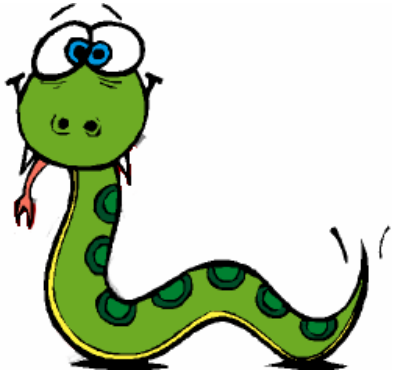
Elargir l'utilisation de PYTHON

Outils et Interface Graphique



Olivier Hargoaa
Informatique et Réseaux, année 3

Comment mieux utiliser PYTHON ?

Plan :

Objectifs.

Présentation du langage.

Quelques librairies.

Choix d'un IDE.

Choix d'un générateur de documentation.

Langage d'interface graphique.

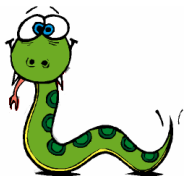
Construction assistée d'IG.

Références.

Qui ? Développeurs JAVA.

Comment ? Apprendre vite…

- Utilisation d'un IDE.
- Génération de code documenté.
- Interface graphique : parallélisme avec JAVA.
- Builder d'interface.



Introduction à PYTHON

Plan :

Objectifs.

Présentation du langage.

Quelques librairies.

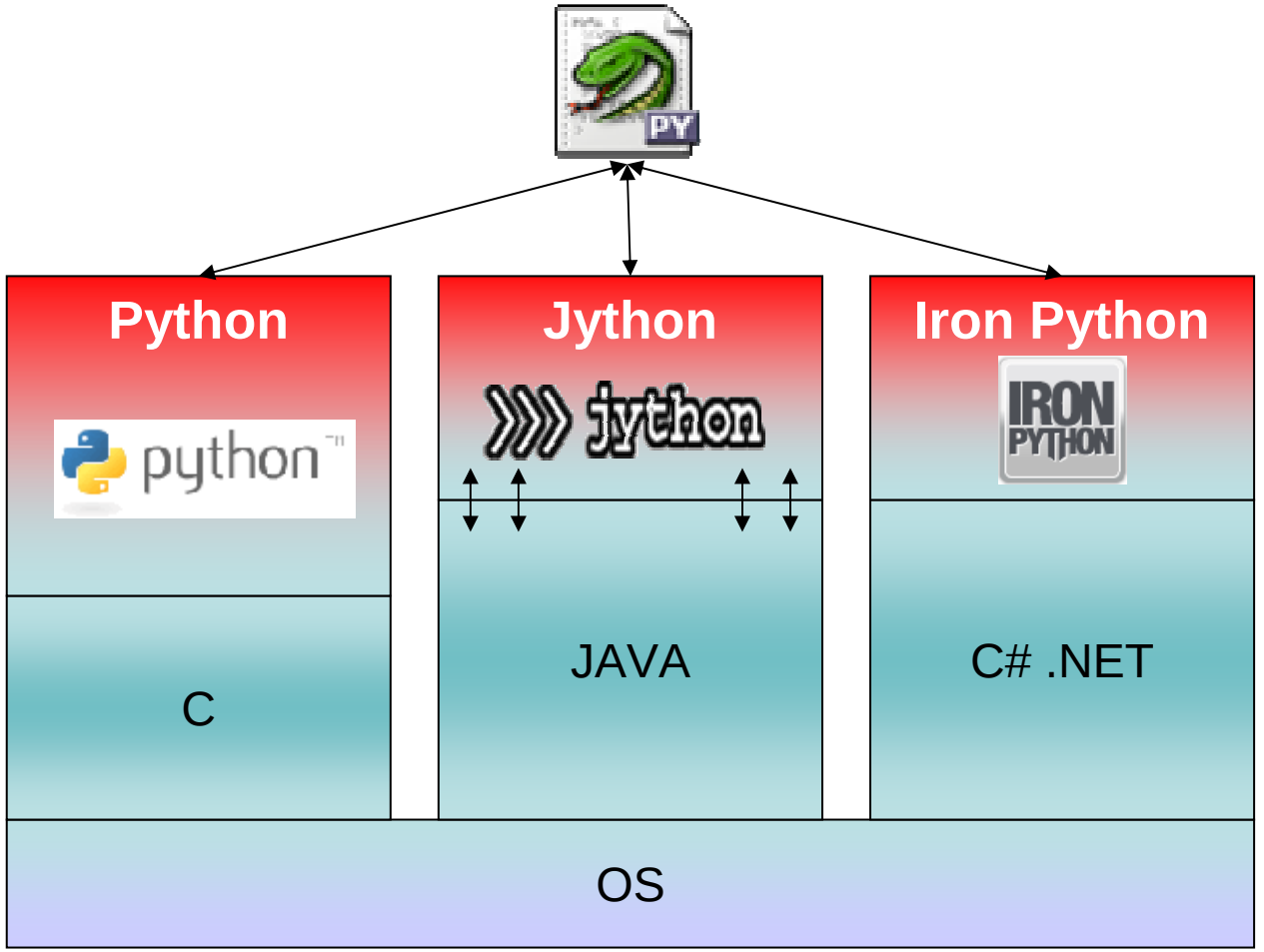
Choix d'un IDE.

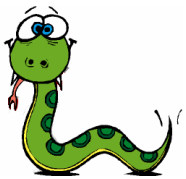
Choix d'un générateur de documentation.

Langage d'interface graphique.

Construction assistée d'IG.

Références.





Introduction à PYTHON

Plan :

Objectifs.

Présentation du langage.

Quelques librairies.

Choix d'un IDE.

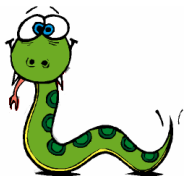
Choix d'un générateur de documentation.

Langage d'interface graphique.

Construction assistée d'IG.

Références.

- Interprété.
- Syntaxe épurée.
- POO.
- Programmation fonctionnelle.
- Introspection.
- Tests unitaires.
- Bibliothèque standard riche.



Syntaxe

Plan :

Objectifs.

Présentation du langage.

Quelques librairies.

Choix d'un IDE.

Choix d'un générateur de documentation.

Langage d'interface graphique.

Construction assistée d'IG.

Références.

- Factorielle

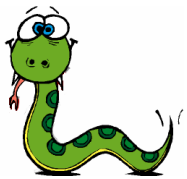
JAVA

```
int factorielle(int n) {
    if(n==0) {
        return 1;
    } else {
        return n * factorielle(n-1);
    }
}
```

Python

```
def factorielle(n):
    if n==0:
        return 1
    else :
        return n * factorielle(n-1)
```

```
def ternaire_factorielle (n):
    return n==0 and 1 or n * factorielle(n-1)
```



Syntaxe

Plan :

Objectifs.

Présentation du langage.

Quelques librairies.

Choix d'un IDE.

Choix d'un générateur de documentation.

Langage d'interface graphique.

Construction assistée d'IG.

Références.

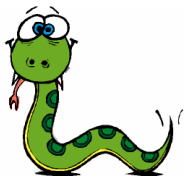
- Factorielle (bis)

Python

```
def gen_factorielle(n):
    fact = 1
    it = 1
    while(it <= n):
        yield fact
        fact = fact*(it+1)
        it = it+1
```

Point d'arrêt.
Pas avec Jython

```
liste=[n for n in gen_factorielle(4)]
print liste
>>>
[1, 2, 6, 24]
```

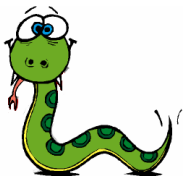


Librairies

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- En standard.
  - Tk (Interface graphique)
  - TelnetLib (Telnet très simplement)
  - …
- Additionnelles
  - PIL (Gestion des images)
  - PyGTK (Interface graphique)
  - PyXML (XML et DTD)
  - 4Suite (XML, XPath, XSLT…)
  - …
- Web
  - Mod_python pour Apache
  - Serveur d'application (ZOPE)

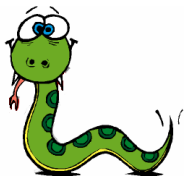


Choix d’un IDE

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d’un IDE.
Choix d’un générateur de documentation.
Langage d’interface graphique.
Construction assistée d’IG.
Références.

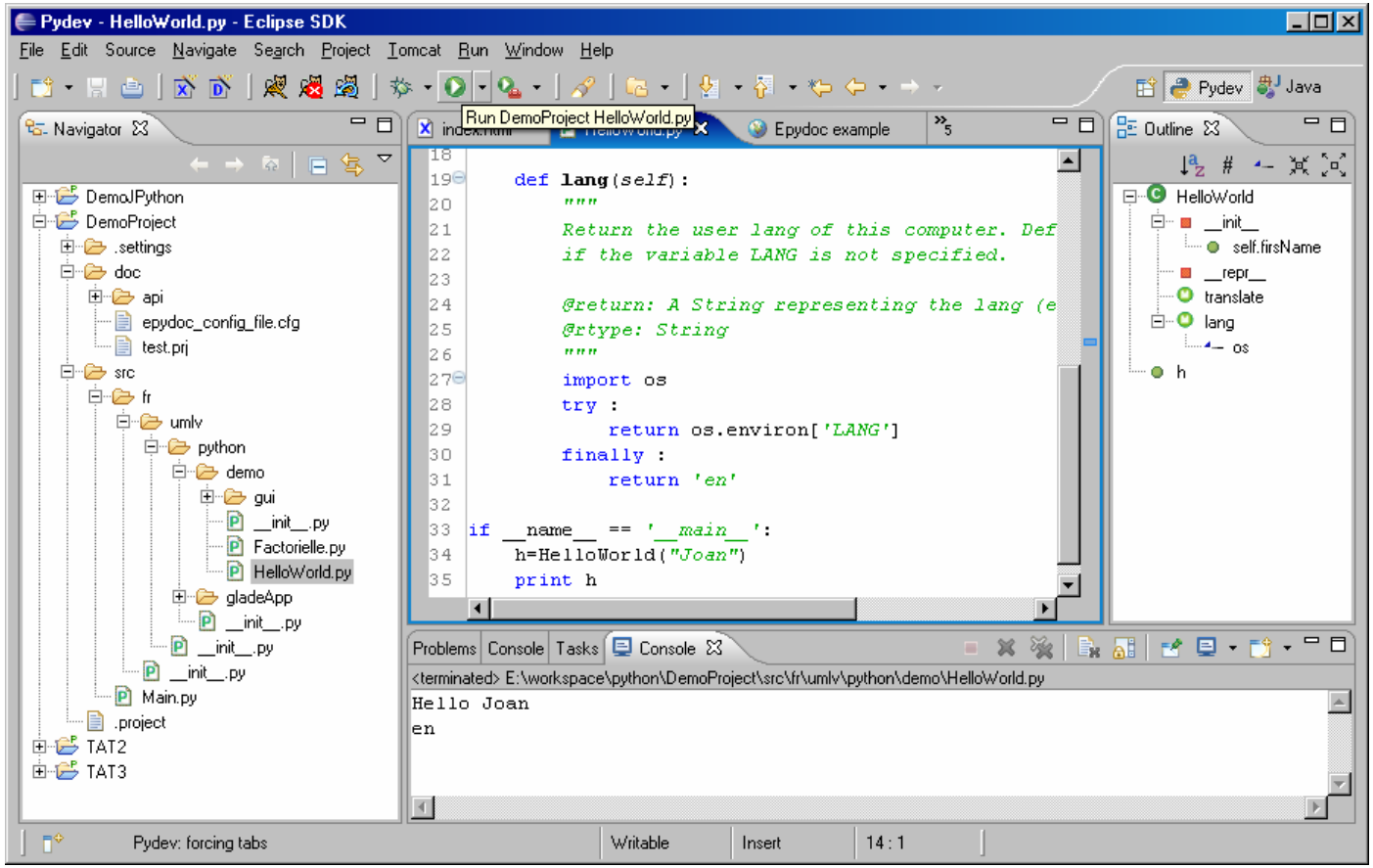
- Eclipse + PyDev
  - Auto complétion.
  - Gestion de packages.
  - Refactoring.
  - Todo.
  - …

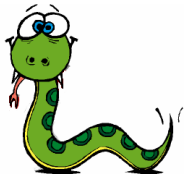


Choix d'un IDE (Eclipse & Pydev)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- **Choix d'un IDE.**
- Choix d'un générateur de documentation.
- Langage d'interface graphique.
- Construction assistée d'IG.
- Références.





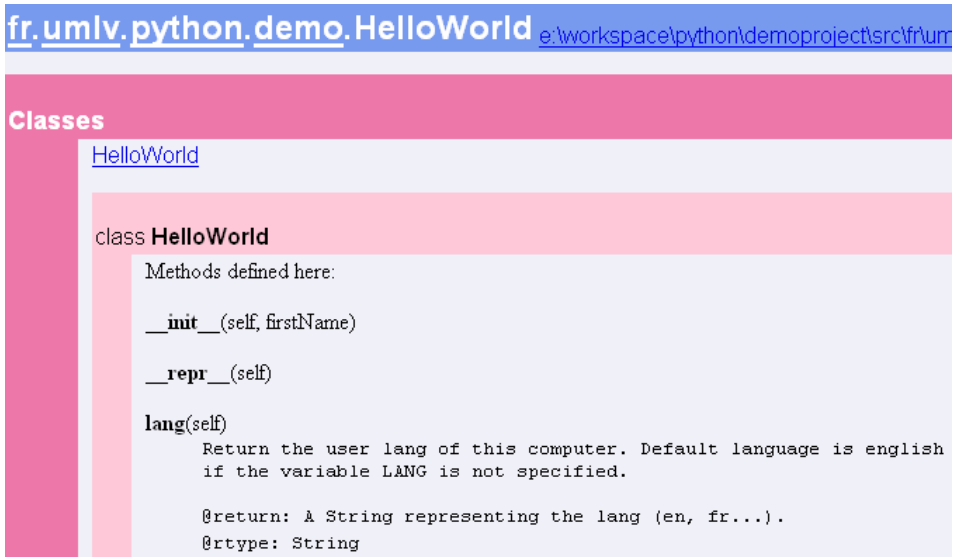
Olivier Hargoaa

Documentation

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- Plusieurs sont concurrents.
- Mon choix : Epydoc.
- Syntaxe Javadoc.
- CSS Javadoc.



fr.umlv.python.demo.HelloWorld e:\workspace\python\demoproject\src\fr\um

Classes

HelloWorld

class **HelloWorld**

Methods defined here:

__init__(self, firstName)

__repr__(self)

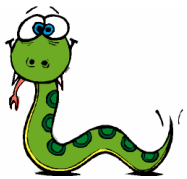
lang(self)

Return the user lang of this computer. Default language is english if the variable LANG is not specified.

@return: A String representing the lang (en, fr...).

@rtype: String

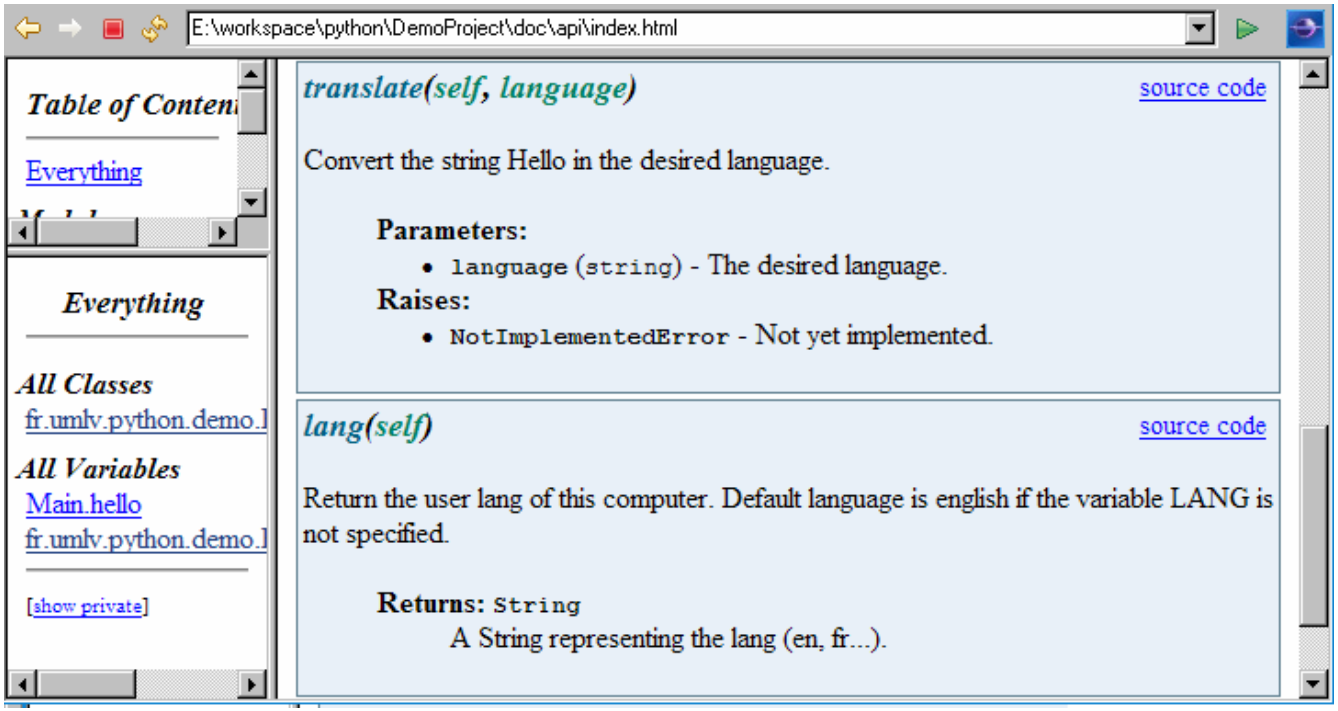
Documentation native python (pydoc)



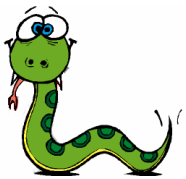
Documentation (Epydoc)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- **Choix d'un générateur de documentation**
- Langage d'interface graphique.
- Construction assistée d'IG.
- Références.



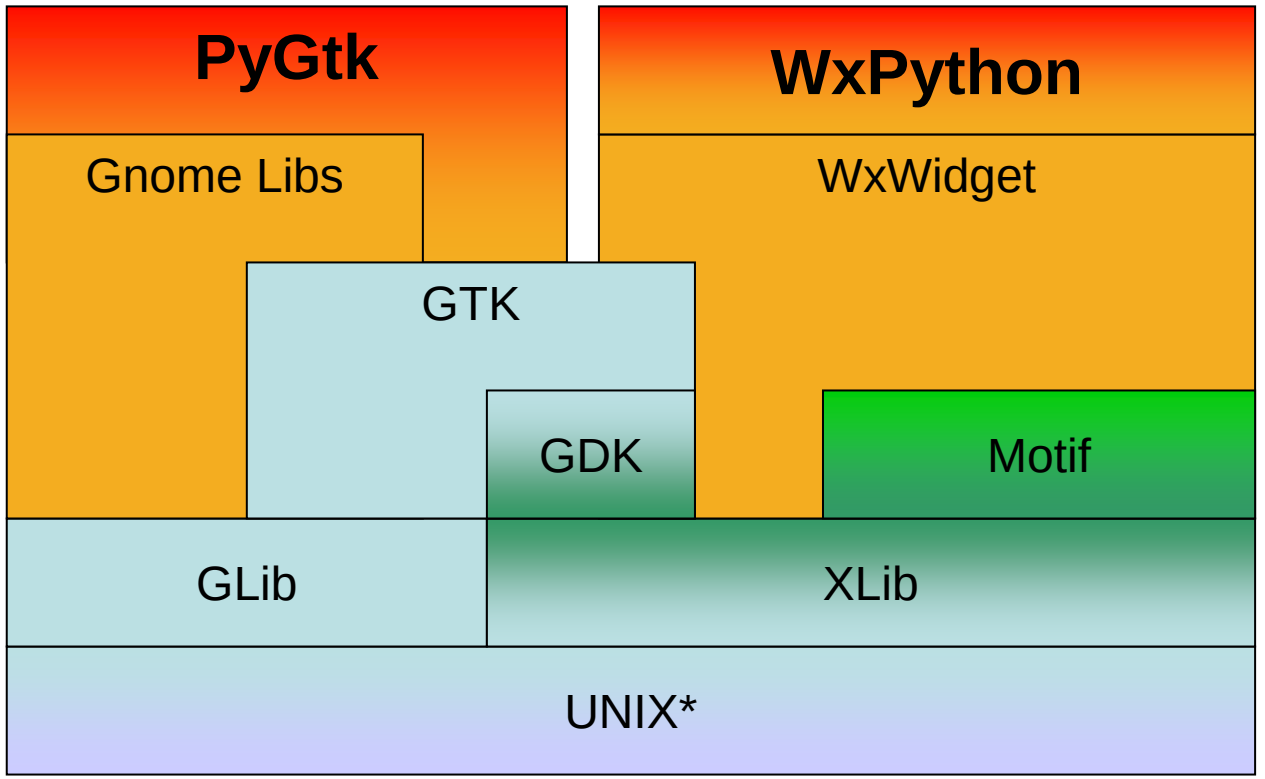

```
def translate(self, language):
    """
    Convert the string Hello in the desired language.
    @type language: string
    @param language: The desired language.
    @raise NotImplementedError: Not yet implemented.
    """
    #TODO: Translate into other language
    raise NotImplementedError("translate")
```



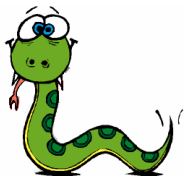
Interface Graphique (1)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- Choix d'un générateur de documentation.
- **Langage d'interface graphique.**
- Construction assistée d'IG.
- Références.



*Librairies portées sous Windows

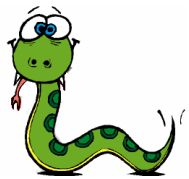


Interface Graphique (2)

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

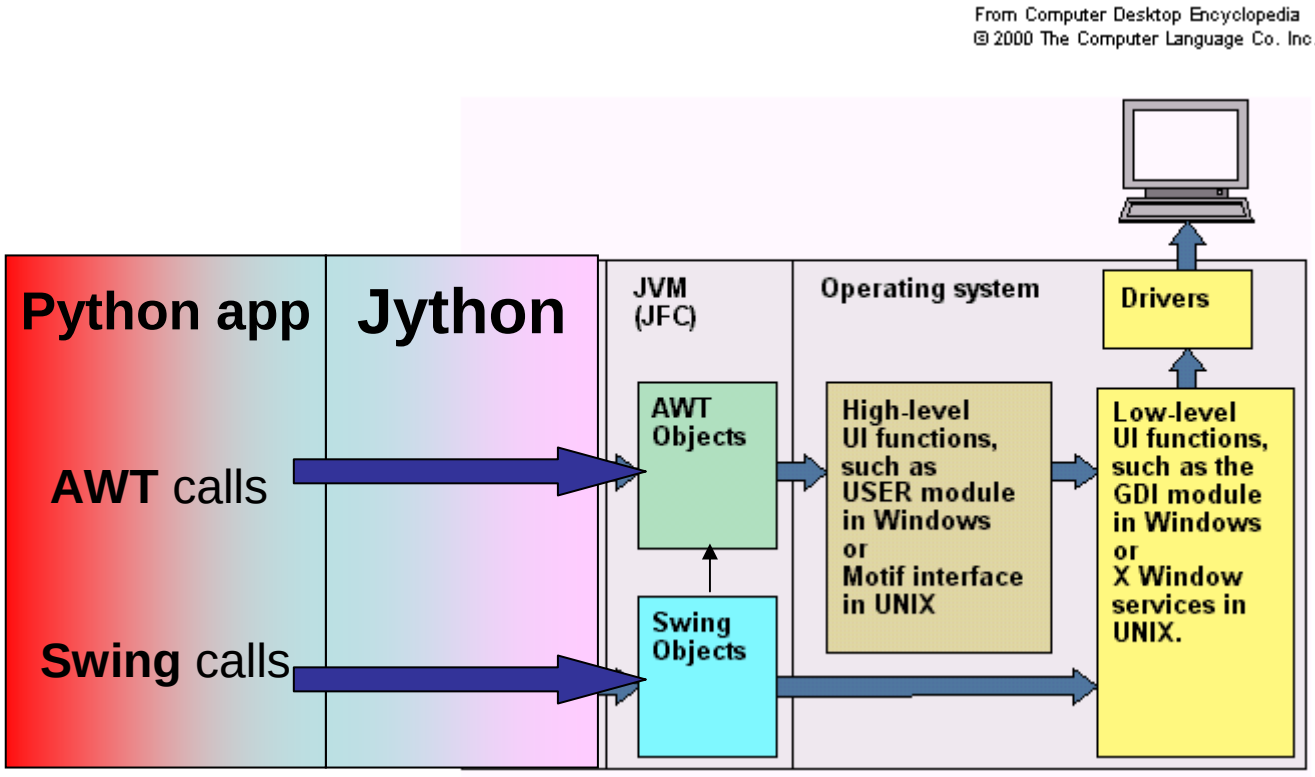
- Choix GTK -> natif gnome
- PyGtk
- Bons tutoriels
- Architecture proche de JAVA Swing
- Beaucoup plus simple (rapide)

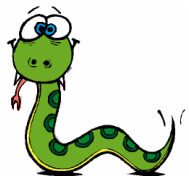


Interface Graphique (3)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- Choix d'un générateur de documentation.
- **Langage d'interface graphique.**
- Construction assistée d'IG.
- Références.



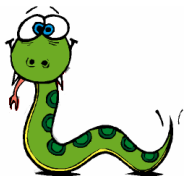


Interface Graphique (4)

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- IG avec Jython
  - Composants JAVA
    - Swing
    - AWT
  - Syntaxe Python
    - Simplicité
  - Métier
    - Python
    - Compact
    - Développement rapide



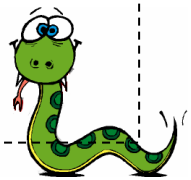
Interface Graphique (5)

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- Hello world Python

```
import gtk, pygtk
import sys

class HelloWorldGUI:
    def __init__(self):
        self.__draw()
    def __draw(self):
        self.window=gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.set_title("Hello world")
        self.window.connect("destroy", self.__on_quit)
        self.button=gtk.Button(stock=gtk.STOCK_CONNECT)
        self.button.connect("clicked", lambda w: sys.stdout.write("Connect python\n"))
        self.window.add(self.button)
        self.window.resize(200, 100)
    def __on_quit(self, widget):
        gtk.main_quit()
    def start(self):
        self.window.show_all()
        gtk.main()

if __name__ == '__main__':
    h=HelloWorldGUI()
    h.start()
```



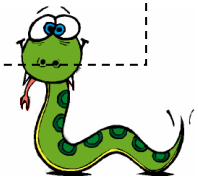
Interface Graphique (6)

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- Hello world Jython

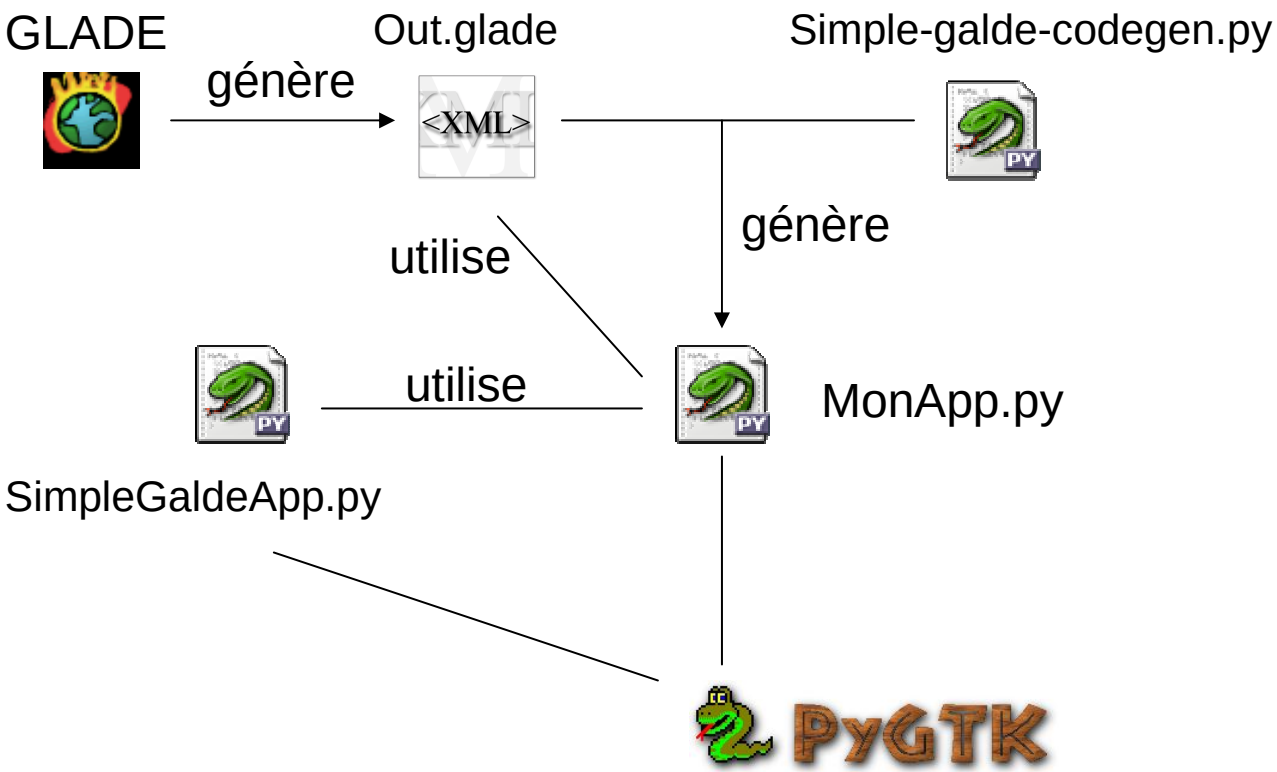
```
from java import awt
from javax import swing
import sys

class HelloWorldGUIJPY:
    def __init__(self):
        self.__draw()
    def __draw(self):
        self.window=swing.JFrame("Hello world Jython")
        self.window.setDefaultCloseOperation(swing.JFrame.EXIT_ON_CLOSE)
        self.button=swing.JButton("Connect", actionPerformed = self.do_connect)
        self.window.setSize(200, 100)
        self.window.add(self.button)
    def do_connect(self, event=None):
        print "Connect Java Python"
    def start(self):
        self.window.setVisible(1)

if __name__ == '__main__':
    h=HelloWorldGUIJPY()
    h.start()
```

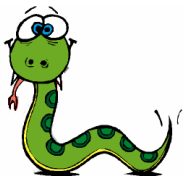


Builder d'interface (GLADE)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- Choix d'un générateur de documentation.
- Langage d'interface graphique.
- **Construction assistée d'IG.**
- Références.





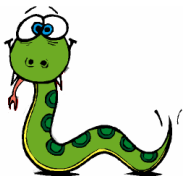
Builder d'interface (GLADE)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- Choix d'un générateur de documentation.
- Langage d'interface graphique.
- **Construction assistée d'IG.**
- Références.

XML généré avec GLADE

```
<?xml version="1.0" standalone="no"?> <!--*- mode: xml -*-->
<!DOCTYPE glade-interface SYSTEM "http://glade.gnome.org/glade-2.0.dtd">
<glade-interface>

<widget class="GtkWindow" id="window1">
  <property name="title" translatable="yes">Hello world with glade</property>
  <property name="visible">True</property>
  <property name="default_width">200</property>
  <property name="default_height">100</property>
  <property name="resizable">True</property>
  <signal name="destroy" handler="gtk_main_quit"/>

  <child>
    <widget class="GtkButton" id="button1">
      <property name="visible">True</property>
      <property name="label">gtk-connect</property>
      <signal name="clicked" handler="on_buttonconnect_clicked"/>
    </widget>
  </child>
</widget>

</glade-interface>
```



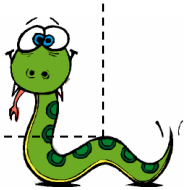
Builder d'interface (GLADE)

Plan :

- Objectifs.
- Présentation du langage.
- Quelques librairies.
- Choix d'un IDE.
- Choix d'un générateur de documentation.
- Langage d'interface graphique.
- **Construction assistée d'IG.**
- Références.

Script python généré avec simple-galde-codegen.py

```
import os
import gtk
from SimpleGladeApp import SimpleGladeApp

glade_dir = ""

class Window1(SimpleGladeApp):
    def __init__(self, path="helloworld.glade", root="window1", domain=None, **kwargs):
        path = os.path.join(glade_dir, path)
        SimpleGladeApp.__init__(self, path, root, domain, **kwargs)
    def new(self):
        #context Window1.new {
        print "A new Window1 has been created"
        #context Window1.new }
    def on_buttonconnect_clicked(self, widget, *args):
        #context Window1.on_buttonconnect_clicked {
        print "Connect from glade hello world"
        #context Window1.on_buttonconnect_clicked }
def main():
    window1 = Window1()
    window1.run()
if __name__ == "__main__":
    main()
```



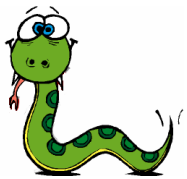
Bilan

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

- Pourquoi Python :
  - 2-10X plus court que JAVA.
  - Facile à apprendre.
  - Eq. Java, PHP.
  - Librairies (PIL, 4Suite, PyGTK, PyXML, Tk…).
  - Vrai langage objet.
  - Existe un serveur d'application (ZOPE).

From makina-corpus.org



Références

Plan :

Objectifs.
Présentation du langage.
Quelques librairies.
Choix d'un IDE.
Choix d'un générateur de documentation.
Langage d'interface graphique.
Construction assistée d'IG.
Références.

Jython :

- http://www.jython.org/Project/index.html
- http://www.oreilly.com/catalog/lpython/chapter/ch10_fel.html
- http://jython.sourceforge.net/docs/differences.html

Python :

- http://4suite.org/index.xhtml
- http://python.developpez.com/outils/PythonZope/

Linux :

- http://ref.web.cern.ch/ref/CERN/CNL/2000/001/kde-gnome2

JAVA :

- http://www.pcmag.com/encyclopedia_term/0,2542,t=AWT&i=38308,00.asp

IDE :

- http://pydev.sourceforge.net/

Documentation :

- http://epydoc.sourceforge.net/

GTK :

- http://www.pygtk.org/
- http://www.pygtk.org/tutorial.html
- http://wikipython.flibuste.net/moin.py/PyGTKhttp://wikipython.flibuste.net/moin.py/PyGTK