# ファイル操作

## ファイルの読み込み

### with openを使う方法

　テキストファイルを読み込んで、内容を書き出すプログラムは、以下のように書きます。input.txtというファイルがプログラムと同一のフォルダに入っていることを仮定します。

with open("input.txt", "r", encoding = "shift\_jis") as f:

for line in f: # 1 行ずつ line に入る

line = line.rstrip("\n") # 行末の改行コードを削除する

print("<" + line + ">") # その内容を書き出す

print("file read end")

　with openでファイルを開きます。1つめの引数はファイル名です。2つめは読み取りのとき "r" を指定します。新規書き込みのとき "w", 追加書き込みのとき "a" です。3つめは読み込むファイルの漢字コードです。シフトJISの場合は "shift\_jis" あるいは "shift-jis", utf-8の場合は "utf-8" と指定します。省略するとデフォルトの値を使います（Windows版IDLEの場合はshift\_jis, wslでpythonを使う場合はutf-8）。漢字コードが誤っている場合は、エラーが出てプログラムが止まります（IDLEの場合は文字化けしてそのまま読み込みます）。

　with openの次の行からインデントします。インデントしている区間はファイルを開いたままにします。インデントが終了すると、ファイルを閉じます。

　with openで開いたファイルオブジェクトをfに代入します。

　for文を使うと、ファイルから1行ずつ取り出すことができます。lineに1行分のデータが入ります。1行の終端に改行記号が入っているので、改行コードを削除します。

line.rstrip("\n")

は文字列が入っている変数 line に対して、右端に改行コードがあれば、それを削除したものを返します[[1]](#footnote-1)。それを自分自身に代入する、すなわち

line = line.rstrip("\n")

とすることで、結果として変数lineに入っている文字列の右端の改行コードを削除します。

　テキストファイルの改行コードはWindowsの場合 0x0d, 0x0a の2バイト、Linuxの場合 0x0a の1バイトです。for line in fでlineに入った時点で、OSの違いはなくなり、"\n" の1文字となるようです。

line = line.rstrip()

のように括弧の中を省略すると、デフォルトで空白文字（スペース、タブ、改行など）が削除されます。

　1行分の内容を、書き出すときに

print(line)

ではなく、

print("<" + line + ">")

としています。これは、行末に空白がある場合に、空白も含めて読み込まれていることを確認するためです。

　ファイルを使い終わったら close() でファイルを閉じます。

### ＜"\n" という表現について＞[[2]](#footnote-2)

　rstrip() において、\n という表現が出てきました。\n は改行コードを表します。「\」をエスケープ文字と呼び、「\」と「次の文字」の2文字で1つのことを表します。

　エスケープ文字は以下のような使い方もあります。

ab"c を出力 → print("ab\"c")

　文字列は ' あるいは " で囲むというルールがありました。文字列を " で囲むことにして、その文字列の中に " が含まれる場合、\" と書くことで " は「文字列の終端」を表すのではなく「" そのものを表す」ことをpythonに知らせます。

line.rstrip("\")

と書くと、2個目の " は " そのものを出力することを意味するので、pythonは文字列がその後も続いていると解釈し、実行時にエラーが発生します。

　この例の場合は、以下のように書くこともできます。

print('ab"c')

### openを使う方法

　以下のように書く方法もあります。

f = open("input.txt", "r"):

for line in f: # 1 行ずつ line に入る

line = line.rstrip("\n") # 行末の改行コードを削除する

print("<" + line + ">") # その内容を書き出す

f.close()

print("file read end")

　この方法はファイルを閉じるf.close() を明示的に書く必要があります。以下の理由により、この方法は推奨しませんが、C言語と同じ書式なので、Cに慣れた人にとっては使いやすいでしょう。

* close() を忘れることがある
* ファイル読み込み中にエラーが発生したとき、ファイルを閉じる処理を明示的に書く必要がある

### ファイル読み込み時の数値の扱い

　input-n.txtは以下のような内容になっています。

10

20

30

(以下略)

　1行に数値が1個入っています。この数値に10を足した数を表示するプログラムは以下のようになります。line = line.rstrip("\n") は省略可能です。

with open("input-n.txt", "r") as f:

for line in f: # 1 行ずつ line に入る

line = line.rstrip("\n") # 行末の改行コードを削除する

a = int(line) # 文字列を数値に変換

b = a + 10

print(b) # 10 を足した数を表示

　ファイルから読み込むとき、読み込んだデータは文字列となります。forループの1回目、変数lineには "10" という文字列が入っています。int( ) は引数として数値を指定したときは、整数にする関数でしたが、文字列を引数としたときは、整数に変換する働きを持ちます。"10.4" のように、実数を表す文字列を引数として与えたときは、エラーになります。

　a = int(line) を省略すると、a + 10の処理をする時点で、文字列と数値は接続できないというエラーが発生します。文字列が小数点以下の数を含むときは、float( ) という関数を使います。

　今回の場合、ファイル中の数値は整数なので、floatを使っても同じ結果となります。

## ファイルへの書き出し

　書き出しは以下のように行います。

with open("output.txt", "w") as f:

for i in range(1, 21):

print(str(i) + " : " + str( i \* 2 ), file = f)

print("file write end", file = f)

　ファイルの開き方は、読み込み時と同じです。書き出しのときは "w" を指定します。既にoutput.txtというファイルが存在した場合は、そのファイルに上書きし、元の内容は失われます。既に存在するファイルに追記するには "a" を指定します。

　print関数を使うときに、コンマで区切って2番目の引数として file = f と指定すると、ファイルオブジェクト f に書き込みます。

注意

　ここではwith openを使っているので、ブロックが終了後ファイルが閉じられます。f = open(filename, "w") を使ってファイルをオープンした場合、f.close() を実行するまで、ファイルには書き込まれません。最後にclose() を呼び出すのを忘れてはいけません。ただし、close() を忘れても、スクリプトが終了すると自動的にclose() が行われます。with openを使ってファイルを開いた場合は、closeが不要なので、より安全です。

【例題】

　くじ引きを作ります。file00.txt, file01.txt, file02.txt .... file20.txtという名前の21個のファイルを作り、そのうち、どれか1つだけ「あたり」と書き込み、その他のファイルは「はずれ」と書き込むプログラムを作りなさい。あたりの番号はあらかじめ決めておくことにします。

atari = 10 # 当たりの番号

for i in range(0, 21):

fname = "file{:02}.txt".format(i) # file○○.txt を作る部分

print(fname) # 確認用

with open(fname, "w") as f:

if i == atari:

print("あたり", file = f)

else:

print("はずれ", file = f)

　ファイル名fnameを作成するところで、formatを使っています。formatについては第4章で詳しく学習します。{:02} は「2桁の整数で表し、空白の桁は0を入れる」ことを意味します。format(i) と指定しているので、変数iの値を00, 01, 02, 03 ..... 09, 10, 11, 12,..... のように表すことを意味します。

　当たりの番号を変数atariに入れています。atariの値を乱数を使って0～20のいずれかに決めるには、以下のように書きます。

import random

atari = random.randint(0, 20)

　importはファイルの先頭に書きます。randomという名前の標準ライブラリを使用することを意味します。これを怠ると、random.randint() を使おうとすると、エラーが発生します。

　ここではrandom.randint(a, b) という関数を使いましたが、random.randrange(a, b) という関数もあります。randint(a, b) がa～bの範囲の整数の乱数を返すのに対して、randint(a, b) はa～(b-1) の範囲の整数の乱数を返します。pythonでfor i in range(a, b): としたとき、iの値はaからb-1まで変化しました。randrangeはこれと同じ指定方法です。

　21個あるファイルのどれに当たりが入っているのか、確かめるプログラムは以下のようになります。

for i in range(0, 21):

fname = "file{:02}.txt".format(i)

with open(fname, "r") as f:

for line in f: # 1 行ずつ読み込む

line = line.rstrip('\n') # 行末の改行コードを削除する

if line == "あたり":

print("あたりを見つけた。ファイル名は " + fname)

【例題】

　input.txtを読み取り、行番号をつけて書き出すプログラムを作りなさい。行番号はファイルの行数が3桁のとき、以下のように付けなさい。

input.txt

abc

def

xxx

output.txt

1:abc

2:def

3:xxx

　input.txtがutf-8で書かれているとき、以下のようになります。

# input.txtの行数を数えてnに入れ、桁数をketaに入れる

with open("input.txt", "r", encoding = "utf-8") as f:

n = 0

for line in f:

n = n + 1

keta = len(str(n))

# format 用の文字列を作る。keta = 2のとき、以下の文字列を作る

# "{:2d}:{}"

format = "{:" + str(keta) + "d}:{}"

print(format) # 確認する

# input.txtから読み込み、行番号をつけてoutput.txtに書き出す

with open("input.txt", "r", encoding = "utf-8") as f:

with open("output.txt", "w", encoding = "utf-8") as f2:

i = 0

for line in f:

i = i + 1

line = line.rstrip("\n")

print(format.format(i, line), file = f2)

【課題3‑1】

(a) ファイルinput2.txtを読み込み、そのままoutput2.txtというファイルに書き出すプログラムを作りなさい。

(b) ファイルinput2.txtを読み込み、1, 3, 5, 7, ...行目はoutput3.txtに書き出し、2, 4, 6, 8, ...行目はoutput4.txtに書き出すプログラムを作りなさい。

（ヒント）読み込んだ行数をカウントする変数を新規導入します。偶数か奇数かは2で割った余りで判別します。

## ファイル名の変更・ファイルの削除

【例題】

　先ほどの練習でfile00.txt, file01.txt, ..... file20.txtというファイルがフォルダ内に作成されました。ファイル名をfile00.txt → file-00.txtのように変更するプログラムを作りなさい。

import os

for i in range(0, 21):

fname1 = "file{:02}.txt".format(i) # 変更前ファイル名

fname2 = "file-{:02}.txt".format(i) # 変更後ファイル名

print(fname1, fname2) # 確認

os.rename(fname1, fname2)

　osというライブラリを使うので、ファイルの先頭にimport osと書きます。

os.rename(fname1, fname2)

と書くと、カレントディレクトリ（通常は上記のpythonプログラムが存在するディレクトリ）に存在するfname1というファイルの名前をfname2に変更します。もしfname1という名前のファイルがないときは、エラーが発生し、プログラムの実行はそこで中断します。ここではファイル名を変更する命令を使いましたが、もし、ファイルを削除するなら

os.remove(fname)

と書きます。

【課題3‑2】

(a) file-00.txt, file-01.txt, .... のファイルをyabu00.txt, yabu01.txtのように、自分の名前をつけたファイル名に変更するプログラムを作りなさい。

(b) yabu00.txt, yabu01.txt, yabu02.txt,... のようにファイル名を変更したファイルのうち、yabu01.txt, yabu03.txt, のように奇数の数値がついたファイルを削除し、偶数がついたファイルのみを残すプログラムを作成しなさい。

1. Windowsのファイル（改行コードは0Dh 0Ah）, Linuxのファイル（改行コードは0Ah）のどちらに対しても "\n" で改行コードが除去できます。Windowsファイルに対して "\r\n" とする必要はありません。"\n" と "\r\n" は等価なようです。Windowsのファイル, Linuxのファイルのどちらに対しても "\r\n" で改行コードを除去できます。 [↑](#footnote-ref-1)
2. \ はバックスラッシュでアスキーコードは5Cです。キーボードのバックスラッシュキー（右のシフトの左）あるいは円マークキー（Back Spaceの左））を押すと、文字コード5Cが入力され、画面上の見かけは円マークに見えます。utf-8という文字コードでファイルを作成した場合、バックスラッシュは5C, 円マークはC2A5の2バイトで区別されます。C2A5はバックスラッシュではないので、C2A5を用いた\nは改行にはマッチしません。pythonプログラムで \n がマッチしないという現象に悩むことがありますが、これが原因だと思われます。Shift JISコードでファイルを作成した場合は、C2A5の円マークは存在しないので、この問題は起こりません。 [↑](#footnote-ref-2)