【課題 03：pokemon.py】

出現したポケモンが図鑑に登録されているかを調べるコードを実装せよ．配布した図鑑用データファイル「poke_names.txt」を読み込み，ポケモン図鑑を作成する．当該ファイルは 1 行が 1 匹のポケモンのデータであり，ポケモン番号，名前，分類，タイプがタブ区切りで書かれている．

採点の都合上，以下の要件を満たすこととする：

- 基本となる Monster クラスを定義する；
  - イニシャライザで，名前を設定する；
  - 等価比較演算子でインスタンスの等価性を評価できるようにする．インスタンスの等価性は，名前を表すインスタンス変数の値が等しいことで定義する；
- Monster クラスを継承した ZukanMonster クラスを定義する；
  - イニシャライザで，名前，分類，タイプを設定する．名前は，親クラスのイニシャライザを呼び出して設定する；
  - インスタンスを文字列化した際に，図 1 のような出力になるように特殊メソッドをオーバーライドする；
- Zukan クラスを定義する；
  - イニシャライザを以下のように定義する；
    - イニシャライザで，図鑑用データファイルのパスを受け取り，ファイル読み込み関数を呼び出す；
    - 読み込み関数が return するリストをインスタンス変数として受け取る；
  - 図鑑用データファイルの読み込み関数を静的メソッドとして定義する；
    - 引数として受け取った図鑑用データファイルを開く；
    - 読み込んだデータに基づき，ZukanMonster クラスのインスタンスを生成する；
    - インスタンスが並ぶリストを return する；
  - 図鑑にポケモンが登録されているかを検索するメソッドを定義する；
    - 引数として受け取ったインスタンスと，図鑑に登録されているインスタンス 1 つずつと等価性を比較する；
    - 図 1 の出力になるように，適切なものを return する；
- adventure.py には変更を加えない；
- 出力の形式は，図 1 のようにする；

```
PS C:\Users\admin\Desktop\ProA2> python .\lec02\kadai03\adventure.py .\lec02\data\poke_names.txt ピカチュウ
なまえ：ピカチュウ
ぶんるい：ねずみポケモン
たいぷ：でんき
PS C:\Users\admin\Desktop\ProA2> python .\lec02\kadai03\adventure.py .\lec02\data\poke_names.txt ポケチュウ
ポケチュウは図鑑に登録されていない新種です
```

図 1　課題 03 の実行例：”ピカチュウ”は図鑑に登録されているが，”ポケチュウ”は登録されていない．

リスト 1　adventure.py

```
1 from pokemon import *
2 import sys
3
4 if __name__ == "__main__":
5     file_path = sys.argv[1] # "lec02/data/poke_names.txt"
6     zukan = Zukan(file_path)
7     poke_name = sys.argv[2]
8     mon = Monster(poke_name)
9     print(zukan.search_monster(mon))
```

リスト 2 pokemon.py

```
class Monster:
    def __init__(self, title):
        self.title = title

    def __eq__(self, other):
        return self.title == other.title


class ZukanMonster(Monster):
    def __init__(self, title, species, types):
        super().__init__(title)
        self.species = species
        self.types = types

    def __repr__(self):
        return f"なまえ: {self.title}\n ぶんるい :{self.species}\n たいぷ :{self.types}"


class Zukan:
    def __init__(self, file_path):
        self.monsters = __class__.read_file(file_path)

    def search_monster(self, mon):
        for monster in self.monsters:
            if mon == monster:
                return monster
        else:
            return f"{mon.title}は図鑑に登録されていない新種です"

    @staticmethod
    def read_file(file_path):
        monsters = []
        with open(file_path, "r", encoding="utf8") as rfo:
            for row in rfo:
                _, tit, spe, typ = row.rstrip().split("\t")
                monsters.append(ZukanMonster(tit, spe, typ))
        return monsters
```

【課題 04：pokemon.py】

自分が選んだポケモンと，ランダムに選ばれたポケモンが勝負をするコードを実装せよ．配布した名前ファイル「poke_names.txt」と種族値ファイル「base_stats.txt」を読み込み，全 251 匹のポケモンインスタンスを要素とするリストを作成する．各ポケモンインスタンスに対して，乱数によりレベルを設定する．そして，コマンドライン引数で指定した番号のポケモンを自分のポケモンとし，リストからランダムに選ばれた敵と戦う．レベルと種族値の合計が高いほうが勝ちとなる．

採点の都合上，以下の要件を満たすこととする：

- Monster クラスを定義する；
  - イニシャライザを以下のように定義する；
    - 名前（title），分類，タイプを設定する；
    - 種族値リストを設定する；
    - 名前マングリングしたレベル用の変数を 0 に初期化する；
  - レベル用の変数の値を取得，設定するプロパティ（ゲッタとセッタ）を定義する．セッタでは，負の値が与えられた時に，図 2 のように「不正の値のため，符号を反転して設定します」と出力するとともに，符号を反転させた正の値を設定する；
  - インスタンスを文字列化した際に，図 2 のように**名前（*title*）とレベル**が出力されるように特殊メソッドをオーバーライドする；
  - インスタンスを大小比較した際に，**種族値の合計とレベルの和**により大小を定義するように特殊メソッドをオーバーライドする；
- battle.py には変更を加えない；
- 名前ファイル「poke_names.txt」と種族値ファイル「base_stats.txt」を読み込む関数は，リスト 4 を使用する（Moodle にある pokemon.py にすでに書かれている）；
- 出力の形式は，図 2 のようにする；

```
PS C:\Users\admin\Desktop\ProA2> python .\lec02\kadai04\battle.py .\lec02\data\poke_names.txt .\lec02\data\base_stats.txt 121
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
不正な値のため，符号を反転して設定します
僕のポケモン： スターミー(Lv.88)
vs アズマオウ(Lv.4)
スターミー(Lv.88) の勝ち！
vs シェルダー(Lv.18)
スターミー(Lv.88) の勝ち！
vs アーボック(Lv.27)
スターミー(Lv.88) の勝ち！
vs バンギラス(Lv.40)
スターミー(Lv.88) の負け(>_<)
vs ヤドキング(Lv.21)
スターミー(Lv.88) の勝ち！
```

図 2　課題 04 の実行例：【上側】レベル用の乱数が負の値になった時の出力である．期待値として 25 回くらい発生する／【下側】121 番のスターミーを自分のポケモンとして指定している．ランダムに選ばれた 5 匹のポケモンと戦い，4 勝 1 敗の状態である．

リスト 3 battle.py

```
from pokemon import *
from random import randint, choice
import sys

if __name__ == "__main__":
    names = read_names(sys.argv[1]) # .\lec02\data\poke_names.txt
    stats = read_stats(sys.argv[2]) # .\lec02\data\base_stats.txt
    monsters = [Monster(n, s) for n, s in zip(names, stats)]
    for mon in monsters:
        mon.level = randint(-10, 100) # 全ポケモンに対して,レベルを乱数で設定する
    ## ここまでが準備

    my_monster = monsters[int(sys.argv[3])-1] # 自分のパートナーポケモン
    print("僕のポケモン:", my_monster)

    for _ in range(5):
        teki = choice(monsters)
        print("vs", teki)
        if my_monster > teki:
            print(my_monster, "の勝ち！")
        else:
            print(my_monster, "の負け (>_<)")
```

リスト 4 pokemon.py

```
1  class Monster:
2      def __init__(self, names, stats):
3          self.title, self.species, self.types = names
4          self.stats = stats
5          self.__level = 0
6
7      @property
8      def level(self):
9          return self.__level
10
11     @level.setter
12     def level(self, val):
13         if val < 0:
14             print("不正な値のため,符号を反転して設定します")
15             self.__level = -val
16         else:
17             self.__level = val
18
19     def __repr__(self):
20         return f"{self.title}(Lv.{self.__level})"
21
22     def __gt__(self, other):
23         return sum(self.stats+[self.level]) > sum(other.stats+[other.level])
24
25
26 def read_names(file_path):
27     names = []
28     with open(file_path, "r", encoding="utf8") as rfo:
29         for row in rfo:
30             _, tit, spe, typ = row.rstrip().split("\t")
31             names.append([tit, spe, typ])
32     return names
33
34 def read_stats(file_path):
35     stats = []
36     with open(file_path, "r") as rfo:
37         for row in rfo:
38             row = row.rstrip()
39             stats.append([int(col) for col in row.split(" ")])
40     return stats
```