Python講座

第六回その他

takumi kaimai

2017 年 7 月 27 日

目次

1　データ構造

1.1　タプル

タプルは，リストと同様にオブジェクトを保存できるオブジェクトの一種である．
リストにかなり似ているが，異なる部分が二点ある．

- 上書きができない(不変性)
- []ではなく，()でオブジェクトを囲んで使用する

他の部分については，リストと同様であり，インデクシング(t[:2]，t[3:7]のような指定した部分の要素だけを取り出す方法)， len関数を使用した長さの確認，+演算子を使用したデータ構造同士の連結等の機能は使用できる．

リストとほぼ同様の機能を持つタプルなので，必要性が感じられない人もいると思うが，不変性を持っているので自作のライブラリ等で変更されたくないで変更を保存する際に使用できる．

```
1   #coding:utf-8
2   #タプルの宣言
3   t = (1,2,"12")
4   #インデクシング
5   print( t[:2])
6   #サイズの確認
7   print(len(t))
8   #タプルの結合
9   print(t+t)
10  #タプルのリスト化
11  l = list(t)
12  print(type(l))
13  #リストのタプル化
14  to_tuple = tuple([1,2,3])
15  print(type(to_tuple))
```

Listing 1: lec6-1.py

1.2　ディクショナリ

ディクショナリ，タプル，リストと同様にオブジェクトを保存できるオブジェクトの一種である．
他のプログラミング言語では連想配列，ハッシュ等と呼ばれている．
リストやタプルとは違い，データにアクセスする際には，対応したキー(文字列)を使用する．

この仕様により，複数のデータをひとまとめにして扱うことや，データ検索が行いやすいといった利点がある．
宣言する際には，{}を使用する．

```
#coding:utf-8
#ディクショナリの宣言
#キー:データでひとかたまりのデータ
#",”で区切って複数のデータを保存
d = {"name":"python","creator":"Guido van Rossum","birth_year":1991}

#キーを指定してデータにアクセス
print(d["name"])

#データの追加
d["version"] = "3.6.1"

#全データの表示
print(d.items())

#キーの確認
print(d.keys())

#中身(バリュー)の確認
print(d.values())
```

Listing 2: lec6-2.py

2　ちょっと便利なプログラムの書き方

2.1　ジェネレーター

　forループを使用して，リストの要素 1 つ 1 つに処理するプログラムを書く機会は頻繁に存在する．
ループで使用するリストの要素が膨大であると，メモリに格納できない可能性があるし，他のソフトウェアの邪魔になる可能性もある．
そんな時にジェネレーターを使用するとメモリを節約したプログラムを書くことができる．
ジェネレーターは使用される瞬間にデータを生成できる．
サンプルソースを確認しながら，ジェネレーターの使い方を説明していく．

```python
#coding:utf-8

def func():
    number = 0
    while True:
        number += 1
        yield number

for var in func():
    if var == 1000:
        break
    print(var)
```

Listing 3: lec6-3.py

ジェネレーターは関数として定義される．
returnは存在していないが，値を返すことができる．
ジェネレーターはreturnと違いyieldキーワードで値を返すことができる．
そして，yieldキーワードは値を返した後に，その関数の処理を止めることができる．
再度，関数を呼ぶことで関数の処理が再開される．
ただpythonでは，yieldの入った関数はイテレータ扱いになり，次の要素にアクセスする為にnext関数を使用する必要がある．
その為，lec6-3.pyのソースでは，for文のループが終わる度に，func()のイテレータが回り，次の値が生成されている．
このようなジェネレーターはrange関数でも使用されている．

また，ジェネレーターを使用するとメモリを気にせず無限に素数を生成することもできる．

```python
#coding:utf-8
def prime_list():
    number = 1
    while True:
        flag = True
        number += 1
        for var in range(2,number):
            if number % var == 0:
                flag = False
        if flag:
            yield number

for var in prime_list():
    if var > 20:
        break
    print(var)
```

Listing 4: lec6-4.py

2.2 演算定義

pythonでは同一クラスでの四則演算を定義できる．方法はクラス内に決められた名前の関数を定義するだけである． 例えば，クラス間の加算を定義したい場合は，__add__という関数を定義する必要がある．この

表1 クラス間の四則演算定義関数

演算子	関数名
+	__add__
-	__sub__
*	__mul__
/	__div__

関数は引数と返り値が決められている．引数にはself(自らのオブジェクト),other(演算する同クラスの別オブジェクト)を指定し，返り値はself,otherと同じクラスを指定する必要がある．下記は加算を定義する際のサンプルソースである．

```python
#coding:utf-8

class data:
    #初期化関数
    def __init__(self,a,b):
        self.a = a
        self.b = b

    #加算を定義する関数
    def __add__(self,other):
        a = self.a + other.a
        b = self.b + other.b
        return data(a,b)
    #表示用の関数
    def values(self):
        return self.a,self.b
#クラスの定義
data1 = data(1,2)
data2 = data(3,4)

#クラス同士の加算
data3 = data1 + data2
print(data3.values())
```

Listing 5: lec6-5.py

2.3 内包表記

[]内にfor文を書き，リストを生成する記法が存在する．
[for文内の処理 for 変数 in リスト]の様に書く．

```python
#coding:utf-8
#二の倍数のリストを作成する
even = [var*2 for var in range(10)]

#内包表記を使用しないと以下のようなスクリプトになる
#even = []
#for var in range(10):
#    even.append(var*2)

for var in even:
    print(var)
```

Listing 6: lec6-6.py

3 演習問題

3.1 フィボナッチ数列

yieldキーワードを使用して，フィボナッチ数列のジェネレーターを作成せよ．フィボナッチ数列は以下のように定義される．

$$\mathrm{F}_0 = 0 \tag{1}$$

$$\mathrm{F}_1 = 1 \tag{2}$$

$$\mathrm{F}_{n+2} = F_{n+1} + F_n \tag{3}$$

3.2 行列の演算定義

2×2の行列クラスを定義したい．
取り敢えず，2×2の値を保存できる機能，行列同士の加算減算機能を持ったクラスを作成せよ．

3.3 九九

内包表記を用いて，九九の二次元リストを作成せよ．

4 環境整備

pythonは使用用途により適した環境が変わる．環境に関連することについて解説していく．

4.1 pip

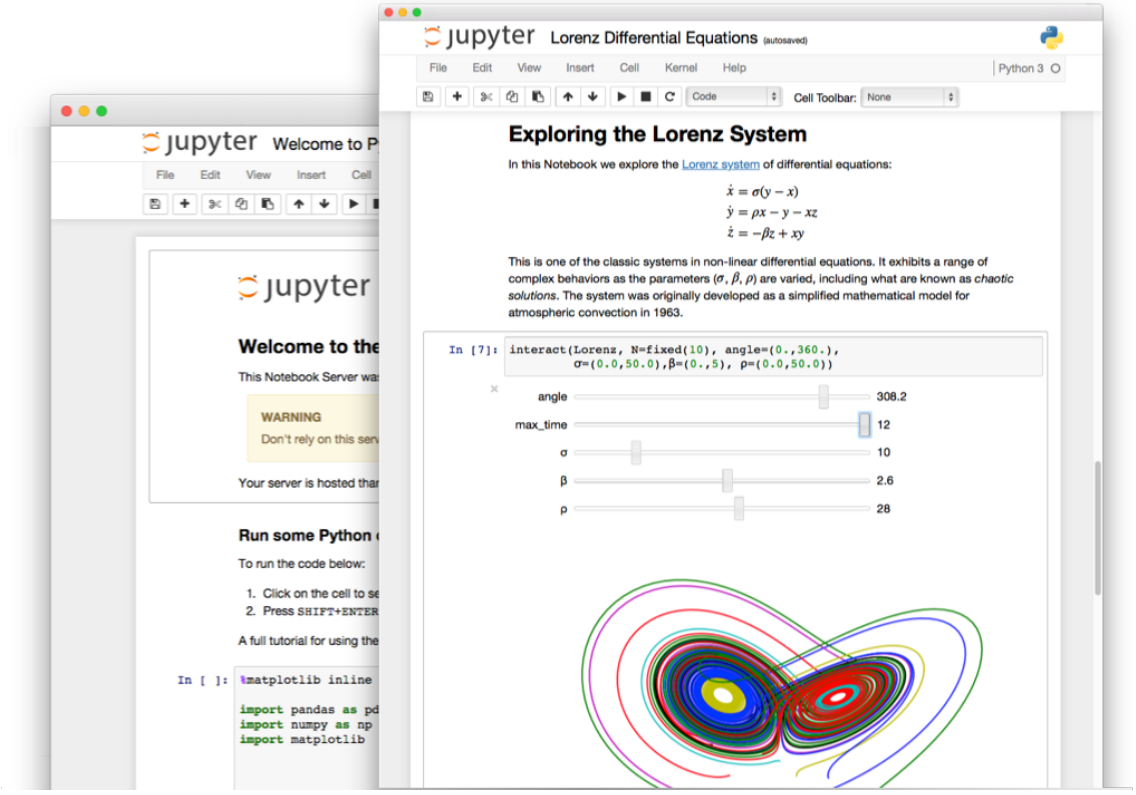
pip(Pip Installs Packages)はpythonに外部ライブラリをインストールする際に使用するパッケージ管理システムである．コマンドライン上で，pip install パッケージ名と打つことでライブラリをインストールすることができる．

4.2 pyenv,virtualenv

pyenv,virtualenvを使用することで、複数python環境を混在させることができる．例えば，python2系とpython3系を切り替えることや，インストールされているライブラリが異なる環境の切り替え，環境変数の異なる環境の切り替えなどに使用する． linuxでは，システムの処理にpythonを使用しているので，できれば仮想環境でプログラミング環境と分けておきたい．

4.3 jupytor notebook

jupytorはブラウザ上でpythonを実行させ，ソースと結果をブラウザ上で確認することができる．ソースと結果を残すことができるので，ノートの様に使用できる．



4.4 anaconda

anacondaは数値計算やデータ解析用のライブラリをインストールしやすいpythonの仮想環境である．中には，ライブラリだけでなくide，jupytor notebookなどもインストールされている．取り敢えず，anacondaをインストールしておけば，困ることはない．

4.5 iron python

iron python は.net framework(VisualBasicやc#のJVMみたいなもの)で動作するpythonのことである．c#で実装されているのでpythonのスクリプトをc#やwindowsの実行ファイルであるexeに変換する際に使用する．

5 ライブラリ

5.1 標準ライブラリ

5.1.1 os

コマンドラインで呼び出せるコマンドの機能をpythonで呼び出せるライブラリディレクトリの移動や，現在のディレクトリのファイル一覧を表示することができる pythonでアプリケーションを作る際には，おそらく重要になる

```
1  #coding:utf-8
2  import os
3
4  #改行文字を取得
5  print(os.linesep)
6
7  #現在のパスを文字列で取得
8  path = os.getcwd()
9
10 #引数で指定したパスにあるディレクトリ，ファイルを表示
11 list = os.listdir(path)
12 print(list)
13
14 #引数で指定したコマンドを実行させる
15 os.system("echo a")
```

Listing 7: lec6-7.py

5.1.2 re

正規表現に関連するパッケージ.

5.1.3 csv

csvとリストの変換に使用するパッケージ

```
1         #coding:utf-8
2         #九九の表を書き込むスクリプト
3         import csv
4         import os
5         mul = [[x*y for x in range(1,10)] for y in range(1,10)]
6         with open("output.csv","w") as f:
7             writer = csv.writer(f,lineterminator=os.linesep)
8             for var in mul:
9                 writer.writerow(map(lambda x:x,var))
```

Listing 8: lec6-8.py

5.1.4 random

乱数生成のパッケージ

5.1.5 tkinter

pythonでGUIのアプリケーションを制作する際に使用するライブラリ

5.2 インストールしておきたいライブラリ

5.2.1 numpy

数値計算用のライブラリ．例えば，ベクトル，行列などを自ら定義することなく計算させることができる

5.2.2 scipy

numpyを基盤に，計算を強化した科学技術計算用のライブラリ

5.2.3 matplotlib

numpyを基盤にしたグラフ描画用ライブラリ

5.2.4 chainer

日本製機械学習ライブラリ.

5.2.5 tensorflow

google社製の機械学習ライブラリ

5.2.6 pygame

pythonでのゲーム制作をサポートするライブラリ

5.2.7 django

webページ制作に使用するライブラリ．rubyで言うところのrailsに当たるもの．