

Comment installer et programmer des scripts Python dans Inkscape ?

VVPix

v0.10 (décembre 2008)

Table des matières

1	Introduction	1
2	Installations	2
2.1	Installation d'Inkscape avec Windows	2
2.2	Installation de nouveaux scripts Python dans Inkscape	2
2.3	Localisation des nouveaux scripts dans Inkscape	2
3	Programmation Python	3
3.1	Saisie de valeurs grâce à une interface utilisateur	3
3.2	Dessin des formes de base	5
3.3	Accès à la documentation	8

1 Introduction

Inkscape est un logiciel de dessin vectoriel, gratuit et multiplateformes (Linux, Mac, Windows).

Ce document indique succinctement comment écrire des scripts Python pour ce logiciel.

J'ai écrit ce document en faisant des tests sur la version 0.46 d'Inkscape, sous Windows 2000 SP4.

2 Installations

2.1 Installation d'Inkscape avec Windows

La dernière version d'Inkscape est disponible ici : <http://www.inkscape.org/>.

L'installation ne pose aucun problème particulier. Python est installé avec Inkscape et il ne faut faire aucune installation complémentaire pour pouvoir bénéficier des scripts Python avec Inkscape.

2.2 Installation de nouveaux scripts Python dans Inkscape

Un script Python pour Inkscape est constitué d'au moins deux fichiers :

- Un fichier .inx :
- Un fichier .py

Ces scripts sont localisés dans le dossier `share/extensions` d'Inkscape.

Par exemple, le chemin par défaut dans Windows est :

`C:\Program Files\Inkscape\share\extensions`

Avec Gimp et Blender, on peut choisir où placer ses greffons pour éviter de les mélanger avec ceux livrés avec le logiciel, mais apparemment ce n'est pas encore le cas avec Inkscape v0.46.

2.3 Localisation des nouveaux scripts dans Inkscape

La localisation d'un script s'effectue grâce au fichier .inx. Un fichier "inx" est un fichier XML dont on trouve un schéma DTD sur le site officiel d'Inkscape.

Voici par exemple le fichier `hello.inx`, dérivé de l'exemple "Hello world" posté sur le wiki de www.inkscape.org.

Listing 1 – Exemple de fichier ".inx" : le fichier "Hello.inx"

```

1 <inkscape-extension>
  <_name>Hello</_name>
3  <id>org.ekips.filter.hello</id>
  <dependency type="executable" location="extensions">hello.py</dependency>
5  <dependency type="executable" location="extensions">inkex.py</dependency>
  <param name="strTexte" type="string" _gui-text="Saisissez le texte a ecrire :">
    Hello</param>
7  <effect>
  <object-type>all</object-type>
9  <effects-menu>
  <submenu _name="Exemples" />
11 </effects-menu>
</effect>
13 <script>
  <command reldir="extensions" interpreter="python">hello.py</command>
15 </script>
</inkscape-extension>

```

La ligne 2 du listing 1 donne le nom d'appel du script dans le menu d'Inkscape ("Hello").

Les lignes 9 et 10 indiquent que le script sera inséré dans le menu "Effect - Exemples" d'Inkscape (voir figure 1 ci-après).

La ligne 4 indique quel fichier Python est relié au fichier .py (hello.py).

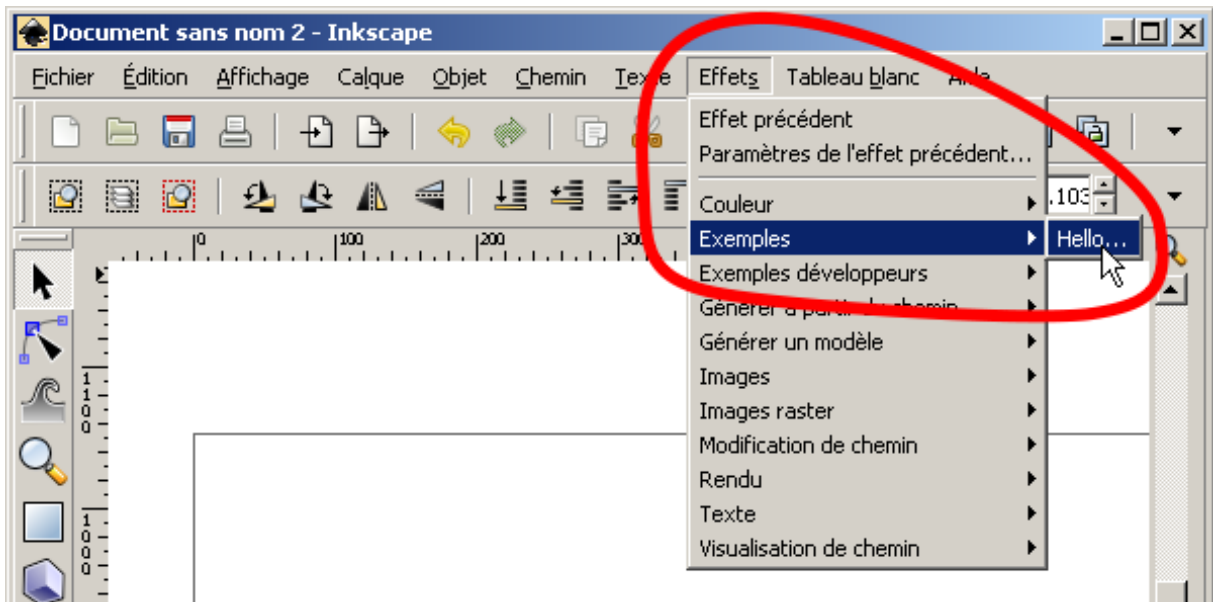


FIG. 1 – Script “Hello” dans le menu “Effect” d’Inkscape

ATTENTION : Un moyen très simple d’écrire un fichier “.inx” qui ne fonctionne pas est d’ajouter un accent dans n’importe laquelle de ses lignes. Le fichier “.inx” que j’ai utilisé est écrit avec le jeu de caractère CP1252 (≈ ISO 8859-15).

3 Programmation Python

Dans les fichiers Python, comme dans les fichiers “.inx”, il ne faut utiliser aucun accent si on utilise le jeu de caractère CP1252 (≈ ISO 8859-15).

3.1 Saisie de valeurs grâce à une interface utilisateur

Le fichier “Hello.inx”, listing 1, permet aussi de créer l’interface graphique figure 2, juste avec la ligne 6.

On voit la récupération de cette variable “strTexte” dans le listing 2, ligne 55 et 56.

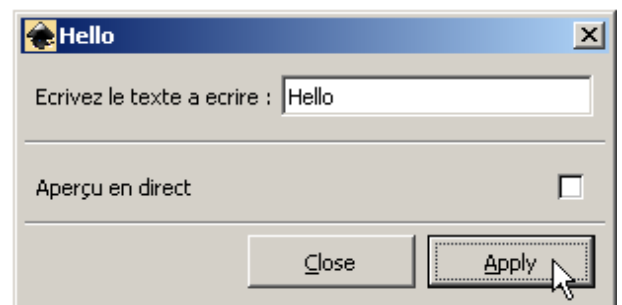


FIG. 2 – Interface utilisateur générée par le fichier “Hello.inx”

Listing 2 – Fichier “hello.py” (partiel)

```

#!/usr/bin/env python
2
# Les deux lignes suivant sont nécessaires seulement si le script n'est pas
4 # directement dans le dossier d'installation
import sys
6 #sys.path.append('/usr/share/inkscape/extensions')

8 # Utilisation du module inkex avec des effets predefinis
import inkex
10 # Le module simplestyle fournit des fonctions pour le parsing des styles

```

```

12 from simplestyle import *
13
14 class CHello( inkex.Effect ):
15     """
16     Exemple Inkscape
17     Cree un nouveau calque et dessine des elements de base
18     """
19     def __init__(self):
20         """
21         Constructeur
22         Definit l'option "--strTexte" du script
23         """
24         # Appel du constructeur.
25         inkex.Effect.__init__(self)
26
27         # Definit la chaine d'option "--strTexte" avec le raccourci "-w" et
28         # la valeur par default "Hello".
29         self.OptionParser.add_option('-w', '--strTexte', action = 'store',
30                                     type = 'string',
31                                     dest = 'strTexte', default = 'Hello',
32                                     help = 'Message a ecrire ?')
33
34     def effect(self):
35         """
36         Fonction principale
37         Surchage la fonction de la classe de base
38         Dessine quelques elements sur le document SVG
39         """
40         # Recupere le document SVG principal
41         svg = self.document.getroot()
42
43         # Recuperation de la hauteur et de la largeur de la feuille
44         width = inkex.unittoou( svg.get('width') )
45         height = inkex.unittoou( svg.attrib['height'] )
46
47         # Creation d'un nouveau calque
48         layer = inkex.etree.SubElement(svg, 'g')
49         layer.set(inkex.addNS( 'label', 'inkscape'), 'Layer texte' )
50         layer.set(inkex.addNS( 'groupmode', 'inkscape'), 'layer' )
51
52         # Creation d'un element texte
53         texte = inkex.etree.Element(inkex.addNS('text', 'svg'))
54         # - Recuperation de la valeur de la variable "strTexte" saisie dans la
55         #   boite de dialogue
56         strVal = self.options.strTexte
57         texte.text = strVal
58
59     [...]
60
61 # Execute la fonction "effect" de la classe "CHello"
62 hello = CHello()
63 hello.affect()

```

3.2 Dessin des formes de base

Le script listing 3 (page 5), associé au fichier inx, listing 1, produit l'image figure 3.

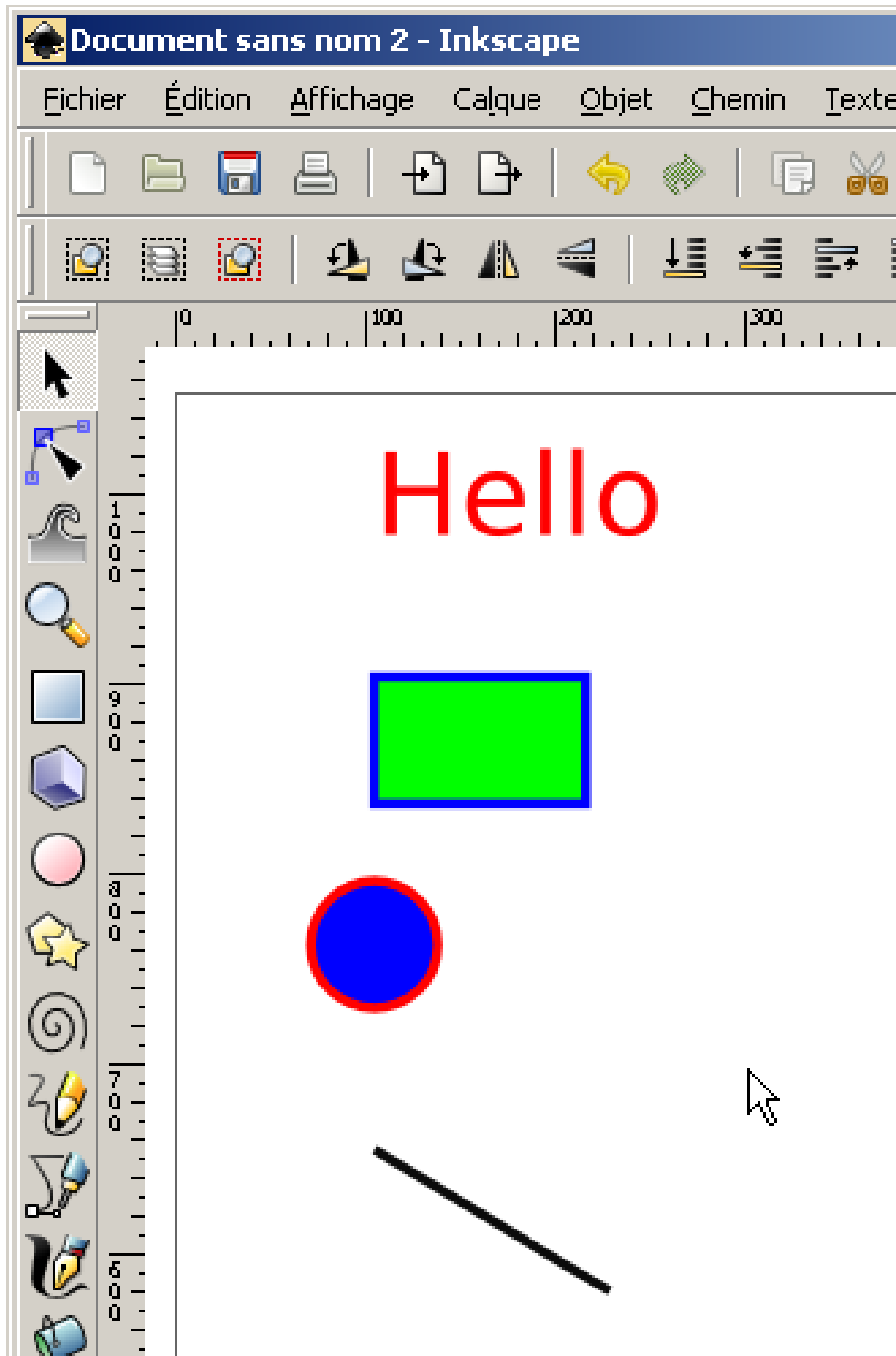


FIG. 3 – Résultat du script “Hello.py”, listing 3

Listing 3 – Fichier “hello.py”

```

1 #!/usr/bin/env python
3 # Les deux lignes suivant sont necessaires seulement si le script n'est pas
  # directement dans le dossier d'installation
5 import sys
  #sys.path.append('/usr/share/inkscape/extensions')
```

```

7 # Utilisation du module inkex avec des effets predefinis
9 import inkex
# Le module simplestyle fournit des fonctions pour le parsing des styles
11 from simplestyle import *

13 class CHello( inkex.Effect ):
    """
15 Exemple Inkscape
    Cree un nouveau calque et dessine des elements de base
    """
17
19 def __init__(self):
    """
21 Constructeur
    Definit l'option "--strTexte" du script
    """
23
25 # Appel du constructeur.
    inkex.Effect.__init__(self)
27
29 # Definit la chaine d'option "--strTexte" avec le raccourci "-w" et
    # la valeur par defaut "Hello".
    self.OptionParser.add_option( '-w', '--strTexte', action = 'store',
31                                     type = 'string',
                                     dest = 'strTexte', default = 'Hello',
                                     help = 'Message a ecrire ?')

33 def effect(self):
    """
35 Fonction principale
    Surchage la fonction de la classe de base
    Dessine quelques elements sur le docuemnt SVG
    """
37
39 # Recupere le document SVG principal
    svg = self.document.getroot()
41
43 # Recuperation de la hauteur et de la largeur de la feuille
    width = inkex.unittoou( svg.get('width') )
    height = inkex.unittoou( svg.attrib['height'] )
45
47 # Creation d'un nouveau calque
    layer = inkex.etree.SubElement(svg, 'g')
    layer.set(inkex.addNS( 'label', 'inkscape'), 'Layer texte' )
    layer.set(inkex.addNS( 'groupmode', 'inkscape'), 'layer' )
49
51 # Creation d'un element texte
    texte = inkex.etree.Element(inkex.addNS('text', 'svg'))
53 # - Recuperation de la valeur de la variable "strTexte" saisie dans la
    #   boite de dialogue
    strVal = self.options.strTexte
    texte.text = strVal
57
59 # Reglages initiaux
    x_org = height / 10
    y_org = width / 10
61 pas_x = width / 6
    pas_y = width / 10
63 nLargeurTrait = 5

65 # Set text position to center of document.
    texte.set('x', str( x_org ) )
    texte.set('y', str( y_org ) )
67
69 # Center text horizontally with CSS style.

```

```

71 style = { 'text-align' : 'left', \
           'text-anchor' : 'top', \
           'font-size' : '48pt', \
73           'fill' : 'rgb(255, 0, 0)' }
texte.set( 'style', formatStyle( style ) )
75
# Ajoute le texte au calque
77 layer.append( texte )
79
# Creation d'un rectangle
y_org += pas_y
81 rectangle = inkex.etree.Element(inkex.addNS( 'rect', 'svg' ) )
rectangle.set( 'x', str(x_org) )
83 rectangle.set( 'y', str(y_org) )
rectangle.set( 'width', str(pas_x - pas_x / 10.0) )
85 rectangle.set( 'height', str(pas_y - pas_y / 10.0) )
rectangle.set( 'fill', 'rgb( 0, 255, 0)' );# couleur de remplissage
87 rectangle.set( 'stroke', 'blue' );# couleur du contour
rectangle.set( 'stroke-width', str(nLargeurTrait) );# largeur du contour
89
# Ajout du rectangle sur le calque
91 layer.append( rectangle )
93
# Creation d'un cercle
y_org += pas_y
95 cercle = inkex.etree.Element(inkex.addNS( 'circle', 'svg' ) )
nRayon = ( pas_y - pas_y / 10.0) / 2.0
97 y_org += 2 * nRayon
cercle.set( 'cx', str(x_org) )
99 cercle.set( 'cy', str(y_org) )
cercle.set( 'r', str(nRayon) )
101 cercle.set( 'fill', '#0000FF' );
cercle.set( 'stroke', 'red' );
103 cercle.set( 'stroke-width', str(nLargeurTrait) );
105
# Ajout du cercle sur le calque
layer.append( cercle )
107
# Creation d'une ligne
y_org += pas_y
109 ligne = inkex.etree.Element(inkex.addNS( 'line', 'svg' ) )
nRayon = ( pas_y - pas_y / 10.0) / 2.0
y_org += nRayon
113 ligne.set( 'x1', str(x_org) )
ligne.set( 'y1', str(y_org) )
115 ligne.set( 'x2', str(x_org + pas_x) )
ligne.set( 'y2', str(y_org + pas_y) );
117 ligne.set( 'stroke', 'rgb(10,10,10)' );
ligne.set( 'stroke-width', str(nLargeurTrait) );
119
# Ajout de la ligne sur le calque
121 layer.append( ligne )
123
# Execute la fonction "effect" de la classe "CHello"
125 hello = CHello()
hello.affect()

```

3.3 Accès à la documentation

Avec Inkscape, le problème est, pour l'instant, que je n'ai pas trouvé de documentation de l'API Inkscape-Python. On trouve seulement quelques listings éparpillés sur internet sans explications. J'ai passé beaucoup de temps juste pour dessiner un rectangle et en changer les couleurs, alors que c'est une opération très simple.

Inkscape repose sur le format SVG et le meilleur moyen que j'ai trouvé pour récupérer la documentation des attributs des formes de base, par exemple, a été de me référer au site du W3C sur le format SVG.

Ici par exemple :

- <http://www.w3.org/TR/SVG/shapes.html>
- <http://www.w3.org/Graphics/SVG/IG/resources/StateOfArt-Dailey.html>

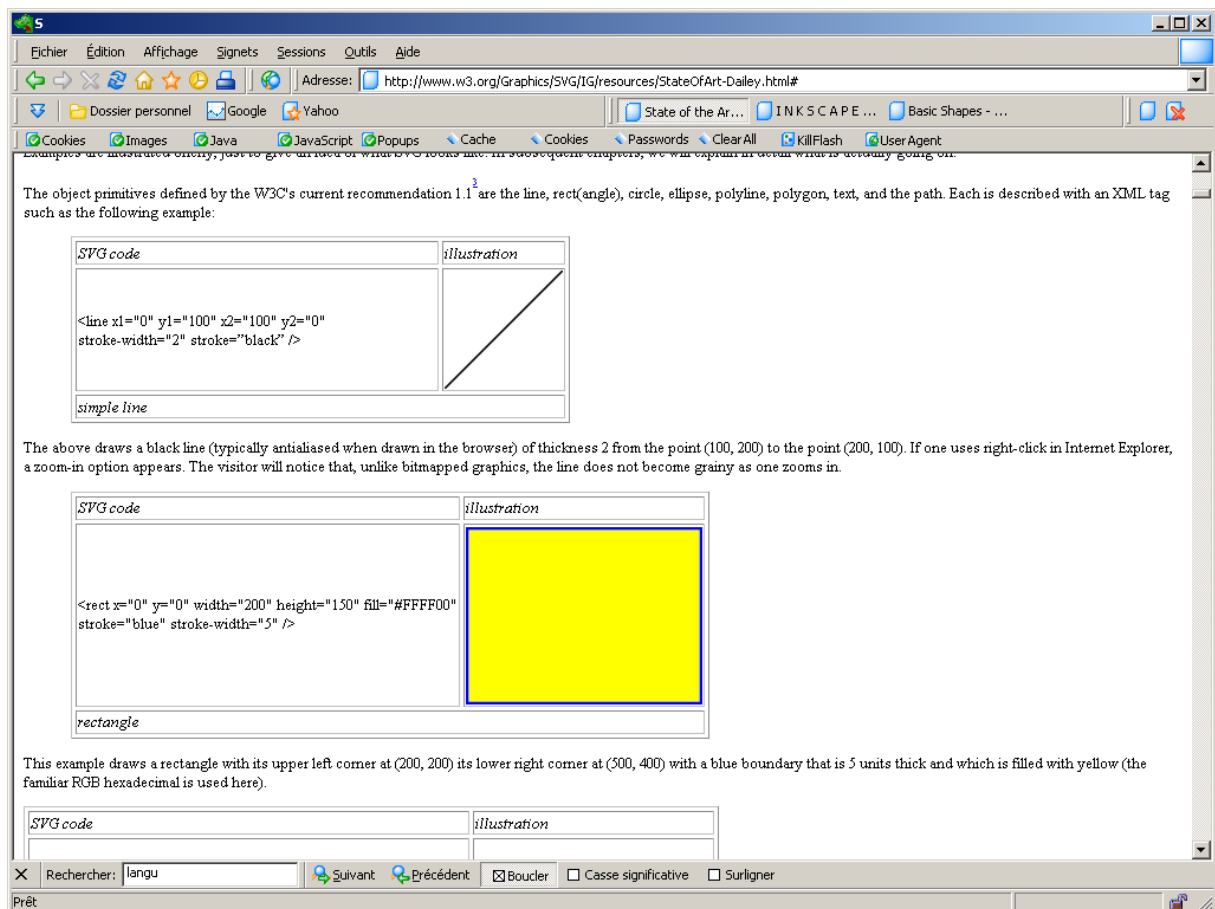


FIG. 4 – Détail des attributs des figures de base SVG sur le site W3C