目次へ

# 県の鳥リスト(HTML)を作ってみる

さて、ここまで読み進んだ方は、HTML文書を作る基本をすでに習得されたものとみなしますよ。よいでしょうか。

ここでやってみようと思うのは、pythonがHTMLを自動生成するという仕事の基本です。

たとえば、下のようなファイルがあったとして…（タブ区切り形式のテキストファイルです。エンコードはシフトJIS)

birds.txt

こいつをもとに、下のような表示をするHTMLをこしらえてみよう、ということです。

# 県の鳥

| | | |
|---|---|---|
| 中国 | 広島 | アビ |
| 近畿 | 兵庫 | コウノトリ |
| 東北 | 岩手 | キジ |
| 近畿 | 京都 | オオミズナギドリ |
| 関東 | 群馬 | ヤマドリ |
| 北陸 | 富山 | ライチョウ |
| 東北 | 青森 | ハクチョウ |
| 中国 | 山口 | ナベヅル |
| 中国 | 島根 | オオハクチョウ |
| 近畿 | 三重 | シロチドリ |
| 北陸 | 新潟 | トキ |
| 四国 | 徳島 | シラサギ |
| 東海 | 長野 | ライチョウ |
| 近畿 | 滋賀 | カイツブリ |

なんか、出現する県の順番がバラバラですが...これはわざとです。あとでこれを並び替えたりする練習がある、かもしれないよ。

## 参考例

この例については、スクリプトを示しながら、どんな手順を踏むとできるかを見てみましょう。

まず、作りたいものはHTMLファイル。HTMLファイルはつまるところテキストファイルなんだから、下みたいな方法で作ることになるでしょう。

```
o = open("birds.html", "w")
...
print >>o, "...てきとうなHTML..."
...
o.close()
```

こんな感じですね。

o.close() って何？　という疑問があるかも知れません。今までは、＞＞o に向かっていろいろとprintすると、勝手にファイルが書き出されると想定していました。これはこれで正しいのですが、厳密には最後に「書き出しが終了したよ」という処理をしなくてはいけなかったのです。まあpythonはそこらへん融通が利きますから、スクリプトが終わった瞬間に自動的にこの処理はやってくれるんですけど、人によっては「プログラマがちゃんと責任持って終わりまで処理すべ

し」という主張を持ってたりもします。もしマジメにこの処理をするとすれば、書き出したいものがすべて終わったら、このような close() っていう命令を実行します。今回はあえて、openとcloseを一対にして書いてみました。

で、HTMLの「お決まりの部分」は、openしてからcloseするまでのタイミングで、下のようにファイルに書き出すんですね。

```
print >>o, '<html>'
print >>o, '<head>'
print >>o, '<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">'
print >>o, '<title>県の鳥</title>'
print >>o, '</head>'
print >>o, '<body>'
...
...
print >>o, '</body>'
print >>o, '</html>'
```

なんか、何行も同じようなものを書かされてめんどい...

ここはちょっとだけ楽にやる方法があります。「"""」という、ダブルクォートを三つ重ねた記号で囲むと、途中の改行も含めてひとつの値として扱ってくれるのです。下のような感じになります。

```
print >>o, """<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=shift_jis">
<title>県の鳥</title>
</head>
<body>"""
...
...
print >>o, """</body>
</html>"""
```

「"""」で始まって「"""」で終わるまでに改行が入ってもいいですから、表示される予定のHTML部分をそのままスクリプトの中に書き込んでしまえばよくて、楽でよいですね。このときは、「"」を「\"」に書きなおしてあいまいさを避けなくちゃ...とかいった「エスケープ」の処理も忘れて、どしどし書いてしまっていいです。

さて、表示する中身のうち、でかでかと「県の鳥」って表示するのは<h1>で行きましょう。下みたいにすればいいですね。

```
print >>o, "<h1>県の鳥</h1>"
```

残るは表本体です。だいたい、下のような感じの出力を目指します。

```
<table border="1">
  <tr>
    <td>中国</td><td>広島</td><td>アビ</td>
  </tr>
  <tr>
    <td>近畿</td><td>兵庫</td><td>コウノトリ</td>
  </tr>
  ...云々...
</table>
```

データファイルは、一行あたりが「中国,広島,アビ」といった情報を含んでいますので、これひとつについて<tr>から</tr>までを出力すればよさそうですね。

で、この繰り返しが始まる前に <table> を一回だけ出力して、繰り返しが終わったあとに </table> を一回だけ出力する。これで大体できあがるでしょう。

これらを踏まえて作成したスクリプトは、全部で下のような感じになります。

```
# 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>
"""

for line in open("birds.txt"):
  a = line[:-1].split("¥t")
  print >>o, "<tr>"
  print >>o, "<td>%s</td><td>%s</td><td>%s</td>" % (a[0], a[1], a[2])
  print >>o, "</tr>"

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

o.close()
```

スクリプトの中に日本語が混じるときは、一行目に #coding: cp932 というオマジナイを混ぜるのを忘れずに。まあ、いつもこの一行を書く、と覚えておいてもいいですね。

ここまで読み進めた方なら、別に何の問題もない簡単なスクリプトに見えると思いますが、いかがでしょう。

## もうちょっと凝りたいな

さて、練習問題としては、この表示をもうちょっとだけ凝ったものに直してみたいと思います。具体的には、さっきまでの表が下のようになったらいいな、と。

# 県の鳥

| 関東 | |
|---|---|
| 茨城 | ヒバリ |
| 群馬 | ヤマドリ |
| 埼玉 | シラコバト |
| 神奈川 | カモメ |
| 千葉 | ホオジロ |
| 東京 | ユリカモメ |
| 栃木 | オオルリ |
| 近畿 | |
| 京都 | オオミズナギドリ |
| 三重 | シロチドリ |
| 滋賀 | カイツブリ |

変化したところは、

- 地域ごとにグループ化された（同一地域内での並び順は、今回はどうなっていてもよい）
- 地域のアタマに、黄色い背景でラベルを表示した

ってところです。地域が出てくる順は、北海等、東北、関東...とかそんな順番のほうがそれっぽいんですが、今は適当な並び順でいいです。もっとはっきりいうと、今から sort というリストの並び替え命令を紹介しますが、これが自動で並び替えをやってくれるままに任せればよいということです。

## ソート（並べ替え)

さて、リストは、並べ替えることができます。下の例を対話シェルで打ち込んでみましょう。

```
>>> a = [46, 20, 9, 33, -6, 322, 61]
>>> a.sort()
>>> a
[-6, 9, 20, 33, 46, 61, 322]
```

これが、リストをsortするということです。a変数にリストが入っていたら、a.sort() という感じに命令を発行します。aの中が、小さい順になりましたね。

文字列にも大小関係があることをすでにご存知のはずですね。だから下のようなリストも小さい順に並べ替えることができます。

```
>>> b = ['sunday', 'monday', 'tuesday', 'wednesday']
>>> b.sort()
>>> b
['monday', 'sunday', 'tuesday', 'wednesday']
```

ここでは辞書順に並びました。文字列の大小関係は、辞書順のことです。

さて、今回使うデータファイルは、冒頭にも挙げましたとおり、　　birds.txt ですね。並び順がわざとバラバラですね。こいつをまずは並べ替えたい、というところから、色々スタートします。

あまりくだくだしくヒントを言いませんよ。つまり、最初にファイルを読むときは、ほかに処理をしないで適当なリストの中にappendで溜め込んでしまえばいいわけです。そしたらあとでsortできる。で、リストに対しても下のような「いっこづつ取り出して処理」という仕事ができるんだったと思い出してください。

```
# ファイルも、リストも、forを使った繰り返し処理が書ける
a = [1, 2, 3, 4, 5]
for num in a:
  print num
```

（pythonの中でソート処理なんかしないで、あらかじめExcelにでもコピペしてソートしておきゃあいいじゃないか、というアイデアも実はあります。今回はあえてこのワザを使いませんが）

## 見出しのタイミング

次に、例では黄色いマスで示したように、地域のカタマリがはじまるタイミングで余分なHTMLを書き出しておきたいときはどうするか。

すでに、行はソートされているとします。だから、隣同士の行では同じ地域がくっついている見込みが高い。なので、「直前と違う地域が出てきたぞ」というタイミングをどこかで見つければいいわけですね。下の擬似コードを参考にしてみてください。

```
for d in some_list:
  if [直前に扱った行と比べて地域が変わってるみたい]:
    print "見出しになる表示"
  ...ここからは通常のデータ行表示...
  [今回のループではこの地域を扱ったぞ、とどこかにメモして、次のループに備える]
```

どうかな。できるかな。できてほしいな。というわけで、

## 練習問題04

**【練習問題：04】今までに説明したとおり、最後の表示例を実現するようなHTMLをつくるスクリプトを書け。データファイルは、　birds.txtである。**

※データファイルは、右クリックから「リンク先を保存」「対象を保存」などを選んで保存してください。