

第Ⅳ編 クラスの応用

1 文字列クラス

(1) エスケープシーケンス (第Ⅰ編で説明済み)

文字列 (彼は"おはよう"と言いました。) を表示するときに

```
print("彼は"おはよう"と言いました。") # ==> エラー
```

とすればエラーになる。ただし、[""]が全角ならば[OK]。

このようなときに下記のエスケープシーケンスを用いてつぎのようにする。

```
print("彼は¥"おはよう¥"と言いました。") # ==> 彼は"おはよう"と言いました。
```

エスケープシーケンス表

記号	内 容
¥¥	「¥」文字そのもの
¥'	シングルクォーテーション
¥"	ダブルクォーテーション
¥b	バックスペース
¥f	改ページ
¥r	キャリッジリターン
¥n	改行
¥t	水平タブ
¥v	垂直タブ
¥0	NULL

上記の**赤い背景色**の部分はよく使うエスケープシーケンスである。

(2) raw文字列 (第Ⅰ編で説明済み)

エスケープシーケンスに代わって、raw文字列を用いてつぎのように書いても同じ結果が得られる。

書 式	r "文字列"
例 題	print(r"彼は"おはよう"と言いました。") # ==> 彼は"おはよう"と言いました。

(3) 文字列操作

(A) 文字列結合

書 式	文字列1 + "文字列2"
例 題	print("おはよう。"+"さようなら") # ==> おはよう。さようなら

(B) 文字列の繰り返し

書 式	文字列 * 回数
例 題	print("おはよう。"*2) # ==> おはよう。おはよう。

(C) 文字列の長さ

書 式	len(文字列)
例 題	print(len("おはよう。"*2)) # ==> 10

(D) 型変換 (数値から文字列)

書 式	str(数値)
例 題	print(str(12.3)) # ==> "12.3"

(E) 型変換 (文字列から数値)

書 式	int(x [, 基底]) # 文字数列 x の基底(2,10,16進数)のこと
	float([x])
例 題	print(int("9",10)) # ==> 9
	print(int("100", 2)) # ==> 4=1*2*2+0*2+0
	print(float("12345678.90123456")) # ==> 12345678.90123456
	print(int("9") + 1) # ==> 10

(F) 文字列の一部抽出

書 式	str0[index] # str0文字列変数(0...len(str0)-1)
	str0[index1:index2] # スライス使用 : index1番地からindex2-1番地
例 題	str0 = "012345"
	print(str0[0]) # ==> "0"
	print(str0[1:3]) # ==> "12"

(G) 検索 (番地と個数) str00 = "abc12345678abcd7654123abckkkk"

リストや文字列に「任意の文字列」が含まれているか否かやその個数を調べることができる。

(場所)	リスト・文字列内で指定した値をもつ要素の最初のインデックス (番地) を取得
	オブジェクト.index(値, n) # n番地以降
	str00.index("b", 2) # 2番地以降で"b"が含まれる番地
(個数)	リスト・文字列内で指定した値をもつ要素の数を取得
	オブジェクト.count(値, n) # n番地以降
	str00.count("b", 2) # 2番地以降で"b"が含まれる個数

(H) 文字列からリスト

文字列から配列を作成する.str00 = "abc123ab54123abk"

(文字列)	文字列を[任意の文字列]ごとに区切って配列を作成
(から)	文字列オブジェクト.split("値")
(リスト)	str00.split("123") # "123"で区切って配列を作成

(I) リストから文字列

上記事項の逆を操作するものである。

(リスト)	リストを「任意の文字列」で結合した文字列を作成
(から)	"値".join(リストオブジェクト) # "値"で各要素を結合した文字列
(文字列)	", ".join(list_vari) # ","で各要素を結合した文字列

(J) 文字列における「任意の文字列」を「任意の文字列」で置換

(置換)	文字列における「任意の文字列」を「任意の文字列」ですべて置換 文字列オブジェクト.replace(置換される文字列, 置換する文字列)
	文字列における「任意の文字列」を「任意の文字列」で前からn個のみ置換 文字列オブジェクト.replace(置換される文字列, 置換する文字列, 置換回数)
	正規表現を用いての置換
	import re 新しい文字列 = re.sub(正規表現, 置換する文字列, 置換される文字列 [, 置換回数])
例題	<pre>str00 = "abc123abc123abc" str00.replace("a","-") # "A"を"-"ですべて置換する str00.replace("a","-", n) # "A"を"-"で前から n 個のみ置換する str11 = re.sub("[a-z]+","A",str00,2)</pre>

正規表現：

メタ文字	意味
^	文字列の先頭をあらわすが、[の次に来る場合や [] の中は除く
\$	文字列の末尾を表わす
+	1 回以上の連続する文字
*	0回以上の連続する文字
?	0回、または1 回だけの文字
{n}	n 回以上の連続する文字
{n,m}	n 回から m 回まで連続する文字
{.m}	0回から m 回まで連続する文字
[]	[] 内のいずれかの1 文字
.	任意の1 文字 (ただし改行文字を除く)
	OR
()	本文を参照する文字
¥	エスケープ文字 (この記号の後の特殊文字をそのまま出力する)
¥n	改行文字

文字クラス	意味
[a-z]	小文字の半角英文字
[A-Z]	大文字の半角英文字
[0-9]	数字
[A-Za-z]	大文字と小文字の半角英文字
[A-Za-z0-9]	数字と大文字と小文字の半角英文字

2 ファイル操作

(A) ファイルオープン・クローズ

書式 1	f p= open(ファイル名, モード', encoding='utf-8')
	モード : r : 読み込み (デフォルト) w : 書き込み a : 追記書き込み b : バイナリモード f = open(ファイルパス, モード)
	fp.close()
書式 2	with文の場合、処理ブロック後にファイルが自動的に閉じられる with open(ファイル名, モード', encoding='utf-8') as fp:

(B) ファイルデータの読み込み

書式	# オープン
	fp = open(ファイル名, 'r', encoding='utf-8')
	# 読み込み
	引数のサイズは省略可能。その場合、全データ読み込み
	fp.read()
	行毎にデータを読み込み、リストに格納する
	fp.readlines()
	1行ずつデータを読み込む
	fp.readline()
	# クローズ
	fp.close()

読み込むファイル : Data¥text.txt

下記をサクラエディットで作成し、文字コードUTF-8で
C:¥PythonPG¥Data¥text.txt として保存しなさい。

```
(Python基礎)ファイル操作
ファイルのオープン
ファイルのクローズ
ファイルのと読み込み
ファイルの書き込み
存在・コピー・削除
```

つぎのプログラムを作成し、[C:¥pythonPG¥sample3410.py]として保存し、実行しなさい。

3通りの読み込み

fp.read()

fp.readlines()

1行ずつ

```
# 例題ファイル: C:¥pythonPG¥sample3410.py

print( "==== ファイル操作 =====" )

print("=====")
print("+++++ すべて +++++")
fp = open('Data¥¥text.txt', "r", encoding='utf-8')
str0 = fp.read()
fp.close()
print(str0)

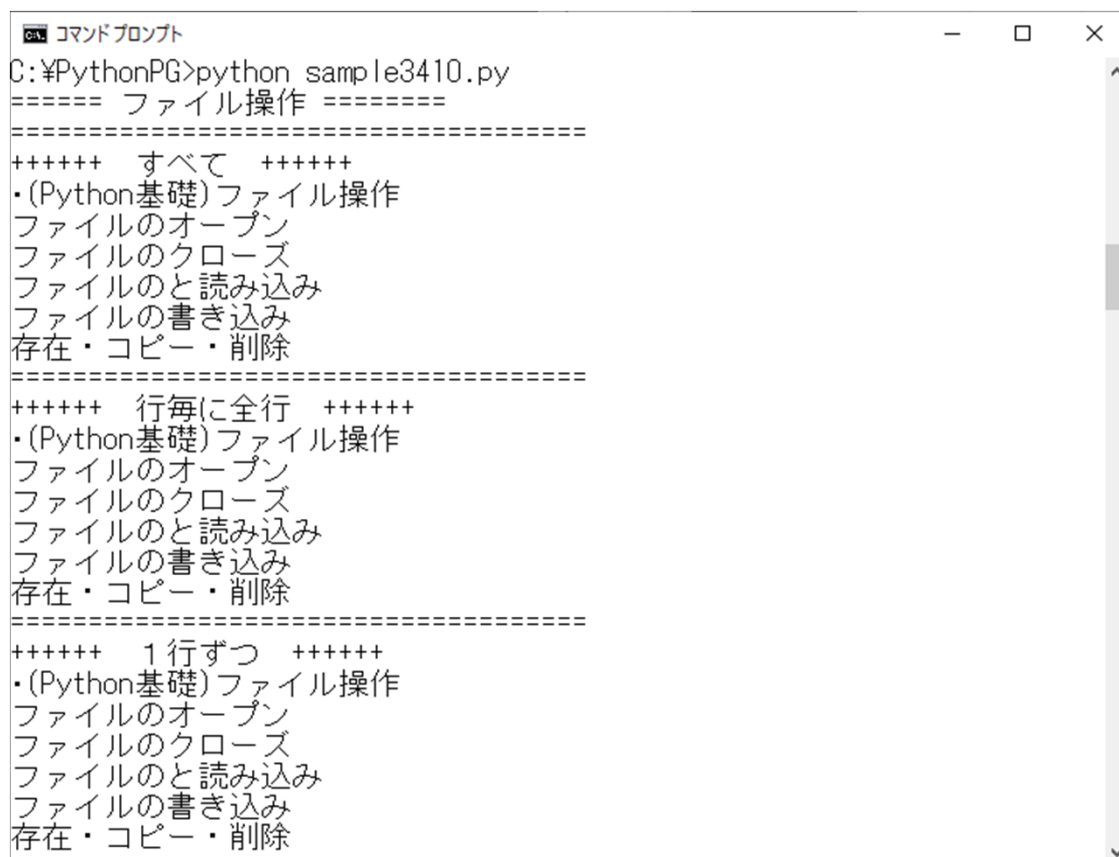
print("=====")
print("+++++ 行毎に全行 +++++")

fp = open('Data¥¥text.txt', "r", encoding='utf-8')
str1 = fp.readlines()
fp.close()
for row in str1:
    print( row.replace( "¥n", "" ) )

print("=====")
print("+++++ 1行ずつ +++++")

fp = open('Data¥¥text.txt', "r", encoding='utf-8')
for row in fp:
    print( row.replace( "¥n", "" ) )
fp.close()
```

実行結果： テキストファイルData¥text.txtを正常に読み込んでいることが分かる。



```
コマンドプロンプト
C:\¥PythonPG>python sample3410.py
===== ファイル操作 =====
++++++ すべて ++++++
・(Python基礎)ファイル操作
ファイルのオープン
ファイルのクローズ
ファイルのと読み込み
ファイルの書き込み
存在・コピー・削除
=====
++++++ 行毎に全行 ++++++
・(Python基礎)ファイル操作
ファイルのオープン
ファイルのクローズ
ファイルのと読み込み
ファイルの書き込み
存在・コピー・削除
=====
++++++ 1行ずつ ++++++
・(Python基礎)ファイル操作
ファイルのオープン
ファイルのクローズ
ファイルのと読み込み
ファイルの書き込み
存在・コピー・削除
```

(C) ファイルデータの書き込み

書式	# オープン
	fp = open(ファイル名, 'w', encoding='utf-8') # 作成または上書き
	fp = open(ファイル名, 'a', encoding='utf-8') # 追加
	# 書き込み
	fp.write(文字列)
	改行を挿入する場合 [\n] を付加する
	fp.writelines(文字列)
	# クローズ
	fp.close()

つぎのプログラムを作成し、[C:¥pythonPG¥sample3420.py]として保存し、実行しなさい。

```
# 例題ファイル: C:¥pythonPG¥sample3420.py

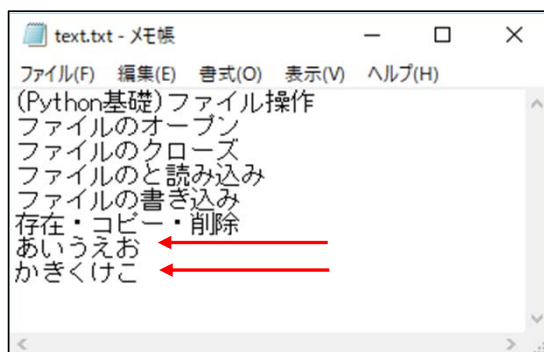
print( "==== ファイル操作_2 =====" )

fp = open('Data¥¥text.txt', "a", encoding='utf-8')
fp.writelines("\nあいうえお")
fp.writelines("\nかきくけこ")
fp.close()

print( "===== " )
```

実行結果：

テキストファイルData¥text.txtに、正しく「2行の文字列」が書き込まれていることが分かる。



3 例外クラス

(1) エラーの種類

エラー（例外クラス）にはつぎのようなものがある。

SyntaxError:	RuntimeError:
ZeroDivisionError:	OSError:
NameError:	KeyError:
TypeError:	FileNotFoundError:
ValueError:	importError:
Exception:	RuntimeError:

(2) 例外処理の書式

書式	try:
	# 処理A
	except 例外クラス名:
	# 処理Aが例外名を出したときの処理
	else:
	# 例外が発生しないときの処理
	finally:
	# Try, exception 等の最後の処理A

except文	
書式	except 例外クラス名:
	except Exception: # システム終了以外の例外クラスの基底クラス
	except: # すべての例外発生時の処理

(3) 例外クラスの定義

書式	except 例外名 :	#発生した例外の処理
	raise	# 自動的に例外を発生させて停止する。

4 日付と時刻

datetime::クラス

書式	import datetime
	datetime(年、月日、時、分、秒、マイクロ秒、タイムゾーン) 作成・取得
	datetime.now() 現在に日時のインスタンス取得
	datetime.today() 現在に日付のインスタンス取得
	datetime.strptime(日時文字列, フォーマット) 指定したフォーマットの日時文字列から インスタンス取得
	datetime.strptime("2018/01/18", "%Y/%m/%d")==>2018-01-18 00:00:00
	日時.date() 日時のdateインスタンス
	日時.time() 日時のtimeインスタンス
	日時.weekday() 曜日(0--6)を取得
	日時.strftime(フォーマット) 指定したフォーマットの日時文字列取得
	dt = datetime.datetime.now() # 2018-01-18 13:20:48 ならば print(dt.strftime("%Y-%m-%d")) # 2018-01-18
	日時.year 年
	日時.month 月
	日時.day 日
	日時.hour 時
	日時.minute 分
	日時.second 秒
	日時.microsecond マイクロ秒
	日時.tzinfo タイムゾーン
	timedelta(属性=値) 日時（属性で指定）の加減算
	str0 = "2018-01-18"
	dt = datetime.datetime.strptime(str0, "%Y-%m-%d") # 2018-01-18 00:00:00
	print(dt+datetime.timedelta(days=1, minutes=5)) # 2018-01-19 00:05:00

日時に関する書式：

表 記	説明
%Y	4桁の年数 (例 2018,1996…)
%y	2桁の年数 (例 18, 96..)
%m	2桁の月 [01,12] (例 01, 07, 12..)
%d	2桁の日付 [01,31] (例 02, 28, 31..)
%H	24時間表記の時間 [00,23] (例 00, 12, 21..)
%I	12時間表記の時間 [01,12] (例 01, 07, 12..)
%M	2桁の分 [00, 59] (例 00, 05, 38, 59..)
%S	秒 [00, 61] (例 00, 15, 39, 60, 61..) 60,61は00,01と同じ
%w	整数表記された曜日 [0(日曜), 6] (例 1(月曜), 4(木曜), 6(土曜)…)
%U	週番号 [00,53] 日曜が週の始めとしてカウントされる。年を通しての週の番号 (例 00, 03, 04, 52..)
%W	週番号 [00,53] 月曜が週の始めとしてカウントされる、年を通しての週の番号
%z	UTC タイムゾーンからのオフセット +HHMMまたは-HHMMの形 (例 +0800, -0925, …)