目次へ

営業日計算・指南編

苦心されている方もいるようですが。

最初の方

全問クリアしたよ(関数の中身のみ引用)

```
def get_n_business_days_forward(d, n):
  while n>0:
    d+=datetime.timedelta(1)
    while (d.weekday() >= 5) or (d in holidays):
      d+=datetime.timedelta(1)
    n-=1
  return d
```

はい、こちらの環境でもこの実装で全問クリアできました。お疲れ様です。

つくった方針を見ると、「1営業日進む」という処理をまず完成させてから、それをn回繰り返すという手順をとったことがわかります。

まずこの部分...

```
    d+=datetime.timedelta(1)
    while (d.weekday() >= 5) or (d in holidays):
      d+=datetime.timedelta(1)
```

この3行が「1営業日進む」という記述に相当します。まずは d を一日進めてみて、あとは「d が土日祝日だったらさらに一日進める」という処理を、whileを使って、条件をみたすかぎり何日でも前方に滑っていくように作りました。

「土日または祝日」という表現はこれでよいですね。土曜日は weekday にして5, 日曜日は weekday にして6 ですからね。

で、あとは、ここを n 回繰り返せば「n営業日後」になりますね。つまり下のコードの部分です。

```
  while n>0:
    ...[1営業日進む]...
    n-=1
```

引数 n を順にひとつづつ減算していって、最後に 0 になったら（n>0 を満たさなくなったら）終了。なるほど。

とてもコードの意図がわかりやすい、鮮やかな回答と思います。

でもいくつか筆者なりのアドバイスをさせていただくと、

関数 get_n_business_days_forward の引数として与えられた d と n は、関数の中では変化させないほうがいいんじゃないかな、と（ちょっと個人的な意見ですが）思っています。n が関数の中で加算や減算されて変化すると、同じ関数の中でさらに n の値を使いたくなったときに、「これって引数のときの値と違っているんじゃないかな」という心配をしなくてはいけなくなります。d も同様。

n を直接減算させながらループを書くのでなくて、ここでは

```
  count = n
  while count>0:
    ...[1営業日進む]...
    count-=1
```

と、一旦別の変数に取り分けてからこいつを変化させるといいんじゃないかと思います。

d も、別の変数にコピーを作ってから（コピーという表現は実は微妙なんですが、今は見逃して！）、それを変化させて、最後にそれを return するのがよいですね。

まあ実際は、pythonだとここはたいした問題ではないのです。でもC言語などの他の言語では、今のような場合に引数の n を直接減算させると、関数を呼び出した側で使った変数も内容がそれ

に伴って変化してしまうということがありうるのです。いつもpythonだけを使うとは限りませんので、ここはまあ、望ましい習慣くらいにでも考えておいてください。

さらにいうと、引数を直接変化させるのがふさわしいときも、実はあります。そんなときがあったらまた説明します。

もうひとつ、n 回繰り返すという処理を while で書いてくれましたが、for を使うと一行少なく書けますよ。

```
  for i in xrange(n):
    ...[1営業日進む]...
```

range（ここではループ用に最適化された xrange）を使ってn回分のリストをつくり、それだけループしました。ループ中の値 i は別に使い道はありません。

あと、ここは筆者の欲張りですが、「1営業日進む」という処理で、timedelta(1) を足し算している記述が二ヶ所ありますね。これはこれで完全に正しいですが、ループ処理をあとほんのちょっと工夫すると、これを一回だけ書けばよくなりますよ。回答がいくつか出揃って、その時点でまだ紹介する余地があったら、それもお示ししましょう…

ともあれ、おみごとでした。

他のかたもどうぞ