An Introduction to Python Part 5

M. Uemoto

Yoshida Lab

31, July, 2013

本日の予定

- 先週のおさらい
- 基本文法
  - 6 章：条件分岐
    - ブール演算
    - 関係演算子
    - if 文
  - 正規表現 (入門)

前回のおさらい

リストのスライス

```
                    0     1     2     3     4     5
data       =  ["H", "He", "Li", "Be", "B", "C"]

data[1:4]  =       ["He", "Li", "Be"]
```

- `data[i:j]`
  - *i* 以上 *j* 未満の添字のリスト要素を抽出した新しいリストを作る

リストメソッド

- `data.append(v)`
  - リスト変数 *data* の最後に要素 *v* を追加する
- `data.pop()`
  - リスト変数 *data* の最後に要素 *v* を取り出す
- その他
  - `insert`, `remove`, `reverse`, `sort` など

ファイル (test.txt) の読み込み

```
for line in open("test.txt", "r"):
    print line
```

変数 `line` に一行ずつテキストを読み込み、直下の処理を繰り返す

特定の文字列が含まれているか判定

```
if "total energy=" in line:
    print line
```

”テキスト” `in` 文字列変数

- 変数に特定の文字列が含まれていれば True を返す

文字列を特定の記号で分解する

```
>>> "total energy= 3.0".split("=")
["total energy", " 3.0"]
```

条件分岐とブール値

ブール代数

- 真 (True)
- 偽 (False)

bool 型

- 真偽値を記録するための Python の変数型
- True と False の二通りの値のみをとる
- not、 and、 or などの演算子がある

否定 (not)

```
>>> not True
False
>>> not False
True
```

- not 演算子
  - 直後の真偽値を反転させる

条件分岐と関係演算子

- 関係演算子
  - 二つの数値を比較した結果を True か False で返す

関係演算子の例

```
>>> 2 < 3
True
>>> 4 < 3
False
>>> 3 <= 3
True
>>> 3 == 3
True
```

- 主な演算子として
  - 大小 (<, >)、以上以下 (<=, >=)、等価 (==)、≠ (!=)
- 比較の結合 "(2.0 < 3.0) and (3.0 < 1.0)" → False
  - $(2.0 < 3.0) \rightarrow \mathit{True}$, $(3.0 < 1.0) \rightarrow \mathit{False}$
  - $\mathit{True} \wedge \mathit{False} \rightarrow \mathit{False}$

条件分岐と論理演算

論理積 (and)

```
>>> True and True
True
>>> True and False
False
>>> False and False
False
```

両方が真のときのみ答えも真になる

論理和 (or)

```
>>> True or True
True
>>> True or False
True
>>> False or False
False
```

一方が真なら答えも真になる

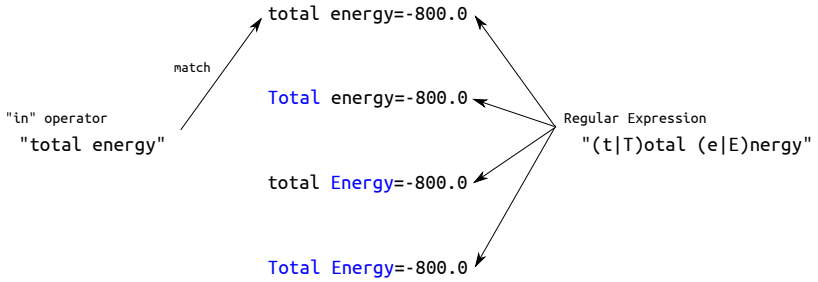
条件分岐

if 文

```
if ph < 7.0:
    print "acidy"
elif ph == 7.0
    print "neutral"
else:
    print "alkaline"
```

- if 条件式 :
  - 条件式が真 (True) なら直下の処理を実行する
- elif 条件式 :
  - 直前の if 文が偽 (False) でなおかつ条件式が真 (True) なら、
- else:
  - すべての条件式が偽 (False) なら、

正規表現 (1)



- 正規表現 (Regular Expression)
  - いくつかの文字列を一つの形式で表現する方法
  - すべての表現を列挙することなくパターンマッチングができる
  - 大量の文書を検索するときに便利
- パターンマッチ
  一定の条件 (パターン) に合致する文字列のみを取り出す

正規表現 (2)

in 演算子による判別

```
>>> text1 = "total energy=-800.0"
>>> text2 = "Total Energy=-800.0"
>>> "total energy" in line1
True
>>> "total energy" in line2
False
```

正規表現による判別

```
>>> import re
>>> re.search(r"(t|T)otal (e|E)nergy", line1)
<_sre.SRE_Match object at 0x.....>
```

- `re.search(r" パターン ", 文字列データ )`
  - マッチングに成功 → `SRE_Match object`
  - マッチングに失敗 → 戻り値なし

正規表現 (3)

- 練習
  - ファイル (file00.out) から特定の文字列 (total energy) を含む行を抽出する

test1.py

```
#!/usr/bin/env python
import re

f = open("file00.out", "r")
for line in f:
    if re.search(r"(t|T)otal (e|E)nergy", line):
        print line
f.close()
```

正規表現 (4)

特殊文字 (メタ文字)

- 正規表現ではいくつかの文字に対して特殊な役割を与えている
- `.`···(すべての文字にマッチングする)
  - `"wh.."`
    - `"what","when","whom"···` にマッチ
- `[]`···(括弧内にある任意の一文字にマッチする)
  - `"[tT]otal"`
    - `"total","Total"···` にマッチ
  - `"atom[0-9]"`
    - `"atom0", "atom1", ···` とすべての数字にマッチ
- `|`··· 何通りかあるパターンのどちらかとマッチング)
  - `"total|Total|TOTAL"`
    - `"total","Total","TOTAL"`にマッチ
  - グループ化 `()` を使って`"(left|right) hand"`
    - `"left hand","right hand"`

おまけ (メタ文字)

繰り返し

- `*`・・・(直前の文字の 0 回以上の繰り返し)
  - `"go*"`
    - `"g"`, `"go"`, `"goo"`, ・・・ とマッチ
- `+`・・・(直前の文字の 1 回以上の繰り返し)
  - `"go+"`
    - `"go"`, `"goo"`, ・・・ とマッチ
- `()`・・・(グループ化、カッコで囲んだ文字をひとまとまりに扱う)
  - `"(ab)*c"` → `"abc"`, `"ababc"`

その他

`\s` スペース・タブなど空白文字とマッチ
`\d` 0 から 9 までの半角数字
`\w` 半角のアルファベット

まとめ

条件分岐

- bool 値を使って真偽を表現できます
- bool 演算子 not/and/or をこれらの値を結合できます
- より大きい、より小さいなどの関係演算子は値を比較してブール値を生成します
- 来週以降の予定を決めておくこと (メモ)