行結果の画面キャプチャを含む文書（PDF ファイル：3rd_rep_kadai2.pdf）

ソースコード 3: 疑似コード

```
import sys
import random
from Student import Student  # Student.py からクラスを読み込み

# 教員データ
teachers_data = [
    {"name": "Yamada Kenta", "field": "A"},
    {"name": "Sato Misaki", "field": "A"},
    {"name": "Suzuki Masato", "field": "A"},
    {"name": "Takahashi Yoko", "field": "B"},
    {"name": "Tanaka Shota", "field": "B"},
    {"name": "Ito Mayu", "field": "B"},
    {"name": "Watanabe Kenichi", "field": "C"},
    {"name": "Nakamura Ryuhei", "field": "C"},
    {"name": "Kinoshita Kenji", "field": "D"},
    {"name": "Kobayashi Sakura", "field": "D"},
]

# 名前から分野を引く辞書
name_to_field = {t["name"]: t["field"] for t in teachers_data}

def assign_evaluators(students):
    """
    学生リストを受け取り、最適なペアを割り当てる
    """
    # 状態の初期化
    assignments = []  # 結果を入れる空リスト
    loads = {全教員名: 0}  # 担当数をカウントする辞書

    # 学生ごとに割り当てを実行
    FOR student IN students:
        # 指導教員の情報を取得
        supervisor = student.supervisor
        sup_field = name_to_field[supervisor]

        # 候補者リストの作成
        candidates = []
        FOR teacher IN teachers_data:
            IF teacher.field != sup_field:
                candidates.append(teacher)

        # ベストなペアの探索（総当たり）
        best_pair = None
        min_cost = 無限大

        # 候補者の中から 2 人組 (t1, t2) をすべて試す
        FOR i FROM 0 TO len(candidates) - 1:
            FOR j FROM i+1 TO len(candidates) - 1:
                t1 = candidates[i]
                t2 = candidates[j]

                # 現在の担当数を取得
                load1 = loads[t1.name]
                load2 = loads[t2.name]

                # 制約チェック: どちらかが 7 件以上ならスキップ
```

```
                IF load1 >= 7 OR load2 >= 7:
                    CONTINUE

                # コスト計算
                # 担当数が少ない人ほどコストが低くなる
                current_cost = (load1 + 1)^2 + (load2 + 1)^2

                # ペナルティ: 分野が被っている場合
                IF t1.field == t2.field:
                    current_cost += 1000

                # 最良ペアの更新
                IF current_cost < min_cost:
                    min_cost = current_cost
                    best_pair = (t1, t2)

        # 割り当ての確定
        IF best_pair IS None:
            RETURN エラー (割り当て失敗)

        # 選ばれた 2 人を確定させる
        eval1, eval2 = best_pair

        # 担当数カウンターを更新 (+1)
        loads[eval1.name] += 1
        loads[eval2.name] += 1

        # 結果をリストに保存
        assignments.append({
            student_id: student.id,
            evaluator1: eval1.name,
            evaluator2: eval2.name
        })

    RETURN assignments, loads


def main():
    # 1. ファイル読み込み
    students = ファイル読み込み ("student_list.txt")

    # 2. ランダム化 (学生の順番で結果が変わる)
    students をシャッフルする

    # 3. 割り当て実行
    results, loads = assign_evaluators(students)

    # 4. 結果の整理
    results を「学生番号」順にソートする

    # 5. ファイル保存
    results をファイル書き出し ("assignment_result.txt")

    # 6. 確認表示
    loads (担当数) を表示して、偏りがないか確認する

# 実行ブロック
if __name__ == "__main__":
    main()
```