An Introduction to Python Part 1

M. Uemoto

Yoshida Lab

July 2, 2013

Python とは

- Python とは
  オランダ人のグイド・ヴァンロッサムが１９８９年から開発しておりオープンソースのプログラミング言語
- 初心者にも扱いやすい言語
- 海外では人気言語に (Google では公式言語に採用される)

- コメディ番組「空飛ぶモンティ・パイソン (Monty Python's Flying Circus)」より命名

Python の特徴

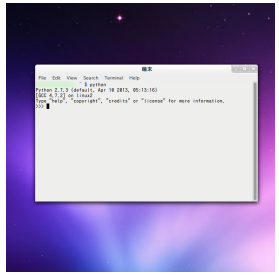
- 軽量 (Lightweight) なオブジェクト指向言語
  - ≠ 処理が早い
  - 短時間でコーディングできる（すぐ書いて, すぐ動く）
- 可読性の高い（よみやすい）文法
- インデントを使ったブロック構造
- コンパイルの必要なし
- 豊富なライブラリ
  - 標準ライブラリ
    正規表現,OS のシステムコール,XML 処理系, シリアライゼーション, HTTP,FTP 等の各種通信プロトコル, 電子メール,CSV ファイルの処理, SQL データベース接続,GUI フレームワーク, 構文解析ツール ··· ect
  - 拡張ライブラリ
    NumPy(行列演算パッケージ), SciPy(科学計算パッケージ）, Python Imaging Library(画像処理), ··· ect
  - (電池が付属しています/Battery Included)

入手方法

- 配布元サイト
  - http://www.python.org/download/
- 最新版（２０１３年７月時点）
  - バージョン２系列
    - Python 2.7.5
      今回の実習ではこのバージョンを推奨します
  - バージョン３系列
    - Python 3.3.2
- Python2 と Python3
  - 文法が微妙に違う（後方互換性がない）
  - 将来的には Python3 のほうが標準になるはずであるが、現時点では、Python2 が一般的である。
- Windows
  - 上記のサイトからダウンロード、または Cygwin を使う
- Mac/Linux
  - 標準でインストール済み

Python の起動

- 対話モード



コマンドライン上で”python”コマンドを実行する。

```
$ python
Python 2.7.3 (default, Sep 26 2012, 21:51:14)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more inform
>>>
```

終了するときは”Ctrl + D”を使う

簡単な計算

```
>>> 1.0+2.0
3.0
>>> 1.0-2.0
-1.0
>>> 1.0*2.0
2.0
>>> 1.0/2.0
0.5
>>> (1.0+2.0) / 3.0
1.0
>>> 12 % 10
2
>>> abs(-3)
3
```

- % あまり
- abs 絶対値

数学関数

```
>>> import math
>>> math.sin(12.0)
```

- import math
  数学関数のモジュール (math) を使用出来るようにする
- その他関数、定数として
- math.sin
- math.cos
- math.log
- math.exp
- math.sqrt
- math.erf
- math.e
- math.pi

複素数

虚数単位として j が使える

```
>>> (1.0+2.0j) * (2.0*3.0j)
(-12+6j)
>>> abs(-12+6j)
13.416407864998739
>>> (-12+6j).real
-12
>>> (-12+6j).imag
6
```

数学関数の中には複素数に対応していないものがある

```
>>> math.exp(1.0j*math.pi) + 1.0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't convert complex to float
```

例 exp が複素数に対応していない

変数

- 変数 x への代入

```
>>> x = 1.0
>>> x = x+1.0
>>> x += 1.0
```

- 入力された数値を x へ代入 (input 文)

```
>>> x=input("Input any number...")
1.0
```

- 変数の内容を表示 (print 文)

```
>>> print x
1.0
```

ループ (for)

- 同じ処理を複数回繰り返す (Fortran の Do 文に相当)

i=0...9 まで 10 回繰り返す

```
for i in xrange(0, 10):
    print "loop"
    print i
```

二重ループ

```
for i in xrange(0, 10):
    for j in xrange(0, 5):
        print i,j
```

- for 直下のインデントブロックが繰り返される
- 最後のコロン「:」をつけること
- $i \cdots$ ループ変数
- ループ回数 xrange(下限、上限) または xrange(上限)

制御構造

- 条件分岐 (if 文)
- ループ (for 文、while 文)
- 関数（def 文)

- インデントブロック
  インデント幅が同じ部分が一塊のソースコードとして処理される
  - タブ
  - 空白（通常は空白４文字）

条件分岐 (if 文)

- 記述した条件が True であれば、条件配下の処理が実行される

```
if value == 1:
    print unicode("value is 1")
elif value == 2:
    print unicode("value is 2")
elif value == 3:
    print unicode("value is 3")
else:
    print unicode("otherwise")
```

- if 条件式 1: 条件式 1 が真の時に実行する処理
- elif 条件式 2: 条件式 1 が偽で条件式 2 が真の時に実行する処理
- else: 全ての条件式が偽の時に実行する処理

- a == b a と b は等しい
- a != b a と b は等しくない
- その他、比較演算子 $(<, >, =<, =>, \cdots)$

関数定義 (def)

```
def polynomial(a, b, c, x):
    first  = a*x*x
    second = b*x
    third  = c
    return first+second+third

（中略）

x = 1.0
y = polynomial(1.0, 2.0, 3.0, x)
```

- 関数の定義
  - def 関数名 (引数)
  - 直下のインデントブロックの処理が実行される
  - 「return 式」を結果として返す
- 呼び出し
  - $polynomial(a, b, c, x)$ 関数名 (引数...) の形式で呼び出す

例（１００未満の偶数の和）

- def f(n):
  n 未満の偶数の和を求める関数 $f(n)$ の定義
- for i in xrange(1, n):
  ループ変数 $0 \leq i < n$ の範囲内で処理を繰り返す
- if i % 2 == 0
  2 で割ったあまりがゼロなら（偶数）
- s += i
  $s + i$ を s に代入する
- return s
  計算結果を関数 $f(n)$ の戻り値として返す
- print f(100)
  関数 $f(n = 100)$ の結果を出力

例（１００未満の偶数の和）

エディタを使ってソースコードを保存

test1.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def f(n):
    s = 0
    for i in xrange(1, n):
        if i % 2 == 0:
            s += i
    return s

print f(100)
```

python のプログラムを実行するときは、

```
python test1.py
```

問題

Multiple 3 and 5

- If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.
- Find the sum of all the multiples of 3 or 5 below 1000.

Even Fibonacci numbers

- Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:
- 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
- By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

- Project Euler(http://projecteuler.net/problems) 例題 1, 2 より
- 教科書 p24 の問題