ファイルからの読込みの方法

ここまでの学習から分かるように、Python でのテキストファイルに対する入出力は、文字の並びである《文字列》として行われます。

ファイルから文字列を読み込むための手段には、豊富なバリエーションがあります。それらを学習していきましょう。

read メソッド

ファイルの末尾である EOF (*end of file*) までを読み込んで、**単一の文字列**として返却するのが、**read メソッド**です。

```
lines = f.read()
print(lines, end='')
```

そのため、右のコードだけで、ファイルの内容がすべて表示されます（'chap13/list1304a.py'）。

▶ 省略可能な引数 size を与えた場合、最大で size バイトを読み込みます。省略時のデフォルト値は -1 であり、負の値あるいは None を受け取ると、ファイルの内容をすべて読み込みます。

readline メソッド

readline メソッドは学習しました（右に示すコードは、前ページのプログラムと同じです）。

```
while True:
    line = f.readline()
    if not line:
        break
    print(line, end='')
```

このメソッドは、行の先頭から改行文字までの 1 行分を読み込んで、**単一の文字列**として返却します。

▶ 省略可能な引数 size を与えた場合、最大で size バイトを読み込みます。省略時のデフォルト値は -1 であり、改行文字までの 1 行を読み込みます。

readlines メソッド

複数の行を読み込むのが **readlines メソッド**です。読み込んだ行は、**文字列のリスト**として返却します。

```
lines = f.readlines()
for line in lines:
    print(line, end='')
```

右のコードは、*lines* に代入されたリスト内の要素（文字列）を 1 個ずつ表示します（'chap13/list1304b.py'）。

▶ 省略可能な引数 hint を与えた場合、最大で hint 行を読み込みます。省略時のデフォルト値は -1 であり、すべての行を読み込みます。

ファイルオブジェクトからの直接の取出し

テキストファイルオブジェクトは、イテラブルオブジェクトであり（p.235）、for 文の走査によって行単位で取り出せる構造となっています。

```
for line in f:
    print(line, end='')
```

そのため、右のコードのように、簡潔に記述できます。なお、このコードは、高速かつ効率のよい方法として知られています（'chap13/list1304c.py'）。

ファイルへの書込みの方法

次は、ファイルへの書込みを行う方法を学習します。

write メソッド

既に学習したとおり、**write メソッド**は文字列を書き込んで、書き込んだ文字数を返却します。末尾に改行文字が出力されませんので、必要に応じて明示的な改行の出力が必要です。

writelines メソッド

writelines メソッドは、文字列のリストを書き込むメソッドです。**write** メソッドと同様に、末尾に改行文字は出力されません。

▶ **List 13-1** と同じように書き込むには、次のようにします（**'chap13/list1301a.py'**）。

```
f.writelines(['Hello!\n', 'How are you?\n'])
```

なお、ファイルへの書込みには **print** 関数も使用可能です。次に示すように、キーワード引数 **file** に書込み先のファイルオブジェクトを与えます（**'chap13/list1301b.py'**）。

```
print('Hello!\nHow are you?', file=f)   # 末尾に改行文字が自動的に出力される
```


Column 13-1 エンコーディングと UTF-8

文字コードと、その一種である ASCII について、**Column 7-3**（p.197）で学習しました。

Python でのテキストファイルの入出力は、その環境に適した文字コードを利用して行われるのが基本です。たとえば、日本語 MS-Windows では、シフト JIS コードを拡張した CP932 が利用されます。

ファイルの読み書きを行うプログラムを、特定の環境でのみ動作させるぶんには問題ないのですが、ある環境で書き込んだファイルを他の環境で読み込む、といった場合に正常な読み書きが行えません。

そのため、**oepn** 関数の第3引数（キーワード引数名は『文字をコード化する』意味の **encoding** です）に、文字コードを指定できるようになっています。以下に示すのが、その一例です。

```
'ascii'   'cp932'   'euc-jp'   'euc-jis-2004'   'utf-8'
```

'ascii' は英語用の文字コード ASCII であり、**'cp932'**、**'euc-jp'**、**'euc-jis-2004'** は日本語用の文字コードです（この他にも中国語、韓国語など、各国語用のエンコーディングがあります）。

（Python で読み書きするファイルではなく）Python のプログラムでは、最後の **'utf-8'** が利用されています。UTF-8 とは、多くの言語の文字を扱えるように作られた世界規格の文字コードであり、広く利用されています。

読み書きするファイルを多くの環境に対応させるには、UTF-8 の採用を検討すべきです。たとえば、**List 13-1** であれば、以下のように **encoding** 引数に **'utf-8'** を指定してファイルをオープンします。

```
f = open('hello.txt', 'w', encoding='utf-8')
```

既に学習したように、モード用の引数のキーワード引数名は **mode** ですから、明示的にキーワード引数を指定するのであれば、以下のように順序を変えることも可能です。

```
f = open('hello.txt', encoding='utf-8', mode='w')
```