

# Chapitre 1 : Introduction

Yves Guidet pour IPSSI

V1.4.7 March 14, 2017

# Remarque préliminaire

Ces transparents sont là pour pallier mon écriture illisible ; ils seront complétés par un livre librement imprimable, comme :

Swinnen <http://inforef.be/swi/python.htm>

Dive into Python <http://diveintopython.adrahon.org/>

Ainsi que par votre support ENI.

- ▶ Chapitre 1 Introduction
- ▶ Chapitre 2 Environnement de travail
- ▶ Chapitre 3 Bases du langage
- ▶ Chapitre 4 Types évolués
- ▶ Chapitre 5 Fonctions et sous-programmes
- ▶ Chapitre 6 Modules
- ▶ Chapitre 7 Classe
- ▶ Chapitre 8 Gestion des erreurs
- ▶ Chapitre 9 Librairies

Python est un langage universel ; citons :

- ▶ exploitation des systèmes informatiques
  - ▶ manipulation de fichiers et répertoires
  - ▶ analyse de logs
- ▶ Web : CGI, Django
- ▶ calculs scientifiques
- ▶ IHM

- ▶ années 1990 Guido van Rossum
- ▶ accent sur la lisibilité
  - ▶ indentation,
  - ▶ accolades -> mots clés
- ▶ langage de script
- ▶ licence libre
- ▶ impératif et orienté objet.
- ▶ Typage fort et dynamique

Ceci peut paraître contradictoire, mais Python est fortement typé, contrairement aux shells Unix et même à Perl, mais en même temps on ne déclare jamais de type (un peu comme en CAML, pour ceux qui connaissent).

# Un typage fort mais inférencé

Un exemple :

```
>>> k = 2
>>> type(k)
<type 'int'>
```

La variable *k* a reçu un *int*, on connaît donc son type sans avoir eu à déclarer celui-ci.

Plus fort :

```
>>> import os
>>> type(os)
<type 'module'>
>>> type('os')
<type 'str'>
```

Notons qu'en Python3k, *class* se substitue à *type*.

# Installation de Python sous Windows.

- ▶ <http://www.python.org>.
- ▶ double clic
- ▶ alternative : Active Python

# Mise en œuvre de Python, éditeurs.

- ▶ mise en œuvre : lancer une fenêtre de commandes

```
c:\pythonxx\python.exe
```

```
c:\pythonxx\python.exe fichier
```

- ▶ et si je ne suis pas anglophone ?

```
# -*- coding: iso-8859-1 -*-
```

- ▶ ou utf-8 (plus rare sous Windows)

# Quel éditeur utiliser ?

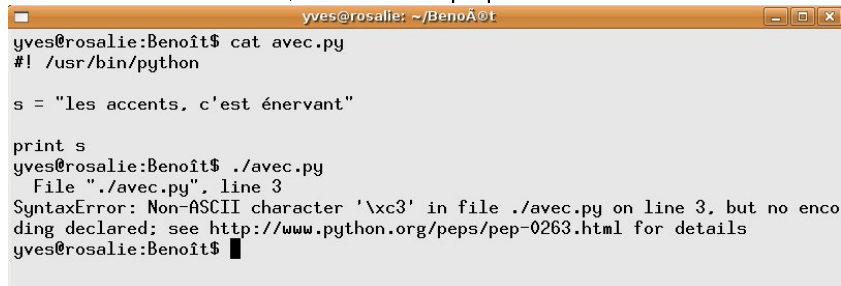
- ▶ IDLE : éditeur fourni avec python
- ▶ XEmacs : éditeur très complet.
- ▶ VIM : éditeur très complet. Coloration syntaxique pour Python.
- ▶ SPE: excellent editeur 100% Python
- ▶ jEdit : éditeur Java (nécessite le JRE 1.3).
- ▶ Komodo : excellent éditeur Python/perl et autres langages  
License: Propriétaire (logiciel commercial).
- ▶ Scite : éditeur léger, simple et efficace.
- ▶ synEdit : éditeur léger, simple et efficace.
- ▶ PSPad : un excellent éditeur, rapide et plus complet que syn.
- ▶ ConTEXT : éditeur simple et efficace.
- ▶ nedit : éditeur comparable à emacs/xemacs
- ▶ Leo : éditeur hiérarchique Python/Tkinter.

On dira un mot sur *IPython* dans le prochain chapitre.

# Encodage : présentation

Jusqu'ici nous n'avons utilisé que des chaînes en anglais, ou, en français, sans caractères accentués.

Si l'on met des accents, tout se complique.

A terminal window titled 'yves@rosalie: ~/Benoît' shows the execution of a Python script. The user runs 'cat avec.py' which displays a script with a string containing accents. Then, the user runs './avec.py', which results in a 'SyntaxError: Non-ASCII character '\xc3' in file ./avec.py on line 3, but no encoding declared' message. The terminal text is as follows:

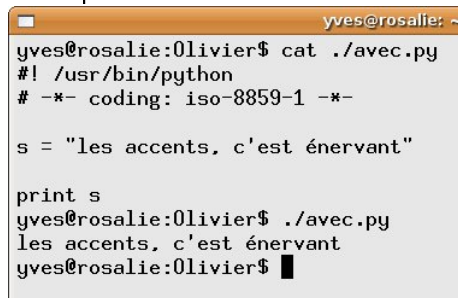
```
yves@rosalie:~/Benoît$ cat avec.py
#!/usr/bin/python

s = "les accents, c'est énervant"

print s
yves@rosalie:~/Benoît$ ./avec.py
File "./avec.py", line 3
SyntaxError: Non-ASCII character '\xc3' in file ./avec.py on line 3, but no encoding declared; see http://www.python.org/peps/pep-0263.html for details
yves@rosalie:~/Benoît$
```

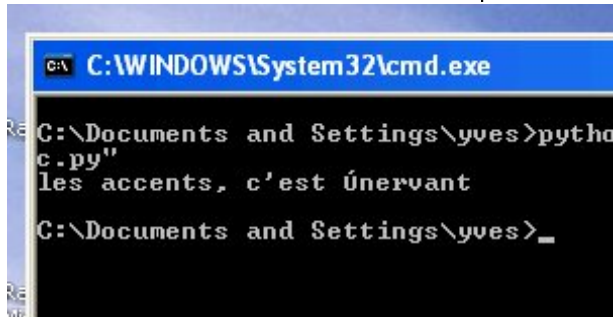
En effet, Python Python2) veut savoir comment sont encodées les chaînes, par défaut il (du moins Python2) considère que c'est de l'ASCII (7bits, donc sans accent aucun).

Un « commentaire magique » (*magic comment*) va résoudre notre problème :



```
yves@rosalie: ~  
yves@rosalie:0livier$ cat ./avec.py  
#!/usr/bin/python  
# -*- coding: iso-8859-1 -*-  
  
s = "les accents, c'est énervant"  
  
print s  
yves@rosalie:0livier$ ./avec.py  
les accents, c'est énervant  
yves@rosalie:0livier$
```

Sous MSWindows maintenant, ce n'est pas absolument idéal :



```
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings\yves>python c.py
les accents, c'est énervant
C:\Documents and Settings\yves>_
```

MSDOS et MSWindows utilisent en effet des « pages de code » distincts.

Et maintenant sous IDLE :



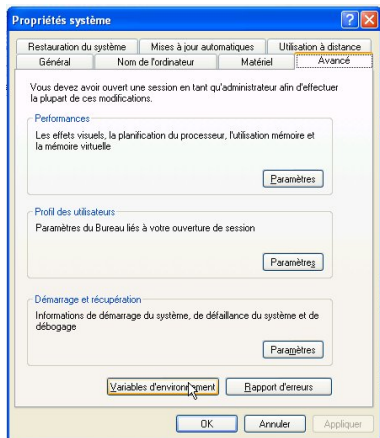
```

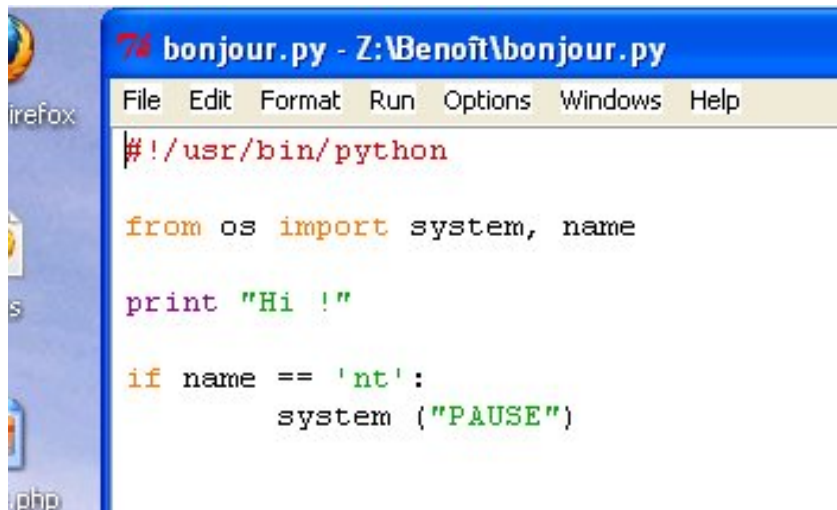
Python Shell
File Edit Shell Debug Options Windows Help
#! /usr/bin/env python
# -*- coding: utf-8 -*-
Type "copyright", "credits" or "license()" for more
>>>
>>>
>>> print 'les accents, c'est énervant'
>>>

```

Ouf.

# La variable PATH





The screenshot shows a Windows desktop environment. On the left, a portion of the taskbar is visible with icons for Firefox and a file named .php. The main window is a code editor titled "bonjour.py - Z:\Benoît\bonjour.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code is written in Python and uses syntax highlighting: red for comments, orange for keywords, green for strings, and purple for print statements.

```
#!/usr/bin/python

from os import system, name

print "Hi !"

if name == 'nt':
    system ("PAUSE")
```

et voilà !



# Et en Python3k ? (I/II)

Reprenons le script avec accents et *magic comment* et lançons-le sous *python3* :

```
yves@bella:python3-magiccomment$ cat avec.py
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-
```

```
s = "les accents, c'est énervant"
```

```
print s
```

```
yves@bella:python3-magiccomment$ python3 avec.py
```

```
File "avec.py", line 6
```

```
    print s
```

```
    ^
```

```
SyntaxError: invalid syntax
```

## Et en Python3k ? (II/II)

Allons bon. Le *print* a changé, mettons le *s* entre parenthèses, et enlevons le *commentaire magique* (qui visiblement ne gêne pas mais était nécessaire en Python 2.7) :

```
yves@bella:python3-magiccomment$ !cat  
cat avec.py  
#!/usr/bin/python
```

```
s = "les accents, c'est énervant"
```

```
print(s)  
yves@bella:python3-magiccomment$ !py  
python3 avec.py  
les accents, c'est énervant
```

Ça marche. Notons que le *shebang* n'est pas utilisé.

# Et sous Linux ?

Zéro souci, Python2 est déjà installé. le script d'installation des RedHat/CentOS/Fedora s'appelle *anaconda* ...  
Le problème de l'encodage subsiste.

# Importation de modules avec from

Revenons sur cette « ruse » ; quand on écrit :

```
from os import system, name
```

on n'importe du module `os` que deux symboles désignant l'un une fonction (*system*) et l'autre une variable (*name*).

On les utilise « tels quels », **sans** les préfixer de leur espace de noms.

```
if name == 'nt':  
    system("PAUSE")  
else:  
    print "youpi je ne suis pas sous Windoze !"
```

On peut même importer **tous** les symboles en utilisant un « joker » :

```
from os import *
```

# Importation de modules sans from

Maintenant, on peut aussi importer l'intégralité du module :

```
import os
```

Dans ce cas en revanche il faudra préciser qu'il s'agit du *name* et du *system* du package *os* :

```
if os.name == 'nt':  
    os.system ("PAUSE")  
else:  
    print "youpi je ne suis pas sous Windoze !"
```

Il existe aussi une forme avec "as" permettant de renommer localement un module.

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> a = np.linspace(0,10,100)
>>> b = np.exp(-a)
>>> plt.plot(a,b)
>>> plt.show()
```

En savoir plus ? *help('import')*.