Formulaire (non exhaustif) de Python

PIERRÉ Jean-Emmanuel

2015-2016

1 Python

Aide	dir(truc), help(truc), truc.__doc__
Types non mutables	**écriture littérale**
NoneType	None
int(), long(), float(), complex()	12, 12L, 12.0, 1+2j
bool()	True, False
str()	"toto", 'toto'
unicode()	u"ça", u'élève'
Types mutables	**écriture littérale**
tuple()	(1, 2, "3", 4.0)
list()	[1, 2, "3", 4.0]
set()	{1, 2, "3", 4.0}
dict()	{'cle' :'valeur', 'a' :1, 'b' :2.0}
Notations	**Exemples**
affectation	truc="contenu" a,b,c=(1,2,3) truc+=2
indiçage	truc['cle'] truc[3]
segmentation	truc[3 :6] # de 3 inclus à 6 exclus truc[3 :6 :2] # de 3 inclus à 6 exclus avec pas de 2
attribut d'objet	truc.attribut
appel	truc(), truc(a,b,c=2,d='z')
Structures de contrôles	**Exemples**

condition	if expression :
	  quelque chose
	elif expression :
	  autre chose
	else :
	  ou alors ça

boucle tant que	while expression : quelque chose
boucle pour	for truc in trucs : quelque chose

gestion d'erreur	try :
	  quelque chose
	except :
	  autre chose
	finally :
	  truc final

gestion de contexte	with expr as truc : quelque chose
Mots clés	print, del, from, import, raise, return, break, continue, assert, pass
Primitives	abs(), max(), min(), sum(), open(), file(), map(), reduce(), filter(), range(), type(), enumerate(), help(), getattr(), hasattr(), delattr(), exit(), id(), dir(), xrange(), all(), append()
Opérateurs	unaires : not, -, + binaires : -, +, *, /, //, %, **, and, or, in, ==, <, >, <=, >=,!= ternaire : a if b else c
Fonctions	def ma_fonction() : return expression

Classe	class MaClasse(objet) :
	  un_attribut=12
	  def __init__(self) :
	  print "constructeur"
	  def une_methode(self) :
	  print "truc"

2 Numpy

Librairie d'analyse numérique de Python.

import numpy as np	import de la librairie que l'on rennomme "np" pour plus de simplicité d'écriture
np.pi, np.exp(), np.abs()...	valeurs et opérateurs mathématiques vectoriels sur les objets numpy. Par exemple, pour un vecteur x, np.sin(x) renvoi le vecteur des sinus des composantes
np.linspace(a,b,n)	création d'un vecteur allant de a (inclus) à b (inclus) avec n valeurs linéairement espacées
np.arange(a,b,h)	création d'un vecteur allant de a (inclus) à b (exclus) avec un pas de h
x=np.array([1,2,3])	création d'un vecteur
A=np.array([[1,2,3],[4,5,6]])	création d'une matrice (ici 2 lignes et 3 colonnes)
A.T ou np.transpose(A)	transposée de A
np.shape(A)	liste des dimensions de A (ici :(2,3))
np.size(A,0)	nombre de lignes de A (ici 2)
np.size(A,1)	nombre de colonnes de A (ici 3)
np.size(A)	nombres d'éléments au total (attention, ici c'est 6 et non (2,3))
np.zeros(n)	vecteur nul contenant n fois 0.
np.zeros((n,p))	matrice de 0. avec n lignes et p colonnes
np.ones(n)	vecteur contenant n fois 1.
np.ones((n,p))	matrice de 1. avec n lignes et p colonnes
np.eye(n)	matrice identité de taille n
np.diag(A)	renvoi la diagonale de la matrice A
np.diag(A,p)	renvoi la $p^{\text{ième}}$ diagonale de la matrice A. Par exemple np.diag(A,1) renvoi la surdiagonale de A tandis de np.diag(A,0)=np.diag(A) renvoi la diagonale
np.diag(x)	crée une matrice dont x est la diagonale
np.diag(x,p)	crée une matrice dont x est la $p^{\text{ième}}$ diagonale
np.dot(A,x)	produit de A et x. Attention, numpy agissant de façon vectorielle, A*x est le produit composante par composante et non le produit "mathématique" entre deux matrices ou vecteurs
np.dot(x,y)=np.inner(x,y)	produit scalaire entre x et y (renvoi le scalaire $x^t y$)
np.outer(x,y)	produit entre deux vecteurs x et y (renvoi la matrice xy^t)
np.reshape(A,(n,p))	redimenssionne A avec n lignes et p colonnes (n×p doit correspondre au nombre d'éléments de A)
np.tile(A,(n,p))	répète A n fois en lignes et p fois en colonnes
z=np.hstack((x,y)) =np.append(x,y,axis=1)	concaténation horizontale de 2 valeurs, vecteurs ou matrices x et y. Par exemple : x=np.hstack((0.1,np.zeros(n-1))) crée un vecteur de taille n avec seulement des 0. sauf la première valeur qui vaut 0.1. Mathématiquement, ce vecteur est noté $x=0.1\delta_{1i}$ Remarque : il est aussi possible de faire : x=np.zeros(n) puis de modifier la première valeur avec x[0]=0.1
z=np.vstack((x,y)) =np.append(x,y,axis=0)	idem en vertical
a=A[np.ix_(u,v)]	a=sous matrice de A selon les listes u et v. Exemple : a=A[np.ix_([0,2],[3,2])]
import numpy.linalg as npl	import de la librairie d'algèbre linéaire de numpy que l'on rennomme "npl" pour plus de simplicité d'écriture
d=npl.det(A)	d=déterminant de A
valp,vecp=npl.eig(A)	valp et vecp sont respectivement les valeurs et vecteurs propres de A
valp=npl.eigvals(A)	valp contient seulement les valeurs propres de A
n=npl.norm(A,p)	n=norme p de A
c=npl.cond(A,p)	c=conditionnement de la matrice A selon la norme p
x=npl.solve(A,b)	x=solution du système linéaire Ax=b

3 Matplotlib.pyplot

Librairie Python pour dessiner des courbes.

import matplotlib.pyplot as plt	import de la librairie que l'on rennomme "plt" pour plus de simplicité d'écriture
plt.clf()	fermeture des courbes existantes
plt.plot(x,y,label='y=f(x)','r+-')	plot de y en fonction de x (x et y sont donc de la même taille). Le label est la légende qui s'affiche si plt.legend() est appelée. 'r+-' correspond au type (ici 'r'=rouge' et '+-'=les points sont des croix et sont reliés les uns aux autres). D'autres exemples de couleurs : r,b,c,m,y,k,g. D'autres exemples de types : +,-,+-,+.-,+–,s,d,o
plt.title('titre')	affichage d'une titre
plt.xlabel('axe x')	affichage d'une légende pour l'axe x
plt.ylabel('axe y')	affichage d'une légende pour l'axe x
plt.legend(loc='upper left')	affichage des légendes associées aux courbes
plt.show()	affichage de la courbe (sans le "show", le plot est créé mais ne s'affiche pas)

4 Scipy

Librairie de calcul scientifique de Python.

import scipy.linalg as spl	import de la librairie d'algèbre linéaire de scipy que l'on rennomme "spl" pour plus de simplicité d'écriture
P,L,U=spl.lu(A)	décomposition A=PLU avec la méthode LU (P étant une matrice de permutation)
L,U=spl.lu(A,True)	décomposition A=$\tilde{L}$U avec la méthode LU (attention, la matrice de permutation existe bien, elle est incluse dans $\tilde{L}$=PL)