An Introduction to Python Part 3

M. Uemoto

Yoshida Lab

July 16, 2013

本日の予定

- 先週のおさらい
- 基本文法
  - 4章：モジュールとインポート文
  - 5章：リスト (前半)
- 演習問題

前回のおさらい (1)

文字列型

```
>>> x = "Text"
>>> print x
Text
>>> print "Hello\nWorld"
Hello
World
>>> print "Today is %d, July." % 17
Today is 17, July.
>>> print "x=%f y=%f z=f" % (1.0, 2.0, 3.0)
x=1.000000 y=2.000000 z=3.000000
```

- シングル・ダブルクオートで文字列を挟む
- エスケープ文字 (`\n`) で改行
- フォーマット記号 (`%d`, `%f`, `%e` ...)
  %以降の数値と置き換えられる

前回のおさらい (2)

ファイル書き込み

```
f = open("test.txt", "w")
f.write("Hello World\n");
f.close()
```

- `f = open("test.txt", "w")`
  - open 命令、ファイル test.txt を書き込みモードで開く
  - 変数 *f* にファイルオブジェクトを代入
    (以降 *f* は"test.txt"を意味する)
- `f.write("Hello World\n")`
  - write 命令、ファイル *f*(test.txt) に書き込む
  - `\n`(行末の改行)
- `f.close()`
  - close 命令
    書き込み終了後は必ずファイルを閉じること

おさらい (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

モジュール

- 一つのファイルにまとめられた関数のコレクション
- math(数学関数)
  - `math.sin, math.cos, math.tan` ...
- os(システムコマンド関連)
  - `os.chdir, os.chmod, os.chown` ...
- time(時刻関連)
  - `time.localtime, time.sleep, time.timezone` ...
- その他モジュール (http://docs.python.jp/2/py-modindex.html)
- 類似した機能を持つ関数群が、モジュールにまとめて格納される
- モジュールの関数を使いたい時
  - `import` 文

モジュールのインポート

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

- `import` モジュール名
  特定のモジュールに含まれる関数群を使用可能にする
- モジュール名. 関数名 (· · ·)
  モジュール名のあとに関数名をつなげて呼び出す

省略バージョン

```
>>> from math import sin, cos, tan
>>> sin(12.0)
```

- ”モジュール名＋ドット”を省略したいとき
- `from` モジュール `import` 関数 1, 関数 2, · · ·
  - 特定の関数のみを使用可能にする
- `from` モジュール `import *`
  - すべての関数を使用可能にする (非推奨)

独自モジュールの定義

- 以前作ったコードを”拡張子 py のファイル”に保存
- あとから再利用したい

code1.py

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

- 外部のスクリプトから code1.py の関数を呼び出したい
- `import` 文を使う

```
>>> import code1
>>> code1.polynomial(1.0, 2.0, 3.0, 4.0)
```

- ファイル名から拡張子 py を取り除いたものが”モジュール名”

リスト

- いままでの変数
  - １つの数値または文字列に限る
  - 大量の数値の管理に複数の変数が必要になる
- リスト・タプル・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

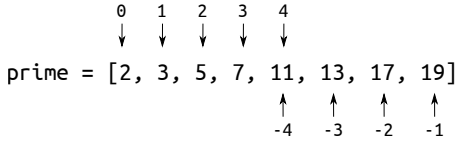
リストへの代入

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

- print prime や print atom でリスト変数の内容を表示

リストのインデックス

- リストに格納された変数へのアクセス
  - アクセスしたい要素を指定する添字 (index)
  - 先頭位置は添字 0
  - 二番目の要素は添字 1



- リストの末尾から数える添字
  - 最後の要素の添字 −1
- 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]
- 通常の変数と同様に扱える

リストの書き換え

リストの書き換え

```
>>> x = ["a", "b", "c"]
>>> print x[0]
a
>>> x[0] = "d"
>>> print x
["d", "b", "c"]
```

- リスト要素は通常の変数と同様に書き換え可能である

リストの連結

```
>>> x = [1, 2]
>>> y = x + [4, 5]
```

- 連結演算子+ ··· リストとリストを連結して新しいリストを作る
- リストに新しい要素を追加する
  - l = l + ["new"]
  - l += ["new"]

リストを使った for ループ

for ループ

```
prime = [2, 3, 5, 7, 11, 13, 17, 19]
for p in prime:
    print p
```

- for ループ変数 in リスト :
  - リストから要素を一つずつとりだして
  - 直下のループブロックを繰り返す
- ループブロックの文はインデントされる必要がある

上と等価な処理

```
p = prime[0]
print p
p = prime[1]
print p
```

(以下同様に続く)

(おまけ) リストに関連する関数 (1)

len 関数

```
>>> prime = [2, 3, 5, 7, 11, 13, 17, 19]
>>> len(prime)
8
```

- `len`
  リストの長さを求める
- `max`
  リストの要素の中で 最大値を求める
- `min`
  リストの要素の中で 最小値を求める
- `sum`
  リストの要素の合計値をもとめる

(おまけ) リストに関連する関数 (2)

文字列の分割

```
>>> text = "energy=1.5"
>>> token = text.split("=")
>>> print token
["energy", "1.5"]
```

- 文字列変数.split(”文字”)
- 特定の文字でテキストを分割する

ファイルの一覧を読み込む

```
>>> import glob
>>> ls = glob.glob("~/*.py")
```

- `glob.glob`(”ディレクトリ”)
  - UNIX の ls コマンドと使い方は一緒
  - 結果 `ls[0]`, `ls[1]`, `ls[2]` にそれぞれファイル名書き込まれる

(おまけ) リストに関連する関数 (3)

ファイルの内容を読み込む

```
f = open("test.txt", "r")
line = f.readlines()
f.close()
```

- `f = open("test.txt", "r")`
  - open 命令、ファイル test.txt を読み込みモード (”r”) で開く
  - 変数 *f* にファイルオブジェクトを代入
    (以降 *f* は”test.txt”を意味する)
- `line = f.readlines()`
  - ファイルの内容を一行一行読み込んで、変数 `line` に代入
  - 行単位でデータが格納されたリストを返す
    - 1 行目 `line[0]`
    - 2 行目 `line[1]`
- `f.close()`
  - 書読み込み終了後は必ずファイルを閉じること

まとめ

今回は 4 章および 5 章 4 節まで説明しました。

- リストは大量の数値・データをまとめて格納することができる
- リストに用いられるオブジェクトのことをリスト要素とよび、添字という位置情報を使って結果を参照できます
- リスト要素は通常の変数と同じように読み書き可能です

次週は、5 章 5 節以降 (リスト構造の続き) および今回最後に説明した、split, glob, readlines を使って、出力ファイルの処理に関する演習を行う予定です。

問題

練習問題

- *alkaline_earth_materials* という変数に５種類のアルカリ土類金属の原子番号、ベリリウム (4), マグネシウム (12), カルシウム (20), ストロンチウム (38), バリウム (56), ラジウム (88) を格納するリストを作ってください。
- ラジウムの原子番号にアクセスするにはどの添字を使ったらいいですか、正の数字と負の数字でそれぞれに東リ答えを書いてください
- *alkaline_earth_materials* に含まれている要素数を教えてくれるのはどの関数ですか？
- *alkaline_earth_materials* に含まれている最も高い原子番号を返すコードを書いてください

(教科書 p92 より)

第一回演習問題 (Fibonacci 級数) (答：4613732)

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

def fibonacci(n):
    if 1<= n <= 2 :
        return n
    else:
        return fibonacci(n-2) + fibonacci(n-1)

Threshold = 4000000
n = 1
s = 0
m = fibonacci(n)

while m <= Threshold:
    if m % 2 == 0:
        s += m
    n += 1
    m = fibbonacci(n)

print s
```

演習問題 1

- prime を [2, 3, 5, 7, 11, 13, 17, 19] とする。すべての値を合計して total 変数に結果を格納する for ループを書いてください
- ヒント
  - total に初期値を 0 を与えてから、ループ本体で total に値を加算します

演習問題 2

- 10 未満の素数は $2, 3, 5, 7$, の 4 つで最大の素数は 7 になります。では 10000 未満の素数の個数、および最大の素数はいくらですか？

ヒント:最初の n 個未満の偶数のリストを作る関数

```
def f(n):
    number = []
    for i in xrange(1, n):
        if i % 2 == 0:
            number += [i]
    return number
```