目次へ

タプル

タプルの話って、普通はpythonのこと始めたらすぐに出てくるものなんです。

でもまあ、「リスト」の使い方だけ覚えておけば、別に無理にタプルを使わなくてはいけない場面ってそんなに出てきませんから、余分に覚えることを少なくしようかなと思って今までちゃんと説明しませんでした。

でも、既存のモジュールが処理結果としてタプルを返してきたりすることもあるわけですから、全然知らないというわけにもいきません。ということでタプルのことを簡単に説明してみようと思います。

実際のところ、なんということはなくて、タプルってのはリストの「変更できない」バージョンです。下の対話プロンプト上での実験をみてください。

```
>>> a = [1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
>>> a[2] = 100
>>> a.append(10)
>>> a
[1, 2, 100, 4, 5, 10]
>>> b = (1, 2, 3, 4, 5)
>>> b
(1, 2, 3, 4, 5)
>>> b[2] = 100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> b.append(10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```

a にはリストを入れてから、中身を変更したり、appendで値を追加したりしました。

b にはタプルを入れました。リストを作る時とちがって、[ ] のかわりに ( ) で囲みます。

で、b の中身を替えようとする命令を書いてみましたが、いちいちエラーを起こして怒られてしまっていますね。まあ、こういうことです。

変更するのはダメですが、そうでなければ、リストを使ってできるような処理は普通にできますよ。

```
>>> b = (1, 2, 3, 4, 5)
>>> for i in b:
...     print i*2
...
2
4
```

```
6
8
10
>>> b[2:4]
(3, 4)
```

タプルを使って繰り返し処理も書けましたし、[:] を使って一部を取り出すこともできました。ここらへんはリストの扱いと変わりがありません。

タプルの中身を変更したいような用事があるなら、新しいタプルを作って、それを丸ごと変数に入れなおせばいいです。

```
>>> b = (1, 2, 3, 4, 5)
>>> b = b[:-1]
>>> b
(1, 2, 3, 4)
```

まあ、こんなことするくらいなら最初からリストを使えばいいんですが。

タプルにあって、リストにない機能は多分ありません。（タプルは辞書のキーになることができる、という特徴が、あるにはあるか。リストは辞書のキーにならないからね）

じゃあそもそもタプルは何のために必要なんだ、という疑問が起こりそうな気がします。これについては、いろいろ思いつくところはありますが、pythonの内部とか、実行効率とか、そんな話になっていきそうです。正直、筆者もそんな詳しいところは語れません。なので、今のところは、タプルが出てきても「ああ、タプルだね」と思ってくれるだけでいいと思いますよ。

どうしてもタプルをリスト的に変更しながら使いたくなったときは、タプルからリストへの変換用関数がありますからこいつを覚えておきましょう。ついでにリストからタプルに変換するものも同時に示します。

```
>>> b = (1, 2, 3, 4, 5)
>>> c = list(b)
>>> c
[1, 2, 3, 4, 5]
>>> d = tuple(c)
>>> d
(1, 2, 3, 4, 5)
```

使う機会、あるかな。そんなにないと思うな。

タプルと文字列は似ている

文字列って、何番目かの文字を取り出すことができるし、[:] の指定で部分文字列を取り出すこともできますよね。それでいて、何番目かの文字だけを入れ替えることはできません。ここで挙げた「タプルの特徴」にダブるところが多いです。

```
>>> s = "ABCDEFG"
>>> s[3]
'D'
>>> s[4:6]
'EF'
>>> s[7] = 'Z'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

ね。

そこでふと思いついて、「文字列がタプルっぽいっていうんなら、他の特徴も似ているのかな」って感じで、まずはループができるのかを試してみます。

```
>>> s = 'HELLO'
>>> for i in s:
...   print i
H
E
L
L
O
```

おおー。

じゃあ、リストへの変換なんていうこともできるのかな。

```
>>> list(s)
['H', 'E', 'L', 'L', 'O']
```

おおー。

なんか不思議な感じですね。

もう一個だけ、文字列がタプルっぽい（変更さえしなければリストっぽく振舞う）という特徴を試して見ましょう。たしか、乱数の出し方のあたりで、リストから無作為にひとつ値を選ぶやつがありましたよね。

```
>>> from ramdom import choice
>>> cards = '123456789TJQK'
>>> choice(cards)
'4'
```

文字列から、ランダムに一文字を選びだすなんてこともできるんですねえ。

こんな特徴がどういう感じで使いこなせるかは、今のところすぐに例は出せませんが、ヒラメキ次第では面白いかもね。