【課題 09：pokemon_dsc.py】

ポケモン 1 体をあらわす Monster クラスに対して，ディスクリプタを介して属性（タイトル，タイプ，レベル）を設定するコードを実装せよ．

各属性には設定できる値に条件があるため，それらを満たした場合は値を設定し，満たさない場合は例外を raise する．

- タイトル：設定できる文字数は，2 以上，5 以下である
- レベル：設定できる値は，1 以上，100 以下である
- タイプ：設定できる値は，以下に含まれるもののみである：$types$ = {”ノーマル”,”ほのお”,”みず”,”くさ”,”でんき”,”こおり”,”かくとう”,”どく”,”じめん”,”ひこう”,”エスパー”,”むし”,”いわ”,”ゴースト”,”ドラゴン”,”あく”,”はがね”,”フェアリー”}

採点の都合上，以下の要件を満たすこととする：

(1) ディスクリプタクラスに継承されることを想定した抽象基底クラス Validator を定義する；
　（a）イニシャライザは，定義してもしなくてもよい；
　（b）ディスクリプタとして必要な特殊メソッドを定義する；
　（c）抽象メソッド validate() を定義する；
(2) Monster クラスのタイトル属性用のディスクリプタとして，抽象基底クラス Validator を継承した Title クラスを定義し，設定できる条件を満たしていない場合は例外を raise する；
(3) Monster クラスのレベル属性用にディスクリプタとして，抽象基底クラス Validator を継承した Level クラスを定義し，設定できる条件を満たしていない場合は例外を raise する；
(4) Monster クラスのタイプ属性用にディスクリプタとして，抽象基底クラス Validator を継承した Type クラスを定義し，設定できる条件を満たしていない場合は例外を raise する；
(5) ポケモン 1 体を表す Monster クラスを定義する；
　（a）タイトル，レベル，タイプ 1，タイプ 2 用の属性を持ち，上記のディスクリプタを介してこれらに値を設定する；
　（b）イニシャライザにおいて，パラメータに代入された値を各種属性に設定する；
　（c）Monster クラスのインスタンスを print した際に，実行結果のようになるように特殊メソッドを定義する；
(6) 配布した pokemon_dsc.py の「if __name__ == ”__main__”」に，適切な例外処理を書き加え，例外発生箇所でプログラムが終了することのないようにする．例外処理以外のコードは書き加えないこと；
(7) デバッグ用の print など必要ないものはコメントアウトし，出力が図 1 になるようにする；

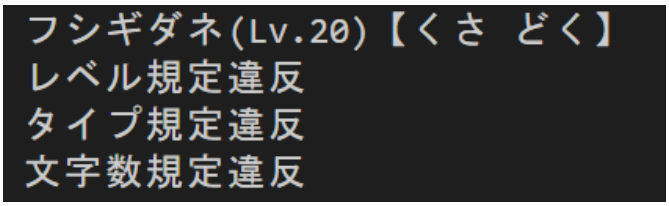

図 1　課題 09 の実行例

リスト 1 pokemon_dsc.py

```
1  from abc import ABC, abstractmethod
2
3  class Validator(ABC):
4      def __init__(self, minval, maxval):
5          self.minval, self.maxval = minval, maxval
6
7      def __set_name__(self, owner, attrname):
8          self.attrname = attrname
9
10     def __get__(self, obj, objtype=None):
11         return obj.__dict__[self.attrname]
12
13     def __set__(self, obj, value):
14         self.validate(value)
15         obj.__dict__[self.attrname] = value
16
17     @abstractmethod
18     def validate(self, value):
19         pass
20
21
22 class Title(Validator):
23     def validate(self, value):
24         if not (self.minval <= len(value) <= self.maxval):
25             raise ValueError("文字数規定違反")
26
27
28 class Level(Validator):
29     def validate(self, value):
30         if not (self.minval <= value <= self.maxval):
31             raise ValueError("レベル規定違反")
32
33 class Type(Validator):
34     def __init__(self, types):
35         self.types = types
36
37     def validate(self, value):
38         if value not in self.types:
39             raise ValueError("タイプ規定違反")
40
41
42 class Monster:
43     types = {"ノーマル", "ほのお", "みず", "くさ", "でんき", "こおり", "かくとう", "どく", "じめ
          ん", "ひこう", "エスパー", "むし", "いわ", "ゴースト", "ドラゴン", "あく", "はがね", "フ
          ェアリー"}
44     title = Title(2, 5)
45     level = Level(1, 100)
46     type1 = Type(types)
47     type2 = Type(types)
48
49     def __init__(self, title, level, *types):
```

```
50          self.title = title
51          self.level = level
52          self.type_ = None
53          if len(types) == 1:
54              self.type1 = types[0]
55              self.type_ = self.type1
56          else:
57              self.type1, self.type2 = types
58              self.type_ = self.type1+" "+self.type2
59
60      def __repr__(self):
61          return f"{self.title}(Lv.{self.level})【{self.type_}】"
62
63
64  if __name__ == "__main__":
65      try:
66          fushi = Monster("フシギダネ", 20, "くさ", "どく")
67          print(fushi)
68      except Exception as e:
69          print(e)
70
71      try:
72          pika = Monster("ピカチュウ", -15, "でんき")
73          print(pika)
74      except Exception as e:
75          print(e)
76
77      try:
78          eevee = Monster("イーブイ", 25, "いろいろ")
79          print(eevee)
80      except Exception as e:
81          print(e)
82
83      try:
84          muu = Monster("ミュースリー", 100, "エスパー")
85          print(muu)
86      except Exception as e:
87          print(e)
```

【課題 10：shuzokuchi.py】

ポケモン 1 体の種族値を表す Shuzokuchi クラスに対して，ディスクリプタを介して属性（HP，こうげき，ぼうぎょ，とくこう，とくぼう，すばやさ）を設定するコードを実装せよ．

種族値には設定できる値に条件があるため，それらを満たした場合は値を設定し，満たさない場合は例外を raise する．

- いずれの種族値も，1 以上，255 以下である

採点の都合上，以下の要件を満たすこととする：

(1) Shuzokuchi クラスの属性用のディスクリプタとして，ShuzokuchiDesc クラスを定義し，設定できる条件を満たしていない場合は例外を raise する；
(2) ポケモン 1 体の種族値を表す Shuzokuchi クラスを定義する；
   （a）HP，こうげき，ぼうぎょ，とくこう，とくぼう，すばやさ用の 6 つの属性を持ち，上記のディスクリプタを介してこれらに値を設定する；
   （b）イニシャライザにおいて，パラメータに代入された値を種族値用の 6 つの属性に設定する；
   （c）Shuzokuchi クラスのインスタンスを print した際に，実行結果のようになるように特殊メソッドを定義する；
   （d）Shuzokuchi クラスのインスタンスに対して インスタンス ["属性名"] のように大かっこを用いて各属性の値を参照できるように，適切な特殊メソッドを定義する；
   （e）Shuzokuchi クラスの 2 つのインスタンスに対して算術演算子 + を用いたときに，同じ種族値同士の和を属性として持つような新たな Shuzokuchi インスタンスを返すような，適切な特殊メソッドを定義する（授業中では扱っていないため，適切な特殊メソッドを調べること）；
   （f）Shuzokuchi クラスの 2 つのインスタンスに対して算術演算子 − を用いたときに，同じ種族値同士の差を属性として持つような新たな Shuzokuchi インスタンスを返すような，適切な特殊メソッドを定義する（授業中では扱っていないため，適切な特殊メソッドを調べること）；
   （g）Shuzokuchi クラスのインスタンスに対して，len() 関数を用いたときに，種族値の合計を返すように，適切な特殊メソッドを定義する；
   （h）Shuzokuchi クラスの 2 つのインスタンスに対して比較演算子 > を用いたときに，自身の種族値の合計が他方より大きいことを真理値として返すように，適切な特殊メソッドを定義する；
   （i）Shuzokuchi クラスの 2 つのインスタンスに対して比較演算子 < を用いたときに，自身の種族値の合計が他方より小さいことを真理値として返すように，適切な特殊メソッドを定義する；
   （j）Shuzokuchi クラスの 2 つのインスタンスに対して比較演算子 == を用いたときに，自身の種族値の合計が他方と等しいことを真理値として返すように，適切な特殊メソッドを定義する；
(3) 配布した shuzokuchi.py の「if __name__ == "__main__"」に，適切な例外処理を書き加え，例外発生箇所でプログラムが終了することのないようにする．例外処理以外のコードは書き加えないこと；
(4) デバッグ用の print など必要ないものはコメントアウトし，出力が図 2 になるようにする；

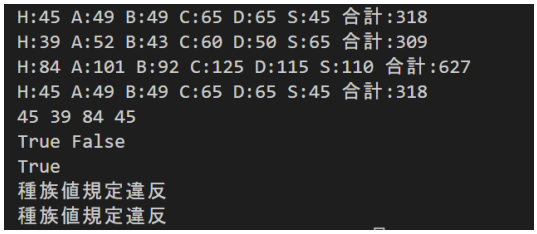

図 2　課題 10 の実行例：1 行目はフシギダネの print 結果；2 行目はヒトカゲの print 結果；3 行目はフシギダネとヒトカゲが合体したポケモンの print 結果；4 行目は合体ポケモンからヒトカゲが剥がれた状態の print 結果；5 行目は上記 4 体のポケモンの hp の print 結果；6 行目は合体ポケモンと剥がれた状態の強さを比較した print 結果；7 行目はヒトカゲが剥がれた状態とフシギダネの強さを比較した print 結果；8 行目はフシギダネ種族値からヒトカゲ種族値を引いた際に例外が投げられた結果；9 行目は全種族値が 0 のザコポケの種族値を設定する際に例外が投げられた結果；

リスト 2 shuzokuchi.py

```
class ShuzokuchiDesc:
    def __init__(self):
        self.minval, self.maxval = 1, 255

    def __set_name__(self, owner, attrname):
        self.attrname = attrname

    def __get__(self, obj, objtype=None):
        return obj.__dict__[self.attrname]

    def __set__(self, obj, value):
        if not (self.minval <= value <= self.maxval):
            raise ValueError("種族値規定違反")
        obj.__dict__[self.attrname] = value


class Shuzokuchi:
    hp = ShuzokuchiDesc() # HP
    atk = ShuzokuchiDesc() # こうげき
    dfs = ShuzokuchiDesc() # ぼうぎょ
    sp_atk = ShuzokuchiDesc() # とくこう
    sp_dfs = ShuzokuchiDesc() # とくぼう
    spd = ShuzokuchiDesc() # スピード

    def __init__(self, hp, atk, dfs, sp_atk, sp_dfs, spd):
        self.hp = hp
        self.atk = atk
        self.dfs = dfs
        self.sp_atk = sp_atk
        self.sp_dfs = sp_dfs
        self.spd = spd

    def __repr__(self):
        return f"H:{self.hp} A:{self.atk} B:{self.dfs} C:{self.sp_atk} D:{self.sp_dfs} S:{self.
            spd} 合計:{len(self)}"

    def __getitem__(self, key):
        return getattr(self, key)

    def __add__(self, other):
        return __class__(**{key: self[key]+other[key] for key in self.__dict__})

    def __sub__(self, other):
        return __class__(**{key: self[key]-other[key] for key in self.__dict__})

    def __len__(self):
        return sum(self.__dict__.values())

    def __gt__(self, other):
        return len(self) > len(other)

```

```
51      def __eq__(self, other):
52          return len(self) == len(other)
53
54      def __lt__(self, other):
55          return len(self) < len(other)
56
57
58  if __name__ == "__main__":
59      p1 = [45, 49, 49, 65, 65, 45]
60      s1 = Shuzokuchi(*p1)
61      print(s1)
62      p2 = [39, 52, 43, 60, 50, 65]
63      s2 = Shuzokuchi(*p2)
64      print(s2)
65      s3 = s1+s2
66      print(s3)
67      s4 = s3-s2
68      print(s4)
69
70      print(s1["hp"], s2["hp"], s3["hp"], s4["hp"])
71
72      print(s3 > s4, s3 < s4)
73      print(s4 == s1)
74
75      try:
76          s5 = s1-s2 # フシギダネーヒトカゲ
77      except Exception as e:
78          print(e)
79
80      try:
81          s0 = Shuzokuchi(*[0 for _ in range(6)])
82      except Exception as e:
83          print(e)
```