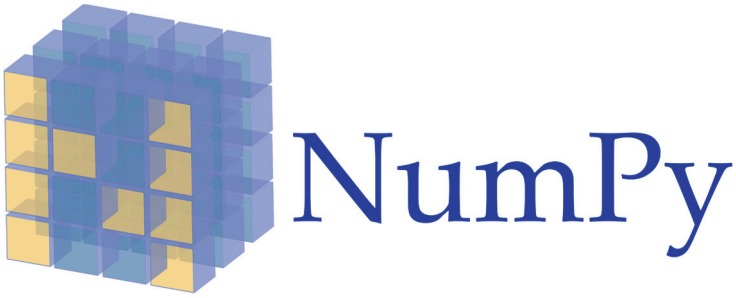
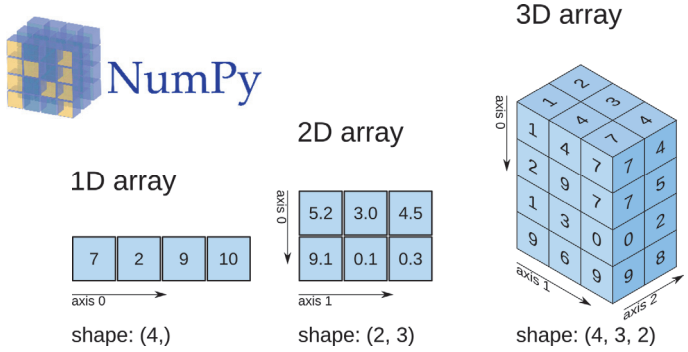


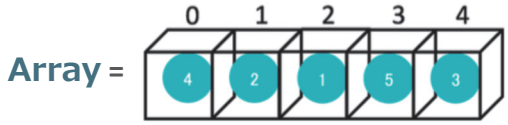
機械学習では、データを「配列」の形で扱うことが非常に多い

Numpyは、**配列データを高速に操作・計算**することが得意



プログラムにおける配列とは

- 配列は**情報を記憶しておくための複数の箱が連なっている**もの
- 全体を「塊」として１つの変数として扱う



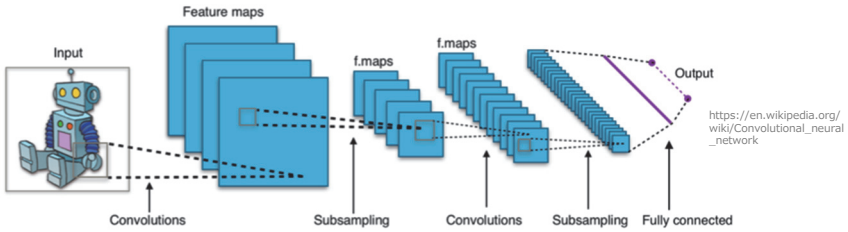
- **同じ意味を持つ複数のデータへの繰り返し処理**（月毎売上、ユーザ毎の属性…）に使うとプログラムが簡略化、労力削減、バグ防止など利点が多い

【例】 24ヶ月に渡る売上の平均を求める場合、

- 配列を使わない場合、変数を24個用意して毎月の売上値を代入し、１個ずつ足し合わせて、さらに24で割る
- 配列を使うと、**array.mean()** の１行だけで済む

※ つまり、配列でデータを表現すると、便利な統計処理演算子が使い放題

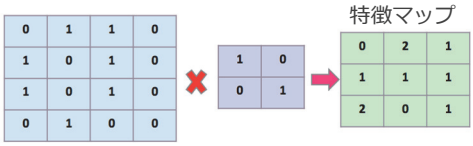
画像データを配列として扱う



- ニューラルネットの学習に必要な大量な**画像データの１枚１枚は２次元配列**
- カラー画像の場合は「色」（RGB）の３次元目も考慮する

CNNの畳み込み層

元画像に対してフィルター（これも2次元配列）を掛けて、行列演算を行うことで凝縮した特徴を取り出す



フィルターと同サイズの部分画像に対する畳み込み演算

Numpyの配列の生成

- Numpyの配列は、”**ndarray**”と呼ばれる
- １次元、２次元、３次元, … と多次元の構造を取れる
- 基本的には数値データを扱う
  ※ 文字列はダミー数値に変換してから機械学習モデルに入力する

■ Numpyは外部パッケージなのでまずコードの中にimport

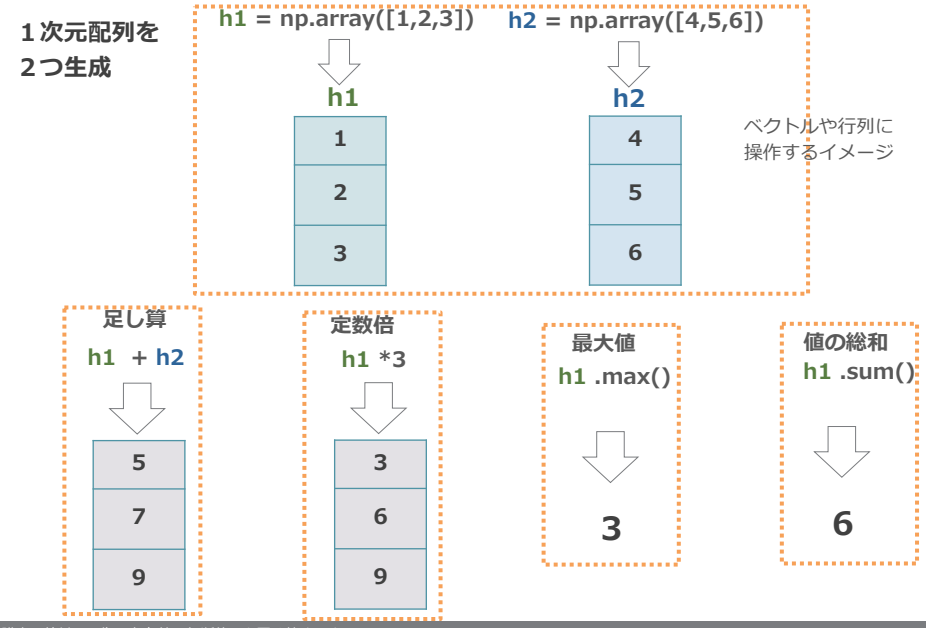
```
import numpy as np
```

■ Numpyで配列を生成する際に **np.array()** メソッドを使用

配列にしたいデータを渡す

```
array = np.array(data)
```


配列データを操作するイメージ



【演習】１次元Numpy配列の生成①

配列にしたいデータを渡す

array = np.array(data)

配列の構造の確認に **shape属性**を利用：**array.shape**

```
#1次元配列を生成
array1 = np.array([0,1,2])
print(array1.shape)
print(array1)
```

np.array() にリストを渡す

```
(3,)
[0 1 2]
```

1d-arrayの場合、shape は (R,)として表される
※ Rは要素の数

【演習】１次元Numpy配列の生成②

連番整数の配列は **np.arange()** を用いて生成出来る

array = np.arrange(連番の個数)

```
#1次元配列を生成
array2 = np.arange(24)
print(array2)
print(array2.shape)
```

np.arange() を用いて、
0～23の整数の１次元配列を生成

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
(24,)
```

24個の要素の１次元配列構造

【演習】１次元配列を２次元配列に整形

- 既存の配列の形状を変更したい場合（学習データの仕様に制限がある時など）、関数 **reshape** を利用:
  整形後配列 ＝ 整形前配列**.reshape**(新しい形)

```
#1次元配列を2次元に整形

array3 = array2.reshape(4,6)

print(array3)
print(array3.shape)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]]
(4, 6)
```

24個要素の1次元配列を (4, 6) の 2次元配列に変形

2次元の場合、shape属性は **(行数,列数)** を表す

【演習】１次元配列から部分要素を取り出す

- 文字列から部分を取り出す時と同じ考え方を適用
  注：インデックスは0から数え始める

```
array1 = np.arange(12)

print(array1)
print(array1[5])  #6番目の要素を取り出す
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
5
```

1d配列の6番目の要素を返す

```
#3番目までの要素を取り出す(スライス法)
array1[0:3]
```

```
array([0, 1, 2])
```

1, 2, 3番目の要素を配列の形で返す