目次へ

指南編（残業時間）

最初のかた

ここまで書けた。

```
for s in open("test.txt"):
  h=int(s[0:2])
  m=int(s[3:5])
  zangyo=h*60+m-1040
```

一日ごとの残業時間を足していくのは、どうやったらいいかわからない。

はい、ここまではとても結構です。

残業時間のファイルは、test.txt と変更して使っているのですね。ここは構いません。 で、これを一行ずつ読み込んで、

h には、その行の頭二文字分を切り出して、それを整数とみなしたものを入れる。
m には、その行の4～5文字目を切り出して、それを整数とみなしたものを入れる。

で、zangyo には、退社時間の零時からの経過分数と、「17:20」の零時からの経過分数である1040との差を計算して、残業した時間（分換算）を入れました。

pythonは、字下げしたこの３行分の命令を、ファイルから一行ずつ読み出すごとにいちいち実行するわけです。
ですので、そのタイミングを捕まえて、「合計」をあらわす変数にzangyo変数の中身を足しこむ処理を入れればできそうです。

その「合計」をあらわす変数は、名前を決めて、最初のデータを読み出す前に、あらかじめ0を入れておきましょう。今回は、for文の前です。

こんなところでどうでしょう。

わかった。ここまで書けた。

```
g=0
for s in open("leavetime.txt"):
  h=int(s[0:2])
  m=int(s[3:5])
  zangyo=h*60+m-1040
  g=zangyo+g
a=g/60
b=g%60
print a, "hours", b, "minutes"
```

合計を入れるための変数を g として、足しこみ処理を書き足してくれました。

```
g = zangyo + g
```

がミソですね。変数の中身を増加させるときのお決まりの書き方です。

で、forのループがひととおり終わったところで字下げをもとに戻して、a と b にそれぞれ「時間、分」に直したときの値を入れて表示させていますね。

これでよいです。

実際の残業時間合計は、各自確かめておいてくださいね。10000時間を少し超えたくらいの値になっているはずですよ。こんなものを手計算でやってたら日が暮れます。10行たらずのpythonスクリプトでみごとに楽しましたね。

べつのかた

できた。

```
kei = 0
for my_line in open ("leavetime.txt"):
    my_line = my_line[:-1]
    i = my_line.find(":")
    h = int( my_line[:i] )
    if h < 17:
```

```
        h = h +24
    m = int( my_line[i+1:] )
    kei = kei + h*60 + m - (17*60+20)

print kei/60, "hours", kei%60, "minutes"
```

お疲れ様です。たいへん結構です。

my_line = my_line[:-1] として、一行ごとに改行記号が含まれるのをあらかじめ取り除いてくださいました。

その後、頭から二文字分が時間、コロンを挟んで次の二文字が分、ということを必ずしも予断しない書き方を採用し、ちゃんとコロン記号がどこにあるかを find で探し、丁寧に区切り処理を行ってくれました。これだと、02:30 というデータが来ても、2:30 というデータが来ても対応ができます。

出色なのは、 if h < 17: h = h + 24 という処理を書き足してくれているところです。今回のサンプルデータには含まれませんが、午前零時を過ぎた出社時間も対応できるようになりました。

kei = kei + h*60 + m - (17*60+20) という部分は完全に正しく動作しますが、kei = kei + (h*60+m) - (17*60+20) とカッコをひと揃い余分に書き足すと、（退社時刻）-（残業開始時刻）という表現がより明快になってよいかな、と思いました。ここらへんは好みの問題ではありますが。

べつのかた

（俺はこれとは違ってこんな風に書けたぞ、という方は、別回答としてどしどしお寄せください。）