第3章 配列を制する者は速度を制する

C以外のチョイスにPython数値計算NumPy

西住 流

リスト1 numpy.array(list,dtype)メソッドにPython標準のリストを渡すことで配列を生成できる

```
# -*- coding: utf-8 -*-        ← 日本語の文字列を扱う場合に必要
import numpy as np

# 1次元配列の生成
list1 = [1, 2, 3]
x = np.array(list1, dtype='int32')
print(x.dtype) # int32
print(x) # [1, 2, 3]

# 2次元配列の生成
list2 = [[1, 2, 3], [4, 5, 6]]
X = np.array(list2, dtype='float32')
print(X.dtype)
print(X)


# 実行結果
# float32
# [[1 2 3]
#  [4 5 6]]
```

Pythonでは，C言語とは異なり，型宣言などを行うことなく変数を扱えます．しかし自由度が高い分，処理が重くなる問題を抱えています．そこで大量のデータを扱う場合は，C言語の配列に近い，NumPy配列を使うと性能が出せます．本稿ではPython基本数値計算ライブラリNumPyを紹介します．

ここからは，NumPyの配列の基本的な扱い方を紹介していきます．なお，本文中に出てくる「配列」は「NumPyの配列」を表すものとします．

配列の扱い方

● 配列の生成

▶初期値がある場合

1次元配列を生成し，配列の中身や属性を確認するサンプル・コードを**リスト1**に示します．

numpy.array(list,dtype)メソッドにPython標準のリストを渡すことで配列を生成できます．1次元配列を生成する場合は，引き数listに1次元リスト，2次元配列を生成する場合は引き数に2次元リストを渡します．

表1 NumPyで扱えるデータ型

データ型を指定しなくても型推定が行われますが，バグの元になりやすいので，きちっと指定した方が安全

種　類	説　明
bool	論理値型
inti	OS依存の整数(64ビットOSならint型64ビット)
int8	8ビットの整数型
int16	16ビットの整数型
int32	32ビットの整数型
int64	64ビットの整数型
unit8	8ビット符号なし整数型
unit16	16ビット符号なし整数型
unit32	32ビット符号なし整数型
unit64	64ビット符号なし整数型
float16	16ビットの実数型
float32	32ビットの実数型
float64	64ビットの実数型
complex64	64ビットの複素数型
complex128	128ビットの複素数型

配列のデータ型はdtypeで指定します．データ型を指定しなくても型推定が行われますが，バグの元になりやすいので，きちっと指定した方が安全です．NumPyで扱えるデータ型を**表1**に示します．

配列の要素を確認したい場合は，リストと同じようにprint関数を用います．

```
print(ndarray)
```

配列のデータ型を確認した場合は，dtype属性を用います．

リスト2 要素の値を初期化する必要がない場合は，numpy.empty(shape)を使った方が高速に配列を生成できる

```
# 2 × 3の空配列を生成
X = np.empty((2, 3), dtype='float32')
print(X)

# 実行結果
# [[  2.31948455e-310   2.31948455e-310
                                  2.31948683e-310]
#  [  2.31948683e-310   7.06652082e-096
                                  7.05479222e-308]]
```