

【課題 05 : sort_name.py】

ポケモンの名前を辞書順にソートするコードを実装せよ。配布したポケモンデータファイル「poke_names.txt」を読み込み、ポケモンの名前を抽出する。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、タイプ、進化先がタブ区切りで書かれている。

辞書順のソートには、名前文字列を構成する 1 文字に対してスコアを算出し、その重み付き和により名前文字列のスコアを算出する。ord(1 文字) 関数を用いることで、「ア」ほど小さい、「ン」ほど大きな値を得ることができる。そして、数値の場合と同様に、先頭文字（上の位の文字）ほど大きな重みを掛けて足し合わせることで、名前文字列のスコアを計算できる。例えば、名前文字列「アイウ」のスコアは、(「ア」のスコア × 1 桁目の重み) + (「イ」のスコア × 2 桁目の重み) + (「ウ」のスコア × 3 桁目の重み) で求める。この時、1 桁目の重みは基数の 5 乗、2 桁目の重みは基数の 4 乗、3 桁目の重みは基数の 3 乗とするのが自然である。基数は、ord() 値の最大とするとシンプルである。ちなみに、10 進数の基数は 10、2 進数の基数は 2 である。

- 今回の対象となる名前文字列の長さの最大値は、「5」である。つまり、5 桁である。
- 今回の対象となる名前文字列に含まれる文字の中で、ord() の値の最大値は、「12540」である（環境によって異なるかも）。

採点の都合上、以下の要件を満たすこととする：

- ファイル読み込み用の関数を定義する；
 - － 読み込み元のファイルパスを受け取るパラメータを持つ；
 - － 2 列目の名前だけを抽出し、リストに格納する；
 - － 名前のリストを返す；
- 結果書き込み用の関数を定義する；
 - － 書き込み先のファイルパスと、書き込み対象のリストを受け取るパラメータを持つ；
 - － リストの要素（この課題ではソート後の名前）を 1 行ずつ出力する；
- 名前文字列のスコアを計算する関数を定義する；
 - － 名前文字列を受け取るパラメータを持つ；
 - － 名前文字列を構成する 1 文字ずつに対して、ord() 関数によりアイウエオ順に高くなるスコアを求める；
 - － 名前文字列の先頭（上の桁）ほど大きな重みを掛けながら、構成文字すべてのスコアを足し合わせることで、名前文字列のスコアを算出する；
 - － ※実際の辞書では、伸ばす音を表す長音符号「ー」は、前の文字の母音により音を決めるが、処理が煩雑なので、今回の課題ではその事実は無視する。すなわち、「ー」は ord("ー") により変換された値をそのまま使う；
 - － 名前文字列のスコアを返す；
- (1) 読み込みファイルと (2) 書き込みファイルのパスは、コマンドライン引数により、この順番で指定する；
- sorted() 関数、または、sort() メソッドに対して、名前文字列のスコアを計算する関数オブジェクトを、キーワード引数により指定する；
- 書き込み関数の 2 つの引数は、キーワード引数により指定する；
- 出力の形式は、Moodle にアップロードしてある「sort_names.txt」と全く同じようにする；

リスト 1 sort_name.py

```
1 def read_names(file_path):
2     names = []
3     with open(encoding="utf8", file=file_path, mode="r") as rfo:
4         for row in rfo:
5             _, nam, *_ = row.rstrip().split("\t")
6             names.append(nam) # 名前の文字列をappend
7     return names
8
9
10 def write_names(file_path, name_lst):
11     with open(encoding="utf8", file=file_path, mode="w") as wfo:
12         for name in name_lst:
13             wfo.write(f"{name}\n")
14
15
16 def name_score(name):
17     s = 0
18     for i, n in enumerate(name):
19         s += ord(n)*12540**(5-i)
20     return s
21
22
23 if __name__ == "__main__":
24     names = read_names(sys.argv[1]) # lec03/data/poke_names.txt
25     chars = [ord(n) for name in names for n in name]
26     print(max(chars))
27
28     names.sort(key=name_score)
29     write_names(file_path=sys.argv[2], name_lst=names) # lec03/kadai05/sort_names.txt
```

【課題 06 : distance.py】

自分が選んだポケモンと最も類似するポケモンを求めるコードを実装せよ。配布した種族値ファイル「base_stats.txt」と名前ファイル「poke_names.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。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、自分のポケモン以外の全ポケモンとの距離を求め、距離が最も小さなポケモンを最類似ポケモンとして出力する。

ポケモン間の距離は、6 次元の種族値ベクトル (6 要素からなるリスト) $\mathbf{x} = [x_1, \dots, x_6]$ と $\mathbf{y} = [y_1, \dots, y_6]$ 間の

- (1) Manhattan : $\sum_{i=1}^6 |x_i - y_i|$
- (2) Euclidean : $\sqrt{(\sum_{i=1}^6 |x_i - y_i|^2)}$
- (3) Chebyshev : $\max_{1 \leq i \leq 6} \{|x_i - y_i|\}$

により求める。そして、各距離定義を表すラムダ式と自分のポケモンを引数として渡すことで、全ポケモンとの距離を計算し最類似ポケモンを求める関数を定義する。

採点の都合上、以下の要件を満たすこととする：

- 指定ポケモンとそれ以外のポケモンの距離を計算し、最類似ポケモンを求める関数 `calc_similar_monster()` を定義する；
 - － 距離関数（ラムダ式）オブジェクトと指定ポケモン（この課題では自分のポケモン）のインスタンスを受け取るパラメータを持つ；
 - － 引数として渡された距離関数に基づき、指定ポケモンとそれ以外との距離を計算する；
 - － 距離が最小のポケモンインスタンスと距離の値をタプルとして返す；
- 距離のラムダ式を `calc_similar_monster()` の引数として渡すことで、各距離定義に基づく最類似ポケモンを求める；
 - － 3 つの距離定義に基づき、引数として渡された 2 つのリスト間の距離を計算するラムダ式を定義する；
 - － 距離の名前（「Manhattan」など）をキー、距離のラムダ式を値とした辞書を構築する；
 - － 辞書に対する `for` 文の中で、ラムダ式と自分のポケモンを引数として指定して `calc_similar_monster()` を呼び出す；
 - － 戻り値として返ってきた最類似ポケモンとその距離を標準出力する；
- `pokemon.py` には変更を加えない；
- 種族値ファイル「base_stats.txt」と名前ファイル「poke_names.txt」を読み込む関数は、リスト 2 を使用する（Moodle にある `pokemon.py` にすでに書かれている）；
- (1) 種族値ファイルのパス、(2) 名前ファイルのパス、(3) 自分のポケモン番号は、コマンドライン引数により、この順番で指定する；
- 出力の形式は、図 1 のようにする；

```

PS C:\Users\admin\Desktop\ProA2> python .\lec03\kadai06\distance.py .\lec03\data\base_stats.txt .\lec03\data\poke_names.txt 2
僕のポケモン：フシギソウ
Manhattan距離：一番類似するのはベイリーフで、距離は34
Euclidean距離：一番類似するのはクサイハナで、距離は22.538855339169288
Chebyshev距離：一番類似するのはフシギダネで、距離は15
PS C:\Users\admin\Desktop\ProA2> python .\lec03\kadai06\distance.py .\lec03\data\base_stats.txt .\lec03\data\poke_names.txt 10
僕のポケモン：キャタピー
Manhattan距離：一番類似するのはビードルで、距離は20
Euclidean距離：一番類似するのはビードルで、距離は10.0
Chebyshev距離：一番類似するのはビードルで、距離は5
PS C:\Users\admin\Desktop\ProA2> python .\lec03\kadai06\distance.py .\lec03\data\base_stats.txt .\lec03\data\poke_names.txt 144
僕のポケモン：フリーザー
Manhattan距離：一番類似するのはカメックスで、距離は50
Euclidean距離：一番類似するのはスイクンで、距離は23.452078799117146
Chebyshev距離：一番類似するのはスイクンで、距離は15
PS C:\Users\admin\Desktop\ProA2>

```

図1 課題06の実行例：

リスト2 pokemon.py

```

1 def read_stats(file_path):
2     stats = []
3     with open(file_path, "r") as rfo:
4         for row in rfo:
5             row = row.rstrip()
6             stats.append([int(col) for col in row.split(" ")])
7     return stats
8
9 def read_names(file_path):
10    names, types, evols = [], [], []
11    with open(encoding="utf8", file=file_path, mode="r") as rfo:
12        for row in rfo:
13            _, nam, typ, *evo = row.rstrip().split("\t")
14            names.append(nam) # 名前の文字列をappend
15            types.append(typ.split(" ")) # タイプのリストをappend
16            evols.append([e for e in evo]) # 進化先のリストをappend
17
18    return names, types, evols
19
20
21 class Monster:
22     def __init__(self, title):
23         self.title = title
24         self.stats = []
25
26     def __repr__(self):
27         return self.title

```

リスト 3 distance.py

```
1 from pokemon import *
2 import sys
3
4 def calc_similar_monster(dist, target):
5     minval = 100000
6     for mon in monsters:
7         if mon == target:
8             continue
9         d = dist(target.stats, mon.stats)
10        if minval > d:
11            minval = d
12            argmin = mon
13    return argmin, minval
14
15
16 if __name__ == "__main__":
17     stats = read_stats(sys.argv[1]) # lec03/data/base_stats.txt
18     names, _, _ = read_names(sys.argv[2]) # lec03/data/poke_names.txt
19     monsters = [Monster(title) for title in names]
20     for i, mon in enumerate(monsters):
21         mon.stats = stats[i]
22     my_monster = monsters[int(sys.argv[3])-1] # 自分のパートナーポケモンの番号
23     print(f"僕のポケモン: {my_monster}")
24
25
26     dist_funcs = {"Manhattan": lambda lst_x, lst_y: sum([abs(x-y) for x, y in zip(lst_x, lst_y
27         )]),
28         "Euclidean": lambda lst_x, lst_y: sum([(x-y)**2 for x, y in zip(lst_x, lst_y
29         )])**0.5,
30         "Chebyshev": lambda lst_x, lst_y: max([abs(x-y) for x, y in zip(lst_x, lst_y
31         )])
32     }
33
34     for dist_name, dist_func in dist_funcs.items():
35         argmin, minval = calc_similar_monster(dist_func, my_monster)
36         print(f"{dist_name}距離:一番類似するのは{argmin}で,距離は{minval}")
```
