ALPS チュートリアル: Python

CMSI 神戸ハンズオン

ALPS Collaboration
http://alps.comp-phys.org/

基本的なデータ型

- **数値**: int, (long), float, complex
- **文字列**: 'hello,python'
- **list**: a = [1, 2, 3]
- **tuple**: b = (1, 2, 3)
- **dictionary**: c = {'apple': 100, 'orange': 200, 'pear': 300}
- **set**: set(1, 2, 3)
- **bool**: True, False
- **array**: numpy array([1,2,3])

数値

- int, long
  - python3 では int,long は int に統合された
  - python2x と python3x では整数同士の演算で計算結果が違う: python2: -1/2 = -1, python3: -1/2 = -0.5
- float は倍精度のみ
- complex
  - 'j' もしくは 'J' で虚数部を表す 2 + 3j
  - 実部 real(2+3j) = 2, 虚部 imag(2+3j) = 3

文字列

```
>>> s = 'hello, world'    # 文字列の定義
>>> s.split(' ')
>>> ['hello,', 'world']
>>> s.split(' ')[0] + ' ' + 'Haruhiko'      # 文字列の結合
>>> 'hello, Haruhiko'
>>> ' '.join(s.split(' ')[0], 'Haruhiko')  #
>>> 'hello, Haruhiko'
```

- シングルクォートとダブルクォートの振る舞いは同じ（シェルでは違いますね）
- 複数行の文字列は "'..."' もしくは """...""" で囲む
- 文字列のスライス: s[1:3] = 'el'
- 文字列の分割 (split) と結合 (+, join)

list, tuple

```
>>>a = [1,2,3,4,5] # list
>>>a[0]    # インデックスは 0 スタート
1          # 要素 1 個だけなら返り値はスカラー
>>>a[2:4] # 2<=,<4 番目のの要素が返される.
[3,4]      # 複数の要素なら返り値はリスト.
>>>b = (1,2,3,4,5) # tuple
>>>b[1:]           # ':' の値を省くこともできる
(2, 3, 4, 5)
```

- ここではリストの要素として数値のみを扱ったが，python オブジェクトなら何でも リストの要素として扱える

list のメソッド

```
>>>a.reverse() # 要素の並びを逆にする
>>>a           # a そのものが変更されるので注意!
[5, 4, 3, 2, 1]
>>>a.pop()  # リストの最後尾の要素を取り出す.
1           # 返り値は取り出された要素
>>>a
[5, 4, 3, 2]
>>>a.sort()
>>>a
[2, 3, 4, 5]
```

- ここではリストの要素として数値のみを扱ったが，python オブジェクトなら何でも リストの要素として扱える

list と tuple の違う点

```
>>>b = (1)  # これは tuple にならない!
>>>type(b)
int          # 整数扱いになる
>>>b = (1,) # 要素 1 個の tuple を定義する
>>>type(b)
tuple        # これはちゃんと tuple になっている
>>>a = [1]  # これは list
>>>type(a)
list
```

list と tuple の違う点

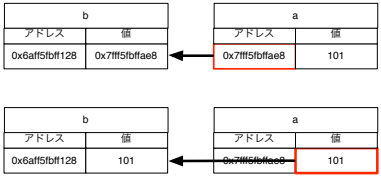
```
>>>b += (2, 3, 3)  # 要素を付け加えることはできる
>>>b
(1, 2, 3, 3)
>>>b[3] = 5        # 要素を変更することはできない
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

- 要素 1 個の場合の扱いが違う
- tuple の要素の値は変更できない.
- tuple は辞書のキーに登録できる. list は不可.

list のコピー

浅いコピーと深いコピーの 2 種類ある

- 浅いコピーではオブジェクトのアドレスをコピーする
- 深いコピーではオブジェクトの値をコピーする




```
>>>a = [101, 102, 103]
>>>b = a      # アドレスをコピー
>>>a[1] = 333
>>>print a, b
[101, 333, 103] [101, 333, 103]
>>>b = a[:]  # 値をコピー
>>>a[1] = 444
>>>print a, b
[101, 444, 103] [101, 333, 103]
```

ネストされた list のコピー

浅いコピーになってしまう例

```
>>>a = [1, [2, 3]]
>>>b = a[:]
>>>a[0] = 4
>>>a[1][0] = 5
>>>print a, b
[4, [5, 3]] [1, [5, 3]]
>>>
```

深いコピー

```
>>>import copy
>>>a = [1, [2, 3]]
>>>b = copy.deepcopy(a)
>>>a[0] = 101
>>>a[1][0] = 102
>>>print a, b
[101, [102, 3]] [1, [2, 3]]
```

- **ネストされているとスライス ':' で返されるのがアドレスになってしまう**
- **ネストされた list で深いコピーをするには copy モジュールの deepcopy を使う**

辞書型の使い方

```
>>>dic = {'key0': 0, 'key1': 1}    # key:value の組を登録
>>>dic['key2'] = 2                 # key2:2 を登録
>>>dic.keys()                      # 辞書に登録されている全ての key
['key2', 'key1', 'key0']
>>>dic.values()                    # 全ての val
[2, 1, 0]
>>>dic.items()                     # 全ての key:val の組を表示
[('key2', 2), ('key1', 1), ('key0', 0)]
>>>del dic['key0']                 # 辞書から key を削除
>>>dic.items()
[('key2', 2), ('key1', 1)]
```

- 辞書の要素はソートされていない．登録順なども関係ない．

if-elif-else 文の使い方

if-elif-else で条件分岐を作れる.

```
>>>if a > 0:
>>>    print '0'
>>>elif a == 0:
>>>    print '1'
>>>else:
>>>    print '2'
>>>
>>>
```

三項演算子

```
>>>val = val1 if cond1 else val2
```

for 文の使い方

```
>>>for i in ('a', 'b', 'c', 'd'):
>>>    print i,    # コンマで改行を抑制している
a b c d
```

アンパック代入と enumerate()

```
>>>for i,j in enumerate(('a', 'b', 'c', 'd')):
>>>    print i, j
0 a
1 b
...
```

アンパック代入は python **で使える一般的なテクニックです.**

```
>>>i,j,k = ['a', 'b', 'c']
```

while 文の使い方

```
a = 0
while a in range(10):
    a += 1
    if a < 3:
        continue
    elif a == 8:
        break
    print a
```

- カレントディレクトリに上の内容で exWhile.py というファイルを作って import してみましょう
- "val in シークエンス:" というフレーズは while だけでなく if, for など至る所で使えます
- continue, break も同じく if, for などでも使えます

関数

```
>>> def f(x, y):
>>> ....z = x * y  # 空白 4 つのインデント!
>>> ....for i in [1, 2, 3]:
>>> ....    z += i
>>> ....return z
>>>
>>> f(2,3)
12
```

- def 関数名 (変数,...): で関数が定義できます
- Python ではインデント (空白 4 つ) によりスコープを制御します

Python プログラムの階層構造

- **プログラムファイルのディレクトリ構成 == 名前空間**
- **モジュール**
  - **1 つの "hoge.py" ファイルが 1 つのモジュール**
- **パッケージ**

ファイルの階層構造

```
pyalps/
    alea.py
    dataset.py
```

モジュールの階層構造

pyalps.alea.class.method
pyalps.dataset.class.method

モジュールの読み込み

```
>>> import fff as f     # 別名をつけてインポート
>>> f.ggg(x, y)
...
>>> from fff import *  # fff 以下のすべてをインポート
>>> ggg(x, y)          # 名前空間 fff が外れる
...
>>> from fff import ggg as g # ggg のみを指定してインポート
>>> g(x, y)
```

- fff モジュール中の ggg 関数を呼び出しています.
- 呼び出し方により名前空間の階層が変わっています.

ファイルの読み書き

ファイルから 1 行ごとにデータを読み込む方法

```
for line in open('dat.txt', 'r'):
    items = line.split('')
    print items[0], float(items[1])
```

入力データ (dat.txt)

```
a 3.432
b 1.42
c 2.159
```

- open で dat.txt ファイルを readonly で読み込む
- 読み込んだファイルを 1 行毎に処理する．読み込まれた行は 1 つのストリングとして扱われるので区切り文字（今の場合空白）で分割している．
- 読み込んだストリングを浮動小数点にキャストしている．

上の例では for 文の終了とともにファイルは自動で閉じられる．しかし，一般的には

- f = open('dat.txt', 'r') として読み込んだ場合，ファイルを読み終えたら f.close() で閉じなければならない．

参考サイト

- **科学技術計算のために** python **を始めよう**
  http://www.ike-dyn.ritsumei.ac.jp/~uchida/scipy-lecture-notes/intro/index.html
- **初心者のはまりどころ**
  http://webtech-walker.com/archive/2010/10/13191417.html
- Doug Hellmann **さん** http://doughellmann.com
  - **標準モジュールなどの使い方が例とともに書いてあり分かりやすい**