目次へ

指南編(県の鳥)

最初のかた

できた。

```
# coding: cp932
o = open("birds.html", "w")
print >>o, """<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">
<title>県の鳥</title>
</head>
<body>
<h1>県の鳥</h1>
<table border="1">"""
list=[]
for line in open("birds.txt"):
  a = line[:-1].split("¥t")
  list.append(a)
list.sort()

area=0
for a in list:
  if a[0] > area:
    area=a[0]
    print >>o, "<tr>"
    print >>o, "<td bgcolor=yellow>%s</td>" % (a[0])
    print >>o, """</tr><tr>"""
    print >>o, "<td>%s</td><td>%s</td>" % (a[1], a[2])
    print >>o, "</tr>"
  else:
    print >>o, "<tr>"
    print >>o, "<td>%s</td><td>%s</td>" % (a[1], a[2])
    print >>o, "</tr>"
```

```
print >>o, """</table>
</body>
</html>"""

o.close()
```

お疲れ様です。筆者の環境でもちゃんと動作して、birds.htmlというHTMLファイルを生成できました。表示内容もばっちりです。

ではスクリプトを見てまいります。

<html> で始まって、 <table border="1"> で始まるところまでが、ひとまとめの文字列として一度に出力されています。

で、birds.txtを一行ずつ読み込みながら、まずは list という名前のリストに順番に読み足していっています。（listという名前でリストを作るのは、もとからあるlistという特殊な関数を塗りつぶしてしまうので、避けておくほうが無難です。実は筆者はこれが動くのを見て、むしろ驚きました。pythonの融通無碍さはすごいな。）

っていうか、'list'って変数を使わないほうがいいよ、って、別のところでも言ったっけ？

ここでは、listに足していくときに、splitを行ってしまってから足しています。これをやると、split自体がリストを返しますから、結果として「リストのリスト」というものが発生しているのですが、そこはご承知の上だったでしょうか？つまり下のようなものがlist変数には入っているのです。

```
[
  ['中国', '広島', 'アビ'],  ←これ全体が、一つ目の要素。
  ['近畿', '兵庫', 'コウノトリ'],  ←これ全体が、二つ目の要素。
  ['東北', '岩手', 'キジ'],  ←これ全体が、三つ目の要素…
  ...
]
```

こんな、「リストのリスト」でもsortできたということに、ちょっと驚いてくださいね。'中国', '近畿', '東北'などの、各々最初の要素がソートの手がかりになりました。（説明してて、ちょっと分かりにくい気もしてきます。まあ、軽く読み飛ばしても可）

で、リストにファイルの中身を全部溜め込んでソートもしたところで、次はこいつをひとつづつテーブルの列に整形して表示していくところですね。

areaという名前の一時変数を設けてくれました。これが、「直前はどこの地域を表示したっけ」というメモ代わりですね。で、if a[0] > area: という条件分岐が、「さっきと違う地域が出現したら…」という判断を行っているわけです。この条件を満たすときは、メモの内容を更新する(area=a[0])ことも怠りありません。これで期待通りの動きが実現できました。

if a[0] > area: という条件分岐は、小さい順にソートをしたのだから確かに有効ですが、「より大きい」という条件ばかりにこだわることもないので、if a[0] != area: という条件にしておくほうが安全でしょうね。大きいか小さいかにかかわらず、違ってさえいれば「見出し行」を出力する条件として充分ですので。

%s をつかった文字列の生成についても、よく呑み込んでいるようで安心です。

改良してみた

「北海道」がトップに来るほうがいいと思って、ちょっと工夫した。

```
# coding: cp932

o = open("birds2.html", "w")
print >>o, """<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">
<title>県の鳥</title>
</head>
<body>
<h1>県の鳥</h1>
<table border="1">"""
bird_list=[]
for line in open("birds.txt"):
  a = line[:-1].split("¥t")
  if a[0] == "北海道":
     a.insert(0, 1)
  elif a[0] == "東北":
     a.insert(0, 2)
  elif a[0] == "関東":
     a.insert(0, 3)
  elif a[0] == "北陸":
     a.insert(0, 4)
  elif a[0] == "東海":
     a.insert(0, 5)
  elif a[0] == "近畿":
     a.insert(0, 6)
  elif a[0] == "中国":
     a.insert(0, 7)
  elif a[0] == "四国":
     a.insert(0, 8)
  elif a[0] == "九州":
     a.insert(0, 9)
  bird_list.append(a)

bird_list.sort()

area=0

for a in bird_list:
  if a[0] !=area:
     area=a[0]
     print >>o, "<tr>"
     print >>o, "<td bgcolor=yellow>%s</td>" % (a[1])
     print >>o, """</tr><tr>"""
     print >>o, "<td>%s</td><td>%s</td>" % (a[2], a[3])
     print >>o, "</tr>"
  else:
     print >>o, "<tr>"
     print >>o, "<td>%s</td><td>%s</td>" % (a[2], a[3])
     print >>o, "</tr>"
print >>o, """</table>
</body>
</html>"""

o.close()
```

工夫大歓迎です。これを実行すると、ちゃんと直感的に正しい感じの地域順でリストが作られるようになりました。これを可能にしたしくみを見てまいります。

さっきまでの例では、ソートに使う手掛かりが、"北海道" とか "東北" とかいう文字列そのものでした。pythonにとっては、漢字の並べ替えもある種の「辞書順」に従います。でもこっちの期待する順番とは違っていました。こいつを、こっちの望む順番に直してやろうとしたのですね。で、北海道は 1 、東北は 2 、というように数字を割り当てることにしました。こっちが並べ替えの手掛かりになればいいと踏んだわけですね。実に的確です。

さっき挙げたリストの中身の例でいえば、下のようにソートすべきデータを直してしまった、と。

```
[
  [7, '中国', '広島', 'アビ'],
  [6, '近畿', '兵庫', 'コウノトリ'],
  [2, '東北', '岩手', 'キジ'],
  ...
]
```

こうしたために、7, 6, 2といった数字が、一行づつを並べ替えるときの最初の手掛かりとしてpythonには採用されるわけです。

これを実現するために、リストに対して insert という命令を発行できることを自力で調べてしまったのですね。今まで紹介していた append ではリストのおしまいにデータが足されるだけですから、ソートの手掛かりとしては優先度が一番低くなります。リストの先頭にデータを足したかった。そうですね、こいつを実現するための命令はinsertです。insert(0, 何か) という命令は、リストの0番目の値を「後ろに押しのけて」指定の何かを付け足すことを意味します。

ここまでやったなら、ついでに、「北海道なら1、東北なら2...」という変換を実現するために、if, elif をたくさん並べずに、辞書を使うとよりスマートな感じにできますから、挑戦してみませんか。

改良してみた(2)

辞書を使ってみたよ

```
# coding: cp932
o = open("birds.html", "w")
print >>o, """<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">

<title>県の鳥</title>
</head>
<body>
<h1>県の鳥</h1>
<table border="1">"""

bird_list=[]

a_dic={"北海道": 1, "東北": 2, "関東": 3, "北陸": 4, "東海": 5, "近畿": 6,
 "中国": 7, "四国": 8, "九州": 9}

a_list=a_dic.keys()
```

```
for line in open("birds.txt"):
  a=line[:-1].split("¥t")
  for a_dic_key in a_list:
    if a[0]==a_dic_key:
      a.insert(0, a_dic[a_dic_key])
  bird_list.append(a)

bird_list.sort()

area=0
for a in bird_list:
  if a[0] != area:
     area=a[0]
     print >>o, "<tr>"
     print >>o, "<td bgcolor=yellow>%s</td>" % (a[1])
     print >>o, """</tr><tr>"""
     print >>o, "<td>%s</td><td>%s</td>" % (a[2], a[3])
     print >>o, "</tr>"
  else:
     print >>o, "<tr>"
     print >>o, "<td>%s</td><td>%s</td>" % (a[2], a[3])
     print >>o, "</tr>"
print >>o, """</table>
</body>
</html>"""

o.close()
```

"北海道"なら1、"東北"なら2…というパターンが出てきたら、辞書にそのルールを丸ごと入れておいて使えばいいわけですね。まさにこのとおりです。

っていうか、筆者は、ホントに「辞書使ってみるといいよ」とアドバイスしただけ。それだけでこれが書けたんだからエライなあ。

下の部分については、苦心のあとが感じられました。

```
for line in open("birds.txt"):
  a=line[:-1].split("¥t")
  for a_dic_key in a_list:
    if a[0]==a_dic_key:
      a.insert(0, a_dic[a_dic_key])
  bird_list.append(a)
```

「北海道」だったら1、「東北」だったら2…というふうにデータを作りたいんだけど、1から9以外の数字が出てきたときは a_dic[a_dic_key] という値の呼び出しがエラーを起こしてしまう。これを避けようとして、いちいちキーの存在をチェックするというコードを書いたのですね。for a_dic_key in a_list: のループの部分。

テキスト内ではまだ説明していない、辞書に対する has_key という命令（メソッド）や、get という命令（メソッド）を使えばもっとうまく書けるのですが、これナシで補おうと工夫するところに感服します。いずれこれらの命令を説明をするときには、「こいつを使えばよかったのか」と思い出してくれると幸いです。

別のかた

> 地域ごとにまとめるのが難しかった

```
# coding:cp932

zone =["北海道","東北","北陸","関東","東海","近畿","中国","四国","九州"]

print """<html>
<header>
<title>県の鳥</title>
<style type="text/css">
table,TH,TD{border-collapse:collapse;
            border-style:solid;
            border-color:black;
            border-width:2px;}
TH {background-color: yellow;}
</style></heder>
<body><table border="3">
"""

try:
  while 1:
    area = zone.pop(0)
    print "<tr><TH>" + area + "</TH></tr>"
    for line in open("birds.txt"):
      a = line[:-1].split("¥t")
      if a[0] == area:
        print "<tr><td>"+ a[1]+"</td><td>"+a[2]+"</td></tr>"

except IndexError:
  print "</table></body></html>"
```

回答を寄せてくださってありがとう！拝見します。

今までの例では、「県の鳥」ファイルを全部リストにいったん読み込んでしまうという特徴がありました。筆者もこれで充分かな、と今のところは考えていましたが… もし元のデータが100万行とかそんな感じだったら、「いったん全部読んで溜めこむよ戦法」だと、いっぱいメモリを消費してしまうことが予想されます。環境によっては耐え切れずにプログラムが異常終了してしまうかもしれません。

この回答は、それを独特の方法で回避したようです。（ひょっとすると、別にそういう意図はなかったかな？）

try, except, while, pop など、まだテキストで触れていない命令が駆使されていますので、この回答が表現した手順上の戦略だけを簡単に確認しておきます。（あ、あと、HTMLにスタイルシートが使ってある。知らない人に一言だけ解説しておくと、HTMLの表示を「化粧」するためのワザですね。styleタグのあたり。）

で、その戦略ってのは下のよう。

- 北海道、東北、北陸…と、ひとつづつ「今回狙う地域」を選んで、そのたびにデータファイルをすべて読み込む。狙ってないデータは読み捨てる。
- 「今回狙う地域」については、最初に地域名のリストを準備しておいて、こいつを一つずつ消費しながら順に指定する。
- 地域名のリストが順に短くなっていってついに「長さゼロ」になると、pop命令に失敗する。ここでexcept文にジャンプして下に実行を続ける。

回答といっしょに、「地域リストを漏れなく準備するのに困った」という感想が挙げてありました。確かに、あらかじめデータを全部観察してみないと、現れる可能性がある地域名がわからないですもんね。

余談ですが、こういうときはsetオブジェクトを使ってあらかじめ調べておくというのも手かな、と思います。下にサンプルスクリプトを挙げておきます。setも未出ですので特に説明はしませんが。

```
zones = set()
for line in open("birds.txt"):
  a = line[:-1].split("¥t")
  zones.add(a[0])
for i in zones: print i,
```

あと、余談その2。これも本テキスト内で紹介してない話なので、用語を知らない人は読み飛ばしてもいいんですが...

この回答は、HTMLのコードを単にprintでコンソール上に吐き出しているだけですが、たぶんリダイレクトして使うことを想定しているんですね。コマンドプロンプトから実行するなら、「＞」を使って適当なHTMLファイルにリダイレクトすることでこの出力をファイルに書き出すことができますからね。

もういっこ、このスクリプトをファイル出力用にしてしまうための、伊藤家の荒技があります。最初にprint文が現れるまでの適当な場所（この場合だとzoneを定義した直後）に、下の二行を書き足してみると...

```
import sys
sys.stdout = open("birds.html", "w")
```

これ以降のすべてのprint文が、自動的に"birds.html"への書き出しに化けてしまいます。「標準出力をファイル出力にジャック」しちゃったんですね。書き捨てタイプのいいかげんなスクリプトを作るときにときどき筆者はこれを使いますが、色々紛らわしくなるので一般的にはオススメしません。あくまで余談、でした...