Appendix 2 これだけでOK！

Python計算ライブラリ NumPyクイック・リファレンス

西住 流

NumPyでよく使う機能を**表1**にまとめます．基本的な配列操作，統計処理，行列計算，ファイル処理を行う際に便利な機能を中心に，筆者が「これだけあればたいていの処理は記述できる」と考えるメソッドをピックアップしています．

またNumPyの配列には，array型とmatrix型の2種類あります．matrix型はあまり使われていないため，特記がない場合は，NumPyの配列はarray型として説明しています．

NumPyには非常に多くの機能があります．所望の処理が表にない場合は，NumPyの公式ドキュメント（http://www.numpy.org/）を参照してみてください．各機能（メソッドや属性）の解説とサンプル・コードが充実しているため理解しやすい内容となっています．

にしずみ・りゅう

表1　NumPyクイック・リファレンス

機　能	記述例	説　明
加算	`z = x + y`	配列x，yの要素同士を加算する
減算	`z = x - y`	配列x，yの要素同士を減算する
乗算	`z = x * y`	配列x，yの要素同士を乗算する（matrix型の行列の場合は内積を求める）
除算	`z = x / y`	配列x，yの要素同士を除算する
剰余	`z = x % y`	配列x，yの要素同士を除算したときの剰余を求める
冪乗	`y = x ** a`	配列xの要素をa乗する
符号反転	`y = -x`	配列xの要素の符号を反転する

（a）算術演算子

機　能	記述例	説　明
1次元配列	`x = numpy.array([1, 2, 3])`	1次元配列xを生成する（array型）
2次元配列	`X = numpy.array([[1,2,3],` `[4,5,6]])`	2次元配列Xを生成する（array型）
1次元行列	`x = numpy.matrix([1, 2, 3])`	1次元行列xを生成する（matrix型）
2次元行列	`X = numpy.matrix([[1,2,3],` `[4,5,6]])`	2次元行列Xを生成する（matrix型）
空配列	`X = numpy.empty([m, n])`	サイズがm×nで空の配列Xを生成する．メモリ領域の確保のみ行い，要素の初期化はしないので高速に配列を生成する
全要素が1①	`X = numpy.ones((m, n))`	サイズがm×nで全ての要素が1の配列Xを生成する
全要素が1②	`X = numpy.ones_like(B)`	配列Bと同じサイズで全ての要素が1の配列Xを生成する
零行列①	`X = numpy.zeros((m, n))`	サイズがm×nで全ての要素が0の配列Xを生成する
零行列②	`Y = numpy.zeros_like(X)`	配列Xと同じサイズで全ての要素が0の配列Yを生成する
全要素が1つの値	`X = numpy.full([m, n], val)`	サイズがm×nで全ての要素がvalの配列Xを生成する
単位行列	`X = numpy.eye(m)`	サイズがm×mの配列X（単位行列）を生成する
三角行列	`X = numpy.tri(m)`	サイズがm×mの配列X（三角行列）を生成する
繰り返し①	`x = numpy.repeat([1, 2, 3], n)`	[1, 2, 3]をn回繰り返した配列xを生成する
繰り返し②	`x = numpy.tile([1, 2, 3], n)`	[1, 2, 3]をn回繰り返した配列xを生成する
範囲・間隔を指定	`x = numpy.arange(a0, an, d)`	a0以上，an未満で，間隔dのデータを要素に持つ配列xを生成する
範囲・要素数を指定	`x = numpy.linspace(a0, an, n)`	a0以上，an未満で，要素数nの配列xを生成する
対数スケール	`x = numpy.logspace(a0, an , n, base=b)`	a0以上，an未満で，要素数nで対数スケール（底はb）で並べられた配列xを生成する
格子状配列	`X = numpy.meshgrid(x, y)`	縦横に等間隔な格子状配列Xを生成する

（b）配列の生成

表1　NumPyクイック・リファレンス（つづき）

機　能	記述例	説　明
一様乱数（整数）①	x = numpy.random.randint(a0, an, n)	a0以上，an未満で，サンプル数nの一様乱数配列xを生成する（値は整数）
一様乱数（整数）②	X = numpy.random.randint(a0, an, (m,n))	a0以上，an未満で，サンプル数n，サイズがm×nの一様乱数配列Xを生成する
一様乱数（実数）①	x = numpy.random.rand(n)	0.0以上，1.0未満で，サンプル数nの一様乱数配列xを生成する（値は実数）
一様乱数（実数）②	X = numpy.random.rand(n)	0.0以上，1.0未満で，サンプル数n，サイズがm×nの一様乱数配列xを生成する
正規分布	x = numpy.random.normal(mu, sigma, n)	平均値mu，標準偏差sigma，サンプル数nの正規分布に従う乱数配列xを生成する
標準正規分布	x = nnmpy.random.randn(n)	サンプル数nの正規分布（平均値0，標準偏差1）に従う乱数配列xを生成する
二項分布	x = numpy.random.binomial(t, p, n)	試行回数t，確率p，サンプル数nの2項分布に従う乱数配列xを生成する
ポアソン分布	x = numpy.random.binomial(lam, n)	試行回数lam，サンプル数nのポアソン分布に従う乱数配列xを生成する
ベータ分布	x = numpy.random.beta(a,b,n)	パラメータa, b，サンプル数nのベータ分布に従う乱数配列xを生成する
χ分布	x = numpy.random.chisquare(df, n)	自由度df，サンプル数nのカイ2乗分布に従う乱数配列xを生成する
ガンマ分布	x =numpy.random.gamma(p,sigma,10)	確率p，標準偏差sigma，サンプル数nのガンマ分布に従う乱数配列xを生成する
F分布	x = numpy.random.f(num, den, n)	分子の自由度num，分母の自由度den，サンプル数nのF分布に従う乱数配列xを生成する

（c）乱数配列の生成

機　能	記述例	説　明
先頭の要素へ代入	x[0] = value	1次元配列xにおける先頭の要素に値を代入する
*i*番目の要素へ代入	x[i] = value	1次元配列xにおけるi番目の要素に値を代入する（先頭は0番目）
末尾の要素へ代入	x[-1] = value	1次元配列xにおける末尾の要素に値を代入する
i～*j*－1番目の要素へ代入	x[i:j] = value	1次元配列xにおけるi～j－1番目の要素に値を代入する
*j*行目，*i*列目の要素へ代入①	X[j][i] = value	2次元配列Xにおけるj行目，i列目にある要素に値を代入する（0番目から開始）
*j*行目，*i*列目の要素へ代入②	X[j, i] = value	2次元配列Xにおけるj行目，i列目にある要素に値を代入する（上の省略形）
指定領域の要素へ代入	X[j:j+h, i:i+w] = value	2次元配列Xにおけるjからj＋h行目，i～i＋w列目にある要素に値を代入する
配列のコピー	y = x.copy()	配列xをコピーした新たな配列yを生成する
末尾に要素追加	z = numpy.append(x, y)	配列xの末尾に配列yの要素を追加した新しい配列zを生成する
次元数の変更	X = x.reshape(m, n)	1次元配列xを2次元配列X（サイズはm×n）に変換する
データ型の変更	y = x.astype(dtype)	配列xのデータ型をdtypeに変換した新しい配列yを生成する
書き換え禁止	x.flags.writeable = False	配列xの要素を書き換え禁止にする
配列の結合（縦）	Z = numpy.vstack([X, Y])	配列X，Yを縦方向に結合する
配列の結合（横）	Z = numpy.hstack([X, Y])	配列X，Yを横方向に結合する
配列の分割（縦）	Y = numpy.vsplit(X, n)	配列Xを縦方向にn個に分割する
配列の分割（横）	Y = numpy.hsplit(X, n)	配列Xを横方向にn個に分割する
配列の比較（完全一致）	z = numpy.allclose(x, y)	配列x，yの要素が完全に一致するかどうか調べる
インデックスの移動	y = numpy.roll(x, n)	配列xの要素を左右にn個分だけ移動する

（d）配列操作

機　能	記述例	説　明
次元数	`dim = X.ndim`	配列Xの次元数を取得する
要素数①	`shape = X.shape`	配列Xの各次元の要素数を取得する
要素数②	`size = A.size`	配列Aの要素数を取得する
バイト数①	`byteX = X.nbytes`	配列Xのバイト数を取得する
バイト数②	`byte = X.itemsize`	配列Xの1要素当たりのバイト数を取得する
データ型	`type = X.dtype`	配列Xのデータ型を取得する
先頭の要素	`a = x[0]`	1次元配列xにおけ先頭の要素を取得する
*i*番目の要素	`a = x[i]`	1次元配列xにおけるi番目の要素を取得する（先頭は0番目）
末尾の要素	`a = x[-1]`	1次元配列xにおける末尾の要素を取得する
i～*j*－1番目の要素	`a = x[i:j]`	1次元配列xにおけるi～j－1番目の要素を取得する
*j*行目，*i*列目の要素①	`a = X[j][i]`	2次元配列Xにおけるj行目，i列目にある要素を取得する（0番目から開始）
*j*行目，*i*列目の要素②	`a = X[j, i]`	2次元配列Xにおけるj行目，i列目にある要素を取得する
指定領域の要素	`a = X[j:j+h, i:i+w]`	2次元配列Xにおけるjからj+h行目，i～i＋w列目にある要素を参照する
行の要素	`a = X[j, :]`	2次元配列Xにおけるj行目にある全ての要素を取得する
列の要素	`a = X[:, i]`	2次元配列Xにおけるi列目にある全ての要素を取得する
複数の条件を満たす要素	`a = numpy.select(condlist, choicelist)`	複数の条件を満たす要素を取得する
行列の対角成分	`a = numpy.diag(X)`	２次元配列Xから対角成分の要素を取得する
条件を満たす要素	`a = x[ numpy.where(condition) ]`	配列xから条件を満たす要素を取得する
ランダム抽出①	`a = numpy.random.choice( x, n, replace=True)`	配列xからn個の要素をランダムに取得する（replaceがTrueなら重複あり）
ランダム抽出②	`a = numpy.random.choice( x, n, p= [0.1, 0.9])`	配列xから出現確率pに基づいてn個の要素をランダムに取得する
シャッフル	`a = numpy.random.shuffle(x)`	配列xの要素の順序をシャッフルしてから取得する
要素のインデックス	`index = numpy.where(condition)`	配列から条件を満たす要素のインデックスを取得する
最小値要素のインデックス	`index = numpy.argmin(x)`	配列xから最小値要素のインデックスを取得する
最大値要素のインデックス	`index = numpy.argmax(x)`	配列xから最大値要素のインデックスを取得する
非0要素のインデックス	`index = numpy.nonzero(x)`	配列xから0以外の値を持つ要素のインデックスを取得する
非0要素数	`n = numpy.count_nonzero(x)`	配列xから0以外の値を持つ要素数をカウントする
条件を満たす要素数	`n = len( numpy.where(condition)[0])`	条件式を満たす要素数をカウントする

（e）データ抽出

機　能	記述例	説　明
合計	`total = numpy.sum(x)`	配列xの要素の総和を求める
平均	`ave = numpy.average(x)`	配列xの要素の平均を求める
分散	`var = numpy.var(x)`	配列xの要素の分散を求める
標準偏差	`std = numpy.std(x)`	配列xの要素の標準偏差を求める
不偏分散	`dvar = numpy.var(x, ddof=1)`	配列xの要素の不偏分散を求める
不偏標準偏差	`dstd = numpy.std(x, ddof=1)`	配列xの要素の不偏標準偏差を求める
偏差値	`y = numpy.round_( 50+10*(x-numpy.average(x))/numpy.std(x))`	配列xの要素の偏差値を求める
最大値	`max = numpy.amax(x)`	配列xの要素の最大値を求める
最小値	`min = numpy.amin(x)`	配列xの要素の最小値を求める
中央値	`median = numpy.median(x)`	配列xの要素の中央値を求める
算術平均	`mean = numpy.mean(x)`	配列xの要素の算術平均を求める
差分	`dx = numpy.diff(x)`	配列xの要素の差分を求める
勾配	`lx = numpy.gradient(x)`	配列xの要素の勾配（傾き）を求める
相関係数	`alpha = numpy.corrcoef(x, y)[0, 1]`	配列x，yの相関係数を求める
ヒストグラム	`hist = numpy.histogram(x)`	配列xのヒストグラムを求める

（f）統計量

表1　NumPyクイック・リファレンス（つづき）

機　能	記述例	説　明
正弦関数	y = numpy.sin(x)	正弦関数sin(x)を求める（xはラジアン）
余弦関数	y = numpy.cos(x)	余弦関数cos(x)を求める
正接関数	y = numpy.tan(x)	正接関数tan(x)を求める
逆正弦関数	y = numpy.arcsin(x)	逆正弦関数arcsin(x)を求める
逆余弦関数	y = numpy.arccos(x)	逆余弦関数arccos(x)を求める
逆正接関数	y = numpy.arctan(x)	逆正接関数arctan(x)を求める
ラジアン→度	deg = numpy.rad2deg(rad)	ラジアン(rad)を度(deg)に変換する
度→ラジアン	rad = numpy.deg2rad(rad)	度(deg)をラジアン(rad)に変換する
平方根	y = numpy.sqrt(x)	配列xの平方根を求める
立方根	y = numpy.cbrt(x)	配列xの立方根(3乗根)を求める
2乗	y = numpy.square(x)	配列xの2乗を求める
絶対値	numpy.absolute(x)	配列xの絶対値を求める
絶対値(省略形)	numpy.abs(x)	配列xの絶対値を求める
符号関数	numpy.sign(x)	配列xの符号を求める
指数関数	y=numpy.exp(x)	指数関数e^xを求める
対数関数(底e)	y=numpy.log(x)	底がeの対数関数を求める
対数関数(底10)	y=numpy.log10(x)	底が10の対数関数を求める
対数関数(底2)	y=numpy.log2(x)	底が2の対数関数を求める
円周率	pi = numpy.pi	円周率πを呼び出す
ネイピア数e	e = numpy.e	ネイピア数e(自然対数の底)を呼び出す
複素数の初期化	x = np.array([1+1j, 1+2j, 1+3j])	複素数を要素に持つ配列xを生成する
実部	numpy.real(x)	配列xから実部のみを取り出す
虚部	numpy.imag(x)	配列xから虚部を取り出す
複素共役	numpy.conj(x)	配列xから複素共役を求める
行方向の和	y = numpy.sum(X, axis=1)	2次元配列Xにおける要素の行方向の和を求める
列方向の和	y = numpy.sum(X, axis=0)	2次元配列Xにおける要素の列方向の和を求める
行方向の差	y = numpy.diff(X, axis=1)	2次元配列Xにおける要素の行方向の差分を求める
列方向の差	y = numpy.diff(X, axis=0)	2次元配列Xにおける要素の列方向の差分を求める
行方向の積	y = numpy.prod(X, axis=1)	2次元配列Xにおける要素の行方向の積を求める
列方向の積	y = numpy.prod(X, axis=0)	2次元配列Xにおける要素の列方向の積を求める

（g）数学関数

機　能	記述例	説　明
単回帰分析	A = numpy.array([x,numpy.ones(len(x))]) a, b = numpy.linalg.lstsq(A.T, y)[0]	配列x(説明変数)と配列y(目的変数)から最小2乗法により回帰直線の傾きaと切片bを求める
重回帰分析	e = numpy.array([x1, x2]) e = numpy.vstack([numpy.ones(e.shape[1]), e]) b, a1, a2 = numpy.linalg.lstsq(e.T, o)[0]	配列x1，x2(説明変数)と配列y(目的変数)から回帰曲線の傾きa1，a2と切片bを求める
直線近似	a, b = numpy.polyfit(x, y, 1)	配列x(説明変数)と配列y(目的変数)から近似直線の傾きaと切片bを求める
曲線近似	a1, a2, b = numpy.polyfit(x, y, 2)	配列x(説明変数)と配列y(目的変数)から2次の近似曲線の係数a1，a2と切片bを求める

（h）回帰分析

機　能	記述例	説　明
加算	z = x + y	2つのベクトルx，yの加算を求める
減算	z = x - y	2つのベクトルx，yの減算を求める
内積	z = x.dot(y) もしくは z = numpy.dot(x, y)	2つのベクトルx，yの内積を求める
外積	z = numpy.cross(x, y)	2つのベクトルx，yの外積を求める
ノルム	norm = numpy.linalg.norm(x)	ベクトルxのノルムを求める
正規化	e = x / numpy.linalg.norm(x)	ベクトルxの単位ベクトルeを求める(正規化)

（i）ベクトル計算

機　能	記述例	説　明
加減算	`Z = X + Y`	2つの行列X，Yの加算を求める
内積	`Z = X.dot(Y)` もしくは `Z = numpy.dot(X, Y)`	2つの行列X，Yの内積を求める．matrix型の場合は算術演算子＊だけで内積を計算できる
外積	`Z = numpy.cross(X, Y)`	2つの行列X，Yの外積を求める
階数（ランク）	`rank = numpy.linalg.matrix_rank(X)`	行列Xの階数（ランク）を求める
行列式	`det = numpy.linalg.det(X)`	行列Xから行列式detを求める
転置行列	`Xt = X.T`	行列Xの転置行列Xtを求める
固有値・ 固有ベクトル	`lam, vec = numpy.linalg.eig(X)`	行列Xの固有値lamと固有ベクトルvecを求める
逆行列	`invX = numpy.linalg.inv(X)`	行列Xの逆行列invXを求める
擬似逆行列	`pinvX = numpy.linalg.pinv(X)`	行列Xの疑似逆行列pinvXを求める
特異値分解 （SVD）	`U, S, V = numpy.linalg.svd(X)`	行列Xを直交行列U，Vと行列Sに分解する
QR分解	`Q, R = numpy.linalg.qr(X)`	行列Xを直交行列Qと上三角行列Rに分解する
コレスキー分解	`L = numpy.linalg.cholesky(X)`	正定値対称行列Xを分解し，下三角行列Lを求める

（j）行列計算

機　能	記述例	説　明
1次元FFT	`f = numpy.fft.fft(x)`	1次元の高速フーリエ変換により，1次元配列xのデータを時間領域から周波数領域に変換する
2次元FFT	`F = numpy.fft.fft2(X)`	2次元の高速フーリエ変換により，2次元配列Xのデータを時間領域から周波数領域に変換する
零周波数成分の移動	`Fs = numpy.fft.fftshift(F)`	1次元配列の場合は，配列要素の右半分と左半分を入れ替える．2次元配列の場合は，第1象限と第3象限，第2象限と第4象限を入れ替える．これにより高速フーリエ変換された配列の中心に低周波数成分が集まる
1次元IFFT	`x = numpy.fft.ifft(f)`	1次元の高速逆フーリエ変換により，配列fのデータを周波数領域から時間領域に変換する
2次元IFFT	`X = numpy.fft.ifft2(F)`	2次元の高速逆フーリエ変換により，配列Fのデータを周波数領域から時間領域に変換する
周波数軸の生成	`freq = numpy.fft.fftfreq(N, d)`	サンプル数N，サンプリング間隔dの周波数軸の値を持つ配列を求める

（k）信号処理

機　能	記述例	説　明

読み込み①	`X = numpy.loadtxt(filename,`
	`delimiter,`
	`skiprows,`
	`dtype)`
		テキスト形式のファイルを読み込んで，データをNumPy型配列Xに格納する（filename：ファイル名，delimiter：区切り文字，skiprows：読み飛ばすヘッダ行の行数，dtype：データ型）

読み込み②	`X = numpy.genfromtxt(filename,`
	`delimiter,`
	`skip_header,`
	`dtype)`
		テキスト形式のファイルを読み込んで，データをNumPy型配列Xに格納する（filename：ファイル名，delimiter：区切り文字，skiprows：読み飛ばすヘッダ行の行数，dtype：データ型）

書き込み①	`numpy.savetxt(filename, x,` `delimiter)`	NumPy型配列xのデータをテキスト形式でファイルに書き込む（filename：ファイル名，delimiter：区切り文字）
書き込み②	`numpy.savez(filename, a, b, …)`	複数の配列a，b，…のデータをバイナリ形式（非圧縮）で書き込む
書き込み③	`numpy.savez_compressed(filename,` `a, b, …)`	複数の配列a，b，…のデータをバイナリ形式（圧縮）で書き込む

（l）ファイル処理