佐藤 聖

8-1 ファイル操作の準備open関数/close関数

表1 ファイルをオープンするときのモード

文字	意味
r	読み出し用に開く（省略時のデフォルト）
w	新規作成，ファイルが存在する場合は上書き
x	新規作成，書き込み用でファイルを開き，ファイルが存在する場合は失敗
a	書き込み用に開き，ファイルが存在する場合は末尾に追記

（a）読み出し/書き込みの指定

文字	意味
b	バイナリ・モード
t	テキスト・モード（省略時のデフォルト）

（b）テキスト/バイナリの指定

文字	意味
+	既存ファイルの上書き

（c）上書きの指定

プログラムの中で行うファイル操作とは，OSが管理するファイル・システム上に存在するバイナリ・ファイル，テキスト・ファイル，画像ファイルなどにアクセスして読み書きする処理です．

Pythonプログラムから直接ファイルを操作できませんので，ファイル・システムを通じてファイルのオープン，クローズ，読み書きを行います．

具体的にはPythonの関数を使って，プログラムとファイル・システム間でストリームと呼ばれる論理インターフェース作ってファイルを操作します．ファイルに対するプログラムとファイル・システム間のストリーム入出力操作が，OS上から見たファイル操作となります．

● ファイルのオープンとクローズ

ファイルを読み書きするためにまずはファイルをオープンします．このためのopen関数は，デフォルトではテキスト・モードでファイルを開きます．

特定のエンコーディングでエンコードされたファイルに対して文字列を読み書きできます．エンコーディングは，Windows，Linux，masOSなどのプラットフォームに依存します．テキスト・モードの読み取りでは，これらのプラットフォーム固有の行末記号が使われます．

バイナリ・データが記録されているファイルはバイナリ・モードで開くことができ，読み書きも可能です．

ファイルの利用が終わったらclose関数で閉じて，ファイルを開放します．

● open関数/close関数の書式

▶open関数

```
open(ファイル名, mode='r', buffering=-1,
encoding=None, errors=None,
newline=None, closefd=True,
opener=None)
```

▶close関数

```
close()
```

▶open関数の書式解説

引数でよく利用されるのが，mode，encoding，newlineです．buffering，errors，closefd，openerについては，通常はデフォルトのまま使うと思います．

・ファイル名

open関数に渡すファイル名は，ファイルが保存されているディレクトリやフォルダを含めたパスを含めて指定できます．例えば，piユーザのホーム・ディレクトリにファイルが置かれている場合，

```
/home/pi/ファイル名
```

と記述できます．

・mode

ファイルが開かれるモードを指定します．省略した場合のデフォルトは“r”になり，読み出し用にテキスト・モードで開きます．モードの一覧を**表1**に示します．

・encoding

ファイルのエンコードやデコードに使われるtext encodingの名前です．このオプションはテキスト・モードでだけ使用できます．デフォルト・エンコーディングはプラットフォーム依存（locale.getpreferredencoding()で調べられる）ですが，Pythonでサポートされているエンコーディングはどれでも使えます．

日本語環境であれば，cp932，euc_jp，shift_jis，utf_8がよく利用されます．

読み書きしようとするファイルのエンコードやデコードに合わせて指定できます．

・newline

ユニバーサル改行モードの動作を制御し，テキスト・モードでだけ動作します．デフォルトではNoneになっており，その他に，

```
'', '¥n', '¥r', '¥r¥n'
```

のいずれかを指定できます．

入出力ではデータを小さい単位が連続したものと捉えて，データの入出力を流れとして抽象化したストリームとして扱います．