Une introduction au langage python

12 septembre 2016

python

Sommaire

Généralités

python, c'est quoi ?

- √ Python est un langage de programmation.
- √ Un programme – on dira un script – écrit en python est interprété : nécessite un interpréteur python.
- √ Un programme (script) python est un fichier texte.
- √ Multiplateforme (Windows, MAC-OS, linux)
- √ Logiciel libre : gratuit, modifiable et re-distribuable (disponible sous licence open source).
- √ Large communauté de développeurs

Généralités

Historique

- Créé en 1991 par Guido van Rossum (Néerlandais). van Rossum est le BDFL (dictateur bénévole à vie !) de la communauté.
- 1993 : sortie de Numerical Python, ancêtre de numpy (calcul numérique, matrices).
- 2016 : versions en cours des branches 2.7.12 et 3.5.2

Quelques caractéristiques de python

- Variables typées (int, float, string, list, dictionnaire) : typage dynamique.
- Langage orienté objet : propriétés, méthodes, héritage.
- Le code peut être groupé en modules thématiques ou en paquets (packages)
- Il existe une gigantesque bibliothèque de modules python (mathématiques, graphiques, orientés vers les applications web, etc...)

Cours, publications

Cours en ligne

√ en français :
- `https://perso.limsi.fr/pointal/_media/python:cours:courspython3.pdf`
- `http://inforef.be/swi/download/apprendre_python3_5.pdf`,
- `http://www.dsimb.inserm.fr/~fuchs/python/`,
- `http://www.courspython.com/`

√ en anglais : `http://www.scipy-lectures.org/index.html`

Tutoriels, web, help

Tutoriels

√ en anglais :
- `https://docs.python.org/3/tutorial/`
- `https://docs.scipy.org/doc/numpy-dev/user/quickstart.html`
- `http://www.scipy-lectures.org/`

web

√ de bons sites d'aide sur le web
- http ://docs.scipy.org/doc/numpy/
- stackoverflow.com (forum question/réponses)

fonction «help»

Pensez à utiliser le help en ligne de commande :
- `help nom_de_fonction` ou `?nom_de_fonction`

Installer un environnement python : anaconda

- Il existe des distributions de python comprenant une collection de ressources (bibliothèques de modules, interfaces utilisateur, éditeur).
- La distribution «anaconda» contient tout ce dont on a besoin pour le calcul, , le graphique, les entrées/sorties : numpy/scipy, matplotlib, ipython, spyder.
- Sur votre ordinateur personnel : télécharger la version 3.5 (64 bits) dédiée à votre système (Mac, Windows, linux) : https ://www.continuum.io/downloads.
- Installer la distribution (instructions suivant le système d'exploitation).

L'environnement python d'anaconda

√ Une fois la distribution anaconda installée :
- vous disposez d'un interpréteur python, de nombreuses ressources (modules), et de différents environnements de travail (console ipython, integrated development environment (IDE) spyder, jupyter notebook)
- l'environnement le plus léger est la console texte ipython.
- Vous aurez également besoin d'un éditeur de texte pour écrire/modifier vos scripts (gedit, kwrite, kate, ...)

√ Nous utiliserons ici la consle ipython («i» signifiant «interactif»), sous linux (Suze)

La console ipython est lancée : premiers pas...

√ Calcul arithmétique de base

```
[1]: 3+2
     5
[2]: 3**3
     27
```

√ Définition de variables :

```
[3]: x=3
[4]: y=2
[5]: z=x+y
[6]: type(z)
     int
```


Premiers pas...

√ Se déplacer dans le système de fichier, définir un répertoire de travail (console ipython)

```
[7]: pwd
     /racine//un_repertoire/
[8]: mkdir maths_SPE
[9]: ls
     maths_SPE/
[10]: ls -l
drwxr-xr-x 2 riw latmos 4096  4 sept. 21:14 maths_SPE/
[11]: cd maths_SPE
[11]: mkdir python
[12]: cd python
```


Premiers pas...

√ La directive «import»

```
[13]: x
      3
[14]: s=sin(x)
NameError: name 'sin' is not defined
[15]: import numpy as np
[16]: s=np.sin(x)
      0.14112000805986721
[17]: type(s)
      numpy.float64
[18]: ly = np.log(y)
[19]: z = np.e**(np.complex(0,1)*np.pi)
      (-1+1.2246063538223773e-16j)
```


Nommage des variables

- Un nom de variable (identificateur) comprend un ou plusieurs caractères alphanumériques
- Un nom de variable ne peut commencer par un caractère numérique
- Un identificateur est sensible à la case (`x` différent de `X`)
- Les 33 mots clés de python sont à proscrire

```
and     del      from     None    True    as
elif    global   nonlocal try     while   assert
else    if       not      break   except  import
or      with     class    False   in      pass
yield   continue finally  is      raise   def
for     lambda   return
```

Types de variables

- Les variables ont une (des) valeur(s). Celles-ci peuvent être de différents types :

type	descriptif	taille en mémoire	valeur min	valeur max
int	nombre entier	variable	illimité	illimité
bool	booléen	1 octet	0 (False)	1 (True)
float	nombre réel	8 octets	10^{-308}	10^{308}
complex	nombre complex	2×8 octets	10^{-308}	10^{308}
str	chaine de char	variable	–	–

- le typage est dynamique (python s'en charge)
- En plus des types de base, d'autres types de variables sont créés par les modules importés. Par exemple le module numpy comprend les types range, fraction, décimal, ndarray, etc...

Types de variables

- Il est posible d'imposer un type de variable :

```
[1]: x = 10     # entier
[2]: x = 10.    # réel
[3]: toto = int("300")  # conversion en entier
[4]: str(300)
     '300'
[5]: float(300)
     300.0
[5]: bool(-123456789)
     True
```


Les listes

- Une liste est une collection ordonnée de variables
- syntaxe : une suite d'éléments, séparés par des virgules, entourés de crochets

```
[1]: feu_de_circulation = ["vert", "orange", "rouge"]
[2]: feu_de_circulation[1]
     'orange'
[3]: liste1, liste2 = ['a','b'],[1, 2]
[3]: liste3 = [liste1, liste2]
[4]: liste3
     [['a', 'b'], [1, 2]]
[5]: liste4 = liste1.extend(liste2)  # une «méthode» associée
     liste4                          # au type list
     ['a', 'b', 1, 2]
[6]: liste3.append(0)                # une autre «méthode» associ
     [['a', 'b'], [1, 2], 0]         # au type list
```


Les listes

- Les éléments d'une liste sont repérés par un indice entier.
  - Le premier élément est repéré par l'indice 0
  - Le premier élément est repéré par l'indice 1
  - Le dernier élément est repéré par l'indice -1

```
[7]: liste3[0]
     ['a', 'b']
[8]: liste3[0][1]
     'b'
# liste contenant le premier et second éléments de lis
[9]: liste3[0:2]
     [['a', 'b'], [1, 2]]
```

Les tuples

Tuple : définition

Une tuple est une collection ordonnée d'objets non modifiables, éventuellement hétérogènes.

Syntaxe : une suite d'éléments, séparés par des virgules, entourés de parenthèses

```
[1]: feu_de_circulation = ("vert", "orange", "rouge")
[2]: feu_de_circulation[1]
     'orange'
[3]: feu_de_circulation[1] = "bleu"
TypeError: 'tuple' object does not support item assignment
```

python

Les tuples sont utiles pour définir des constantes.

Les dictionnaires

Définition

Un dictionnaire python est un tableau associatif. C'est un type de données permettant de stocker des couples (cle : valeur).
syntaxe : Collection de couples clé : valeur entourée d'accolades.

√ Un dictionnaire est itérable (on peut le parcourir) mais n'est pas ordonné.

```
[1]: ma_voiture = {
        "marque" : "Renault",
        "couleur": "rouge",
        "consommation": 8.5,
        "vitesse_max" : 150
      }
[2]: ma_voiture['vitesse_max']
     150
```

python

Les chaînes de caractères

- Plusieurs façons de définir des chaines de caractères

```
[1]: str1 = "une chaine avec un retour ligne \n"
[2]: str1
     "une chaine avec un retour ligne \n"
[3]: print(str1)
une chaine avec un retour ligne

[4]: str1[0:5]
     'une c'
[5]:  k,l = 1,2; print("k = %d et l = %d" %(k,l))
k = 1 et l = 2
[6]: str2 = """ une chaîne
     sur plusieurs
     lignes """
```

python

Les instructions de contrôles

- Un script python est constitué d'une suite d'instructions qui seront exécutée séquentiellement
- Certaines instructions peuvent être exécutées sous condition, ou peuvent être répétées plusieurs fois.
- On utilisera pour ce faire une «instruction composée»

instruction composée : définition

Une instruction composée se compose :

– d'une ligne d'en-tête terminée par deux-points « :»

– d'un bloc d'instructions indenté par rapport à la ligne d'en-tête. On utilise habituellement trois/quatre par indentation.

Toutes les instructions au même niveau d'indentation appartiennent au même bloc.

thon

Les instructions de contrôles

```
# saisir la température en C
[1]: T = input("Entrez la température extérieure : ")
[2]: if float(T) <= 0:
        print("il gèle")
     else:
        if float(T) > 5:
           print("T = %s > 5 C" % T)
        else:
           print("0 < T < 5 C")
```


Les instructions de contrôles

```
[3]: toto = input("Calculer la table de "); toto = int(toto)
[4]: print("table de multiplication de %d\n" %toto)
[5]: for i in range(1,11):
         print("%d x %d = %d " %(toto,i,i*toto))
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```


Les instructions de contrôles

- Dans la version basique, le langage python est extrêmement pauvre.
- Il est nécessaire de charger les ressources logicielles dont on aura besoin.
- Les ressources logicielles sont organisées en modules thématiques.
- Un module est un fichier — ou un ensemble de fichiers — contenant du code python logiquement organisé.
- Un module comprend généralement des définitions de variables (typées), des fonctions, des classes (objets).
- Les modules numpy et scipy (calcul), et matplotlib (graphique) contiennent la quasi totalité des outils dont nous aurons besoin pour le calcul et la présentation de données.

Les modules numpy et matplotlib

```
[1]: import numpy as np
[2]: from matplotlib.pyplot import *
[3]: np.sin(1)
[4]: X = np.arange(0,100,0.1)
[5]: Y = np.cos(X)*np.sin(X)
[6]: plot(X,Y)
[7]: show(block=False)
```

Les modules numpy et matplotlib

Il existe un racourcis permettant de charger à la fois matplotlib.pyplot et numpy : pylab lors d'une session ipython :

```
[1]: from pylab import *
[2]: ...
```

ou la commande «magique» :

```
[1]: %pylab
[2]: ...
```

ou encore, au lancement de ipython depuis un shell de commande :

```
> ipython - - pylab
```


Les tableaux numpy

- Le type de base de numpy est le tableau (array).
- un tableau est une collection ordonnée de variables homogènes
- Il existe plusieurs façon de créer des tableaux :

```
[1]: %pylab
[2]: X = array([1, 2, 3, 4, 5])
[3]: type(X)
     numpy.ndarray
[4]: Y = ndarray(len(X),dtype=float)
     # Le tableau Y est créé mais n'est pas initialisé
[5]: for i in range(len(X)):
        Y[i] = log10(X[i])
[6]  Z = ones((len(X),len(Y)),dtype=int)
```


Attributs des tableaux numpy

Quelques attributs des tableaux numpy :

```
[7]: X.shape
     (5,)
[8]: X.ndim
     1
[9]: X.dtype
     dtype('int64')
[10]: X.min()
     1
[11]: X.sum()
      15
[12]: X.cumsum()
      array([ 1,  3,  6, 10, 15])
```

et d'autres encore :

```
[13]: X.mean()
      3.0
[14]: X.std()
      1.4142135623730951
```


Attributs des tableaux numpy

Une propriété troublante des tableaux numpy :

```
[15]: Y = X
[16]: X
      array([1, 2, 3, 4, 5])
[17]: Y
      array([1, 2, 3, 4, 5])
[18]: X[0] = -1
[19]: X
      array([-1,  2,  3,  4,  5])
[20]  Y
      array([-1,  2,  3,  4,  5])
# le tableau Y est identique
# au tableau  X !!!
```

```
[21]: Y = X.copy()
[22]: X[1] = 10
[23]: X
      array([-1, 10, 3, 4, 5])
[24]: Y
      array([-1, 2, 3, 4, 5])
# Cette fois, le tableau Y
# n'est pas modifié si X
# est modifié.
```

numpy : autres fonctionnalités

Numpy contient bien d'autres fonctionnalités

- Lecture/ecriture de fichiers textes
  `np.loadtxt()`
  `np.save()`
- interpolation
  `np.interp()  # interpoklation linéaire`
- calcul d'histogramme
  `np.istogram() np.istogram2()`

scipy

Le paquet scipy s'appuie sur numpy et contient de nombreuses fonctions mathématiques :
Les fonctions scipy sont groupées en sous-modules :

- stats (scipy.stats)
- ndimage (scipy.ndimage)
- io (scipy.io)
- signal (scipy.signal)
- interpolate ...
- linsolve, ode
- fftpack
- integrate
- ...

Lecture de fichier texte : loadtxt

La fonction numpy.loadtxt() permet de lire des fichiers textes contenant des tableaux et éventuellement un en tête descriptif

- stats
- ndimage
- io
- signal
- interpolate
- linsolve, ode
- fftpack
- integrate
- ...

Structure d'un script

- Un script python est constitué d'une suite d'instructions python
- Les deux premières lignes sont des directives indiquant l'interpréteur du script (python) et le codage des caractères (utf-8)
- Les lignes suivantes sont les «import» des modules dont on aura besoin
- La suite du fichier contient vos instructions
- Sauver le fichier qui suit sous le nom : `log_vs_dl3.py`
- Pour l'exécuter

```
run log_vs_dl3.py      \# depuis la console ipython
python log_vs_dl3.py   \# depuis un shell de commande
./log_vs_dl3.py        \# depuis un shell de commande si le
                          fichier est exécutable.
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-0.5,1,100)
l1 = np.log(1+x)
d3 = x - x**2/2 + x**3/6
plt.figure(1)
plt.clf()
plt.plot(x,l1,'blue',linewidth=2,label="log(1+x)")
plt.plot(x,d3,'red',label="DL ordre 3")
plt.xlabel('x',fontsize = 14)
plt.ylabel('log(1+x)/DL(3)',fontsize = 14)
titre = "comparaison log(1+x)/DL(3)"
plt.title(titre,fontsize=15)
plt.grid(); plt.legend()
plt.show(False)
```


Ecriture d'un script

- Un dernier exemple avec lecture d'un fichier et tracé d'une colonne de données en fonction de l'autre
- Le fichier «exoplanets.tsv» se trouve dans le répertoire courant
- Sauvez le script sous le nom «exoplanets.py»,
- puis exécutez le.


```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

fich = "exoplanets.tsv"  # nom du fichier
data = np.loadtxt(fich,skiprows=2)
R = data[:,1]; P = data[:,2]
plt.figure(2)
plt.clf()
plt.loglog(R**3,P**2,'x',color='blue',linewidth=2)
plt.xlabel('$R^3$',fontsize = 14)
plt.ylabel('$T^2$',fontsize = 14)
titre = r"T$^2$ vs R$^3$ (troisième loi de Kepler)"
plt.title(titre,fontsize=15)
plt.grid();
plt.show(False)
```