Petit document sur Pythontex

Yves Moncheaux

16 avril 2019

Table des matières

1 Structure basique d'un document

```
\documentclass[12pt,a4paper]{article}
\usepackage[utf8]{inputenc}
\usepackage[french]{babel}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{pythontex}

\begin{document}
... commandes LaTeX ou Python
\end{document}
```

2 Insertion de code Python avec coloriage syntaxique

Code

```
\begin{pyverbatim}
xA = float(input(" Entrez xA "))
yA = float(input(" Entrez yA "))
xB = float(input(" Entrez xB "))
yB = float(input(" Entrez yB "))
if xA <> xB:
    a=(yB - yA)/(xB - xA)
    b=yA - a*xA
    print("La fonction affine dont la courbe est (AB) est")
    print("f(x)=",a,"*x+",b)
else:
    print("(AB) n'est pas la courbe d'une fonction affine.")
\end{pyverbatim}
```

Résultat

```
xA = float(input(" Entrez xA "))
yA = float(input(" Entrez yA "))
xB = float(input(" Entrez xB "))
yB = float(input(" Entrez yB "))
if xA <> xB:
    a=(yB - yA)/(xB - xA)
    b=yA - a*xA
    print("La fonction affine dont la courbe est (AB) est")
    print("f(x)=",a,"*x+",b)
else:
    print("(AB) n'est pas la courbe d'une fonction affine.")
```

3 Exécution de code Python

3.1 La commande \py

La commande \py renvoie le résultat de son argument, sous forme de texte.

Code

```
$2^{10} = \py{2**10}$
```

Résultat

$2^{10} = 1024$

Code

```
\py{'Bonjour'.upper()}
```

Résultat

BONJOUR

On peut utiliser un autre délimiteur que les accolades (comme la commande \verb) :

Code

```
\py+'Bonjour'.upper()+
```

Résultat

BONJOUR

3.2 La commande \pyc

La commande \pyc exécute du code Python.

Code

```
\pyc{var='Bonjour';print(var,' chez vous !');}
```

Résultat

Bonjour chez vous !

On peut utiliser ensuite la commande \py :

Code

```
\pyc{var='Bonjour'}\py{var}
```

Résultat

Bonjour

3.3 Les commandes \pyb, \pyv

La commande \pyb met en forme (comme l'environnement `pyverbatim`) et exécute le code.

Code

```
\pyc{var='Bonsoir'}
\pyb{var='Bonjour';print(var,' chez vous !');}
\py{var}
```

Résultat

```
var='Bonjour';print(var,' chez vous !');
```
Bonjour

On voit ici que la commande print() n'a pas été exécutée mais que l'affectation a bien eu lieu.

La commande \pyv (pour "verbatim") met seulement en forme son argument (comme l'environnement \pyverbatim).

Code

```
\pyc{var='Bonsoir'}
\pyv{var='Bonjour';print(var,' chez vous !');}
\py{var}
```

Résultat

```
var='Bonjour';print(var,' chez vous !');
```
Bonsoir

On voit ici que la commande print() n'a pas été exécutée et que l'affectation n'a pas eu lieu.

Enfin, la commande \pys (pour "substitute") permet de remplacer la partie d'un texte contenue dans !{...} par la valeur d'une variable et affiche le texte obtenu.

Code

```
\pyc{var=42}
\pys{\verb|toto = !{var}|}
```

```
>
```

Résultat

```
toto = 42
```

3.4 Les environnements `pycode`, `pyblock` et `pysub`

Ils correspondent aux commandes `\pyc`, `\pyb`, `\pys`.

Code

```
\begin{pycode}
print(r'\begin{center}')
print(r'\textit{Un grand bonjour !!!}')
print(r'\end{center}')
\end{pycode}
```

Résultat

Un grand bonjour!!!

4 L'environnement `pyconsole`

Il permet de simuler une console Python.

Code

```
\begin{pyconsole}
var=2**8
var
3.14+2.05
\end{pyconsole}
```

Résultat

```
>>> var=2**8
>>> var
256
>>> 3.14+2.05
5.1899999999999995
```

5 Calcul formel avec Sympy

Code

```
\begin{pycode}
from sympy import *
x= symbols('x')#x, y = symbols('x y') pour plusieurs variables
exp1 = (2*x-1)*(4-3*x)
exp2 = -6*x**2+11*x-4
\end{pycode}
Un d\'eveloppement :
\[(2x-1)(-3x+4)=\py{expand(exp1)}=\py{latex(expand(exp1))}.\]
\newline
Une factorisation :
\[-6x^2+11x-4 = \py{factor(exp2)} = \py{latex(factor(exp2))}.\]
```

Résultat

Un développement :

$$(2x-1)(-3x+4) = -6*x**2+11*x-4 = -6x^2+11x-4.$$

Une factorisation :

$$-6x^2+11x-4 = -(2*x-1)*(3*x-4) = -(2x-1)(3x-4).$$

Code

```
\begin{pycode}
f = x*ln(x) + sin(x)**4
g = Integral(f, x)
\end{pycode}
\[\py{latex(g)}=\py{latex(g.doit())}\]
```

Résultat

$$\int \left(x\log\left(x\right)+\sin^4\left(x\right)\right)\,dx = \frac{x^2}{2}\log\left(x\right)-\frac{x^2}{4}+\frac{3x}{8}-\frac{1}{4}\sin^3\left(x\right)\cos\left(x\right)-\frac{3}{8}\sin\left(x\right)\cos\left(x\right)$$

On peut utiliser d'autres variables que x :

Code

```
\begin{sympycode}
theta = Symbol(r'\theta')
p = 1/(sqrt(2*pi))*Integral(exp(-theta**2/2), (theta, -oo, oo))
\end{sympycode}
\[\sympy{p}=\sympy{p.doit()}\]
```

Résultat

$$\frac{\sqrt{2}}{2\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\frac{\theta^2}{2}}\, d\theta = 1$$

Remarque : les commandes `\sympy`, `\sympyc`, etc., permettent normalement d'écrire `\sympy{g}=\sympy{g.doit()}` au lieu de `\py{latex(g)}=\py{latex(g.doit())}` mais ne marchent pas systématiquement chez moi (voir les deux exemples ci-dessus)...

6 Calcul approché avec Scipy

Code

```
\begin{pycode}
from math import exp,inf
from scipy.integrate import quad
myintegral = quad(lambda x: exp(-x**2/2), 0, inf)[0]
\end{pycode}
\[\int_0^\infty e^{-x^2/2}\,dx = \py{myintegral}\]
```

Résultat

$$\int_0^{\infty} e^{-x^2/2}\, dx = 1.2533141373154997$$

7 Valeurs aléatoires

Code

```
\begin{pycode}
from sympy.stats import DiscreteUniform, sample
xa = sample(DiscreteUniform('a', range(-10, 11)))
ya = sample(DiscreteUniform('b', range(-10, 11)))
xb = sample(DiscreteUniform('c', range(-10, 11)))
yb = sample(DiscreteUniform('c', range(-10, 11)))
\end{pycode}
Soient \(A\,(\py{xa};\py{ya})\) et \(B\,(\py{xb};\py{yb})\).
\newline
Alors :
\(AB
= \sqrt{(\py{xb}-\py{xa})^2+(\py{yb}-\py{ya})^2}
= \sqrt{(\py{xb-xa})^2+(\py{yb-ya})^2}
= \sqrt{\py{(xb-xa)**2+(yb-ya)**2}}
\).
```

Résultat

Soient $A\,(-5;-10)$ et $B\,(-7;-2)$.
Alors : $AB = \sqrt{(-7--5)^2+(-2--10)^2} = \sqrt{(-2)^2+(8)^2} = \sqrt{68}$.

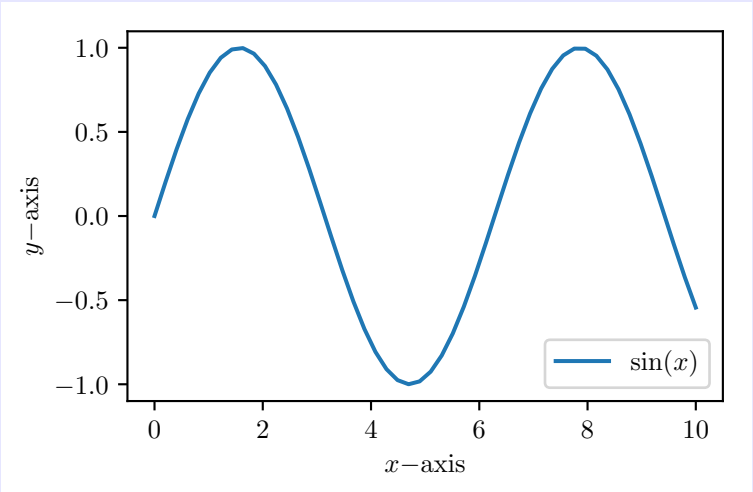
8 Création de graphiques

En utilisant la bibliothèque matplotlib, Python peut générer des graphiques. L'intérêt ici est que les textes inclus dans le graphique sont au format LaTeX.

Code

```
\begin{pylabcode}
rc('text', usetex=True)
rc('font', family='serif')
rc('font', size=10.0)
rc('legend', fontsize=10.0)
rc('font', weight='normal')
x = linspace(0, 10)
figure(figsize=(4, 2.5))
plot(x, sin(x), label='$\sin(x)$')
xlabel(r'$x\mathrm{-axis}$')
ylabel(r'$y\mathrm{-axis}$')
legend(loc='lower right')
savefig('myplot.pdf', bbox_inches='tight')
\end{pylabcode}
\begin{center}
\includegraphics{myplot.pdf}
\end{center}
```

Résultat



9 Définitions de macros LaTeX avec Python

On peut utiliser Python pour définir de nouvelles macros LaTeX avec une syntaxe beaucoup plus simple.

Par exemple, pour retourner une chaîne de caractères :

Code

```
\newcommand{\retourne}[1]{\py{"#1"[::-1]}}
\retourne{'Bonjour tout le monde !'}
```

Résultat

'! ednom el tuot ruojnoB'

Ou pour créer un polynôme du second degré à coefficients aléatoires :

Code

```
\begin{pycode}
from sympy import *
from sympy.stats import DiscreteUniform, sample
x= symbols('x')
a = DiscreteUniform('a', range(-10, 11))
b = DiscreteUniform('b', range(-10, 11))
c = DiscreteUniform('c', range(-10, 11))
def randquad():
    return Eq(sample(a)*x**2 + sample(b)*x + sample(c))
\end{pycode}
\newcommand\randquad{\ensuremath{\py{latex(randquad())}}}
\randquad
```

Résultat

$6x^2 - 8x + 4 = 0$

D'autres exemples dans une galerie Pythontex, par exemple une table de dérivées automatisées.

10 Configuration sous linux

10.1 Installation de Python et Pythontex

Pythontex était déjà installé sur mon ordinateur sous Xubuntu. J'ai dû installer python3-pygments, python3-scipy, python3-matplotlib et python3-sympy, en tapant dans une fenêtre de terminal (Ctrl Alt t sous Xubuntu) :

Code

```
sudo apt install python3-pygments python3-sympy python3-scipy python3-matplotlib
```

10.2 TeXworks

Créer un fichier `~/TeXworks/scripts/pythontex.js` (on peut aussi, dans TeXworks, aller dans Scripts -> Scripts pour TeXworks, Montrer le dossier des scripts puis copier-coller un script existant pour le modifier, penser à Recharger la liste des scripts) qui contient :

Code

```
// TeXworksScript
// Title: Pythontex
// Description: Compilations avec inclusion de scripts ou code Python
// Author: prof.math
// Version: 0.1
// Date: 2019-04-11
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Ctrl+Alt+P

// Dans Edition / Préférences / Script /
// Autoriser les scripts à écrire dans les fichiers

var nomfic = TW.target.fileName;

// compilation du document, pythontex puis recompilation
var cmd1="pdflatex --shell-escape -synctex=1 -interaction=nonstopmode ";
cmd1=cmd1+nomfic;
var cmd2="/usr/share/texlive/texmf-dist/scripts/pythontex/pythontex3.py ";
cmd2=cmd2+nomfic;
TW.system(cmd1);
TW.system(cmd2);
TW.system(cmd1);
```

10.3 Texmaker

Dans Utilisateur -> Commandes utilisateurs -> Éditer les commandes utilisateurs, choisir une commande, puis dans Item menu, taper : Pythontex et la commande est (en une seule ligne) :

```
pdflatex --shell-escape -synctex=1 -interaction=nonstopmode %.tex
|pythontex %.tex|pdflatex --shell-escape -synctex=1
 -interaction=nonstopmode %.tex
```

11 Bibliographie

Présentation rapide
Engendrer une feuille d'exercices aléatoires avec Python
Python Quickstart
Diaporama : A Gentle Introduction to PythonTEX
Galerie Pythontex
La documentation officielle
Deux vidéos en anglais : la une et la deux.