12 Python基礎

入出力

- 教科書 P.73-78までを、良く読んでください。

出力時の書式

- まずは文字列メソッドを理解しよう

> https://docs.python.org/ja/3.9/library/stdtypes.html#string-methods

  - `print()` 以外の代入などでも使用可能であることに注意！

【str_format.py】

```
# format method
print('x={} , y={}'.format(10, 20))
print('x={0} , y={0} , z={1}'.format(10, 20))

s = 'a = {0} , b = {1} , c = {2}'.format(100, 200, 300)
print(s)

# f-string python3.6以降で利用可
x = 10
y = 20
print(f'x={x} , y={y}')
print(f'x={x} , y={x} , z={y}')

s = f'a = {100} , b = {200} , c = {300}'
print(s)
```

- 次にように出力したい場合、どうすればよいだろうか？

  - 文字列は、`'` で囲って出力し、数値はそのまま出力する

【実行結果】

```
$ python str_format_2.py
引数 = 'abc'
引数 = 'あいう\nかきく'
引数 = 3.1415
```

【str_format_2.py】

```
def output(var):
  # 各自作成


output('abc')
output('あいう\nかきく')
output(3.1415)
```

- 知っていれば、コードは簡単に書けるようになる

> https://qiita.com/Kawausomando/items/5199d059b644c8dbf773

練習1

- 以下のプログラムを作成してください。
    - 書式フォーマットを使用して、期待する出力になるようにしてください。

1. 【期待する出力】

```
名前: 田中太郎, 年齢: 30歳
```

【ren1_1.py】

```
name = "田中太郎"
age = 30

# ここにコードを書いてください
formatted_string = ...
print(formatted_string)
```

2. 【期待する出力】

```
数値: 123.46
```

【ren1_2.py】

```
  value = 123.456789

  # ここにコードを書いてください
  formatted_value = ...
  print(formatted_value)
```

3.【期待する出力】

```
数値： 0000000042
```

【ren1_3.py】

```
value = 42

# ここにコードを書いてください
formatted_value = ...
print(formatted_value)
```

4. 【期待する出力】

```
商品： りんご， 価格： ¥3,000
```

【ren1_4.py】

```
item = "りんご"
price = 3000

# ここにコードを書いてください
formatted_string = ...
print(formatted_string)
```

5. 【期待する出力】

```
日付： 2024年5月22日
```

【ren1_5.py】

```
year = 2024
month = 5
day = 22

# ここにコードを書いてください
formatted_date = ...
print(formatted_date)
```

6. 【期待する出力】

日付：2024年5月22日(水)

【ren1_6.py】

```
import locale
import datetime

year = 2024
month = 5
day = 22

# ここにコードを書いてください



formatted_date = ...
print(formatted_date)
```

ファイル操作

open( ) / close()

- 他の言語と同様、ファイルを開く→使用→閉じる の流れ

```
open(ファイル名,モード,encoding='文字コード') -> ファイルオブジェクト
```

```
ファイルオブジェクト.close()
```

モード	意味
r	読み込み用（デフォルト）
w	書き込み用
a	追記用
b	バイナリモード
t	テキストモード（デフォルト）
+	更新用（r,w,aと同時に使用）
x	排他的書き込み用（すでにファイルがあるとエラー）

ファイルオブジェクトのメソッド

メソッド名	働き
read( )	ファイル全体を文字列として読み込む
readlines( )	ファイル全体をリストとして読み込む
readline( )	1行読み込む
write( )	文字列の書き込み
writelines( )	リストの書き込み
seek()	アクセス位置の移動

- 詳細は、公式ドキュメントを参照

> https://docs.python.org/ja/3.9/library/io.html#i-o-base-classes

- 利用可能な文字コード

> https://docs.python.org/ja/3.9/library/codecs.html#standard-encodings

利用例

- 1行だけ読み込む例

```
f = open('sample.txt')
line = f.readline().strip()
print(line)
f.close()
```

```
with open('sample.txt','r') as f:
    line = f.readline().strip()
    print(line)
```

※ with文を使用することで、ファイルオブジェクト使用後の `close()` 忘れを防ぐことができる。
withを使用しない場合は、try～finallyを使用する。

> https://www.python.jp/pages/with-statement-3.9.html

- メソッド `strip()` は、何を行うのか？　なぜ必要なのか？

- すべての行を読み込む例（1行を繰り返す）

```
with open('sample.txt','r') as f:
    for line in f:
        print(line.strip())
```

- まとめて読み込む例1

```
# 1行ずつ出力
with open('sample.txt','r') as f:
    for line in f:
        print(line.strip())

# リストに格納してから出力
with open('sample.txt','r') as f:
    lines = [line.strip() for line in f]
print(lines)
```

- まとめて読み込む例2

```
# readlines()を使用する
with open('sample.txt','r') as f:
    lines = f.readlines()
print(lines)    # 改行文字が含まれる

lines = [i.strip() for i in lines]
print(lines)    # 改行文字が取り除かれる
```

練習2

- 以下のプログラムを作成してください。【ren2.py】
    - word.txtファイルが存在しているかを調べ、存在していれば、そのファイルを読み込む。
        - ファイルは英単語リストのデータ
        - 1行に1単語が記述されている
        - 句切りは改行とする
        - 読み込んだデータは、リストに格納する
        - ファイルが存在しない場合、空のリストを作成する。
    - プログラム終了時に、英単語が格納されたリストをword.txtに書き込む。
    - リスト作成後以下の動作を行う。
        - キーボードから英単語を入力し、リストに存在していなければ、追加する

        - 存在している場合は、その旨を出力し、追加はしない
        - 空文字を入力したら、終了する
        - 大文字でコマンド `LIST` と入力したら、その時点でのリストを出力する

【実行例】

```
単語を入力してください：hello
単語を入力してください：world
単語を入力してください：LIST
単語リスト： ['hello', 'world']
単語を入力してください：hello
すでに登録済です
単語を入力してください：Good
単語を入力してください：bye
単語を入力してください：
終了します
これまでに覚えた単語： ['hello', 'world', 'Good', 'bye']
```

練習3

- 次のプログラムを作成してください。【ren3.py】
    - 先のプログラムで生成した word.txt を読み込む。
        - word.txt が無い場合は、他のテキストファイルでもOK
    - 各行の先頭に行番号を4桁で出力する。

【実行例】

```
0001:hello
0002:world
0003:Good
0004:bye
```

練習4

- 次のプログラムを作成してください。【ren4-1.py】
    - 試験結果が記録されたファイル test.csv を読み込む
    - 「,」で区切られているので、分割する
    - 各人の合計点を出力する

【test.csv】

```
氏名,国語,算数,理科,社会
A,50,80,90,90
B,70,70,70,80
C,90,90,70,70
D,60,70,80,90
```

- 文字コードは、UTF-8、改行コードは「\n」とする。
    - EXCELで作成し、保存すると上記条件と異なるので注意！
        - 文字コード：shift_jis（cp932,ms932など）
        - 改行コード：\r\n
    - https://docs.python.org/ja/3.7/library/codecs.html#standard-encodings

【実行結果】

```
A 310
B 290
C 320
D 300
```

- 出力結果を以下のように変更したい【ren4_2.py】
    - 結果を、ファイルresult.csvに出力してください。

【実行結果】

```
氏名,国語,算数,理科,社会,合計
A,50,80,90,90,310
B,70,70,70,80,290
C,90,90,70,70,320
D,60,70,80,90,300
```

検証

- Pythonで入出力するファイルは、現在UTF-8が主流
    - Windowsでは、CP932（日本語Shift-JIS）
    - Linux・MacはUTF-8
- CSVがUTF-8だと、Windowsユーザは不便
    - といって、Shift-JISで保存すると、vscodeでは確認するのが面倒になる
    - Excelでインポートすると文字化けする

- 良い方法はないだろうか？

【プログラム例】（for_win.py）

```
# 読み込み
read_filename = 'foo.csv'
with open(read_filename,'r',encoding='utf_8_sig') as f:
    print(f.read())

# 書き出し
write_filename = 'bar.csv'
csv_data = ["abc",10,"xyz",20]
with open(write_filename,'w',encoding='utf_8_sig') as f:
    f.write('sample')
```

調査事項

- encoding指定で使用している `utf_8_sig` とは何か？

検証手順

- 以下の2種類のファイルを、それぞれ `UTF-8` 、 `utf_8_sig` で読み込んだ時、正常に処理できるだろうか？
    - UTF-8のファイル（foo.csv)
    - UTF-8 with BOMのファイル(foo_bom.csv)

検証プログラムの作成

- 上記の2種類のファイルを用意し、それぞれ1行目だけ読み取り、比較してみる。
- 出力するプログラムを作成してください。（diff_comp.py）

【実行結果】

```
utf8でopenした場合
foo.csv         ['氏', '名', ',', '国', '語', ',', '算', '数', ',', '理', '科', ',', '社',
'会', ',', '合', '計', '\n']
foo_bom.csv     ['\ufeff', '氏', '名', ',', '国', '語', ',', '算', '数', ',', '理', '科',
',', '社', '会', ',', '合', '計', '\n']

utf_8_sigでopenした場合
foo.csv         ['氏', '名', ',', '国', '語', ',', '算', '数', ',', '理', '科', ',', '社',
'会', ',', '合', '計', '\n']
foo_bom.csv     ['氏', '名', ',', '国', '語', ',', '算', '数', ',', '理', '科', ',', '社',
'会', ',', '合', '計', '\n']
```

バイナリエディタで各ファイルの先頭を確認してみる

- Linuxの場合

```
$ od -tx1 foo_bom.csv
$ od -tx1 -N3 foo_bom.csv # バイト数を指定する場合
```

- Windowsの場合

```
> certutil -dump foo_bom.csv
```

まとめ

- この作業を通じて、分かったことを各自まとめておいてください。