Pythonで学ぶ
実践画像・音声処理入門

博士(工学) 伊藤　克亘
工学博士 花泉　弘 共著
博士(工学) 小泉　悠馬

コロナ社

まえがき

本書の目的は，音声や画像などのティジタルメディアを処理するためのプログラミングの基礎を学ぶことです。ディジタルメディア処理の最先端技術の多くは高度な数学に基づいています。それらの技術を習得する第一歩として本書では，理系の大学 1，2 年生までに学ぶような数学でもさまざまな処理に役立つことを学びます．その際に，なるべく実際の音声データや画像データを処理して実践的な知識を習得することを目指しています。また，コンピュータによる音声や画像の簡単な加工，分析，生成方法を音声処理と画像処理の共通性を意識しながら学ぶことで，データサイエンスの基礎を学ぶことを目的としています。

対象読者

本書は，Python 3 による Python プログラミングの基本知識を備えていることを想定して執筆しました。具体的には，関数，関数の引数，ループを理解していることを想定しています。

数学に関しては，高校数学および大学レベルの微積分，線形代数，統計学の基礎知識があることが必要ですが，必要に応じて，基礎的なレベルの教科書を参照すれば十分でしょう。またフーリエ変換については，どのようなものか知っている方がサンプルプログラムを理解しやすいでしょう。

本書の構成

本書では，章の最初に，その章のプログラムに必要なパッケージを示します。

説明内容に合わせて，数行のプログラムが示されます。これらのプログラムは，章内では，全部続けて実行することを想定しています。つまり，章の最初の方で値を設定された変数が，後のプログラムで断りなしに使われることもあります。

また，章の最初にキーワードを示します。それに関連する数学的な事項を思い出すとよいでしょう。

その章で学んだことを定着させるための章末問題も用意しています。これらの問題は，学んだ知識を自分なりに応用するためのヒントになっています。定着させるためには，最後まで自力で考えた方がよいのですが，巻末にはヒントを掲載しました。

本書で想定する Python 環境

本書のサンプルプログラムは，Mac OS Sierra 上の Anaconda 4.4.0 をベースとした Python 3.5 の環境で動作確認しています。Windows など異なる環境では，出力などが多少異なるのでご留意ください。パッケージのインストール方法などは，サポートサイトに掲載しています。本書で用いた音声や画像データ，本書のために開発したパッケージもサポートサイト（www.coronasha.co.jp/np/isbn/9784339009026/）からダウンロードできます。

本書で用いるパッケージの一覧を次頁に示します。破線の下は `cis.py` だけで用いるもので，サンプルプログラムなどで直接使うことはありません。

科学技術プログラミングの分野では，本書で紹介した NumPy を基盤とする，さまざまなパッケージが作られています。本書が，そのようなパッケージの利用につながるような音声・画像処理の実践的な Python プログラミングを習得する一歩になれば幸いです。

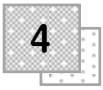
2018 年 2 月

著　　者

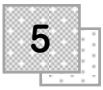
本書で用いるパッケージの一覧

パッケージ名	説　明
numpy	数学的なアルゴリズムを式に似た形式でプログラミングできる関数
numpy.matlib	プログラミング言語・環境の MATLAB に含まれる便利な関数
numpy.linalg	線形代数
scipy.fftpack	高速フーリエ変換（FFT）
scipy.io	ファイル入出力
scipy.signal	信号処理
scipy.ndimage	画像処理
scipy.ndimage.filters	画像処理用フィルタ
scipy.spatial	空間処理
scipy.stats	統計処理
skimage.transform	画像の幾何学的変換
skimage.feature	画像の特徴抽出
skimage.util	画像処理ツール
sklearn.cluster	クラスタリング
sklearn.neighbors	機械学習の最近傍法
matplotlib.pyplot	MATLAB のようなグラフ描画
matplotlib.mlab	MATLAB と同一の名称を持つプロット関連の互換関数
cv2	画像処理ライブラリ OpenCV の利用
cis	本書のために作成
simpleaudio	音声入出力
mpl_toolkits	matplotlib 用ツール
plotly	データ可視化

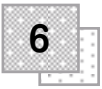
目　　次



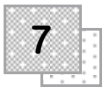
4　フィルタ（音声）



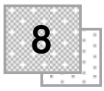
5　画像の周波数領域処理



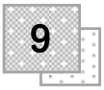
6　画像の空間領域処理



7 音声データの相関



8 画像データの類似度



9 複 素 信 号

10 画像の幾何学的処理

11 分類

12 音声・画像処理の応用

1 簡単な音声処理

まず，1 次元のディジタルデータの代表として音データを取り上げる。この章ではコンピュータに読み込んだ音声データを用いて，本書で対象とする中心パッケージである numpy と matplotlib の基本機能を習得する。

利用するパッケージ

```
import numpy as np
import matplotlib.pyplot as plt
import cis
```

キーワード 振幅，周波数，サンプリング周期，サンプリング周波数，離散的，スカラ，ベクトル，可視化

1.1 波形データの生成

最も単純な音の一つに**純音**がある。純音は**正弦波**で表される。物理の教科書では，純音は式 (1.1) のように表される。

$$y = A\sin(2\pi f t) \tag{1.1}$$

ここで，y は音圧，A は**振幅**，f は**周波数**，t は時間である。時間を横軸にとった y の変化のグラフが**図 1.1** である。

グラフは 2 次元であるが，y という変数が各時刻で一つの値をとり，その値が時間で変化するので 1 次元データと呼ぶ。

式 (1.1) に基づいて，Python で音のディジタルデータを生成するプログラムを作成する。データを生成するためには，まず，式 (1.1) の変数の値を決めなけ

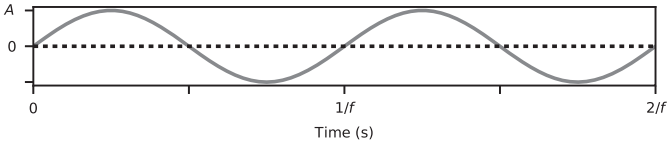


図 1.1　正弦波のグラフ

ればならない。A，f は，それぞれ一つの値を定めればよい。例えば，$A = 1$，$f = 50$ などである。このように大きさのみを持つ量のことを**スカラ**と呼ぶ。

一方，t については，例えば，1 秒間のデータを作成するときには，開始時刻を 0 秒とすると，時刻 0 秒から 1 秒まで変化する。式で書くと $0 \leqq t \leqq 1$ となる。つまり一つの値ではない。

物理的な世界では，t は連続的に変化する。このような連続的な量をコンピュータで扱う最も一般的な方法は，均等で微小な間隔の値の列として表現することである。例えば，1/8 000 秒の間隔とすると，$0 \leqq t \leqq 1$ という範囲は，0，1/8 000，2/8 000，3/8 000，$\cdots$，7 999/8 000，1 という数の列で表される。この 1/8 000 秒を**サンプリング周期**と呼ぶ。また，この周期に対応する周波数を**サンプリング周波数**と呼ぶ。周波数は周期の逆数なので，この場合は 8 000 Hz となる。

このようにとびとびの値で表すことを**離散的**であるという。つまり，コンピュータの中では正弦波は，**図 1.2** の丸印のところだけで表現されている。

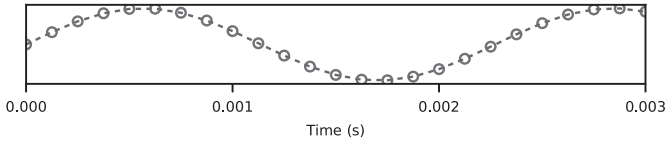


図 1.2　離散的な音声データ

このようなデータをプログラミングするときには，t をひとまとまりで扱うと便利である。そのような場合，数学的には**ベクトル**として扱うことがある。プログラムでは，ベクトルをそのままの形で扱えると便利である。

● **正弦波の生成**　ベクトルをそのまま扱える numpy パッケージを利用して 440 Hz の音を 1 秒間生成してみる。**プログラム 1-1** のように入力する（先頭に数字がある場合は行番号を表す。また，>>>は，Python が表示するプロンプトである。したがってどちらも入力しなくてよい）。

プログラム 1-1（正弦波の生成）

```
1  >>> t=np.arange(0,1,1/8000)
2  >>> a=0.8
3  >>> f=440
4  >>> y=a*np.sin(2*np.pi*f*t)
```

このように，ベクトルを用いた計算を，非常に簡単に，ほとんど数式と同じ形で記述するだけでプログラミングできる。1 行目では，時間を表す数列を作成している。np.arange の np. の部分は，章の冒頭で示した numpy というパッケージであることを示している。つまり，np.arange とは numpy パッケージの arange という関数であることを意味する。arange の使い方を示す。

```
>>> np.arange(0,10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

arange は，このように，数列を作成する関数である。第 1 引数は，数列の開始値であり，第 2 引数の一つ前の値で終了するように指定する。この例の場合，0 から 10 の一つ前の 9 までの整数列が生成され，n に代入される返り値には，データ型が array であることが明示されている。

プログラム 1-1 の arange の例では引数が三つある。この場合，第 3 引数が生成する数列の間隔となる。間隔を指定する例を示す。

```
>>> np.arange(-1,2,0.5)
array([-1. , -0.5,  0. ,  0.5,  1. ,  1.5])
```

指定した通りに 0.5 刻み間隔の数列が生成される。第 2 引数は 2 なので，2 の一つ前の 1.5 が最後の要素となる。

プログラム 1-1 の 2 行目は，振幅を設定している。ここでは，振幅が指定さ

れていないので，適当な値としている。4 行目は，式 (1.1) を NumPy でプログラミングしている。数式とほとんど同じ形で表現できることに注目してほしい。この pi は，NumPy であらかじめ用意されている変数で π の値を持つ。

sin は，三角関数の sin である。NumPy で用意されている関数については，info 関数で説明を見ることができる。

```
>>> np.info(np.sin)
sin(x[, out])

Trigonometric sine, element-wise.

Parameters
----------
x : array_like
(以下略)
```

この説明に x : array_like とあるように，sin 関数は引数に配列（数列）をとれる。引数に数列をとった場合の例を示す。

```
>>> np.set_printoptions(precision=3)
>>> y2=np.sin(np.arange(0,1,0.1))
>>> y2
array([ 0. ,  0.1 ,  0.199,  0.296,  0.389,  0.479,  0.565,  0.644,
        0.717,  0.783])
```

1 行目 set_printoptions は出力を制御する関数である。ここでは，最長で小数点以下 3 桁になるように抑制している。この例では，sin の引数は数列である。その場合，計算結果の y2 も数列となり，その値は引数の sin の値である。このように NumPy の多くの関数は数列や行列を引数にとることができる。この機能を活用すると，NumPy を用いて数式とほぼ同じ形でプログラミングできるようになる。

プログラム 1-1 の 4 行目では，t は数列である。t に掛けられている 2*np.pi*f は，この場合 f は 440 なので，$2 \times 3.14 \times 440 = 2\,763.2$ となりスカラである。NumPy では，ベクトルにスカラが掛けられている場合はベクトルのそれぞれ

の要素をスカラ倍する。したがって，2*np.pi*f*t は，数列 t のおのおのの要素を $2\pi f$ 倍する。すべての要素が $2\pi f$ 倍された数列に対して sin を計算し，その結果の数列を a 倍している。

作成した音データを出力するための関数が audioplay である。プログラム 1-1 で作成した y はつぎのようにして出力する。

```
>>> cis.audioplay(y,8000)
```

第 1 引数が出力したい数列，第 2 引数はサンプリング周波数である。周波数が 440 Hz の「ラ」の音が 1 秒間聞こえるはずである。

1.2　1 次元データの可視化

音を生成したり，加工したり，録音した場合には，もちろん，音を再生，出力して確認すべきである。しかし，音は聞こえ方が人によってかなり異なるし，プログラムに失敗していたら，聞こえる音にならなかったり，デバイスに悪影響を与えるようなデータになることもある。したがって，聞く以外の方法でも確認した方がよい。普通には見ることができないデータを見えるようにすることを**可視化**という。

まず，時間に対する音圧の変化のグラフで確認する方法を取り上げる（**音声波形**，**時間波形**のプロットと呼ばれることが多い）。

```
>>> plt.plot(y)
[<matplotlib.lines.Line2D object at 0x110d515c0>]
>>> plt.show()
```

1 行目の plot は，数列をプロットする関数である。特に指定しなければ，横軸を数列の**インデクス**（番号）とし，縦軸を数列の値として，直線でつないでプロットする。ただし，この段階では，グラフは表示されず，システムの出力が返される（2 行目，at 以降の 16 進数は実行環境により変化するため，この例とは異なった値となる）。3 行目の show によりグラフが表示される（**図 1.3**）。

2 簡単な画像処理

2 次元，もしくは多次元のディジタルデータの代表として画像データを取り上げる。この章では，コンピュータに読み込んだ画像データを用いて，画像データの構造や基本的な扱いを説明する。

利用するパッケージ

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.matlib as mlb
import cv2
import cis
```

キーワード 座標，画素，RGB，バンド

2.1 画 像 の 構 造

〔1〕 **グレイスケール画像**　画像も音と同じようにコンピュータの中では数字の組で表現される。まず，簡単な例として，**グレイスケール画像**（白黒画像）の例を見てみる。

```
>>> x=np.linspace(255,0,12)
>>> x
array([ 255. , 231.81818182, 208.63636364, 185.45454545,
        162.27272727, 139.09090909, 115.90909091, 92.72727273,
        69.54545455, 46.36363636, 23.18181818, 0. ])
>>> x=x.astype(np.uint8)
>>> x
array([255, 231, 208, 185, 162, 139, 115, 92, 69, 46, 23, 0],
```

```
 9  dtype=uint8)
10  >>> x=x.reshape(3,4)
11  >>> x
12  array([[255, 231, 208, 185],
13         [162, 139, 115, 92],
14         [ 69, 46, 23, 0]], dtype=uint8)
15  >>> plt.imshow(x,cmap='gray')
16  <matplotlib.image.AxesImage object at 0x116c1b208>
17  >>> plt.show()
```

1 行目の `linspace` は，開始値と終了値を指定して，その間に指定した個数の等間隔の点からなる配列（ベクトル）を生成する。ここでは，255 から 0 までの 12 点を生成している。6 行目の `astype` は配列の型を変更するメソッドである。ここでは，元の float 型の値を 8 ビットの符号なしの整数型（`uint8`，0〜255 の範囲の整数）に変換している。10 行目の `reshape` は配列の形状を変更して出力するメソッドである。それを用いて配列を 3×4 の 2 次元配列に整形している。11 行目の出力結果からわかるように，2 次元の場合は，行方向に順に並ぶように整形される。15 行目で画像を描画しているが，17 行目を実行するまで表示されない（**図 2.1**）。

この画像は，12 個の**画素**から構成される。実際の画素は非常に小さいので，ディスプレイ上では，実寸で表示すると見えないくらい小さい。しかし，`imshow` は画像を適当に拡大，縮小して描画する関数である。この例の場合は，実際の画

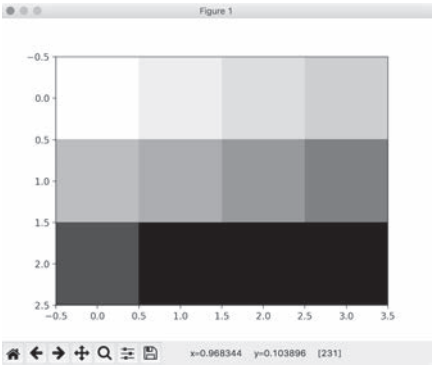


図 2.1　画素の情報表示

素よりはるかに大きく表示される。実際には，左上隅の白い部分は画素一つ分で，その座標は $(0,0)$ であり，その下は $(0,1)$，右は $(1,0)$ となる。`imshow` では，マウス位置の座標とその場所の画素の値がウィンドウ下部のメニューバーの右側に表示される。図 2.1 は $(1,0)$ の部分を表示させたところで，画素の値が 231 と表示されている。この例の画像はグレイスケール画像と呼ばれ，画素の値が大きいほどその画素は明るくなり，小さいほど暗くなる。ここでは，描画するときに，`cmap` 引数を `gray` とすることでグレイスケール画像として描画されている。

画像の大きさは，横方向の画素数を幅（width）と呼び，縦方向の画素数を高さ（height）と呼ぶ。図 2.1 では，幅が 4 で高さが 3 である。

〔**2**〕 **RGB 画 像** **RGB** と呼ばれるカラー画像では，画素は赤，緑，青（red，green，blue，略して RGB）の三つの値の組（ベクトル）で表現される。**プログラム 2-1** に色の指定方法を示す。

プログラム 2-1（さまざまな色の生成）

```
1  bar0=np.zeros((500,100),np.uint8)
2  bar1=255*np.ones((500,100),np.uint8)
3  Col=np.zeros((500,800,3),np.uint8)
4  Col[:,:,0]=mlb.repmat(np.hstack((mlb.repmat(bar1,1,2),
5  mlb.repmat(bar0,1,2))),1,2)
6  Col[:,:,1]=np.hstack((mlb.repmat(bar1,1,4),mlb.repmat(bar0,1,4)))
7  Col[:,:,2]=mlb.repmat(np.hstack((bar1,bar0)),1,4)
8  plt.imshow(Col)
9  plt.show()
```

左から白，黄，シアン，緑，マゼンタ，赤，青，黒の帯が生成される。NumPy では，`Col[:,:,1]` のように，三つの添え字を持つ配列で 3 次元配列を表す。この例の場合，`Col` は，500×800 の 2 次元配列三つから構成される。それぞれの 2 次元配列は，プレーン，**バンド**などと呼ばれる。

NumPy では，この例のように配列の要素を指定する部分に「`:`」だけを書いた場合は，開始値 0 と終了値が最後の要素までという指定を省略したことになるのですべての要素を指定したことになる。RGB 画像では，3 次元配列の最初

のバンドが赤，つぎが緑，最後が青に対応する。そのことは，表示された画像の赤の部分でマウスボタンを押しながら少し動かしてみて，画素の値を表示させると [255,0,0] と表示されることでわかる。

1 行目の zeros，2 行目の ones は配列を 0 や 1 で初期化するために使われる。第 1 引数は，作成したい配列の大きさをタプルで指定する。(500,100) は 2 次元配列で第 1 次元が 500 個，第 2 次元が 100 個，行列としては 500×100 であることを示す。第 2 引数は配列の（数）値の型を指定する。ここでは符号なし 8 ビット整数である。ones，zeros の使い方をつぎに示す。

```
>>> np.ones((2))
array([ 1., 1.])
>>> np.zeros((2,2),np.uint8)
array([[0, 0],
       [0, 0]], dtype=uint8)
```

プログラム 2-1 の 4 行目の repmat は，行列を繰り返して大きな行列を作成する関数である。

```
 1  >>> x1=np.array([[1],[2]])
 2  >>> x1
 3  array([[1],
 4         [2]])
 5  >>> x1.shape
 6  (2, 1)
 7  >>> mlb.repmat(x1,1,2)
 8  array([[1, 1],
 9         [2, 2]])
10  >>> mlb.repmat(x1,2,1)
11  array([[1],
12         [2],
13         [1],
14         [2]])
15  >>> mlb.repmat(x1,2,2)
16  array([[1, 1],
17         [2, 2],
18         [1, 1],
19         [2, 2]])
```

上記 1 行目の array は numpy の多次元配列を作成する関数である。ここで作成している x1 は 2 行 1 列の行列である。repmat は行列を第 1 引数に指定し，その行列を行方向（縦）に第 2 引数の回数，列方向（横）に第 3 引数の回数だけ繰り返した行列を作成する。7 行目では列ベクトルの x1 を行方向に 1 回，列方向に 2 回，つまり列方向に 2 回繰り返して 2×2 の行列を作成している。

プログラム 2-1 の 6 行目の hstack は，ベクトルだけでなく行列も横に並べることができる。つぎの例の 6 行目が行列を並べる例である。

```
1  >>> x2=np.array([[3],[4]])
2  >>> x12=np.hstack((x1,x2))
3  >>> x12
4  array([[1, 3],
5         [2, 4]])
6  >>> np.hstack((x12,x12))
7  array([[1, 3, 1, 3],
8         [2, 4, 2, 4]])
```

2.2 画像・ビデオの読み込み

〔1〕 **画像の読み込み** Python では，つぎのように OpenCV の関数を用いて画像を読み込める。

```
1  >>> I=cv2.imread('paprika-966290_640.jpg')
2  >>> plt.imshow(cv2.cvtColor(I, cv2.COLOR_BGR2RGB))
3  <matplotlib.image.AxesImage object at 0x1141b3438>
4  >>> plt.show()
```

OpenCV の関数でカラー画像を読み込むと，プログラム 2-1 と違って，バンドが青，緑，赤の順で並べられる。このような画像を BGR 画像と呼ぶ。2 行目では，cvtColor 関数で BGR から RGB に変換している。

この画像の大きさは，shape メソッドで調べられる。

```
1  >>> sz=I.shape
```

索　　引

—— 著 者 略 歴 ——

伊藤　克亘（いとう　かつのぶ）

1993 年　東京工業大学大学院理工学研究科博士課程修了（情報工学専攻）
　　博士（工学）
1993 年　電子技術総合研究所研究員
2003 年　名古屋大学大学院助教授
2006 年　法政大学教授
　　現在に至る

花泉　弘（はないずみ　ひろし）

1981 年　東京大学大学院工学系研究科博士課程中退（計数工学専攻）
1981 年　東京大学助手
1987 年　工学博士（東京大学）
1987 年　法政大学専任講師
1989 年　法政大学助教授
1996 年　法政大学教授
　　現在に至る

小泉　悠馬（こいずみ　ゆうま）

2014 年　法政大学大学院情報科学研究科博士前期課程修了（情報科学専攻）
2014 年　日本電信電話株式会社 NTT メディアインテリジェンス研究所研究員
　　現在に至る
2017 年　電気通信大学大学院情報理工学研究科博士後期課程修了（情報学専攻）
　　博士（工学）

Python で学ぶ実践画像・音声処理入門

Introduction to Media Computing in Python : A Practical Approach

2018 年 4 月 27 日　初版第 1 刷発行　★

検印省略

著　者　伊藤克亘
　　花泉弘
　　小泉悠馬
発行者　株式会社　コロナ社
　　代表者　牛来真也
印刷所　三美印刷株式会社
製本所　有限会社　愛千製本所

112–0011　東京都文京区千石 4–46–10
発行所　株式会社　コロナ社
CORONA PUBLISHING CO., LTD.
Tokyo Japan
振替 00140–8–14844・電話 (03) 3941–3131 (代)
ホームページ　http://www.coronasha.co.jp

ISBN 978–4–339–00902–6　C3055　Printed in Japan　(三上)