目次へ

練習問題・残業時間ファイナル

架空のお話

ボス「忙しいかね」
あなた「はあ、まあまあで」
ボス「君に至急頼みたいことがある」
あなた「どうしたんです」
ボス「このデータを見よ。去年度一年分の、社員の退社時間の記録だ」

leavetime3.txt

あなた「はあ」
ボス「このなかでもっとも残業時間が多いものを探したい」
あなた「定時は5時20分でしたっけ。どれどれ、わあ、午前様な人までいるよ。かわいそうだなあ」
ボス「余計なことは言わんでよい。ひと月あたりの合計残業時間が最も多いものは誰だ。また、それは何月のことか」
あなた「その答えはいつまでに必要ですか」
ボス「すぐだ。一時間で答えを出せ」

という謎の任務に応えようというのが、今回の最終的な目的です。まあ、一時間以上かかってもいいですけど。

最初は観察

さて、何はともあれ、データがどんな形になっているかを観察することにしましょう。

```
20080401,Ito,21:10
20080401,Kobayashi,17:20
20080401,Nakamura,17:20
20080401,Saito,18:00
...
```

まず、データはコンマ記号「,」で分けられて一行ずつの単位で成り立つもののようですね。これが、いわゆるCSV(Comma Separated Value)と呼ばれるものです。

一行は、三つの部分に分けられるようです。最初が年月日。2008年4月1日なら20080401という8桁の数字にまとめられているようです。次が従業員名、最後が退社時刻ですね。

任務は、「月ごと・人ごとの集計」ですから、「何年何月何日」のうち「何日」は省いてもよさそうです。20080401という日付が与えられたら、200804の部分だけ使えばよさそう。

最初の一行分を、対話シェルをいじりながら、どう処理するか検討しましょう。

```
>>> s = "20080401,Ito,21:10¥n"
>>> s[:-1]       ←改行記号は省く…
'20080401,Ito,21:10'
>>> s[:-1].split(",")      ←コンマ記号を使ってsplit
['20080401', 'Ito', '21:10']      ←三つに分かれるね
>>> a = s[:-1].split(",")
>>> a[0]
'20080401'
>>> a[0][:6]  ←日付データの最初の6桁だけ使う書き方はこう
'200804'
```

とりあえず、一行づつデータを読み込みながら、「年月、名前、残業時間」をそれぞれ得ることはできそうですね。

月ごと、人ごと

さて、前回までの練習問題では、「人ごと」の集計のために辞書を使いました。

今回は、それだけでは済みません。YamamotoさんならYamamotoさんの4月分の残業時間集計が必要です。プログラム的にはどう表現しましょうねえ。

いくつか方法はありますが、ここでは一番安易なやりかたを採用してみましょう。安易というと聞こえは悪いですが、「シンプルに済ます」と言い換えればよりいい感じでしょうか。シンプルさはきわめて大事です。

どうシンプルにするかというと、辞書にこんなキーを使って値を格納しようというのです。

```
'201004,Yamamoto'
```

二つの属性（年月、名前）をコンマ記号でくっつけた文字列を、辞書のキーにする。こうすれば、Yamamotoさんの4月の集計と5月の集計は別のキーを持ちますから、辞書にそれぞれ入りますね。

従業員が五人いて、12ヶ月分の集計を作るとすれば、5*12 = 60 個のキーをもった辞書を管理すればいいわけです。この方法を今回は採用しましょう。

こんな感じのキーを作るには、「文字列の足し算」を使えばよいでしょう。下の例を参考に。

```
>>> name = 'Yamamoto'
>>> yearmonth = '201004'
>>>  k = yearmonth + "," + name
>>> k
'201004,Yamamoto'
```

最大値を得る

さて、データをひととおり処理しおわって、集計結果が辞書に入った状態になったとします。今回の任務は、この中で誰の何月分が「最大」だったかを調べよということでした。

辞書の内容をそのまま全部printしてしまって、あとは人間が眺めて調べても、まあ悪くはありません。ただ、従業員が100人くらいいたとしたら、これは苦痛な作業です。間違いも生じるでしょう。ここでは最大値を得る作業もスクリプトを使って表現しておきたいところです。

どうやるか。これはプログラムのテクニックというより、人に作業を頼むとしたらどういう指示になるかなあ、と想像してみると、うまくいくものです。

- ひとつづつ、辞書の中身を改めよ。順番は問わないが、必ず全部調べるよう注意せよ。
- ただし、それをはじめる前に、手元にメモを準備せよ。メモには、最初にゼロと記しておけ。
- 辞書の中身をひとつ調べて、その「値」の部分が手元のメモ上の数値より大きかったら、メモの内容をそれに更新せよ。そのときの「キー」も併記しておけ。そうでなかった場合は何もしなくてよい。
- 辞書の中身をすべて調べ終わったときに手元に残っているメモが、つまり最大値である。

どうでしょう。いかにも「手取り足取り」な感じの指示ですが、プログラムを書くというのはこんなものです。

辞書の内容をひとつづつ調べるには、これも何種類か方法がありますが、ここでは keys を使った方法を採りましょう。

```
>>> a = {'Yamamoto': 100, 'Okada':200, 'Sato':300}    ←テスト用に適当に辞書を作った
>>> k = a.keys()
>>> k
['Yamamoto', 'Okada', 'Sato']
```

辞書.keys() という書き方をすることで、キーだけの一覧をリストの形で得ることができます。この例では、こいつを変数 k に入れました。

で、kの中身をひとつづつ使って何かをする基本パターンがこうです。

```
>>> for i in k:
...   print i

Yamamoto
Okada
Sato
```

こいつを少し書き換えて、ついでに辞書の中身もひとつづつ表示しようと思えば、こうです。

```
>>> for i in k:
...   print i, a[i]

Yamamoto 100
Okada 200
Sato 300
```

おわかりでしょうか。k には、a辞書のキー一覧が入った。それをひとつづつ使いながら、もとのa辞書にキーを指定して値を調べるわけなので、この例のような結果が得られるわけです。

かなり基本的なテクニックなので、よく納得しておいてください。

あとは、これを応用して、最大値をどう算出するかを工夫してみましょう。

練習問題03

さあ、ここまで説明すれば、あとはスクリプトを組み立てるのみです。

【練習問題：03】退社時間ファイル・バージョン3(leavetime3.txt)をすべて読み込んで、（年月, 従業員）ごとに残業時間の集計をし、月単位で最長時間の残業は誰が何年何月に行ったものかを調べ、答えよ。後の模範解答では、スクリプト名を zangyo_sum3.py とする予定である。

　leavetime3.txt ←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。

※このデータは、ランダムな値をもとに作ったもので、完全にフィクションです。