# An Introduction to Python Part 6

M. Uemoto

Yoshida Lab

7, Aug, 2013

## 本日の予定


- 最終回
- 辞書型オブジェクト

## これまでのおさらい

コレクション

- リスト・辞書型・集合型
  - データのコレクション
  - すべての数値をまとめて一つの変数に格納できる
- リスト型
  - カッコの中にカンマで区切られた値を入れる

リストへの代入

```
>>> prime = [2, 3, 5, 7, 11, 13, 17, 19]
>>> atom = ["H", "He", "Li", "Be"]
>>> print atom[3]
Li
```

- リスト要素へのアクセス
- 角カッコで添字を囲む
  prime[0],prime[1], prime[2] ··· 変数名 [添字]

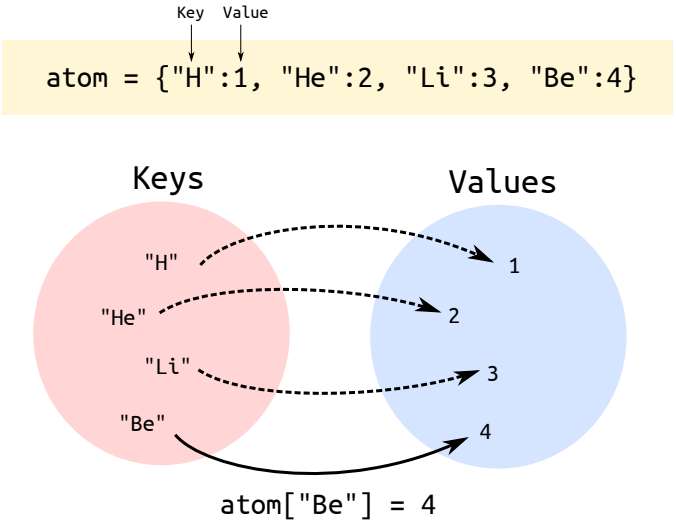
- 正規表現 (Regular Expression)
  - いくつかの文字列を一つの形式で表現する方法
  - すべての表現を列挙することなくパターンマッチングができる
  - 大量の文書を検索するときに便利
- パターンマッチ
  一定の条件 (パターン) に合致する文字列のみを取り出す

正規表現によるパターンマッチングの例

```
>>> import re
>>> line1 = "Total energy 1.00 Ryd"
>>> 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

# 辞書型オブジェクト (1)




- 辞書型 (あるいはマップ型)
  - Perl ではハッシュないし連想記憶と呼ばれることも
- キーと値のペアを格納する

# 辞書型オブジェクト (2)

- 辞書型の定義
  - 丸カッコの中にキーと値の対を入れて作る
- 辞書の要素へのアクセス
  - キーを角カッコの中に入れて辞書に添える

辞書の操作

```
>>> atom = ["H":1, "He":2, "Li":3, "Be":5]
>>> atom["Be"] = 4
>>> print atom["Be"]
```

- キーが辞書に含まれているかどうかテストするには in 演算子

in 演算子

```
>>> if "He" in atom:
>>>     print "He is atom"
```

# 辞書型オブジェクト (3)

- for ループで内容を処理したい場合

辞書型に対する for ループ

```
>>> atom = ["H":1, "He":2, "Li":3, "Be":5]
>>>
>>> for x in atom:
>>>     print x, atom[x]
>>>
>>> for (x, y) in atom.items():
>>>     print x, y
```

- for ループ変数 in 辞書オブジェクト :
  - ループ変数には辞書のキーが次々代入される
  - リスト型ではリスト要素がループ変数に代入されるのに対し, 辞書型ではキー変数が代入される
- for ( 変数 1, 変数 2 ) in 辞書オブジェクト .items():
  - リストのキーと要素のペアをループ変数に入れる (items() メソッド)

# 辞書型オブジェクト (4)

- その他メソッド

辞書メソッドの例

```
>>> atom = ["H":1, "He":2, "Li":3, "Be":5]
>>>
>>> print atom.keys()
>>> print atom.values()
>>> atom.clear()
>>> print atom
```

- 辞書オブジェクト.keys()
  - キーの一覧表を返す
- 辞書オブジェクト.values()
  - 値の一覧表を返す
- 辞書オブジェクト.clear()
  - 辞書のデータをすべて消去する

# 外部プログラムとの連携 (1)

- os.system("UNIX コマンド")
- subprocess.call("UNIX コマンド")
  - コマンド・外部プログラムの実行

外部プログラムの呼び出し (1)

```
>>> import os
>>> os.system("emacs")
```

- command.getoutput("UNIX コマンド")
  - 実行した結果 (標準出力) も欲しい場合は getoutput を使う

外部プログラムの呼び出し (2)

```
>>> import command
>>> result = command.getoutput("ls -al")
>>> print result
```

# 外部プログラムとの連携 (2)

plot.py

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

command = """gnuplot << EOF
plot sin(x) title "sin function"
set terminal postscript enhanced
set output "sample.eps"
replot
quit
EOF"""

os.system(command)
```

- command = """複数行のテキスト"""
  - """から次に"""が現れるまでの複数行の文字列を変数に代入する

# qsub との連携 (1)

job-sge.py

```
#!/bin/python
#$ -S /bin/python
#$ -cwd
#$ -N name
#$ -l h_vmem=1024M
#$ -l h_cpu=99:00:00
#$ -p -10
#$ -V

os.system("./specx < ./in/fe > ./out/fe")
```

- os.system("コマンド")
  - コマンド・外部プログラムの実行

ジョブを投げる場合は

- qsub job.py

# qsub との連携 (2)

job-pbs.py

```
#! /bin/python
#$ -cwd
#PBS -l nodes=1:ppn=1
import os

os.chdir(os.environ["PBS_O_WORKDIR"])
os.system("./specx < ./in/fe > ./out/fe")
```

- os.environ["環境変数"]
  - システムの環境変数を読みに行く
  - PBS Torque では PBS_O_WORKDIR にワークディレクトリ名が保存される
- os.chdir(ディレクトリ)
  - UNIX の cd コマンドと同じ

## おまけ (メール送信)

mail.py

```
import smtplib
import email

text = """
Hi ....,
Hello
"""
msg = email.mime.text.MIMEText(text)
msg['Subject'] = "Subject of Email"
msg['From'] = ".......@aquarius.mp.es.osaka-u.ac.jp"
msg['To'] = ".......@aquarius.mp.es.osaka-u.ac.jp"
s = smtplib.SMTP()
s.connect()
s.sendmail(msg['From'], [msg['To']], msg.as_string())
s.close()
```

## 課題

### 問題 1

Python の数学関数を利用して,$y = \exp(-x * x)$ を計算し, その結果を gnuplot でプロットして `plot.eps` として保存するスクリプトを作ってください

ヒント (データの書き出しを行うサンプル)

```
import math

f = open("data.d", "w")
for i in xrange(-30, 30):
    x = i*0.1
    f.write("%f %f\n" % (x, math.exp(x*x))
f.close()
```

## 課題

### 問題 2

- `d1={"H": "1", "He": "2", ··· }`
- `d2={"Ne": "10", "Ar": "18", ··· }`

二つの辞書の和集合 (双方の要素を併せ持った辞書) を作る関数 `addDict(d1, d2)` を考えてください

### 問題 3

`addr={"name1": "email_address1", ··· }`と各個人の名前とメールアドレスが格納された辞書データがある. 全員に対して以下のような内容のメールを送信するスクリプトを考えてください.

```
Hi %s,

Hello World
```

`%s` は送信先の各人の名前と置き換えられる

## 課題

### 問題 4

疎ベクトルは

$$(1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 3 \quad 0 \quad 0 \quad 0) \tag{1}$$

のように, エントリの大半が 0 になっているベクトルです. それらの 0 をすべてリストに格納してもメモリの無駄ですから, プログラマはよく, 辞書を使って 0 以外のエントリを管理します. たとえば, 例として示したベクトルは, 第 0 要素の値が 1 で第 6 要素の値が 3 ですから, `{0:1, 6:3}`と表現することができます.

- 辞書形式で格納されている二つの疎ベクトルを引数にして, その和である新しい辞書を返す `sparse_add` という関数を考えてください
- 二つの疎ベクトルのドット積を返す `sparse_dot` という関数を考えてください