

【課題 11 : count_pairs.py】

`collections` モジュールの `Counter` クラスを利用して、課題 02 と同様に、配布した「poke_names.txt」を読み込み、3 列目に書かれたタイプのペアをカウントし、出現回数が多いペアから順に出力するコードを実装せよ。当該ファイルは、各行において番号、名前、タイプ、進化先がタブで区切られており、3 列目のタイプは半角スペースで複数のタイプが区切られている。この課題では、3 列目のタイプが 2 つある行が対象となる。

採点の都合上、以下の要件を満たすこととする：

- `collections.Counter` を用いる；
- 読み込みファイルのパスは、コマンドライン引数により指定する；
- 不要なモジュールは `import` しない；
- 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；

```
PS C:\Users\admin\Desktop\ProA2> python .\lec06\kadai11\count_pairs.py lec06\data/poke_names.txt
[(frozenset({'ひこう', 'ノーマル'}), 10),
 (frozenset({'どく', 'くさ'}), 9),
 (frozenset({'じめん', 'いわ'}), 8),
 (frozenset({'むし', 'どく'}), 7),
 (frozenset({'ひこう', 'むし'}), 5),
 (frozenset({'みず', 'いわ'}), 5),
 (frozenset({'エスパー', 'みず'}), 4),
 (frozenset({'ひこう', 'ほのお'}), 3),
 (frozenset({'フェアリー', 'ノーマル'}), 3),
 (frozenset({'ひこう', 'どく'}), 3)]
```

図 1 課題 11 の実行例

【課題 12 : evolution.py】

list クラスを継承し、進化先リストを表す EvolvList クラスを定義せよ。list クラスを継承しているので、リストとしての性質、機能を持つ。例えば、append メソッドにより当該クラスのインスタンスに要素を追加できる。

【「if __name__ == "__main__"」以下の説明】

- (1) フシギダネの進化先リスト tane をインスタンス化し、tane に進化先のフシギソウを表す kusa を append メソッドにより追加している；
- (2) フシギダネはフシギソウにしか進化できないため、リスト tane には 1 つの要素のみ append している；
- (3) この例は実際と異なり、フシギバナはさらにフシギスギに進化するが、フシギスギはこれ以上進化しないため、フシギバナを表す hana には、名前文字列"フシギスギ"を append している；
- (4) 当該リストクラスの evolution() メソッドにより、図 2 のように、進化先の名前を進化の段階に合わせてインデントして print している；

上記のように直接進化先を要素に格納する特殊なリストを表す EvolveList クラスを実装せよ。採点の都合上、以下の要件を満たすこととする：

- (1) list クラスを継承する；
- (2) このクラスのインスタンスは、直接進化先（フシギダネの直接進化先はフシギソウのみ）を要素として持つリストである；
- (3) ただし、直接進化先がさらに進化先を持っている場合は、進化先ポケモンの EvolvList インスタンスを、さらなる進化先がない場合は、進化先ポケモンの名前文字列を要素とするリストである；
- (4) イニシャライザでは、インスタンス変数 title に名前を設定する；
- (5) evolution() メソッドを定義する；
 - (a) 進化の段階（例：フシギダネは 0、フシギソウは 1、フシギバナは 2）を表すパラメータ dankai を持つ；
 - (b) dankai のデフォルト値を 0 とする；
 - (c) dankai の値に合わせたインデント（全角スペース）を定義する；
 - (d) 定義したインデントと名前を print する；
 - (e) 当該リストの要素（EvolvList インスタンス or 文字列）に対して、
 - i. EvolvList インスタンスなら（さらに進化するなら）、その evolution() メソッドを呼び出す；
 - ii. 文字列なら（それ以上進化しないなら）、その名前文字列を print する；
- (6) evolution.py にあらかじめ書かれている部分は、変更しない；
- (7) 不要なモジュールは import しない；
- (8) 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；

```

フシギダネ/
フシギソウ/
  フシギバナ/
    フシギスギ
[[['フシギスギ']]
ナゾノクサ/
クサイハナ/
ラフレシア
クレイハナ
[['ラフレシア', 'クレイハナ']]
イーブイ/
シャワーズ
サンダーズ
ブースター
エーフィ
ブラッキー
リーフィア
グレイシア
ニンフィア
['シャワーズ', 'サンダーズ', 'ブースター', 'エーフィ', 'ブラッキー', 'リーフィア', 'グレイシア', 'ニンフィア']

```

図 2 課題 12 の実行例

リスト 1 evolution.py

```
1 if __name__ == "__main__":
2     tane = EvolvList("フシギダネ")
3     kusa = EvolvList("フシギソウ")
4     hana = EvolvList("フシギバナ")
5     tane.append(kusa)
6     kusa.append(hana)
7     hana.append("フシギスギ")
8     tane.evolution()
9     print(tane)
10
11    nazo = EvolvList("ナゾノクサ")
12    kusai= EvolvList("クサイハナ")
13    nazo.append(kusai)
14    kusai.append("ラフレシア")
15    kusai.append("キレイハナ")
16    nazo.evolution()
17    print(nazo)
18
19    eevee = EvolvList("イーブイ")
20    eevee.append("シャワーズ")
21    eevee.append("サンダース")
22    eevee.append("ブースター")
23    eevee.append("エーフィ")
24    eevee.append("ブラッキー")
25    eevee.append("リーフィア")
26    eevee.append("グレイシア")
27    eevee.append("ニンフィア")
28    eevee.evolution()
29    print(eevee)
```

【課題 13 : unique.type.py】

dict クラスを継承し、異なるタイプからなるポケモンのチームを表す Team クラスを定義せよ。Team クラスは、キーをポケモンの名前、値をタイプ (set で表す) とした dict である。通常の dict は、キーは一意である必要があるが、値が重複しても問題ない。一方、今回実装する Team クラスは、値も一意である必要がある。もし、すでに登録されている値を再度登録しようとした場合、ValueError が発生するように実装する。これにより、一意なタイプパターンからなるチームを作ることができる。なお、単独の「set("どく")」タイプと「set("どく", "くさ")」タイプは異なるタイプパターンとする。

上記のように、キーだけでなく、値にも一意性の制約を持つ特殊な辞書を表す Team クラスを実装せよ。採点の都合上、以下の要件を満たすこととする：

- (1) dict クラスを継承する；
- (2) 「辞書[キー] = 値」の形式で辞書に値を設定できるように、特殊メソッドを定義する；
- (3) この特殊メソッドでは、以下の条件のとき、ValueError を raise する：
 - (a) 「値」がすでに登録されており、「キー」が既に登録されている、かつ、「辞書[キー] != 値」の場合（※「辞書[キー] == 値」なら、同じエントリーを再度登録しようとしているだけなので、エラーではない）；
 - (b) 「値」がすでに登録されており、「キー」がまだ登録されてない場合；
- (4) unique.type.py にあらかじめ書かれている部分は、変更しない；
- (5) 読み込みファイルのパスは、コマンドライン引数により指定する；
- (6) 不要なモジュールは import しない；
- (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 3 になるようにする；

```
このタイプのポケモンはすでにチームに存在します
このタイプのポケモンはすでにチームに存在します
このタイプのポケモンはすでにチームに存在します
60
{'アーボ': {'どく'},
 'イシツブテ': {'じめん', 'いわ'},
 'ウソッキー': {'いわ'},
 'ウパー': {'じめん', 'みず'},
 'ウリムー': {'こおり', 'じめん'},
 'エアームド': {'はがね', 'ひこう'},
 'オムナイト': {'みず', 'いわ'},
 'カイリリュー': {'ひこう', 'ドラゴン'},
```

図 3 課題 13 の実行例：例外が発生した箇所の一部と最後の pprint の一部を表示している。pprint で出力しているため、キーがあいいうえお順で表示されている。タイプパターンの異なり数は 60 である。

リスト 2 unique_type.py

```
1 import sys
2 from pprint import pprint
3
4 def read_file(file_path):
5     titles, types = [], []
6     with open(file_path, "r", encoding="utf8") as rfo:
7         for row in rfo:
8             _, tit, typ, *_ = row.rstrip().split("\t")
9             titles.append(tit)
10            types.append(set(typ.split()))
11    return titles, types
12
13
14 if __name__ == "__main__":
15     titles, types = read_file(sys.argv[1])
16     team = Team()
17     for title, type_set in zip(titles, types):
18         try:
19             team[title] = type_set
20         except Exception as e:
21             print(e)
22
23     print(len(team))
24     pprint(team)
```

【課題 14 : zukan.py】

list クラスを継承し、Monster クラスのインスタンスのみを append できる特殊なリスト Zukan クラスを実装せよ。
採点の都合上、以下の要件を満たすこととする：

- (1) 出力結果を参考にして、ポケモン 1 体を表す Monster クラスを、本日習った、namedtuple または dataclass を使用して定義する；
- (2) Monster クラスのインスタンスのみを append できる Zukan クラスを定義する；
 - (a) list クラスを継承する；
 - (b) Monster クラスのインスタンスのみを追加できるように、append メソッドをオーバーライドする；
 - (c) Monster クラスのインスタンス以外が append されそうになった時、ValueError を raise する；
- (3) zukan.py にあらかじめ書かれている部分は、変更しない；
- (4) 読み込みファイルのパスは、コマンドライン引数により指定する；
- (5) 不要なモジュールは import しない；
- (6) 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 4 になるようにする；

```
Monster(title='スイクン'),  
Monster(title='ヨーギラス'),  
Monster(title='サナギラス'),  
Monster(title='バンギラス'),  
Monster(title='ルギア'),  
Monster(title='ホウオウ'),  
Monster(title='セレビィ')]  
Monsterクラスのインスタンス以外は登録できません
```

図 4 課題 14 の実行例

リスト 3 zukan.py

```
1 import sys
2 from pprint import pprint
3
4
5 def read_names(file_path):
6     titles = list()
7     with open(file_path, "r", encoding="utf8") as rfo:
8         for row in rfo:
9             _, tit, _, *_ = row.rstrip().split("\t")
10            titles.append(tit)
11    return titles
12
13
14 if __name__ == "__main__":
15     titles = read_names(sys.argv[1])
16     zukan = Zukan()
17     for title in titles:
18         zukan.append(Monster(title))
19     pprint(zukan)
20     try:
21         zukan.append(251)
22     except Exception as e:
23         print(e)
```
