Guide de démarrage pour Boa Constructor

Installation et présentation de Boa constructor

Boa s'exécute sous Windows ou sous UNIX avec GTK+.

Téléchargement : http://sourceforge.net/projects/boa-constructor/files/Boa-Constructor/

Pré-requis

Il faut avoir installé :

- une version récente de **Python**
- sous Linux :
  - **GTK+**
  - **wxWidgets** (version pour GTK+ ; il existe plusieurs versions, par exemple la version pour Microsoft Windows)
  - **wxPython** (version correspondant à celle de wxWidgets, c'est à dire 2.5.x pour wxWidgets 2.5.x.), instructions détaillées : voir http://wxPython.org).
  - éditeur Scintilla (stc) et composants Open GL (OGL) de wxWidgets

Le programme d'installation standard pour Windows installe les bibliothèques **wxWidgets** et l'environnement **wxPython**.

Installation

Sous Windows

Deux possibilités :

- Lancer l'installateur de Boa (fichier se terminant par “win32.exe”)
- ou à partir du fichier source zip de Boa : l'extraire dans un répertoire (par exemple ..\site-packages\boa) et créer sur le bureau un raccourci vers le fichier boa.py. Pour lancer Boa, double-cliquer sur ce raccourci.

Sous UNIX

- installer le paquet **Boa**
- ou à partir des sources téléchargés sous forme de fichier zip :
  - créer un répertoire pour Boa par exemple /usr/local/boa, y extraire les sources de Boa
  - créer sur le path un lien symbolique vers boa.py par exemple,

```
ln -s /usr/local/boa/boa.py /usr/bin
```

- Dans le shell, lancer boa avec la commande

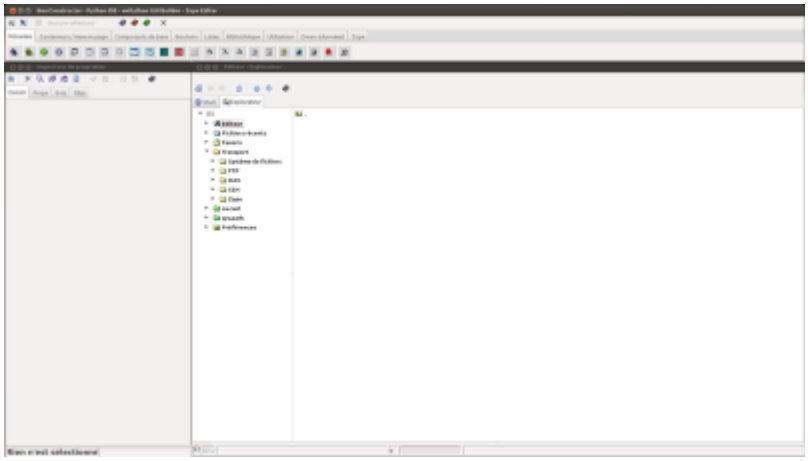
```
boa.py
```

A partir de CVS

Voir le guide d'installation de Boa http://wiki.wxpython.org/BoaInstallGuide sur la page wiki de wxPython.

Présentation de l'EDI Boa

Quand on le lance, l'EDI affiche ceci :



Il comporte trois fenêtres :

en haut, la **palette** avec :
- Une barre d'outils pour ouvrir ou fermer les fenêtres des applications
- Plusieurs onglets
  - **Nouveau** - créer de nouveaux modules, des applications etc.
  - **Conteneurs/ Mise en page**, **Composants de base** etc. donnent accès aux différents composants

2. en bas à gauche, l'**inspecteur de propriétés**
   - Cette section permet aussi d'afficher par exemple le Debugger ou le Class Browser
   - Elle sert à modifier les propriétés des composants et à définir des événements.
3. en bas à droite, l'**éditeur - Explorateur**, avec plusieurs plusieurs onglets :
   - **Shell** - donne accès à un interpréteur Python, ce qui facilite et rend confortable le test de petites sections de code.

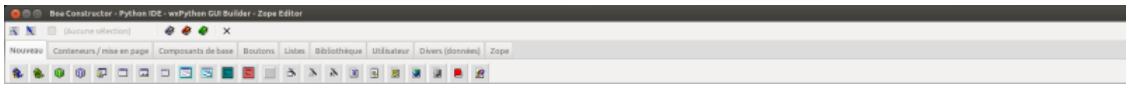
- **Explorateur** - permet de naviguer dans le système de fichiers, donne accès à la démo wxPython (via un plug-in), permet de modifier les préférences de Boa, et il y aura des onglets supplémentaires, un pour chaque fichier sur lequel on travaille.

La fenêtre de la palette

C'est la fenêtre principale du programme Boa : sa fermeture termine le programme.

Elle se compose de deux parties :

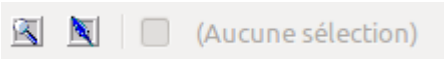
- une barre d'outils
- et un ensemble d'onglets



La barre d'outils et ses icônes

La barre d'outils permet d'accéder aux autres fenêtres de Boa. En survolant les différentes icônes avec la souris, des bulles d'aide expliquent ce que font les boutons.

La barre d'outils comporte 3 boutons sur la gauche, concernant l'**application** :



Le premier met la fenêtre **Inspecteur** à l'avant-plan.

Le second met la fenêtre de l'**éditeur** à l'avant-plan.

La case à cocher indique le nom du composant sélectionné (**Aucune sélection** si rien n'est sélectionné)

Les 3 boutons suivants concernent l'**aide** :

aide de Boa ou aide pour le composant sélectionné

aide de wxPython

aide de Python

Les onglets

Les onglets fournissent des composants regroupés logiquement, pour construire les applications.

Les composants du premier onglet (**Nouveau**) permettent de créer de nouveaux fichiers : une application Python, un module Python, un fichier de configuration, une

application de Boa ou autre.

La nouvelle fonctionnalité **modèle de code** permet d'insérer du code, par exemple pour une boîte de dialogue. A l'endroit du où un dialogue doit être inséré, presser Alt+T et sélectionner dans la liste déroulante.

L'icône  permet de créer une nouvelle application Boa (sa bulle d'aide indique **wx.App**, en référence à sa classe de base). La nouvelle application se compose du fichier source de l'application (**App1.py**) et du cadre initial (**Frame1.py**).

Le fichier d'application est le point de départ de l'application. Par défaut, la nouvelle application se contente de charger le formulaire, qui est blanc. On peut lui ajouter des composants.

Boa ouvre le code source de la nouvelle application et le nouveau formulaire dans la fenêtre de l'éditeur.

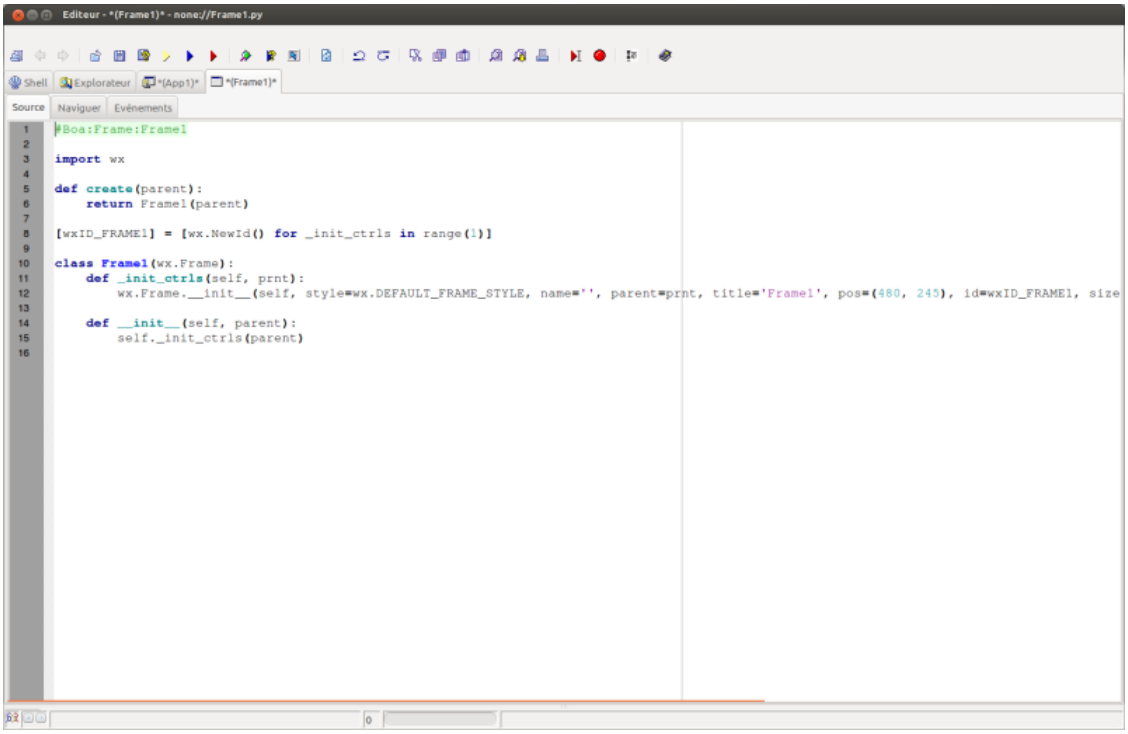
La fenêtre de l'éditeur

C'est l'une des trois fenêtres de Boa. Si on la ferme, on peut la rouvrir avec le bouton  de la palette.

L'éditeur comprend deux onglets : **Shell** et **Explorateur**.

Pour ouvrir un fichier source dans l'éditeur de Boa, on peut :

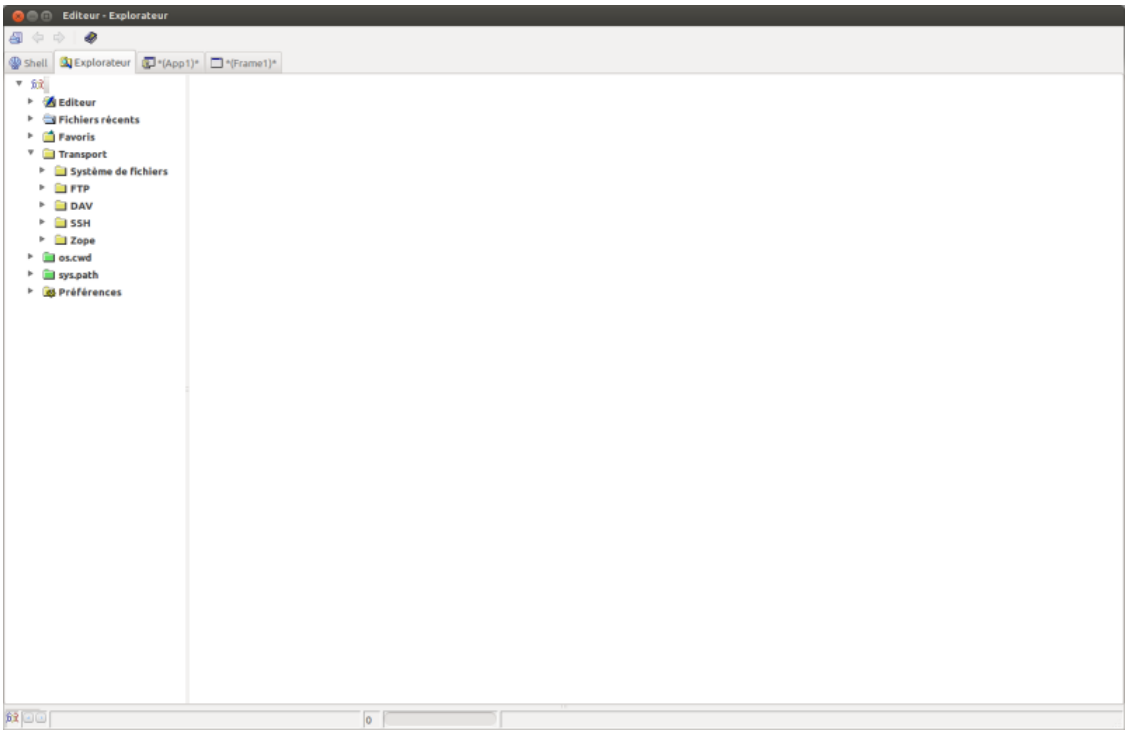
- cliquer sur le bouton **Ouvrir un module** de la barre d'outils de l'éditeur
- sélectionner le menu **Fichier → Ouvrir** (ou raccourci Ctrl+O)
- Glisser-déposer un fichier sur l'éditeur.
- utiliser la fenêtre Inspecteur de propriétés
- utiliser un fichier application
- lancer Boa en ligne de commande avec en paramètre le fichier source.
- utiliser l'une des options de l'onglet **Nouveau** dans la palette.



Quand on ouvre un fichier, l'éditeur crée une nouvelle page pour le module, avec plusieurs onglets qui présentent plusieurs onglets du fichier, selon le type de fichier. Ici : **App1** et **Frame1**. Pour **Frame1**, trois onglets : **Source**, **Naviguer** et **Événements**, Pour **App1**, trois onglets **Application**, **Source** et **Naviguer**.

<note>Les astérisques sur les onglets **App1** et **Frame1** indiquent que des modifications ont été apportées mais ne sont pas sauvegardées.</note>

L'onglet **source** (ci-dessus) montre le code source et permet de le modifier, sauf si l'éditeur graphique est ouvert ou actif pour ce fichier (dans ce cas, le code source a un fond bleuâtre).

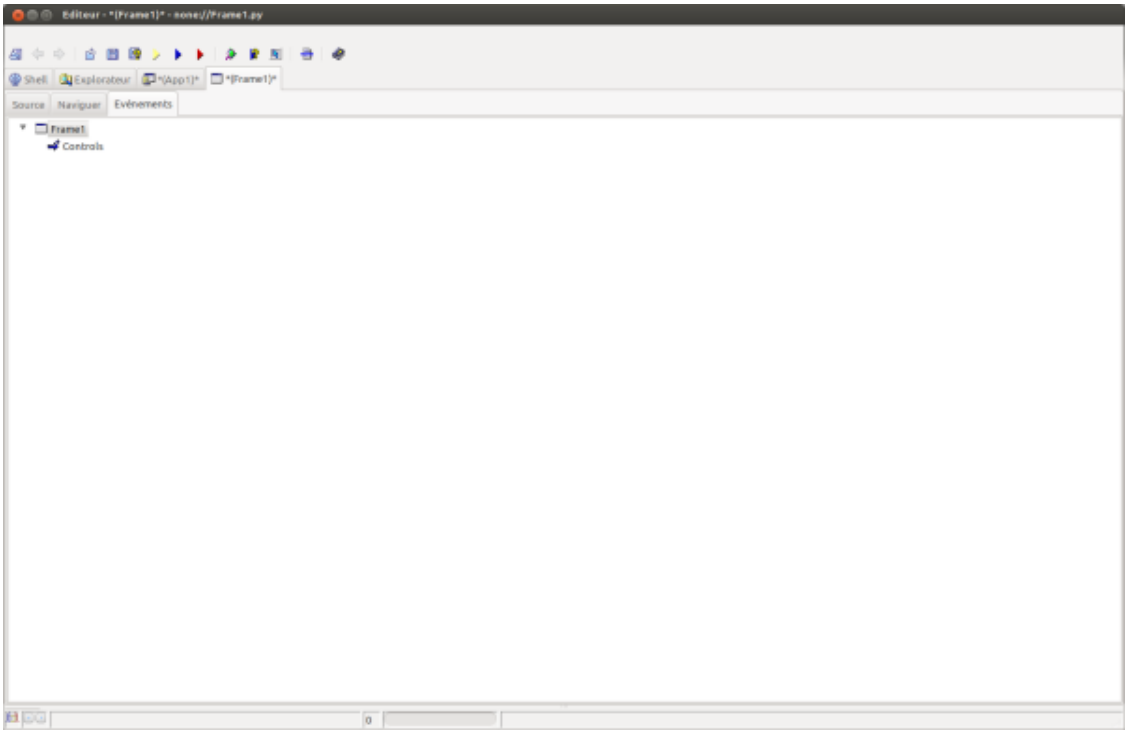


L'onglet **explorateur** fournit une vue arborescente du module avec les icônes

suivantes :

- Classes
- Méthodes
- Fonctions
- Événements
- Global

L'onglet **explorateur** présente une vue du source basée sur les classes. Les méthodes et attributs de chaque classe dans le source sont affichés. Un double-clic sur un attribut ou une méthode affiche dans l'onglet **Source** le code source pour ce composant.



L'onglet **Événements** montre une vue basée sur le contrôle des événements définis dans l'inspecteur de Boa. Les événements ajoutés manuellement (en dehors du code généré par Boa) ne s'affichent pas ici.

Des affichages supplémentaires, comme **A faire**, **Documentation**, **UML** etc, peuvent être ouverts via l'Éditeur ou le menu **Affichage** ; certains affichages ne sont disponibles que pour certains types de fichiers.

Le menu **Affichage** de l'éditeur fournit d'autres vues sur la source : **Hiérarchie**, **Documentation**, **A faire**, Application ToDo, Imports, **UML**, CVS conflicts, Readme.txt, Changes.txt, Todo.txt, Bugs.txt.

- La vue **Hiérarchie** montre les classes du source comme une hiérarchie. Les relations d'héritage entre les classes sont clairement indiquées.
- La vue **Documentation** montre la documentation produite automatiquement à partir du code source par les chaînes de documentation standard de Python

pour les classes et méthodes (c'est à dire une chaîne immédiatement après la déclaration).

- La vue **A faire** sert au suivi des listes de tâches à faire. On ajoute des items *A faire* au code (en commentaire, suivis de trois caractères X, par exemple

```
# XXX My todo item
```

). C'est utile pour suivre plus tard des items.

- La **vue Application ToDo**, disponible seulement pour un fichier de type wx.App, montrera tous les fichiers du projet avec le nombre de todos par fichier.
- La **vue UML** montre les relations entre les classes.
- La **vue Imports** montre les relations entre les modules.
- La **vue Diff with:** est créée en sélectionnant l'option de menu File/NDiff files, elle compare le fichier déjà ouvert avec celui sélectionné.
- Les autres Vues devraient être évidentes.

Les icônes disponibles sur la barre d'outils de la fenêtre de l'éditeur pour un fichier wxFrame sont :

<note>Les icônes disponibles sur cette barre d'outils changent en fonction du fichier actif.</note>

Les icônes ci-dessus permettent d'ouvrir, fermer, enregistrer, enregistrer-sous et parcourir en avant / en arrière pour sauter à des positions marquées (CTRL+M).

Les icônes ci-dessus permettent d'exécuter l'application, lancer le module ou déboguer l'application.

Les icônes ci-dessus permettent de profiler le module, vérifier le source, ou démarrer le cadre éditeur graphique (parfois aussi appelé GUI Editor).

Les icônes ci-dessus permettent de rafraîchir l'écran, Annuler, Rétablir, Couper, Copier et Coller.

Les icônes ci-dessus permettent de rechercher / remplacer, rechercher à nouveau et d'imprimer le source.

Les icônes ci-dessus permettent d'exécuter jusqu'au curseur (dans le débogueur), mettre ou retirer (basculer) un point d'arrêt (on peut aussi utiliser la touche F5), insérer des informations du module (Nom de l'auteur, etc, ce qui peut être personnalisé, voir Module Info), et aider.

Quelques raccourcis utiles :

- CTRL+Space met en œuvre la complétion de code, par exemple après la saisie “wx.F” presser CTRL+Space
- CTRL+SHIFT+Space donne des conseils, par exemple après la saisie “wx.Frame”, pressser CTRL+SHIFT+Space
- ALT+O (uniquement disponible avec le plug-in ErrOutShortcut) montre le notebook avec les tracebacks, sorties, les erreurs et onglets Tâches ou s'il est visible, il le cache.

La fenêtre de l'éditeur graphique

On accède à la fenêtre de l'Éditeur graphique par le bouton Éditeur graphique sur la barre d'outils de la fenêtre de l'éditeur. L'éditeur graphique ne peut être démarré que si le fichier actif est un formulaire (wx.Panel, wx.Frame etc) ou un fichier de type Dialog.

L'éditeur graphique affiche le cadre sur l'écran. L'éditeur graphique affiche le cadre tel qu'il apparaîtra à l'exécution. Ce cadre est désigné comme l'**Éditeur graphique**. L'éditeur graphique crée également deux nouvelles pages dans l'éditeur. Le premier est la vue **Données**, le second est la vue **Sizer**.

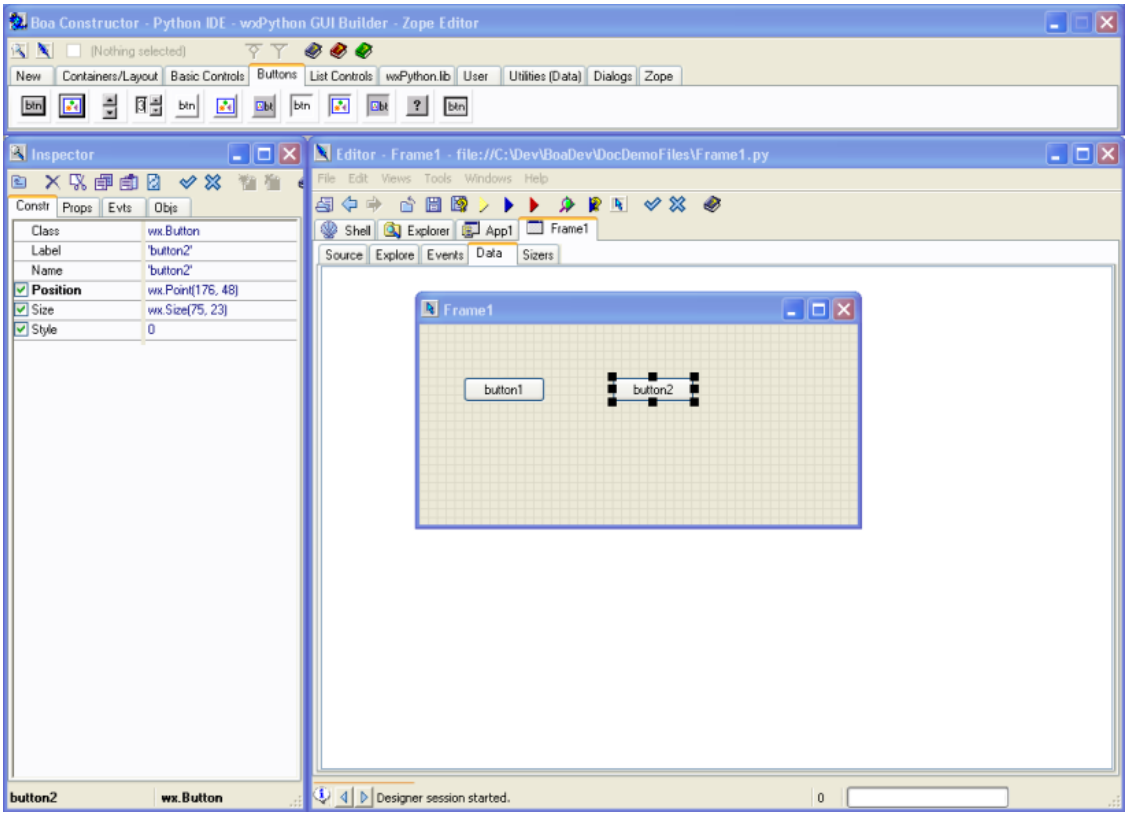
L'**Éditeur graphique** sert à concevoir la présentation du cadre ou du dialogue. On peut placer des composants dans le volet, les redimensionner, les déplacer ou les supprimer. On peut aussi placer des composants dans d'autres composants.

Pour créer un composant, sélectionner le composant approprié dans la palette. Les composants sont regroupés pour un accès facile, par exemple les commandes de base sont regroupées. Une fois un contrôle sélectionné, la barre d'état dans la palette montre le contrôle sélectionné.

Pour placer le composant déplacer le curseur sur l'Éditeur graphique. Un clic de souris à l'endroit où doit être inséré le composant. Une fois le composant sur le formulaire, il peut être déplacé et redimensionné. On le déplace en plaçant la souris à dans le composant et en le faisant glisser. On le redimensionne en faisant glisser l'un des huit marqueurs sur le bord d'un contrôle sélectionné.

Les modifications apportées dans l'éditeur graphique sont enregistrées dans le code source. Les modifications sont enregistrées quand on clique sur le bouton Post . Il ya deux boutons Post, un sur la barre d'outils de l'éditeur et l'autre sur la barre d'outils de l'inspecteur. Un clic sur l'un ou l'autre ferme l'éditeur graphique et la vue Données et génère le code source pour les changements. Les modifications peuvent aussi être postées en fermant le cadre.

Pour annuler toutes les modifications effectuées depuis l'ouverture de l'éditeur graphique, cliquer sur le bouton Annuler sur la barre d'outils de l'éditeur ou de l'inspecteur.

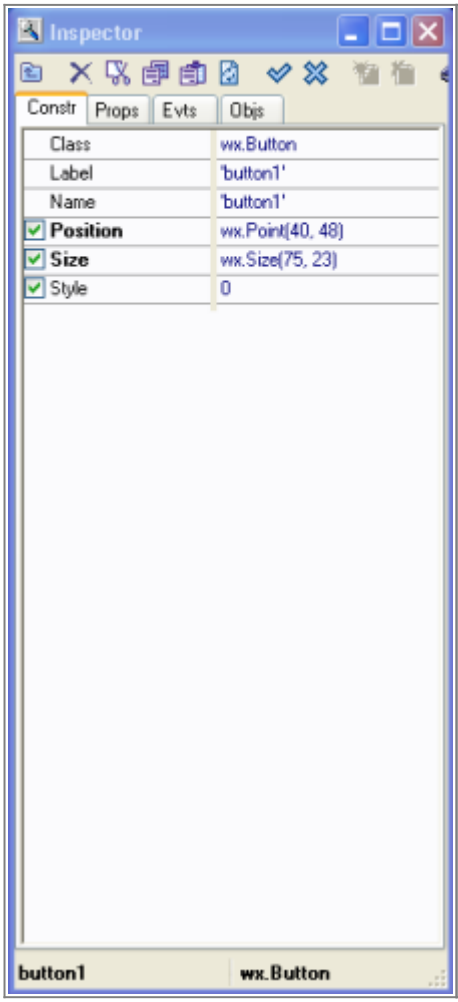


La fenêtre intitulée Frame1 ci-dessus est la fenêtre graphique de l'éditeur d'images, un wx.Panel (trouvé sur la palette Conteneurs / mise en page) a été placé dans le wx.Frame et dans ce dernier deux contrôles wx.Button (trouvé sur la palette Boutons). Le contrôle button2 est actuellement sélectionné et on peut pouvez voir ses propriétés dans la fenêtre Inspecteur de propriétés.

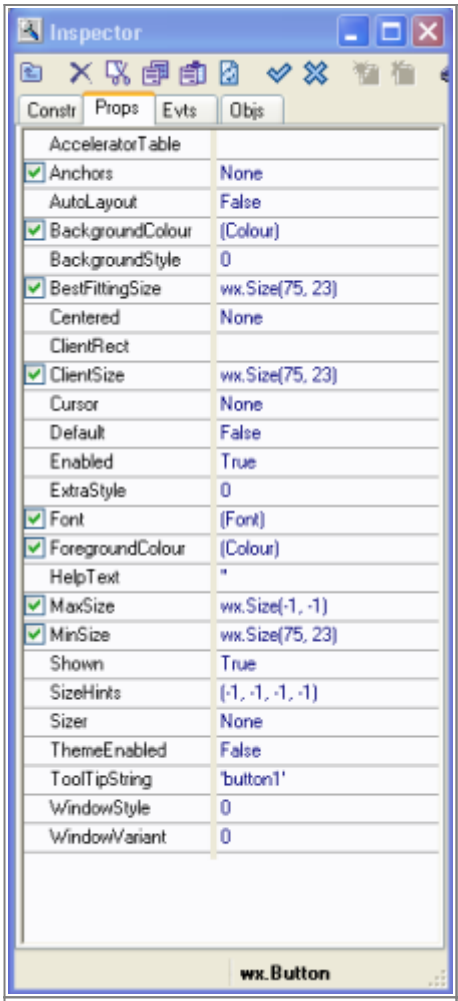
<note>Vous pourriez déposer les contrôles de l'onglet Divers (données), sur l'onglet Affichage des données et pour tout conteneur de type sizer (onglet conteneurs / Mise en page), sur la vue Sizer.</note>

La fenêtre Inspecteur de propriétés

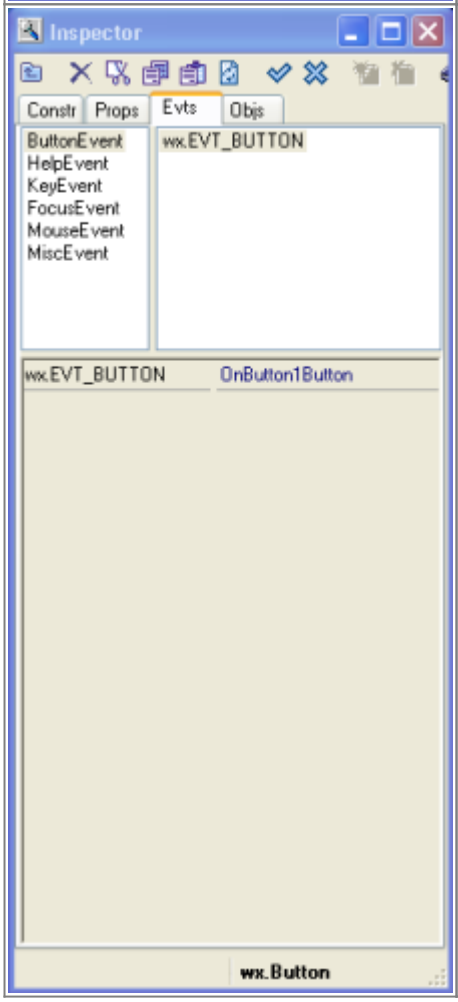
L'Inspecteur affiche toujours la configuration du composant actuellement sélectionné. L'Inspecteur contient 4 pages, la page du constructeur ('Constr'), la page Propriétés ('Props'), la page Événements ('Evts') et la Page objets ('Objs').



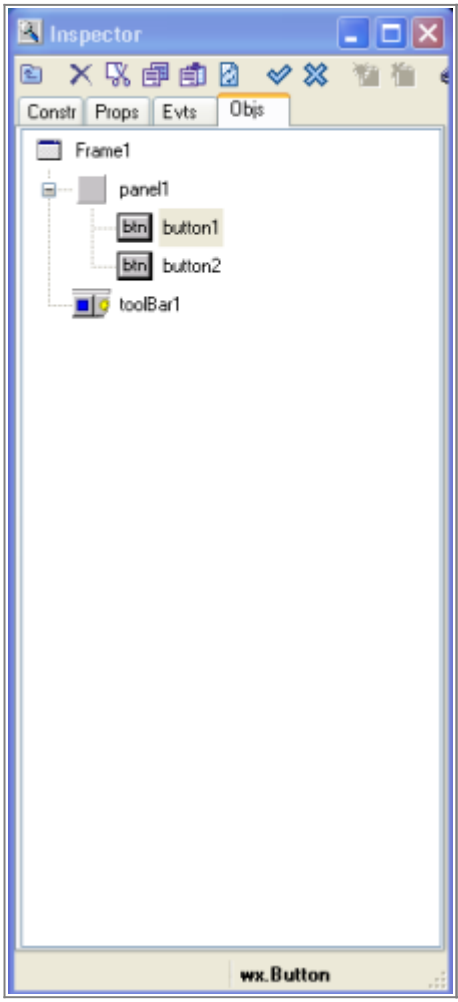
La page **constructeur** permet d'éditer les propriétés nécessaires au moment de la construction de l'objet, par exemple, le nom et le style du composant.
Voir l'aide de wxPython pour les valeurs passées au constructeur pour un contrôle. Les styles dans le manuel sont d'une importance particulière.
Noter que les modifications apportées à certains paramètres du constructeur ne prennent effet que lorsque le contrôle est créé.
Modifier la propriété “Nom” en quelque chose qui gardera donner un sens pour quelqu'un d'autre ou pour vous dans 3 mois !



La page **Propriétés** donne un contrôle fin pour toutes les propriétés de ce composant, par exemple, polices et couleurs.



La **page Événements** permet de sélectionner les événements à gérer dans le code. Les événements sont regroupés en groupes logiques. La création d'un événement est facile, il suffit de cliquer par exemple sur “ButtonEvent” et de cliquer sur “wx.EVT_BUTTON” et Boa va créer un événement “OnButton” appelé “OnControlNameButton”, Évidemment, il faut ajouter au code source généré les actions à lancer lorsque l'utilisateur appuie sur ce bouton.

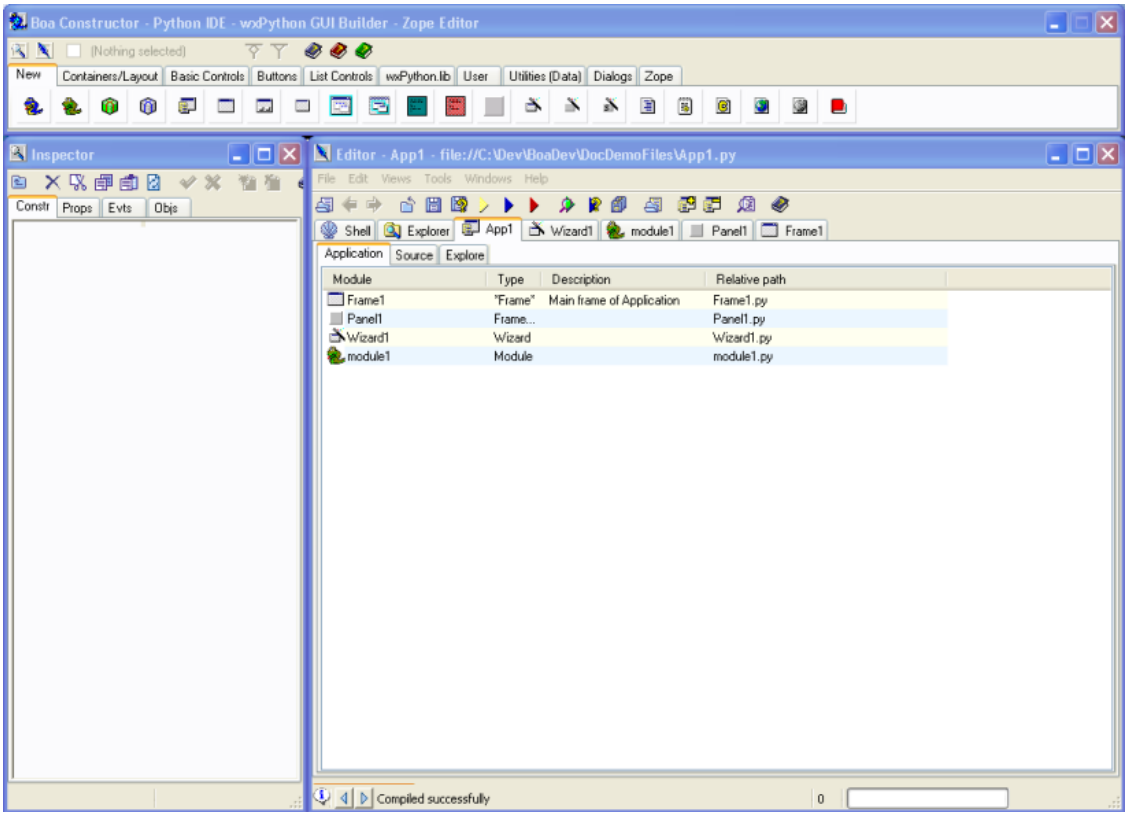


La page **Objets** permet de naviguer à travers les composants dans un autre format pour le volet de l'éditeur graphique. C'est particulièrement utile pour des éléments invisibles ou qui se chevauchent. Certains composants comme le wxStatusBar ne traitent pas les événements de clic (sur MSW) : il faut donc les choisir dans la page des Objets.

Gestion de l'application

Quand le module en cours d'édition est une application, l'éditeur offre une vue spéciale de l'application. Cette vue application permet de suivre facilement les fichiers dans l'application et d'ajouter de nouveaux modules, dialogues et autres types de fichiers à la demande.

Lors de l'affichage d'une application (l'une de ses vues), on peut sélectionner un élément du volet «Nouveau» dans la palette et il sera ajouté à l'Application.



La première des icônes ci-dessus permet d'ajouter un fichier existant à cette application.

La seconde permet de supprimer un fichier de l'application.

<note>Le Panel1 répertorié comme un module ci-dessus n'est pas le même que le Panel1 contenu dans Frame1 !</note>

A propos de l'utilisation de l'explorateur

La deuxième page de l'éditeur notebook contient une page appelée l'Explorateur. On utilise cette page pour trouver les différentes sources de données, y compris le système de fichiers, CVS, Zope, WebDAV, comptes SSH, FTP.

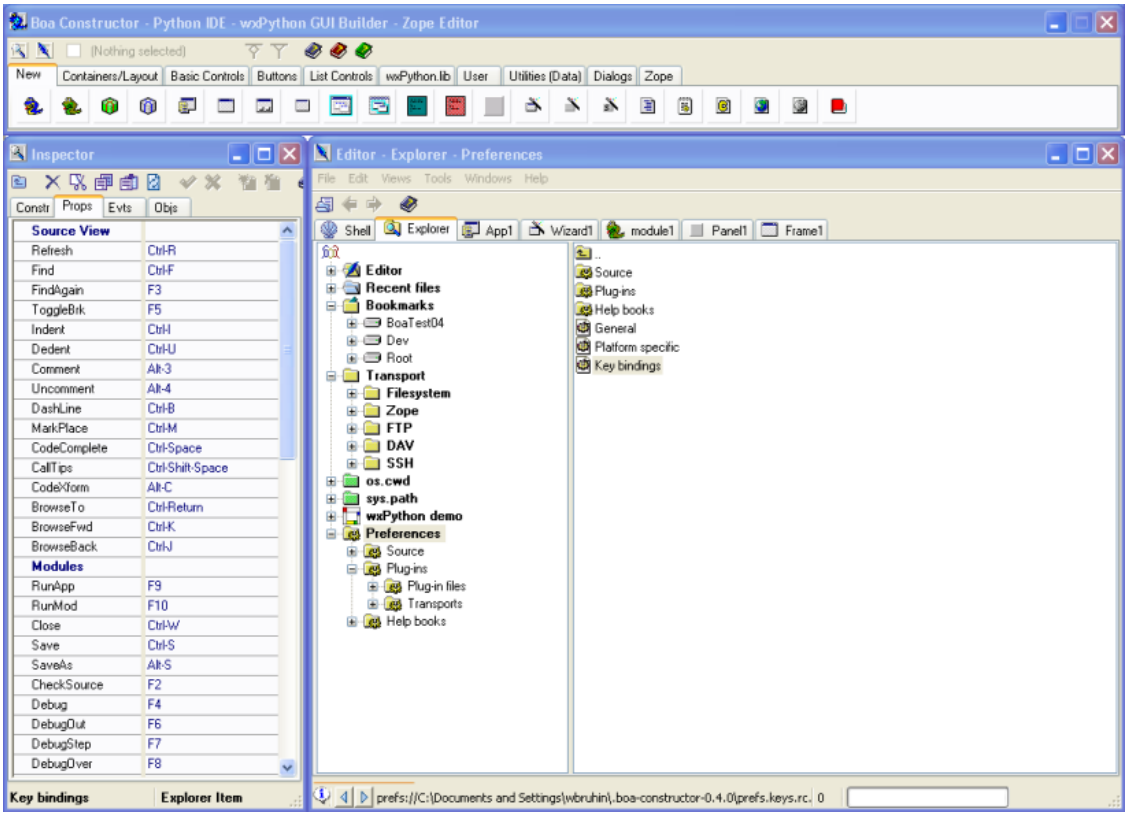
L'Explorateur affiche les systèmes de fichiers et répertoires du système d'exploitation. A la première exécution de Boa ces systèmes de fichiers et répertoires de développement sont ajoutés. Vous pouvez également reconfigurer où chercher voir Setting Preferences.

Les répertoires python sont déduits de la variable d'environnement PYTHONPATH et les emplacements par défaut tels que compilés dans le système d'exécution Python.

Les répertoires utilises souvent peuvent être ajoutés à la section des signets en les sélectionnant et en cliquant sur l'outil Signets  ou en utilisant le menu Edition ou par un clic droit de souris sur le répertoire. Noter que cet item va s'afficher au

prochain démarrage de Boa ou lorsque on sélectionne dans le menu Editer / rechargement.

Si on utilise Zope, on peut accéder à des projets dans le serveur Server en utilisant l'option Zope. Les informations de connexion TCP / IP sont configurées par défaut dans le panneau de l'inspecteur ou dans le fichier de configuration de l'explorateur, Explorer.msw.cfg sous Windows ou sous UNIX Explorer.gtk.cfg.



Il permet également d'accéder aux préférences de paramètres, au-dessus, des raccourcis clavier sont sélectionnés dans l'explorateur et les détails sont présentés dans l'Inspecteur.

<note>La démo de wxPython s'affiche dans l'Explorateur comme cela a été activé par le Plug-ins.</note>

Utilisation de l'aide

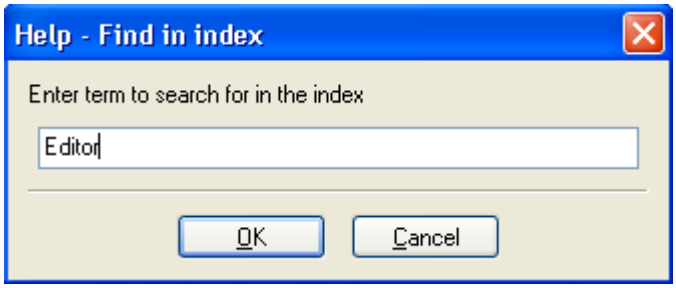
L'EDI Boa fournit des liens vers différents fichiers d'aide de l'EDI.

Pour Boa 0.6.0 les fichiers d'aide suivants sont inclus:

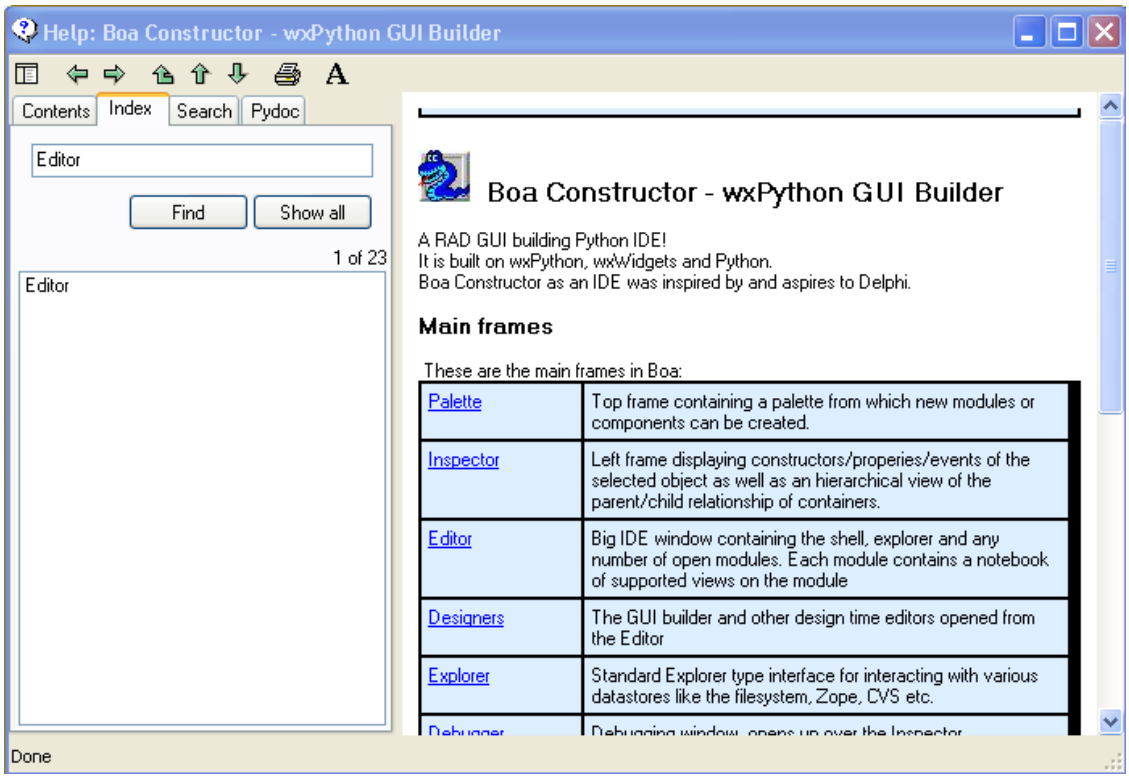
- Aide de Boa
- Guide de démarrage de Boa
- Documentation Python 2.5
- wxWidgets 2.8.4
- Documenation wxPython 2.8.4 API

- Documentation wxStyledTextCtrl
- Object Graphics Library 3.0

Des livres d'aide supplémentaires peuvent être ajoutés (voir Livres d'aide).



L'accès le plus simple et le plus rapide à l'aide depuis à peu près partout dans Boa est CTRL+H, puis entrer ce qui est cherché ; Boa trouvera toutes les références dans TOUS les livres d'aide qu'il connaît (par exemple wxPython, Python etc) et le montre comme ci-dessous.



La fenêtre d'aide permet une navigation HTML standard avec un outil de recherche.

Si l'aide ne s'affiche pas correctement, configurer l'environnement.

- L'aide Python ou wxPython risque de ne pas afficher si le chemin vers le répertoire Python installé est incorrect. Le chemin vers l'aide de Python dépend du fichier de préférences de la plate-forme. C'est PrefMSW.py pour Windows et PrefGTK.py pour UNIX.
- L'aide de wxPython peut ne pas s'afficher si elle n'est pas sous forme HTML. La distribution de wxPython peut stocker l'aide sous forme HTML compilé (chm) plutôt que HTML (html / htm). Boa ne peut pas afficher du HTML compilé. Au lieu de cela, il faut télécharger la version HTML de la

documentation sur le site wxPython.

D'autres bons endroits pour chercher de l'aide sont :

- Les fichiers de démonstration fournis avec wxPython (à télécharger séparément selon le programme d'installation utilisé sur le site wxPython).<note tip>Vérifier le plug-in Boa correspondant, l'activez à partir de Préférences / Plug-ins / fichiers plug-in / wxPythonDemo et redémarrer Boa</note>
- Les pages wiki sur wxPython à : http://wiki.wxpython.org/
- Il y a une communauté très utile qui peut aider.
  - Poster la question sur Boa-constructor-users@lists.sourceforge.net si elle est spécifique à Boa
  - Ou envoyer la question à wxPython-users@lists.wxwidgets.org si elle n'est pas spécifique à Boa

Tutoriel - Créer votre première application

Cette section présente un bref didacticiel pour familiariser avec l'EDI Boa constructor. Ce tutoriel guide étape par étape à travers le processus de construction d'un éditeur de texte simple, appelé Notebook. Après avoir travaillé ce tutoriel, on ensait assez pour être productif avec Boa Constructor.

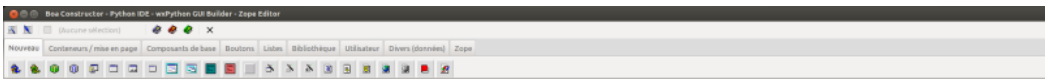
On apprendra à:

- Créer une application.
- Créer des cadres, des menus et des barres d'état.
- Créer des contrôles tels que des boutons, des champs de saisie de texte et des étiquettes.
- Configurer les contrôles selon les besoins.
- Travailler avec des boîtes de dialogue courantes.
- Concevoir des boîtes de dialogue.

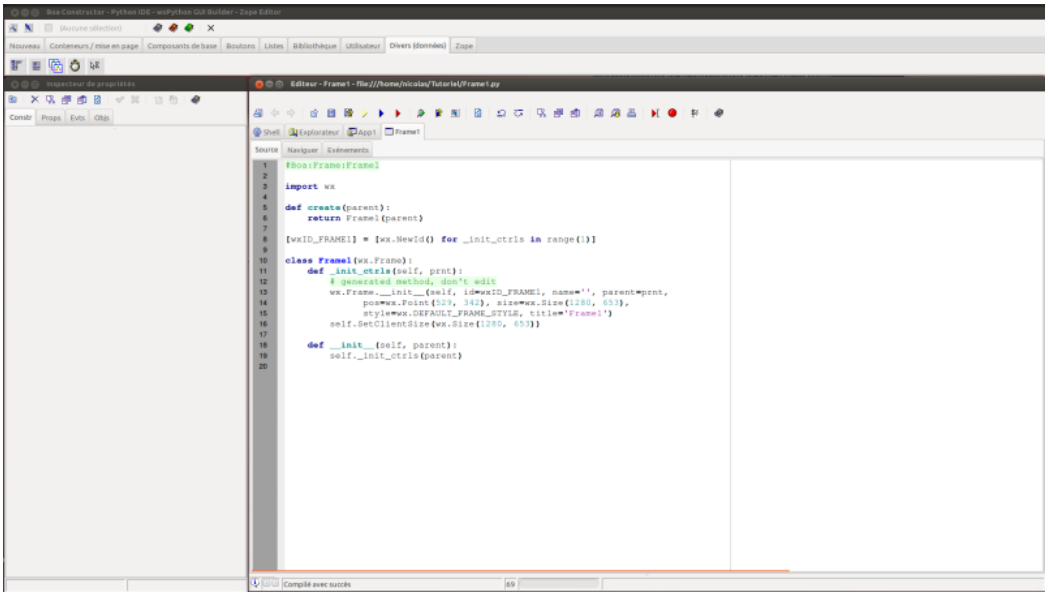
Création d'une nouvelle application

- Décider d'un répertoire destiné à contenir l'application. Si nécessaire, créer le répertoire.
- Créer une nouvelle application avec le bouton nouvelle application ci-dessous à partir de la palette.

 Application - info-bulle wx.APP



- Enregistrer à la fois le fichier App1.py et le fichier Frame1.py dans le répertoire créé plus tôt. On peut utiliser le bouton «Enregistrer» sur la barre d'outils de l'éditeur. Noter que les astérisques (*) disparaissent du nom quand il est sauvé. Elles signalent qu'il y a des modifications non sauvegardées dans le fichier.
- On a maintenant une application qui montre juste un cadre vierge. Utiliser le bouton 'Run' sur la barre d'outils de l'éditeur pour exécuter l'application.

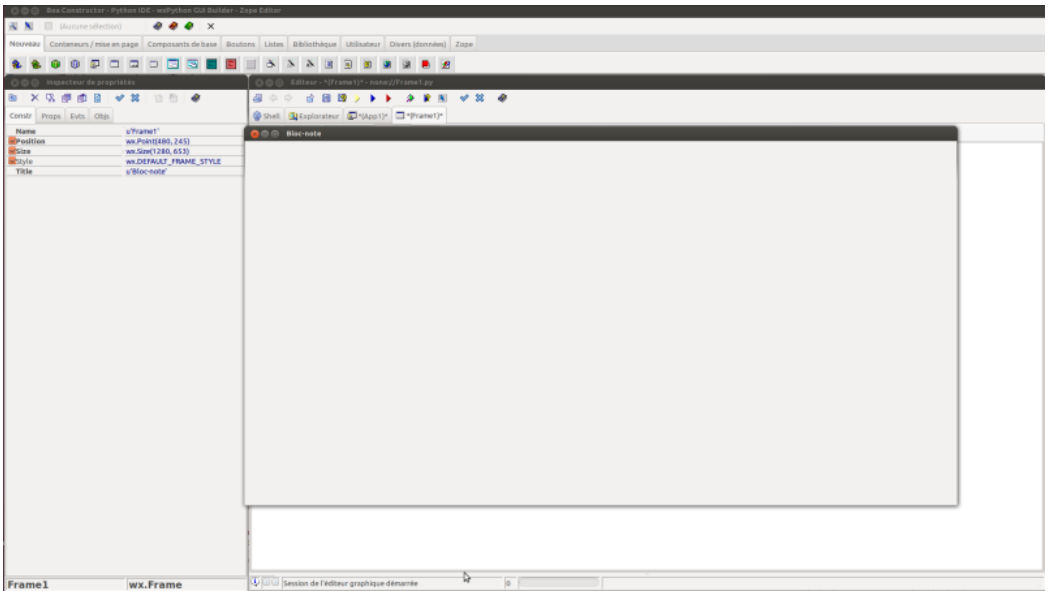


On voit ci-dessus, dans la section de l'éditeur les deux nouveaux fichiers créés et enregistrés.

Un clic sur le bouton de lancement (jaune) montre le résultat de la programmation, c'est-à-dire juste un cadre vide.

Utilisation de l'éditeur graphique pour mettre le titre

- Sélectionner l'onglet Frame1 dans l'éditeur pour s'assurer que nous éditons le cadre.
- Démarrer l'éditeur graphique, en cliquant sur le bouton éditeur graphique à partir de la barre d'outils de l'éditeur. 



- Le cadre sera affiché comme une zone de dessin (initialement le titre indique Frame1
- Deux nouveaux volets sont affichés dans l'éditeur, étiquetés volet **Données** et volet **Sizer**.
- La fenêtre Inspecteur de propriétés affiche le volet 'cons' ou constructeur. Ce volet permet de modifier la taille, la position, le style, les noms de variables et le titre d'un composant. Modifiez le champ appelé «Titre». Donnez au cadre le nom 'Bloc-Notes', en appuyant sur Retour, le titre de la fenêtre Éditeur graphique a changé.
- Il faut enregistrer les modifications en utilisant le bouton 'Post'. Appuyer sur le bouton 'Post' sur la barre d'outils de l'inspecteur ou de l'éditeur.
- La fenêtre de l'éditeur graphique se ferme.
- Remarquer que le code source a été mis à jour pour refléter le titre.
- L'éditeur indique que le code source est modifié par des astérisques sur l'onglet Bloc-notes, il faut donc appuyer sur le bouton Enregistrer.

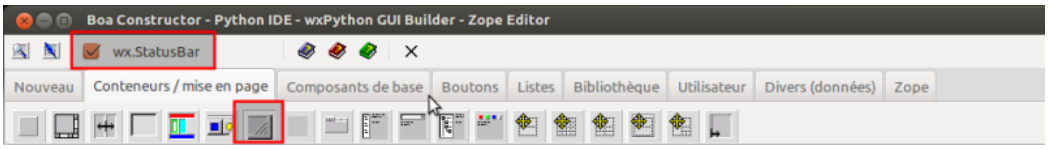
Ajouter une barre d'état

Le premier composant que nous allons ajouter à l'application sera une barre d'état. Une barre d'état donne des informations sur un programme lors de son exécution. Nous allons utiliser la barre d'état pour indiquer à l'utilisateur ce qui se passe lorsque des actions lentes se produisent, de donner des messages d'aide simples ou toute autre information à montrer.

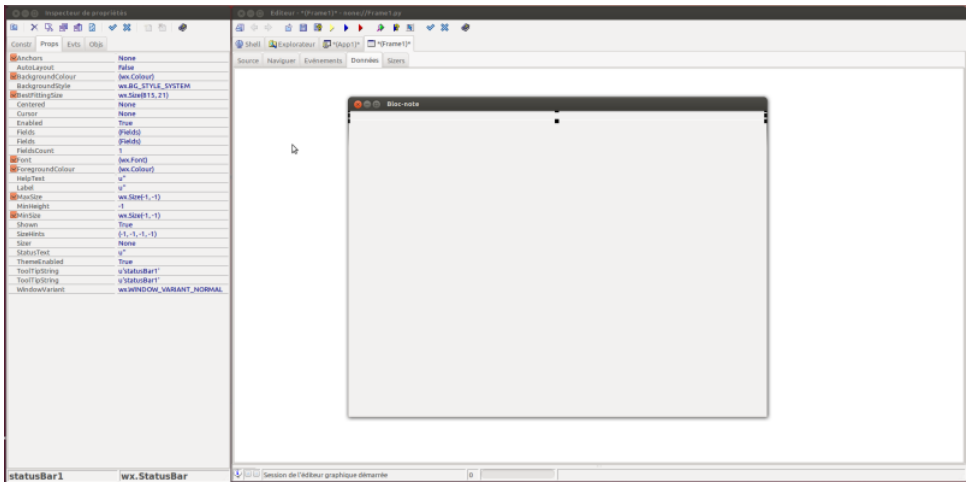
- Sélectionnez l'onglet Frame1 dans l'éditeur pour s'assurer que nous éditons le cadre.
- Démarrer l'éditeur graphique en cliquant sur le bouton Éditeur graphique à partir de la barre d'outils de l'éditeur.
- Le cadre s'affiche comme une zone de dessin.
- Dans la palette sélectionner l'onglet appelé 'Conteneurs/mise en page'.

Cet onglet contient des entrées pour les composants qui sont utilisés avec des cadres, dont la barre d'état.

- Déplacez la souris sur les boutons. Les bulles d'aide montrent qu'un de ces boutons représente le contrôle wx.StatusBar. Il s'agit du contrôle que nous voulons. Cliquer sur ce bouton.
- Le bouton change en devenant ombré pour indiquer qu'il est pressé. La palette contient une case à cocher pour indiquer le type de composant sélectionné. Elle devrait afficher wx.StatusBar.

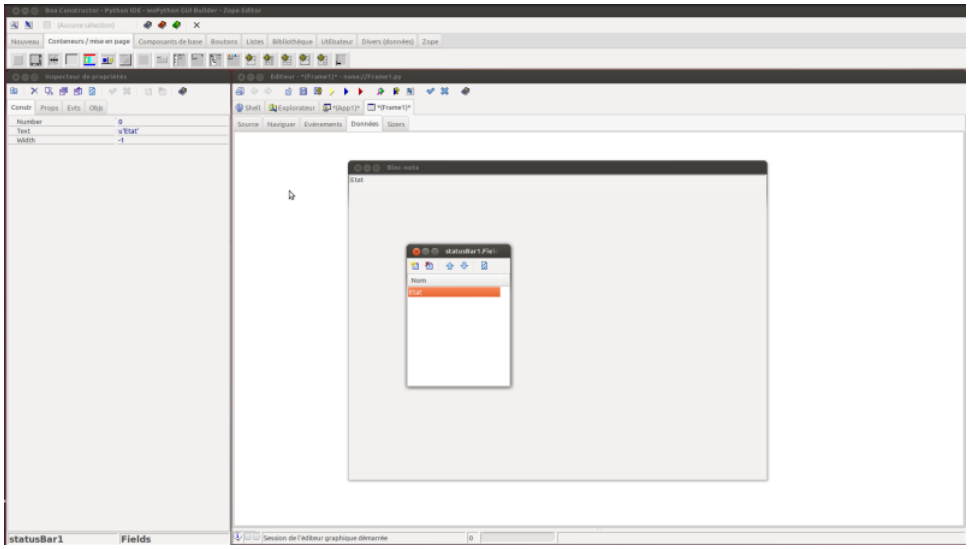


- Maintenant, déplacer le curseur de la souris sur le cadre de dessin. Cliquer (gauche) dans la zone de dessin. Cela crée une barre d'état dans le cadre.
- La ligne de statut dans l'Inspecteur affiche sur la gauche du nom du widget actuel 'statusBar1 » et sur la droite il montre que la classe wxWidget est dérivée de wx.StatusBar.
- Dans l'inspecteur, sélectionner le panneau 'Propriétés'. Ce volet permet de configurer les propriétés de notre barre d'état.
  - Cliquer pour modifier la valeur de 'Fields'. Le champ affiche un bouton «+ + +». Cliquer sur le bouton. Cela ouvre l'éditeur de collection.

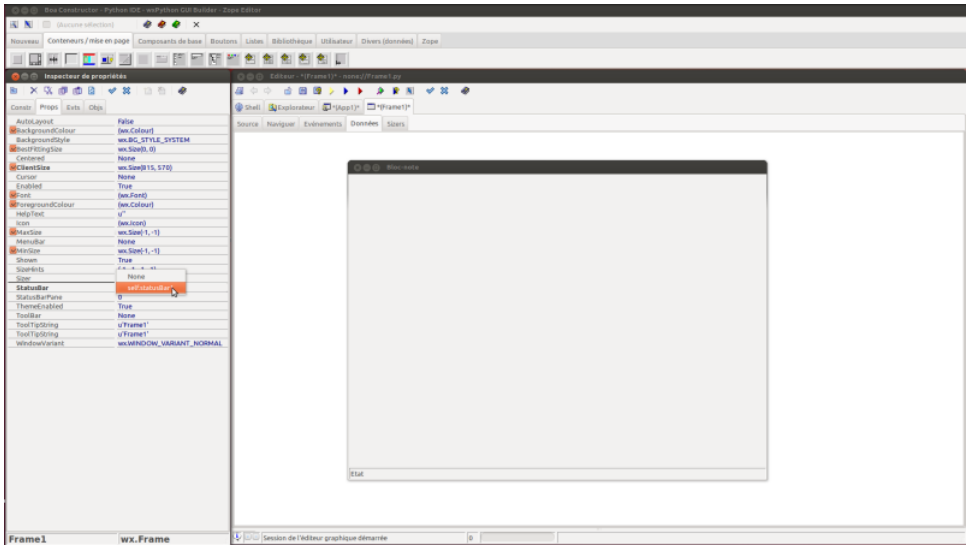


  - L'éditeur de collection est un outil qui sert à ajouter plusieurs sous-composantes à des composants. Nous allons ajouter un champ à la barre d'état, cependant on peut ajouter plusieurs champs.
  - Appuyez sur le bouton 'Nouveau'  de l'éditeur de collection. Ceci crée un nouveau champ dans la barre d'état. Il devient le champ courant de l'inspecteur.
  - Modifier le champ texte. Définir le nom de «Fields0» à «statut».
  - la barre d'outils de l'éditeur de collection contient un bouton “Refresh”  Appuyer sur ce bouton pour voir l'évolution de

l'inspecteur dans l'éditeur de fenêtre de collection.



- Fermer la fenêtre de l'éditeur de collection. Sélectionnez la fenêtre de l'éditeur graphique. Cliquer n'importe où dans la zone de dessin pour faire du cadre (Frame1) le composant courant.
- Sélectionner le volet de propriétés dans l'inspecteur.
- Modifier la propriété 'StatusBar. Le menu déroulant montre notre nouvelle barre d'état. Sélectionner la barre d'état, c'est nécessaire pour que le cadre gère la barre d'état, c'est à dire qu'il soit positionné au bas de l'écran et que sa taille soit défiinie.



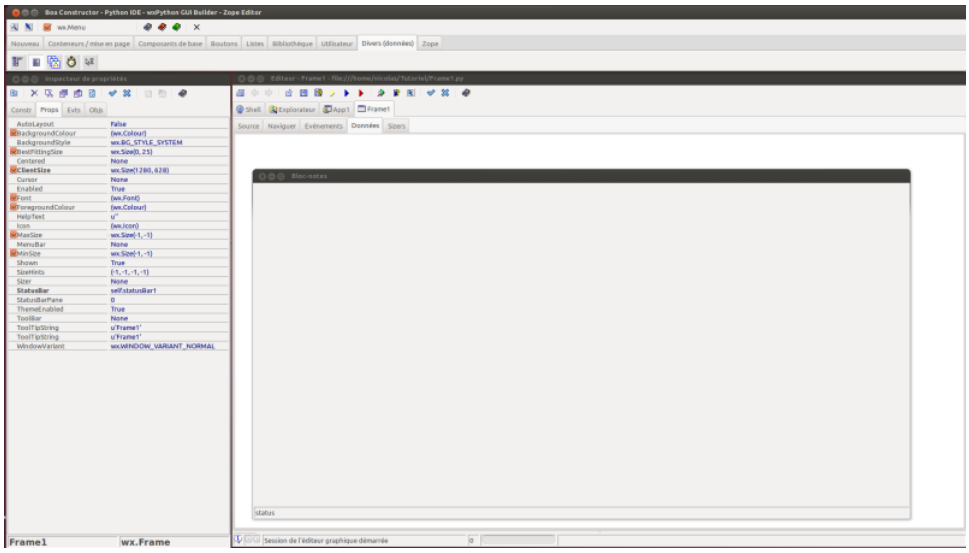
- Mettre à jour le code source avec un bouton Post 
- Enregistrer les modifications du code source en utilisant le bouton Enregistrer de la barre d'outils de l'éditeur.

Ajouter une barre de menus

La composante suivante que nous allons ajouter à l'application est une

barre de menus, un élément courant pour les programmes Windows. Notre barre de menu contiendra deux entrées, Fichier et Aide. La sélection de l'une d'elles va afficher un menu déroulant dans lequel l'utilisateur peut sélectionner une option.

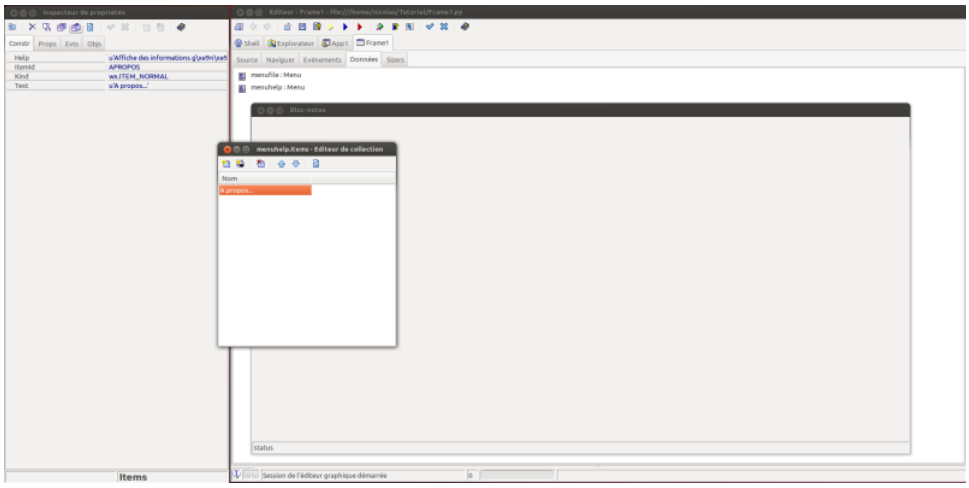
- Sélectionnez l'onglet Frame1 dans l'éditeur afin de s'assurer que nous éditons le cadre.
- Démarrez l'éditeur graphique, en cliquant sur le bouton éditeur graphique  à partir de la barre d'outils de l'éditeur.
- L'éditeur graphique ajoute deux volets supplémentaires dans la fenêtre de l'éditeur, les vues 'Données' et 'Sizers'. Dans la palette sélectionner l'onglet appelé 'Divers (données)'. Le menu déroulant (wx.Menu) est l'une des composantes répertoriées dans cet onglet.
- Déplacer la souris sur les boutons. Les bulles d'aide montrent que l'un de ces boutons représente le contrôle wx.Menu. C'est le contrôle que nous voulons. Cliquer sur ce bouton.
- Le bouton devient ombré pour indiquer qu'il est pressé. La palette contient une case à cocher pour indiquer le type de composant actuellement sélectionné, ici wx.Menu.



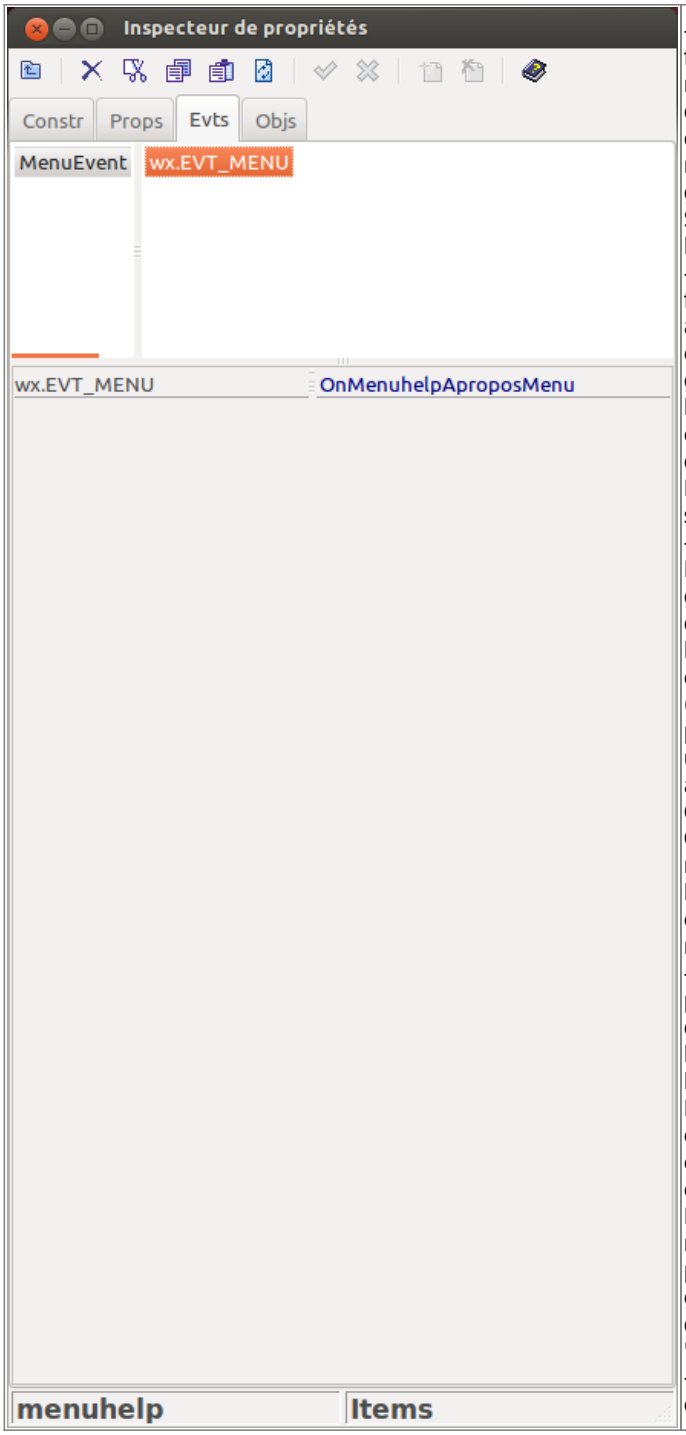
- Maintenant cliquer dans la vue données de l'éditeur ou dans l'éditeur graphique, Dans ce cas, s'assurer de cliquer sur la zone *Frame1* et non sur *barre d'état*.
- Le menu n'est pas visible sur le cadre. Mais il y a une entrée sur la vue de données.
- Répéter la procédure. Il ya maintenant deux **wx.Menus** dans la vue de données, appelés menu1 et menu2. Sélectionner menu1 à la souris. L'inspecteur (onglet Props) affiche le nom et le titre de menu1.
- Modifier le nom du premier wx.Menu en menufile. Nommer le deuxième wx.Menu en menuHelp. Définir les titres, respectivement Fichier et Aide.
- Double-cliquer sur l'entrée menuHelp de la vue données. Cela

ouvre l'éditeur de collection. L'éditeur de collection permet d'ajouter des éléments à nos menus.

- Cliquer sur le bouton Nouveau de l'éditeur de collection. Cela crée un nouvel élément de menu dans le menu déroulant, qui devient l'élément actif dans l'inspecteur.
- Modifier le champ 'Text' de 'Items0' en 'A propos...'. Il est aussi recommandé de changer ItemId de 'ITEMS0' en 'APROPOS'.
- Modifiez le champ 'Help' en 'Affiche des informations générales sur le Bloc-Notes'.
- La barre d'outils de l'éditeur de collection contient un bouton “Refresh”. Appuyez sur ce bouton pour voir les changements de l'inspecteur dans la fenêtre de l'éditeur de collection. Garder l'éditeur de collection ouvert.



- Dans l'inspecteur, sélectionner le Panneau Événements, qui permet de configurer les événements. Il nous faut configurer l'action qui se produit lorsque l'élément de menu 'A propos' est sélectionné. Lorsque l'élément de menu 'A propos' est sélectionné, un événement appelé EVT_MENU est généré et envoyé à notre programme. Nous devons ajouter une méthode à notre classe pour gérer cet événement.



- Le côté gauche de la fenêtre des événements montre les groupes d'événements sont disponibles. Pour menuitem, il n'y a qu'un groupe 'MenuEvent'. Sélectionner ce groupe à la souris.
- Le côté droit de la fenêtre d'événements affiche maintenant les événements dans le groupe sélectionné. Pour le menu, il n'y a qu'un seul événement EVT_MENU dans le groupe MenuEvent. Double-cliquer sur cet événement.
- Le bas du volet Evénements affiche les gestionnaires d'événements de l'application pour le composant actuel (l'élément de menu A propos). Il y a maintenant un nouveau gestionnaire appelé OnMenuHelpAproposMenu. C'est le nom de la méthode qui sera appelée lorsque l'option 'Apropos' est sélectionnée dans le menu Aide.
- Notez la méthode utilisée par le gestionnaire d'événement pour générer les noms. L'événement est la dernière partie (Menu). Le composant est la partie centrale, ici la sous-composante Apropos de la composante menuHelp. Enfin, Boa constructor respecte la convention de préfixer tous les gestionnaires d'événements avec le mot 'On'.
- Fermer l'éditeur de collection.

Maintenant, il faut répéter le processus pour ajouter des options au menu fichier.

- Dans l'affichage des données de l'éditeur, double-cliquer sur l'item 'menufile' pour ouvrir l'éditeur de collection.
- Ajouter cinq nouveaux éléments.
- Sélectionner successivement chaque élément de menu et les étiqueter 'Ouvrir', 'Enregistrer', 'Enregistrer sous', 'Fermer' et 'Quitter' ; ici encore, il est recommandé de changer aussi ItemId. Entrer du texte d'aide pour chaque élément de menu. Appuyez sur le bouton Actualiser de l'éditeur de collection pour afficher les nouvelles étiquettes.
- Sélectionner successivement chaque élément de menu. Pour

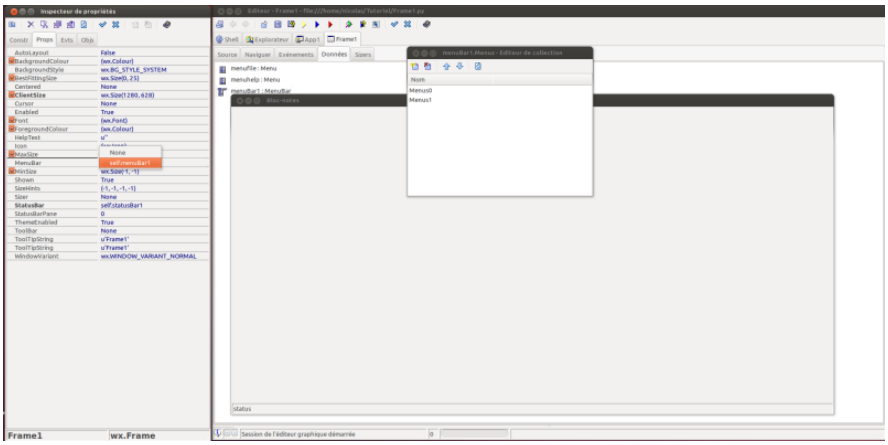
chaque élément, sélectionner le volet Événements de l'inspecteur, et ajouter un gestionnaire d'événements pour EVT_MENU à chaque élément.

- Fermer l'éditeur de collection.

Maintenant, nous allons créer la barre de menus.

- Dans la fenêtre Palette, sélectionner le panneau Divers (données). Sur ce volet, sélectionner le composant barre de menus (wx.MenuBar).
  - Déplacer le curseur sur l'affichage des données de l'éditeur. Cliquer sur ce volet pour ajouter une barre de menu appelé MenuBar1 à l'application.
  - Double-cliquer sur l'item MenuBar1 pour ouvrir son éditeur de collection (comme on peut le voir sur l'image ci-dessous on peut en garder plusieurs ouverts).

2. Ajouter deux éléments à la barre de menu en utilisant l'éditeur de collection. Sélectionner Menus0 ; dans le volet constructeur de l'inspecteur, modifier le champ 'Menu'. c'est un menu pop-up, avec trois articles, le constructeur de wx.Menu() et nos deux menus déroulants, sélectionner l'élément self.menuFile et définir le titre en 'Fichier'. Cela fait du menu menufile le premier menu déroulant de la barre de menus.
3. Dans l'éditeur de collection, sélectionner le deuxième élément. Répéter le processus pour relier Menus1 au menu d'aide déroulant, avec l'étiquette Aide.
4. Sélectionner l'image principale, Frame1 sur l'éditeur graphique. Le cadre est maintenant le contrôle actuel de l'inspecteur.
5. Sélectionner le volet Propriétés (Props) dans l'Inspecteur. Modifiez le champ MenuBar. Il s'agit d'un menu pop-up. Sélectionner la nouvelle barre de menu self.menuBar1. Cette propriété définit quelle menuBar associer au cadre.



- Enregistrer les modifications à l'aide d'un bouton Post pour fermer l'éditeur graphique et laisser Boa générer le code source.

- Enregistrer le code source du fichier source généré Frame1.py
- Exécuter le programme.
  - On voit maintenant les menus et la barre d'état.
  - Quand on sélectionne une option de menu, le texte d'aide apparaît dans la barre d'état.

Sous windows, cela donne :

Ajout du contrôle texte

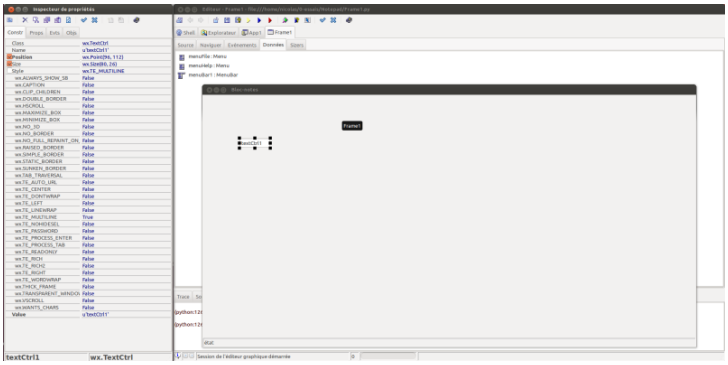
Nous allons maintenant ajouter un contrôle texte pour l'édition à notre cadre. Ce contrôle est appelé wx.TextCtrl.

- Rouvrir l'éditeur graphique pour modifier le cadre, Frame1.py.
- Dans la palette, sélectionnez le panneau Composants de base. Sélectionner wx.TextControl.<note tip>Vous pouvez maintenir le pointeur de la souris sur chaque contrôle pour connaître son nom.</note>
- Déplacer le pointeur de la souris sur la fenêtre de l'éditeur graphique, veiller à ce qu'une bulle d'aide affiche «Frame1», puis cliquer. Un nouveau contrôle de texte est dessiné. Inutile de définir la taille du contrôle. Par défaut, il remplit toute la surface disponible, c'est à dire entre la barre d'état et la barre de menus.
- Le wx.TextControl par défaut est une entrée d'une seule ligne. Nous devons lui indiquer que nous voulons que ce soit un contrôle de saisie multi-ligne. Pour cela éditer la propriété 'style' de l'inspecteur, sur le volet constructeur.
- Modifier le style et le définir en wx.TE_MULTILINE. On peut taper cela dans le champ de valeur du style ou

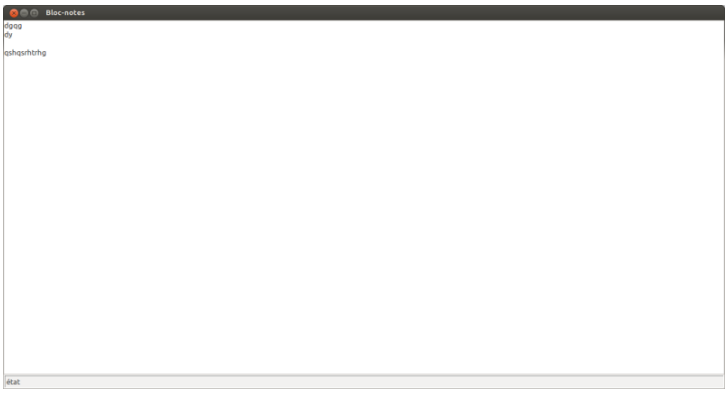
cliquer sur la case à gauche de style et Boa montre tous les styles disponibles. Définir les styles que l'on veut utiliser à Vrai en cliquant dessus.

- Le champ style contient du code Python valide. Pour définir deux styles logiques, les séparer par un '|'. On peut voir tous les styles disponibles pour wx.TextControl dans l'aide en ligne wxPython pour la classe wx.TextControl.

<note tip>Utiliser Ctrl+H et entrer textctrl pour obtenir de la documentation et des descriptions des différents styles, notez que certains d'entre eux pourraient ne pas être montrés pour wx.TextCtrl car ils sont hérités, par exemple wx.Window. Actuellement Ctrl-H ne fonctionne pas dans la fenêtre de l'éditeur graphique, mais à peu près partout ailleurs dans de Boa. </note>



- Renommer le champ de texte. Le nom par défaut est textCtrl1. Le changer en TextEditor.
- Dans le volet constructeur il y a un champ appelé valeur qui contient la valeur par défaut du contrôle. Le mettre à blanc.
- Mettre à jour le code source avec un bouton Valider.
- Enregistrer les modifications du code source.
- Exécuter l'application.



- Le champ éditeur de texte est automatiquement redimensionné à l'espace disponible.
- Si on redimensionne le cadre, le contrôle se redimensionne.
- Remarquez que wxWidgets offre une barre de défilement. Le champ défile automatiquement si on dépasse le bas. Si on tape la ligne au-dela de la largeur de la fenêtre d'édition, il se produit un passage à la ligne.
- Il y a également la fonctionnalité couper / coller et le marquage par défaut à la souris.

Ajout de fonctionnalités au menu Fichier

La tâche suivante consiste à interagir avec l'utilisateur pour implémenter les fonctionnalités du menu. Les boîtes de dialogue sont utilisées pour recueillir l'entrée de l'utilisateur. Les boîtes de dialogue sont modales, c'est à dire qu'on ne peut pas utiliser les autres fenêtres de l'application en cours jusqu'à ce que le dialogue soit fermé.

- Les boîtes de dialogue sont placées directement dans le code source. Elles ne sont pas placées dans l'éditeur graphique. Elles sont placées avec l'éditeur. Dans le code source de Frame1.py, aller dans le gestionnaire d'événements pour l'événement ouvert. Cette méthode est appelée OnMenuFileOpenMenu. Nous allons placer le dialogue Ouvrir un fichier dans cette méthode. Placez le curseur de clavier directement devant 'event.Skip()'. Nous allons introduire notre nouveau code ici.
- Appuyer sur Alt+T et sélectionner wx.FileDialog dans le menu déroulant : Boa Constructor colle le code squelette directement dans la méthode du gestionnaire d'événements.
- Notez le code 'dlg.Destroy ()', il est très important que les dialogues sont détruits !
- La section de code source doit maintenant se présenter comme suit:

```
def OnMenuFileOpenMenu(self, event):
    dlg = wx.FileDialog(self, "Choose a
file", ".", "", "*.*", wx.OPEN)
    try:
        if dlg.ShowModal() == wx.ID_OK:
```

```
            filename = dlg.GetPath()
            # Your code
    finally:
       dlg.Destroy()
    event.Skip()
```

- Ce code crée un squelette de dialogue. Il interagit avec l'utilisateur. Quand le dialogue est terminé, il est détruit.
- Les mots '# Your code' marquent la position où insérer notre code. Ce code est déclenché quand la boîte de dialogue retourne 'wx.ID_OK, c'est-à-dire lorsque l'utilisateur a cliqué sur le bouton 'Open'. Nous allons insérer notre code ici. Le contrôle wx.TextCtrl a une méthode que nous allons utiliser pour charger un fichier dans la fenêtre d'édition, pour cela, nous utilisons la méthode 'LoadFile'.
- On peut supprimer le 'event.Skip()' restant car aucun autre événement ne sera appelé dans ce cas. Le nom 'event.Skip()' C'est un peu confus, on doit appeler 'event.Skip()' pour les événements où d'autres gestionnaires d'événements doiventt AUSSI être exécutés.
- Il a été nécessaire lorsque le code a été généré parce que Python signale une erreur s'il y a une méthode sans corps.
- Dans le cadre de toutes les fonctionnalités de notre application, nous devons être en mesure d'accéder au nom de fichier de sorte que l'option “Enregistrer” de menu puisse enregistrer dans ce fichier, donc nous avons ajouté la ligne 'self.FileName=filename'.
- La ligne 'self.SetTitle(('Notebook - %s') % filename)' change le titre pour montrer sur quel fichier on travaille.
- Le listing ci-dessous présente notre nouveau code.

```
def OnMenuFileOpenMenu(self, event):
    dlg = wx.FileDialog(self, "Choose a
file", ".", "", "*.*", wx.OPEN)
    try:
        if dlg.ShowModal() == wx.ID_OK:
            filename = dlg.GetPath()
            # Your code
self.textEditor.LoadFile(filename)
            self.FileName=filename
```

```
            self.SetTitle(('Notebook -
%s') % filename)
    finally:
        dlg.Destroy()
```

- Nous devons répéter l'exercice pour fournir la fonctionnalité «Enregistrer sous». Insérez une boîte de dialogue de fichier dans le corps de 'OnFileSaveasMenu'.
- Il s'agit d'une boîte de dialogue Enregistrer. Changer le prompt, le paramètre 2 de wx.FileDialog en “Enregistrer le fichier sous”. Changer le style, paramètre 6 en wx.SAVE. La méthode pour enregistrer le fichier s'appelle SaveFile.
- Encore une fois, nous sauvons la valeur de nom de fichier utilisé par l'option du menu “Enregistrer”.
- La liste ci-dessous montre le code.

```
def OnMenuFileSaveasMenu(self, event):
    dlg = wx.FileDialog(self, "Save
file as", ".", "", "*.*", wx.SAVE)
    try:
        if dlg.ShowModal() == wx.ID_OK:
            filename = dlg.GetPath()
            # Your code
self.textEditor.SaveFile(filename)
            self.FileName=filename
            self.SetTitle(('Notebook -
%s') % filename)
    finally:
        dlg.Destroy()
```

- Ensuite, nous allons mettre en œuvre l'option 'Fermer' du menu. Dans cette méthode, nous allons simplement effacer le contrôle éditeur de texte, la variable membre de nom de fichier et réinitialiser le titre..

```
def OnMenuFileCloseMenu(self, event):
    self.FileName = None
    self.textEditor.Clear()
    self.SetTitle('Notebook')
```

- Ensuite, nous mettrons en œuvre l'option de menu “Quitter”. Dans cette méthode, nous devons mettre fin à l'application. Toutes les méthodes de wxPython sont terminées par la fermeture de la fenêtre de premier niveau.

Dans notre cas, nous avons seulement la fenêtre Frame1. Pour mettre fin à l'application, nous invoquons la méthode Close() pour Frame1.

```
def OnMenuFileExitMenu(self, event):
    self.Close()
```

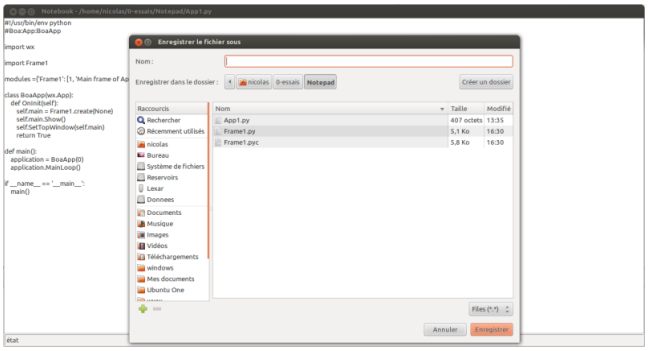
- Ensuite, mettons en œuvre l'option de menu “Enregistrer”. Cet élément de menu enregistre le fichier en utilisant le nom actuel, qui est stocké dans la variable self.FileName.
- Quand il n'y a pas de fichier actuel, la variable self.FileName est réglée sur None. Dans ce cas, l'option de menu «Save» doit agir comme l'option de menu “Enregistrer sous”.
- Le nom de fichier variable doit être créé lorsque Frame1 est construit. Nous devons l'ajouter au constructeur. On peut ajouter le code d'application à la fin du constructeur par défaut (init).

```
def __init__(self, parent):
    self._init_ctrls(parent)
    self.FileName=None
```

- Maintenant, nous pouvons implémenter la fonctionnalité Enregistrer. Nous vérifions s'il y a un nom de fichier courant. S'il existe, nous pouvons enregistrer le contenu de ce fichier. Sinon, il suffit d'appeler la méthode «Enregistrer sous».

```
def OnMenuFileSaveMenu(self, event):
    if self.FileName == None:
        return
self.OnFileSaveasMenu(event)
    else:
self.textEditor.SaveFile(self.FileName)
```

- Nous avons maintenant mis en œuvre toutes les fonctionnalités de l'éditeur. Nous pouvons ouvrir des fichiers, les modifier et les sauvegarder.
- Votre éditeur doit ressembler à ce qui est montré sur l'image ci-dessous.
    - Le fichier App1.py a été ouvert
    - Ensuite, l'option de menu “Fichier / Enregistrer sous” a été sélectionnée



Créer une fenêtre de dialogue

Les boîtes de dialogue sont utilisées pour interagir avec l'utilisateur et récupérer des entrées spécifiques. Dans les sections précédentes, nous avons utilisé la boîte de dialogue pré-intégrée wx.FileDialog. Nous allons maintenant développer notre propre boîte de dialogue pour l'option A propos du menu.

- Le dialogue que nous allons créer nécessite une nouvelle fenêtre. Ce n'est pas une composante de la fenêtre Frame1. Il formera dans notre application un fichier Python séparé. Sélectionner le module d'application App1 dans l'éditeur. Choisir le volet 'Application'.
- Dans la palette, sélectionner le volet Nouveau. Sélectionner le bouton wx.Dialog. Cela crée un nouveau fichier source Dialog1.py, et ajoute automatiquement ce nouveau fichier source au module d'application.
- Sélectionner le volet Frame1. Nous voulons écrire le code de l'option A propos du menu, qui est utilisé pour afficher la boîte de dialogue. Cette option est mise en œuvre par la méthode OnHelpAboutMenu. Le code est le suivant:

```
def OnMenuHelpAboutMenu(self,
event):
    dlg = Dialog1.Dialog1(self)
    try:
        dlg.ShowModal()
    finally:
        dlg.Destroy()
```

- Ce code fait référence au module Dialog1.

Pour que ce code fonctionne, il faut importer le module Dialog1. Par convention, nous plaçons les importations au début du source. Ajouter la ligne 'import Dialog1' à Frame1.py après la ligne 'import wx'.

```
import wx
import Dialog1
```

- Enregistrer les trois fichiers source. On peut maintenant lancer l'application. Lorsqu'on sélectionne 'A propos' dans le menu Aide, la nouvelle boîte de dialogue apparaît. Notez que la boîte de dialogue est modale, c'est à dire qu'il faut la fermer avant de pouvoir accéder à la fenêtre Frame1. Quitter l'application et revenir à Boa Constructor.
- Maintenant, nous allons ajouter des champs de la boîte de dialogue. Pour cet exercice, nous aurons besoin d'un fichier bitmap. Pour la démonstration j'en ai utilisé un appelé Boa.jpg. On pouvez créer son propre bitmap en utilisant un utilitaire comme paint. Copier le bitmap dans le répertoire de l'application.
- Sélectionner le volet Dialog1.py. Démarrer l'éditeur graphique en cliquant sur le bouton éditeur graphique  .
- D'abord, nous allons ajouter une étiquette à la boîte de dialogue. Dans la palette sélectionner 'Basic Controls'. Sur ce volet, sélectionner le contrôle wx.StaticText. Cliquer sur l'éditeur graphique pour créer le contrôle.
- Sur l'inspecteur modifier le champ 'Label'. Réglez la valeur de 'Note book - Simple Text Editor'. Notez que l'étiquette dans l'éditeur graphique grandit pour accueillir le texte.
- Nous utilisons la propriété style pour configurer l'alignement du texte dans l'étiquette. Définissez la propriété style 'wx.ALIGN_CENTRE' ou sélectionner ce style après avoir cliqué sur la case à gauche du style.
- Sélectionner le panneau Propriétés de

l'inspecteur. Modifiez le champ appelé 'font'. Définir la police à une assez grande police, e.g. 12 or 14 point. Notez que vous pouvez modifier à la fois la police et la taille en point avec cette propriété.

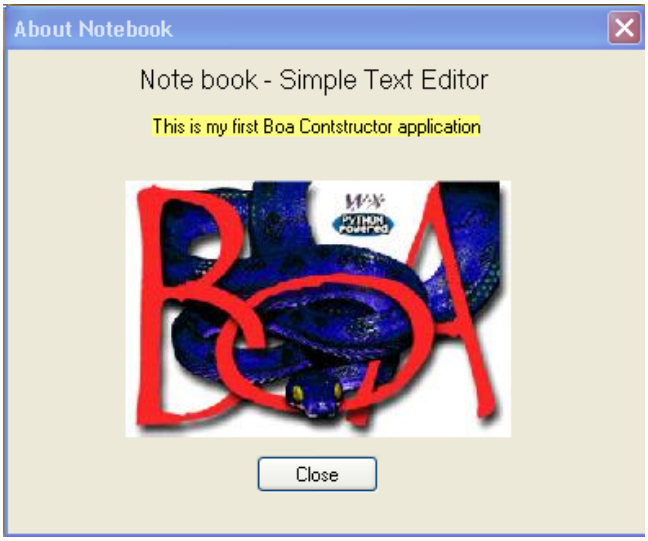
- Dans la fenêtre de l'éditeur graphique votre étiquette s'affiche avec huit marqueurs sur les bords. Vous cliquez sur le bouton gauche de la souris (en le maintenant enfoncé) sur l'un de ces marqueurs puis déplacez la souris pour redimensionner la boîte. Vous pouvez également cliquer au centre de l'étiquette, et maintenir enfoncé le bouton de la souris pour déplacer l'étiquette. Placez l'étiquette en haut au milieu de la boîte.
- Maintenant ajouter une autre étiquette en dessous de la première. Réglez le texte «Ceci est ma première application Boa Contstructor». In the Inspector, sélectionner le panneau 'Propriétés'. Modifiez la valeur 'BackgroundColour'. Choisissez une couleur dans l'ensemble disponible et appuyez sur OK. Maintenant repositionner et redimensionner votre étiquette jusqu'à ce qu'elle semble équilibrée.
- Ensuite, nous allons ajouter le bitmap. Dans les 'Basic Controls', sélectionnez le contrôle wx.StaticBitmap. Placez le sous la seconde étiquette de votre dialogue. Dans l'inspecteur sélectionner le volet constructeur. Modifiez le champ Bitmap. Cela vous donnera un dialogue 'Open File'. Choisissez le bitmap que vous avez créé plus tôt. Le champ wx.StaticBitmap dans l'éditeur graphique va changer pour accueillir votre bitmap. Déplacez le bitmap jusqu'à ce qu'il soit équilibré en dessous des deux étiquettes.
- Enfin, nous allons ajouter un bouton à la boîte de dialogue. Dans la palette sélectionner le panneau Buttons. Sélectionnez le type de bouton de base, wx.Button. Le Placer dans le formulaire en dessous du bitmap. Dans le volet constructeur de l'inspecteur, modifier le 'Label'. Réglez-le sur «Fermer». Sélectionnez le volet Événements de

l'inspecteur. Ajoutez un gestionnaire pour le type d'événement EVT_BUTTON.<note tip>sélectionner le groupe d'événements d'abord, puis l'événement.</note>

- Ce sont tous les contrôles que nous allons ajouter à la boîte de dialogue. Régler la taille de la boîte de dialogue pour accueillir les contrôles. Repositionner et redimensionner les contrôles jusqu'à ce que vous sentiez qu'ils sont bien équilibrés.
- Sélectionnez Dialog1 dans l'éditeur graphique. Dans le volet constructeur de l'inspecteur, modifier le champ Titre. Lui donne la valeur 'About Notebook'.
- Appuyez sur un des boutons valider pour mettre à jour le code source de vos modifications.
- Enfin, nous devons mettre en œuvre le gestionnaire d'événements pour le bouton Fermer. Dans l'éditeur, sélectionner le source de Dialog1. Allez au source de votre méthode OnButton1Button. Nous allons utiliser la même méthode «Fermer» que nous avons utilisé dans le menu 'Quitter'. Notez que cela ferme la fenêtre. La fermeture de la fenêtre rootferme l'application. Toutefois, la fermeture d'une fenêtre enfant va simplement revenir à la fenêtre parent.

```
def OnButton1Button(self, event):
   self.Close()
```

Lancez l'application. Votre nouveau dialogue devrait ressembler à ceci.



Félicitations: Vous avez construit votre première application utilisant Boa Constructor. Votre éditeur est terminé. Dans ce tutoriel, vous avez utilisé les composants de base de Boa.

Prenez le temps de revoir ce que vous avez fait jusqu'ici. Vous avez appris à:

- Créer une application.
- Créer des cadres, des menus et des barres d'état.
- Créer des contrôles tels que des boutons, des champs de saisie de texte et les étiquettes.
- Configurer les contrôles à vos besoins.
- Travailler avec dialogues communs.
- Concevez vos propres boîtes de dialogue.

Création d'une fenêtre d'application en utilisant sizers

Sizers est un excellent moyen d'obtenir une configuration graphique agréable et propre. C'est pratique quand on ignore l'espace dont a besoin un contrôle ou celui qu'il peut utiliser (c'est par exemple le cas quand on internationalise une application (I18N) ou pour des contrôles comme les listes ou les grilles auxquels on veut donner autant d'espace que possible (ou au contraire aussi peu d'espace que possible).

<note> La suite explique comment utiliser les **sizers** avec Boa (version 0.6.x). Pour plus de

détails sur les **sizers**, consulter la documentation de wxPython, la démo wxPython et les liens suivants.

- http://wiki.wxpython.org/index.cgi/UsingSizers
- http://wiki.wxpython.org/index.cgi/LearnSizers1
- http://wiki.wxpython.org/index.cgi/wxDesigner_20Sizer_20Tutorial

</note>

Nous allons utiliser un contrôle **wx.Frame** et créer un écran de saisie d'informations d'adresse.

Fermer tous les fichiers source dans l'éditeur, pour ne pas ajouter la nouvelle application à la précédente.

Dans la palette, volet **Nouveau**, Sélectionner le bouton **wx.Frame**. Cela crée un nouveau fichier source (**Frame1**).

Cliquer sur le bouton **Enregistrer** (ou menu **Fichier / Enregistrer**) et l'enregistrer sous le nom **AddressEntry.py**.

Dans le menu **Edition**, sélectionner l'option **Ajouter le module runner**. Cela ajoute un peu de code au fichier, ce qui permet de l'exécuter sans nécessiter un fichier wx.App séparé.

Sauvegarder le fichier. En exécutant l'application, on voit seulement Frame1 dans la barre de titre et un fond gris.

Sélectionner le volet **AddressEntry**. Démarrer l'éditeur graphique en cliquant sur le bouton .

Dans la palette, sélectionner le volet **Conteneurs / mise en page**. Cliquer sur le bouton **wx.Panel** pour le sélectionner et cliquer n'importe où dans le cadre de **AddressEntry** pour déposer le panneau sur le cadre.

Sur le même volet de la palette, cliquer sur le bouton **wx.BoxSizer** pour le sélectionner et

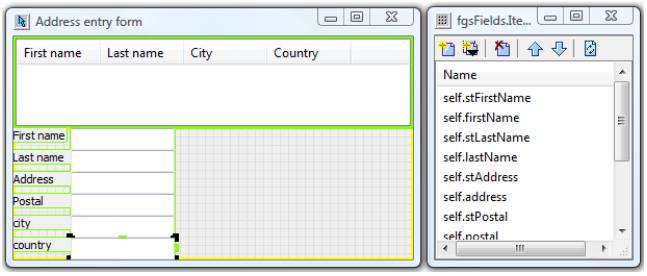
cliquer n'importe où sur le wx.Panel qui vient d'être ajouté à l'image. Une ligne jaune apparaît autour du panneau.

Valider ces changements, enregistrer le fichier et rouvrir l'éditeur graphique.

- Dans le volet sizer, cliquez sur le boxSizer1 et renommez-le, par exemple en bsMain.
- Amener l'éditeur graphique à l'avant-plan (par exemple, il suffit de cliquer sur le bouton de l'éditeur graphique de la barre d'outils).
- Sélectionnez le contrôle wx.ListCtrl sur le volet Liste contrôles et déposez-le sur l'éditeur graphique, cela va automagiquement l'ajouter au sizer bsMain.
- Dans le volet “Conteneurs / mise en page” sélectionner le wx.FlexGridSizer et déposez-le également sur le wx.Panel, which again will automagically add it to the bsMain sizer.
- Click on the “Sizers” pane and select the flexGridSizer1 and rename it to e.g. fgsFields.
- In the Inspector change the 'Cols' setting from '0' to '2' and the 'Rows' setting from '1' to '0', as we will have to columns of controls/widgets in this sizer.
- Post the changes, save the file and open l'éditeur graphique again. I do this quit regurarely to ensure that I don't loose too much of my work if something should go wrong. It is also a good idea to just run the application to see how it looks.
- In the Inspector change the Name from 'Frame1' to 'AddressEntry' and the Title from 'Frame1' to 'Address entry form'.
- Select the wx.ListCtrl in l'éditeur graphique and change the style from wx.LC_ICON to wx.LC_REPORT and on the “Props” pane click on (Columns) and then on the “…” to open l'éditeur de collection for the listctrl. Create the columns “First name, Last name, City and Country”.
- Click on the “Sizers” pane and double click on bsMain to open it's éditeur de

collection. Then click on the self.listCtrl1 and change the Border from 0 to 2 (or what you find appropriate for a border around this control) and change Flag from 0 to wx.ALL | wx.EXPAND and change Proportion from 0 to 1. These changes will ensure that you have 2 pixels space around the listctrl and that it will use up as much space as is available. If you run this little application now you will see that the listctrl takes up all the available space.

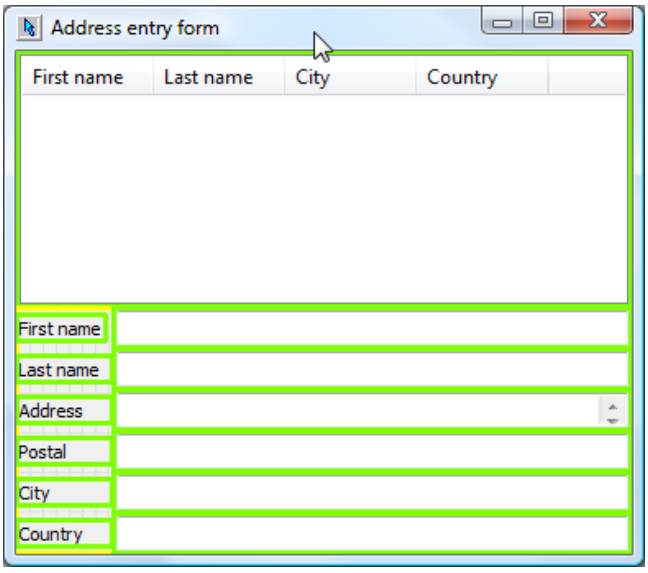
- On the “Sizers” pane open l'éditeur de collection for the fgsFields sizer and add 12 new items, when you now look at l'éditeur graphique it will show these items in red.
- From the “Basic Controls” Palette pane select the wx.StaticText control and drop it onto the top left red area and to the right of it drop a wx.TextCtrl and then repeat this until your éditeur graphique screen and the fgsFields éditeur de collection look something along these lines.



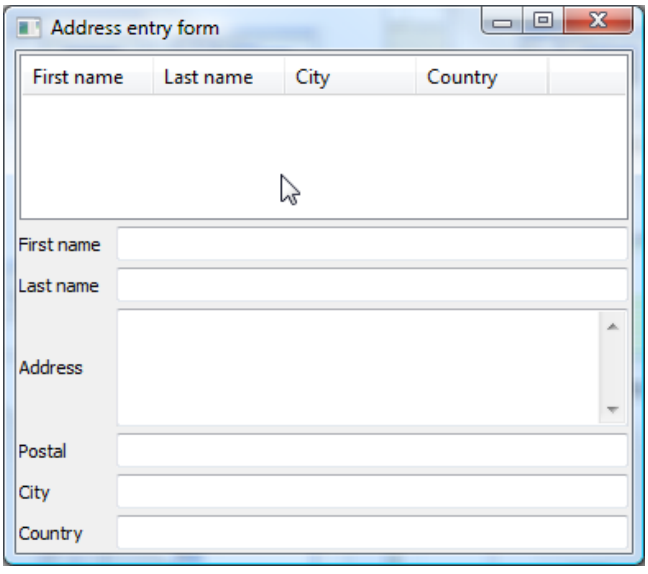
- Make sure to rename the controls to names which make sense (i.e. firstName, lastName, address, postalCode, city and country).
- Now we need to set the Border, Flag and Proportion for each of these controls.
- For wx.StaticText I suggest: 2, wx.ALL | wx.ALIGN_CENTER_VERTICAL and 0
- For wx.TextCtrl I suggest: 2, wx.ALL | wx.EXPAND and 0
- On the “Sizers” pane you need to select the fgsFields sizer and make the second column growable which you can do from the Inspector “Props” pane by clicking on

the "..." next to "(Growables)".

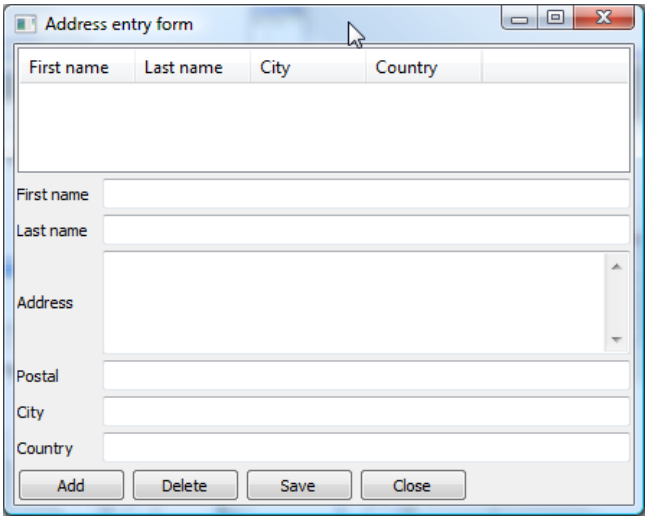
- And for this to take any effect you need to change the Flag for the fgsFields sizer in the bsMain sizer from 0 to wx.EXPAND.
- So, now you should see something like this in l'éditeur graphique.



- If you run it at this point and resize the window you can see the sizers at work.
- You might also notice that you see scrollbars on the Address field and it is larger then other fields. For this you need to change it's style from 0 to wx.TE_MULTILINE and in l'éditeur graphique you enlarge it to the hight you want to allocate for it.
- When you run the application you should see something along these lines.



- We will also need some buttons for this, so we can add, delete, save and close this form.
- For this open l'éditeur graphique again and drop another flexGridSizer (I will name it fgsButtons) onto the “Sizers” tab and then add it to the bsMain sizer.
- Then add four items to the fgsButtons sizer and then drop wx.Button controls onto the red squares on l'éditeur graphique.
- In the sizer éditeur de collection change the Border to 2, the Flag to wx.ALL for all these buttons.
- Then select the first button by double clicking its entry in l'éditeur de collection and in the Inspector “Constr” pane change the label from button1 to “” (blank) and the name from button1 to “add” and the Id to wx.ID_ADD.
- Repeat this for the others but name them delete, save and close and use the appropriate wx.ID_ entries (having access to the stock button ID's is new in Boa 0.6.0, it will only work if you blank the label.)



- You should now see something like the above when you run it.
- Obviously you only have the GUI code at this point and one would have to flesh all this with code for each of the buttons, but for the moment this goes beyond this tutorial.

Please note that the file generated during this example is also available in the directory “Examples\guide” under your Boa installation directory.

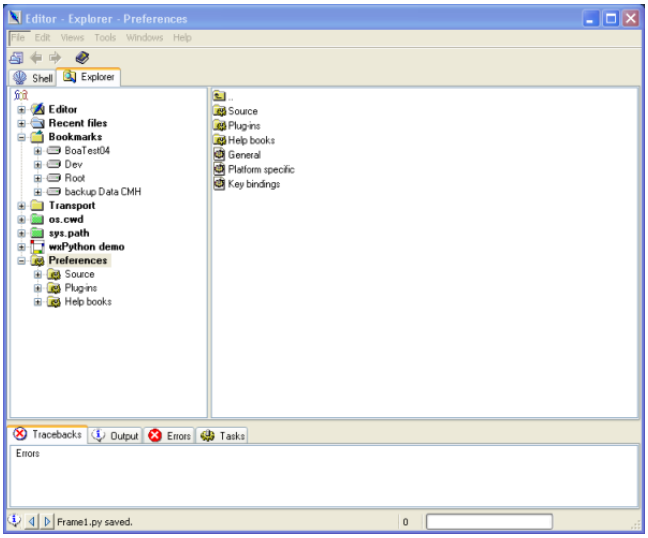
For coding guide lines you might also want to consult the wxPython style guide http://wiki.wxpython.org/index.cgi/wxPython_Style_Guide.

Other Useful Items

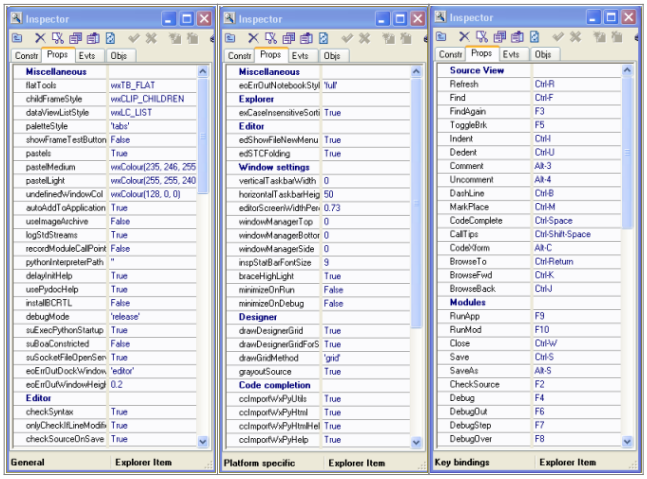
Setting Preferences

The Boa Constructor tool provides a number of features which can be customised by you.

Most of the customization settings can be set en utilisant la vue Explorateur de l'éditeur. Click on Preferences and it shows you something similar to the image below.



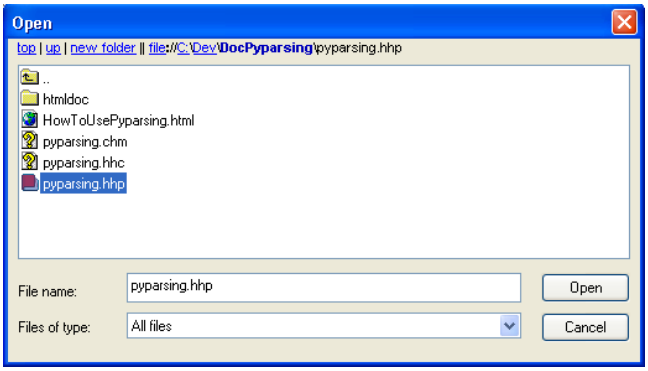
To change settings double click on either 'General', 'Platform specific' or 'Key bindings' and the properties of each will be shown in the Inspector as shown below:



Help books

Boa by default includs its help books and the ones for wxPython and Python.

However if you like to add others you can do so by selecting Preferences/Help Books and right mouse click in the right pane of the Explorer. Select 'Add new Item' and browse to find the '.hpp' file.



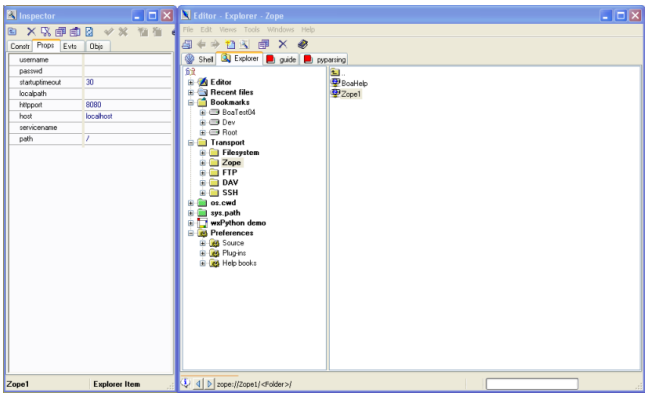
Bookmarks

If you like to add additional bookmarks just right mouse click on the folder you like to add within the Explorer.

Transport

Adding additional transports works similar to the bookmarks, select the transport type on the left hand side of the Explorer view, e.g. 'Zope' and then right mouse click in the right hand

side and select 'New' and then complete the information in the Inspector.



Module Info

You might want to change the following section in the file 'prefs.rc' stored in your user preference directory, on a Windows system this is by default in 'driveletter:\Documents and Settings\username\.boa-constructor'.

```
# Info that will be filled into the
comment block. (Edit->Add module
info)
# Also used by setup.py
staticInfoPrefs = { 'Purpose':
'',
                'Author':    '<your
name>',
                'Copyright': '(c)
2004',
                'Licence':   '<your
licence>',
                'Email':     '<your
email>',
              }
```

- Replace '<your name>' with guess what your name.
- Change the '© 2004'
- Replace '<your license>' with something along these lines 'Shareware - see license.txt for details'
- Replace '<your email>' with the email you want to use

<>If you like to edit the 'CustomModuleInfo.plug-in' file right mouse

clicking the corresponding entry in 'Preferences/Plug-ins/Plug-in files' and select 'Open Plug-in file'.

If you don't want to loose your changes when you upgrade Boa you might want to copy the 'CustomModuleInfo.plug-in' file and call it 'MyCustomModuleInfo.plug-in' in the Plug-ins directory. After restarting Boa you should disable the standard one by right mouse clicking and selecting the appropriate option.

I changed it as follows:

```
""" Demonstrates how to change
system constants as a plug-in """

import sourceconst
# The order of (Name)s may change
and lines may also be removed
sourceconst.defInfoBlock = '''# -*-
coding: iso-8859-1 -*-#
#----------------------------------
----------------------------------
--------
# Name:        %(Name)s
# Purpose:     %(Purpose)s
#
# Author:      %(Author)s
#
# Created:     %(Created)s
# RCS-ID:      %(RCS-ID)s
# Copyright:   %(Copyright)s
# Licence:     %(Licence)s
#----------------------------------
----------------------------------
--------
'''

import Preferences
# (Name)s not in the original
dictionary needs to be added
# New field:   %(NewField)s
#Preferences.staticInfoPrefs['NewFi
eld'] = 'Whatever'
```

- Added the '# -*- coding: iso-8859-1 -*-#'
- Moved the 'New field:'
- Commented the

'Preferences.staticInfoPrefs' line as I don't need it

After restarting Boa and clicking on the button  Boa will insert the following into the selected file (in this case Frame1.py).

```
# -*- coding: iso-8859-1 -*-#
#-----------------------------------
-----------------------------------
--------
# Name:        Frame1.py
# Purpose:
#
# Author:      Werner F. Bruhin
#
# Created:     2005/12/03
# RCS-ID:      $Id: node31.html,v
1.1.2.1 2005/03/14 09:23:09 wbruhin
Exp $
# Copyright:   (c) 2004 - 2005
# Licence:     Shareware, see
license.txt for details
#-----------------------------------
-----------------------------------
--------
```

Main frames

These are the main frames in Boa:

Palette	Top frame containing a palette from which new modules or components can be created.
Inspector Left frame displaying constructors/properies/events of the selected object as well as an hierarchical view of the parent/child relationship of containers.	

Editor	Big IDE window containing the shell, explorer and any number of open modules. Each module contains a notebook of supported views on the module
éditeurs graphiques	The GUI builder and other design time editors opened dans l'éditeur
Explorer	Standard Explorer type interface for interacting with various datastores like the filesystem, Zope, CVSetc.
Debugger	Debugging window, opens up over the Inspector. Supports source code tracing, breakpoints and watches.
Help	Boa, wxWidgets & Python help.

Other sections:

- Key definitions
- Windowlayout
- Different ways start or open files
- Command−line
- switches
- Preferences and Configuration
- Notes on specific components
- Support for Non−ASCIIsystems
- Zope support

- Mixing your code with generated code
- Extending Boa by writing new components, models, views and explorers
- The road ahead
- Philosophy behind Boa
- Glossary

Main windows

Palette

Creation starts here. From the first page titled Newyou can create new modules, after clicking on one un nouveau module s'ouvre dans l'éditeur. The other pages work more on the principal of cocking your cursor like in Delphi. By clicking on e.g. wxStaticText, it becomesdepressed.While an object is selected on the Palette any click in a éditeur graphique will create that object in the container or the container of the object that you clicked on of thet éditeur graphique.

Also the statusbar of the Palette shows the current selected object as a checkbox in the second column. To unselect an object uncheck this checkbox.

Dialogs will be created from a template directly in the source code of the currently active Source page at the point where the cursor is.

Note this palette page should not be used while editing in l'éditeur graphique.

Objects created from the 'Utilities' DataView. palette page should be created in the Zope objects created should be created in the Zope Explorer's list view.

Inspector

The Inspector displays the constructor parameters, properties and events of the selected object in l'éditeur graphique or

DataView. It also displays the parent child relationship of container controls when appropriate.

The selected object can be changed in the Parent view.

Whena property is selected in the inspector, an associated Property Editor will open in the Inspector for that property. Le type le plus courant de l'éditeur est un contrôle de texte used for editing strings and other builtins.

Pages

These are the pages in the inspector:

Constructor	Parameters for the constructor of the object. Someof these parameters are also accessable at run time and these will reflect changes at design−time. Other parameters like for example Style will only reflect changes the next time the frame is recreated from source, so don’t worry if you don’t see your changes immediately. This will change in the future.
Properties	This is usually the list of Get* / Set* methods an object supports. Additional properties may be defined in a objects Companionclass.
Events	This page is divided in three parts: - Categories (top left) : Logical groupings of related macros - Macros(top right) : Event connection functions of the form EVT_* - Definitions (bottom) List of defined (connected) events for the selected control. To delete select (delete) from the drop−downand it will be deleted on posting of the frame.
Parent	This page shows the parent/child relationship of container controls. By selecting a different item the Inspector and l'éditeur graphique will change their selection to the newly selected item.

Éditeur

L'éditeur est l'IDE. The top level notebook represent open modules. A module is a source code file like a python module, a Python package, an wxAppfile or any of the wxFramevariants (To create a module see Palette)

For each module the views supported by the module will be hosted on a second level notebook. Any actions applicable to the view like refreshing the source code or adding a module will be accessible by right clicking on the view or selecting from the 'Edit' menu. Most of these options are also available from the toolbar. Usually a default action will be triggered by double clicking on a view.

Modules

These are the supported modules:

Python module	Standard python source code file.
Python package	Standard python package, loads any init.py and shows it's directory as a package of modules as well as sub−directories containing a init.py file.
wxApp	The module that hosts the wxAppobject. A module list is maintained in this module. The App module controls which is the main frame of an application and currently maintains relative paths for the module list. It has a specialised Application View
PythonApp	This type is like wxApp, but for plain Python applications without wxPython file types.

wxFrame variants	Currently wrapped wxFrametype modules are wxFrame, wxDialog, wxMiniFrame wxMDIParentFrame, wxMDIChildFrame wxPopupWindow wxTransientPopupWindow. These modules support l'éditeur graphique view for visual frame creation.
Setup (Distutils)	Very basic support for Distutils. From the File menuthe following options are available: - build - clean - install - sdist - bdist - bdist_rpm - bdist_wininst py2exe is also supported (only Win32 ofcourse). Remember to add the line: import py2exe below the distutil import in your Setup.py. Add your main script, e.g. 'wxApp1.py' to the scripts argument of setup().
Other filetypes	Text − Plain text file, .txt and CAPS filenames Config

Éditeur graphique

Les éditeurs graphiques are visual design time editors for creating and manipulating components and properties.

Each éditeur graphique maintains it's own method in the generated source.

Clicking on an item in a éditeur graphique will select display that item's properties and events in the Inspector.

Double clicking will open l'éditeur par défaut (like a éditeur de collection) or define the default event for the control.

The following éditeur graphiques are currently available:

- éditeur graphique
- DataView
- éditeurs de collection

The FrameDesigner is a GUI building view on wxFrametype modules that cooperates closely with the Inspector. It is used to select, size and position controls.

It maintains the _init_ctrls method.

Multiple selection is supported by holding down shift while selecting controls. Only sibling controls can be in a multiple selection.

Selections can be cut, copied and pasted to and from the clipboard in the Frame Designer and the DataView. Whena selection is copied, the equivalent code is put on the clipboard and can also be pasted directly into your source code.

Layout Anchors have been integrated with the Frame Designer. If a control has any anchors set, it will show the selection tags as a shade of blue instead of black.

Right click on a top, bottom, left or right selection tag to set or unset the anchor.

Whena éditeur graphique is opened a session is started. Whenyou end the session, either by posting or cancelling, all your changes will be applied to the source.

Closing l'éditeur graphique frame also ends the session and posts the changes.

While l'éditeur graphique is open for a module, the source code will be read−only.

A Data View will also be opened dans l'éditeur associated with l'éditeur graphique and closed on posting / cancelling.

This will usually be the last view of the module in the éditeur named ’Data’.

It maintains the _init_utils method.

Non visual controls located on the Utilities page of the Palette must be created/inspected in this view.

Les éditeurs de collection maintain list like properties.

Selecting an item will display it’s properties and events in the Inspector.

Each maintain a _init_coll_* method.

From:
https://www.nfrappe.fr/doc-0/ - **Documentation du Dr Nicolas Frappé**

Permanent link:
https://www.nfrappe.fr/doc-0/doku.php?id=logiciel:programmation:python:boa:help:aideboa

Last update: **2022/08/13 22:27**