2018年航空宇宙情報システム学第２
第2部「プログラミングと数値計算」

第6回 Python入門 5
～ファイル入出力・クラス～

2018年6月19日

9章（ケーススタディ）言葉あそび
＋
14章ファイル

単語リスト words.txt

- 前回の宿題で使用
- クロスワードパズルで有効とされる113,809個の単語のリスト
- http://thinkpython2.com/code/words.txt からダウンロード可
- Spyder の作業ディレクトリに置いておく
  - os.getcwd() で作業ディレクトリを確認できる
- プレインテキストで書かれているファイルなので、テキストエディタで開いて見られる

ファイルの読み込み方（1）

- 読み込みモードで”words.txt”をオープン

  >>> fin = open('words.txt‘)

  – 正式には、fin = open(‘words.txt’ , **‘r’**)

  – 変数 fin はファイルハンドルなどと呼ばれる

- readlineメソッド：（先頭から）1行ずつ読み込む

  >>> line = fin.readline()

  >>> line

  ‘aa¥r¥n‘　　# ¥r¥n は改行文字

  >>> line.strip()　# stripメソッドで取り除くと

  'aa'

(注) ¥マークは本当は半角の＼（バックスラッシュ）

ファイルの読み込み（2）

- （例）先頭から末尾まで1行ずつ読み込んで表示

```
>>> fin = open('words.txt','r')
>>> for line in fin:    # 1行ずつ読み込む for 文
...         word = line.strip()
...         print(word)
 :
>>> fin.close()    # 開いたファイルは最後に閉じる
```

- 全行をリストに読み込んでしまうことも可

```
>>> lines = open('words.txt').readlines()
>>> len(lines)
113809
```

練習問題 9.1

words.txt に含まれる20字より長い単語を表示せよ

（解答例）

```
>>> fin = open('words.txt')
>>> for line in fin:
        word = line.strip()
        if len(word) > 20:
                print(word),
>>> fin.close()
```

忘れがちだが、ファイルは使い終わったら閉じよう

練習問題 9.2 (改)

‘e’を含まない単語の割合を計算せよ。

```
>>> ntot,nnoe = 0,0
>>> fin = open('words.txt')
>>> for line in fin:
        ntot += 1
        if not 'e' in line.strip():
                nnoe += 1
>>> print float(nnoe)/ntot
0.33073834231
```

別解も色々あるので考えよう。

カンマ区切りデータの読み込み（1）

- tokyo-weather-20170601-20180531.csv : 2017年6月から2018年5月までの東京の気象情報（年月日、最高気温、最低気温、降雨量、日照時間、平均風速）をカンマ区切りで列挙
  - 気象庁HP（http://www.data.jma.go.jp/gmd/risk/obsdl/）よりダウンロードしたデータから生成

テキストエディタで見てみると..

```
年月日,最高気温(℃),最低気温(℃),降水量の合計(mm),日照時間(時間),平均風速(m/s)
2016/6/1,26.4,18,0,5.7,3.3
2016/6/2,26,16.3,0,13,4.8
2016/6/3,24.2,14.2,0,12.7,4
2016/6/4,27.2,16.3,0,8,4.5
2016/6/5,23.4,16.4,3,2.8,2.8
2016/6/6,23.9,16.3,0,0.5,2.6
```

先頭行は項目名

Excel で開いて見てみると..

年月日	最高気温(℃)	最低気温(℃)	降水量の合計(mm)	日照時間(時間)	平均風速(m/s)
2016/6/1	26.4	18	0	5.7	3.3
2016/6/2	26	16.3	0	13	4.8
2016/6/3	24.2	14.2	0	12.7	4
2016/6/4	27.2	16.3	0	8	4.5
2016/6/5	23.4	16.4	3	2.8	2.8
2016/6/6	23.9	16.3	0	0.5	2.6

カンマ区切りデータの読み込み(2)

- (例) 年月日、最高気温、最低気温、降雨量、日照時間を、それぞれリストに格納する

```
>>> fin = open(' tokyo-weather-20170601-20180531.csv ')
>>> fin.readline()  # 先頭行(ヘッダー)を読み飛ばすため
>>> date,hitmp,lotmp,rain,sun = [],[],[],[],[]  # 空のリスト
>>> for line in fin:
        lst = line.strip().split(',') # 行をカンマで区切る
        date.append(lst[0])  # 文字列のままリストに追加
        hitmp.append(float(lst[1])) # 数値(実数値)に変換
        lotmp.append(float(lst[2]))
        rain.append(float(lst[3]))
        sun.append(float(lst[4]))
>>> fin.close()
```

- もっとエレガントな方法も考えてみよう

（補足）日本語（マルチバイト）文字が含まれるテキストファイルの読み書き

- 前スライドの、fin.readline() でエラーが出る場合
- csvファイルの先頭行の日本語文字列の文字コードが Shift JIS であるため
  - Windows 環境では（多分）問題なし
- 対処法１（推奨）: codecs モジュールを使う

```
>>> import codecs
>>> fin = codecs.open("' tokyo-weather-20170601-20180531.csv '","r","shift_jis")
```

  - readline(), readlines()メソッド等も同様に使える
- 対処法2 : ファイルの文字コードを変換する
  - nkf コマンドやテキストエディタの機能を使う

ファイルを丸ごと読み込む

- readlinesメソッドを使ってまとめて読み込んでも良い

```
>>> lines = cocecs.open(' tokyo-weather-201606-201705.csv','r','shift_jis').readlines()
>>> date,hitmp,lotmp,rain,sun = [],[],[],[],[]
>>> for line in lines[1:]:    # 先頭行を飛ばすため
        lst = line.strip().split(‘,’)
        date.append(lst[0])
        hitmp.append(float(lst[1]))
        lotmp.append(float(lst[2]))
        rain.append(float(lst[3]))
        sun.append(float(lst[4]))
```

ファイルへの書き込み

- ファイルを書き込みモードで開き、文字列を書き込む

```
>>> fout = open('output.txt','w') # 書き込みモード
>>> fout.write('This is the first line.¥n') # writeメソッド
>>> line = 'This is the last line. ¥n'
>>> fout.write(line)
>>> fout.close() # 必ず閉じよう
```

- writelinesメソッドでまとめて書き込む方法もある

```
>>> lines = ['First line¥n','Second line¥n','Last line']
>>> open('output.txt','w').writelines(lines)
```

(補足) 日本語文字列のファイル書き込み

- 日本語文字が含まれる文字列をファイルに書き込む場合も、codecsモジュールを用いて、文字コードを指定する

```
>>> lines = ['これは1行目です。¥n','これは2行目です。¥n']
```

- Unicode (UTF-8) で書き込む場合（Mac向き）

```
>>>  codecs.open('test-utf.txt','w','utf-8').writelines(lines)
```

- Shift JIS で書き込む場合（Windows向き）

```
>>> codecs.open('test-sj.txt','w','shift_jis').writelines(lines)
```

書式演算子

- 数値をテキストファイルに書き込むには
  – まず、文字列に変換しなければならない

  方法 1 : str 関数を使う（数値→文字列変換）

```
>>> x, y = 2, 10.5
>>> line = str(x)+’,’+str(y) + ’¥n’  # 文字列の連結
>>> fout.write(line)
```

  方法 2 : %演算子を使う（文字列中に数値を埋め込む）

```
>>> line = ’%d,%f¥n’ % (x,y)  # 複数の数値の埋め込み
>>> fout.write(line)
```

  方法 3 : format メソッド を用いる

```
>>> print("{}! is {}".format(10,math.factorial(10)))
```

書式文字列

% 演算子の使用例

```
>>> ‘My name is %s.’ % ‘Bob’  # 文字列の埋め込み
'My name is Bob.‘
>>> math.pi
3.141592653589793
>>> ‘%.2f’ % math.pi  # 小数点以下桁数の指定
'3.14‘
>>> x = 0.5
>>> ‘cosine of %f radian is %f’ % (x,math.cos(x)) #タプル
'cosine of 0.500000 radian is 0.877583'
```

15章 クラスとオブジェクト

ユーザー定義の型（クラス）

- Python 組込みの型 : int, char, list, dict, など
- ユーザが独自の型（クラス）を作成することも可
- 例題として、2次元平面上の点を表す Point型 を作成してみる

```
class Point: #ヘッダー
        """Represents a point in 2-D space.""" #本文
        # 説明文だけ
>>> blank = Point() # Point型のオブジェクトを生成
>>> blank
<__main__.Point at 0x1e23dc40048>
```

- blankに代入されたオブジェクト は Pointクラス のインスタンスである

属性（Attributes）

- ドット記法を用いてインスタンスに値を代入

```
>>> blank.x = 3.0
>>> blank.y = 4.0
```

- この場合の x や y を属性（attributes）と呼ぶ
  - 他の言語では、メンバ（変数）などと呼ばれる
- 例：Point インスタンスの属性x,yを表示する関数

```
def print_point(p):
        print('(%g, %g)' % (p.x, p.y))
>>> print_point(blank)
(3,4)
```

矩形クラスの定義

- Pointクラスに加えてRectangleクラスを定義
  - どんな属性を持つべきかは必ずしも一意ではない
  - (1) 頂点の1つと、幅と高さによって指定
  - (2) 2つの対角の頂点によって指定
  - どちらでも良いがここでは(1)を採用

```
class Rectangle:
        """Represents a rectangle.
        attributes: width, height, corner.
        """
```

- 3つの属性(width, height, corner) を持つ
- corner 属性は、左下の点を表すPointオブジェクトとする

矩形オブジェクトの生成と属性値の代入

```
>>> box = Rectangle()  # Recangleクラスのインスタンス
>>> box.width = 100.0
>>> box.height = 200.0
>>> box.corner = Point() #corner属性にPointオブジェクト代入
>>> box.corner.x = 0.0
>>> box.corner.y = 0.0
```

- 属性の値をひとつひとつ代入するのは面倒？
  - タプルを使えば1行にはできるが、実質的には変わらない
  - 属性の値をまとめて代入する関数を作れば良い
  - 関数ではなく、メソッドを作る方がオブジェクト指向的にはより良いが、詳細は割愛。教科書 17.5節 ("The init method") 等を参照

戻り値としてのインスタンス

関数はインスタンスを戻り値として返すことができる

- 中級以上には便利なテクニック

例：矩形の中心点を求める関数：

```
def find_center(rect):
        p = Point()
        p.x = rect.corner.x + rect.width/2
        p.y = rect.corner.y + rect.height/2
        return p
>>> center = find_center(box)
>>> print_point(center)
(50, 100)
```

クラスオブジェクトは変更可能

- オブジェクトの属性の値は変更可能

```
>>> box.width = box.width + 50 # 幅を拡大
```

- 例：Rectangleオブジェクトを拡大する関数

```
def grow_rectangle(rect, dwidth, dheight):
        rect.width += dwidth
        rect.height += dheight
>>> box.width, box.height #変更前のサイズ
(150.0, 300.0)
>>> grow_rectangle(box, 50, 100) # 幅・高さを変更
>>> box.width, box.height
(200.0, 400.0)
```

オブジェクトのコピー

- クラスを使う人にとっては非常に重要
- = による代入は、別名を付けるだけであって複製にはならない（リストの場合と類似）

```
>>> p1 = Point()
>>> p1.x,p1.y = 3.0,4.0
>>> p2 = p1   # 複製ではなく、別名の生成!
>>> p2.x = -1.0 # 別名の方の属性値を変更すると..
>>> print('(%g,%g)' % (p1.x,p1.y)) # p1 のx,yを表示
(-1,4)
>>> p1 is p2 # p1 と p2 は同一か？
True
```

オブジェクトのコピー（続き）

- 複製を作る1方法: copy モジュールを利用

```
>>> import copy
>>> p3 = copy.copy(p1) # p1の中身を複製しp3に代入
>>> print('(%g,%g)' % (p1.x,p1.y))
(-1,4)
>>> print('(%g,%g)' % (p3.x,p3.y))
(-1,4) # 属性の値はp1と同じ
>>> p3 is p1
False # 別個のオブジェクト
```

- この後、p3 の属性(x,y)の値を変更しても、p1には影響が無い（逆も同じ）

浅いコピーと深いコピー

- 属性として、他のクラスオブジェクトやリストを持つオブジェクトをコピー（複製）する場合はさらに注意が必要

```
>>> box2 = copy.copy(box) #「浅い」コピー
>>> box3 = copy.deepcopy(box) #「深い」コピー
>>> box.corner.x = 1.0  #0.0から変更
>>> print(box.corner.x,box2.corner.x,box3.corner.x)
1.0 1.0 0.0
```

クラス，属性のチェック

- あるオブジェクトがあるクラスのインスタンスかどうかをチェック : isinstance 関数

```
>>> isinstance(p1,Point)
True
```

- （クラス）オブジェクトがある属性を持っているかどうかをチェック: hasattr 関数

```
>>> hasattr(p1,'x')  # x ではなく 'x' (文字列)
True
```

今日の課題

tokyo-weather-20170601-20180531.csv を読み込んで、以下を計算するプログラムを作成せよ。

(1) この1年間の、平均最高気温と平均最低気温

(2) 最も雨が多く降った日とその量

(3) 各月の平均日照時間

- 例: ”月” をキーとして辞書配列を使う（それ以外の方法でももちろんOK）

- 余裕のある人は、クラスを使って書いてみよう
- プログラムファイルに homework05.py という名前を付けて、ITC-LMSから提出すること
- 締切：6月26日午前8時（次回授業の直前）