eBook Gratuit

APPRENEZ wxpython

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#wxpython

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: wxpython

It is an unofficial and free wxpython ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official wxpython.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec wxpython

Remarques

Qu'est-ce que wxPython

En termes simples, wxPython est un ensemble de liaisons à la bibliothèque d'interfaces graphiques C ++ Cross Platform de **wxWidgets** .

Ok ce qui est wxWidgets

La bibliothèque wxWidgets fournit un ensemble gratuit, gratuit et gratuit, d'abstractions pour les différents éléments de l'interface graphique, afin que les contrôles natifs soient toujours utilisés, le cas échéant, en conservant l'aspect, la rapidité et l'aspect natifs. En tant que tel, il fournit une abstraction pour la création d'interface graphique et de nombreux autres utilitaires sur une plate-forme qui permet aux développeurs de créer des applications pour Windows, Mac OS X, Linux et d'autres plates-formes utilisant une seule base de code. wxWidgets a été lancé en 1992 et vous pouvez voir un historique détaillé ici . La bibliothèque wxWidgets est distribuée sous la licence wxWindows, qui est basée sur la **L-GPL mais avec une clause d'exception** . *La clause d'exception vous permet de lier dynamiquement ou statiquement votre application à wxWidgets* ***sans*** *devoir distribuer la source pour votre propre application. En d'autres termes, vous pouvez utiliser wxWidgets pour* ***des*** *projets* ***gratuits ou commerciaux*** *,* ***sans frais*** *. La licence vous encourage à redonner des améliorations à la bibliothèque wxWidgets elle-même.*

Les points saillants, *notez que wxWidgets comprend des centaines de classes pour le développement d'applications multiplate-forme* :

- Disposition de la fenêtre à l'aide de Sizers
- Contextes de périphériques (avec stylos, pinceaux et polices)
- Système complet de gestion des événements
- Visualiseur d'aide HTML
- Lecture sonore et vidéo
- Prise en charge d'Unicode et d'internationalisation
- Architecture de document / vue
- Archiecture d'impression
- Douilles
- Multithreading
- Manipulation de fichiers et de répertoires
- Aide en ligne et contextuelle
- Rendu HTML
- Conteneurs de base
- Chargement, sauvegarde, dessin et manipulation d'images
- Date et heure de la bibliothèque et des minuteries

- La gestion des erreurs
- Presse-papiers et glisser-déposer

Notez que certaines de ces fonctionnalités, *par exemple le threading,* ne sont pas réellement liées à l'interface graphique, mais fournissent une abstraction utile entre plates-formes afin que, par exemple, un ensemble de code d'application fonctionne sur n'importe quelle plate-forme.

Pendant de nombreuses années, la bibliothèque wxWidgets a produit 4 versions distinctes, *en plus des versions* de *débogage* d'un ensemble de code source, de bibliothèques statiques et dynamiques conçues pour ASCII et Unicode. Il est généralement disponible pré-construit dans les variantes les plus courantes et en tant que code source à construire *avec les différentes options* pour l'environnement cible et avec la chaîne d'outils C ++ des développeurs avec de nombreuses chaînes d'outils prises en charge.

Les liaisons python pour cette bibliothèque et certains ajouts forment wxPython.

Retour à Qu'est-ce que wxPython, (qu'est-ce que ça me donne)?

wxPython permet au développeur de bénéficier d'une bibliothèque d'interface graphique multi-plateforme, avec une licence claire, tout en offrant les avantages de Python. Comme wxWidgets et Python, wxPython est gratuit, gratuit et open source, et peut être utilisé et distribué dans des projets gratuits et commerciaux *sans qu'il soit nécessaire de distribuer votre code source* .

- Suite graphique complète incluant (mais sans s'y limiter):
  - Windows (y compris MDI Windows)
  - Des magiciens
  - Cadres et MiniFrames
  - Dialogues, Standard, Avancé et Personnalisé
  - Livres, arbres, grilles et contrôles d'affichage des données
  - Jauges, Sliders, Spinners, Animations, Presse-papiers, Drag & Drop
  - Prise en charge du visualiseur HTML, PDF et image
  - Les composants de l'interface graphique peuvent être positionnés de manière absolue, mais il est fortement recommandé d'utiliser une mise en page basée sur le dimensionnement qui prend en charge le dimensionnement automatique, etc.
- Cross Platform - Support des interfaces graphiques pour Windows, OS-X et Linux avec une base de code unique *sans instructions conditionnelles dans votre code*
- Vitesse native, look & feel.
- Prototype rapide, test et débogage - *rappelez - vous qu'il s'agit de python*
- Exécuter et éditer des échantillons de presque tout dans le package de démonstration.
- Licence claire pour une utilisation gratuite même dans les produits commerciaux.
- Si nécessaire, votre interface graphique python peut être modifiée en une interface graphique C ++ wxWidgets ultérieurement, car elle l'utilise déjà.
- Large communauté d'utilisateurs et de développeurs, active et utile, à la fois sur StackOverflow et sur les listes de diffusion .

Notez que lorsque python lui-même fournit un mécanisme multi-plateforme pour implémenter les fonctions utilitaires de wxWidgets, le *nouveau threading étant un bon exemple* , il est **intentionnellement** omis de wxPython.

wxPython possède également une très grande suite de démonstrations pouvant être exécutées, testées et éditées à partir du package Documents and Demo.

Saveurs de wxPython

ASCII vs Unicode :

Pendant de nombreuses années, *comme avec wxWidgets* , les développeurs devaient choisir entre les versions ASCII et Unicode, ainsi que leur version spécifique de python et les options 32/64 bits. À partir de wxPython 2.8.9, la version ASCII uniquement de wxPython a été supprimée. Le support Unicode est donc toujours disponible.

Classique vs. Phoenix :

Depuis wxPython 3.0.0, il y a eu la *version* "Classic" de wxPython et une version Phoenix *inédite* . La version classique a tendance à prendre du retard par rapport aux versions wxWidgets des mêmes numéros et le package de documentation est C ++ - il est disponible en téléchargement pour différentes plates-formes (voir Installation de Classic), dans le cas de Windows en tant qu'installateur exécutable. Les liaisons Phoenix, générées en grande partie automatiquement, devraient suivre de plus près les versions de wxWidgets et inclure également la documentation spécifique à wxPython - elle peut être générée à partir de builds sources ou nocturnes *car des roues* peuvent être obtenues avec **pip** (voir Installation de Phoenix).

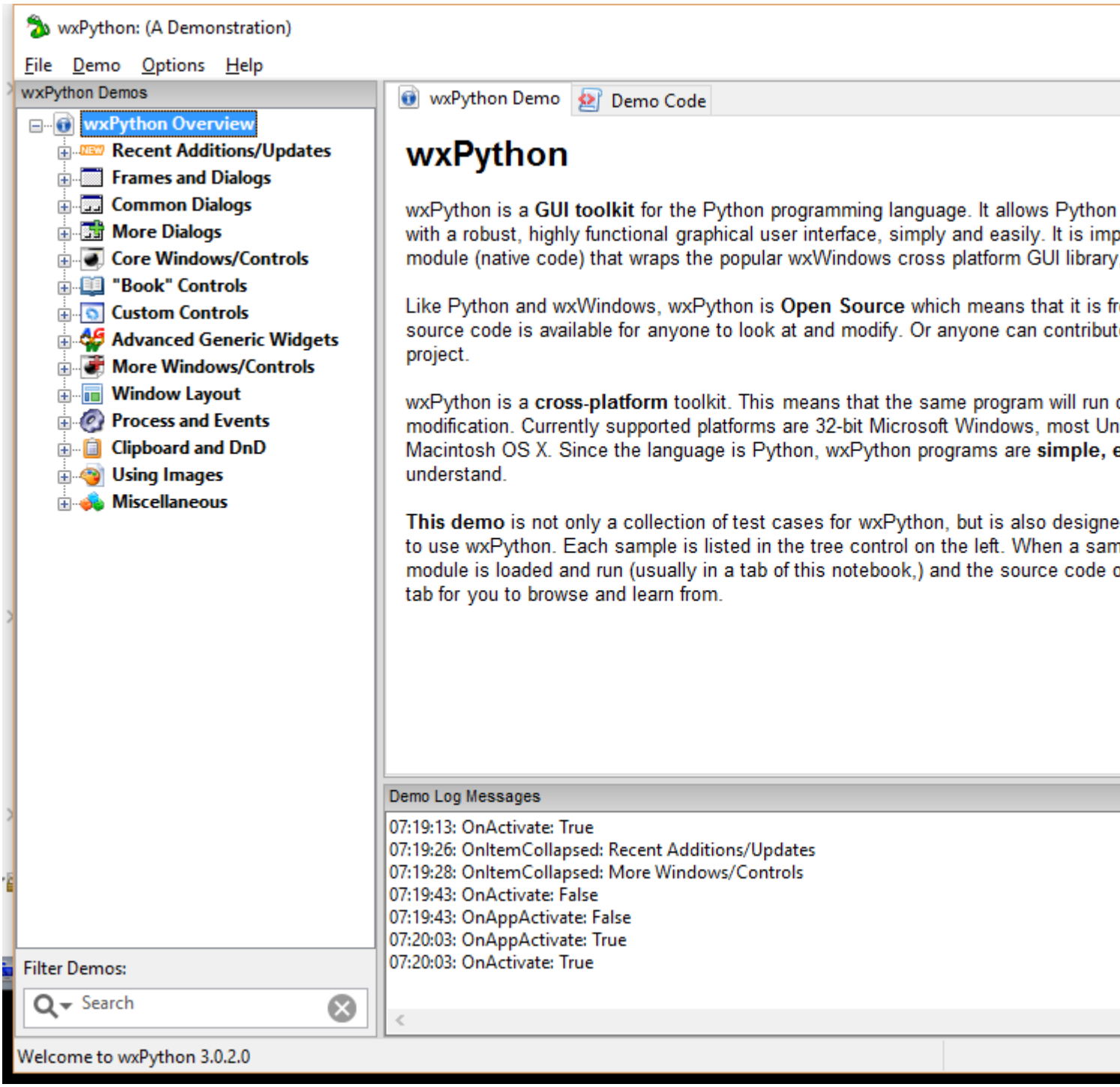
Dans wxPython mais pas wxWidgets

wxPython étend la bibliothèque wxWidgets avec un certain nombre de fonctionnalités, *les suivantes ne sont que quelques unes,* non disponibles dans wxWidgets:

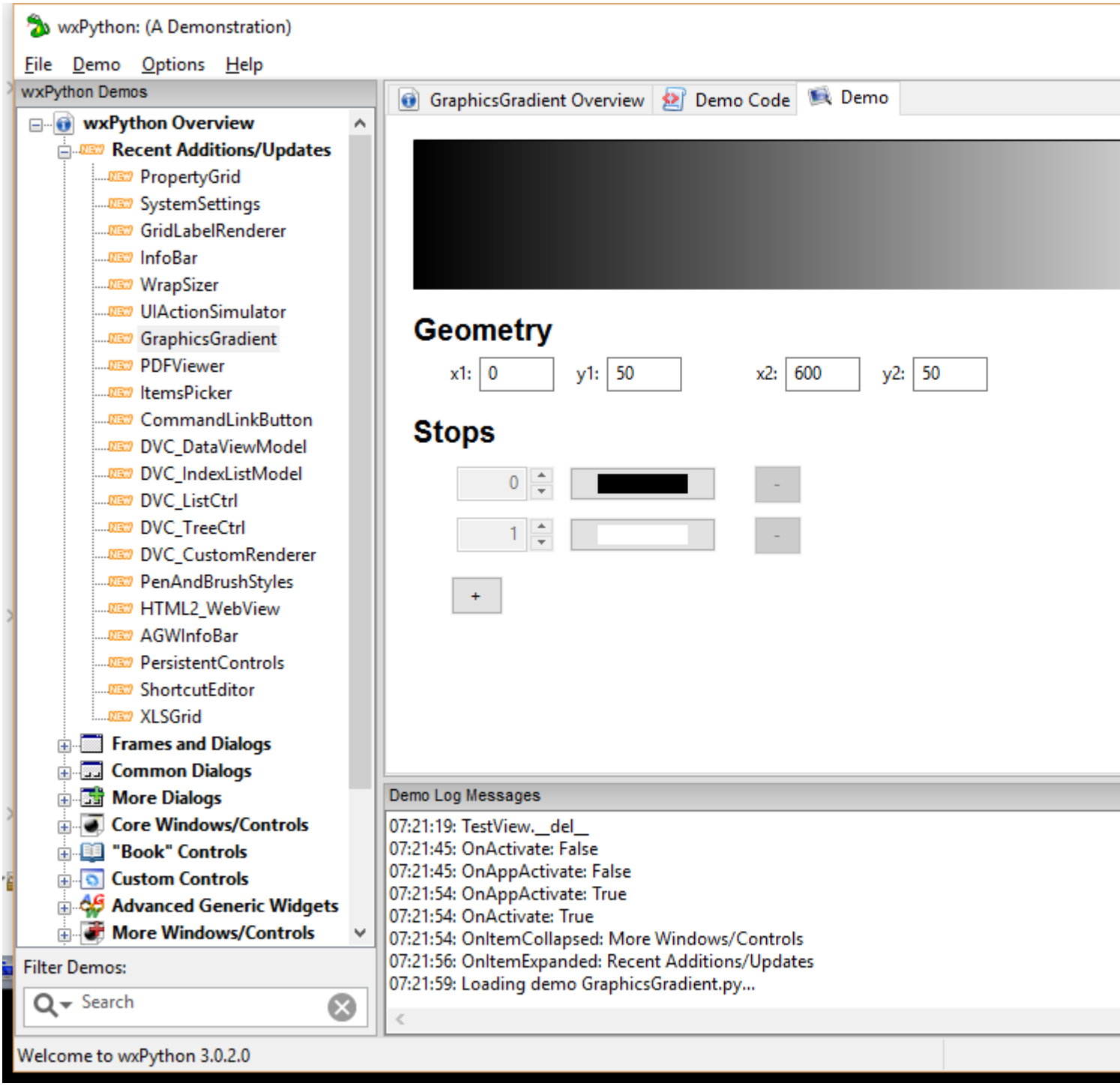
- Programmeurs Editeurs & Coquilles: crust , crustslices , AlaCart & AlaMode , AlaModeTest
- Interprète et magie
- Inspection - cela vous permet de lancer une fenêtre pour parcourir tous les composants de votre interface graphique.
- Un ensemble complet de démos

Captures *d'* écran de démonstration *sur Win10*

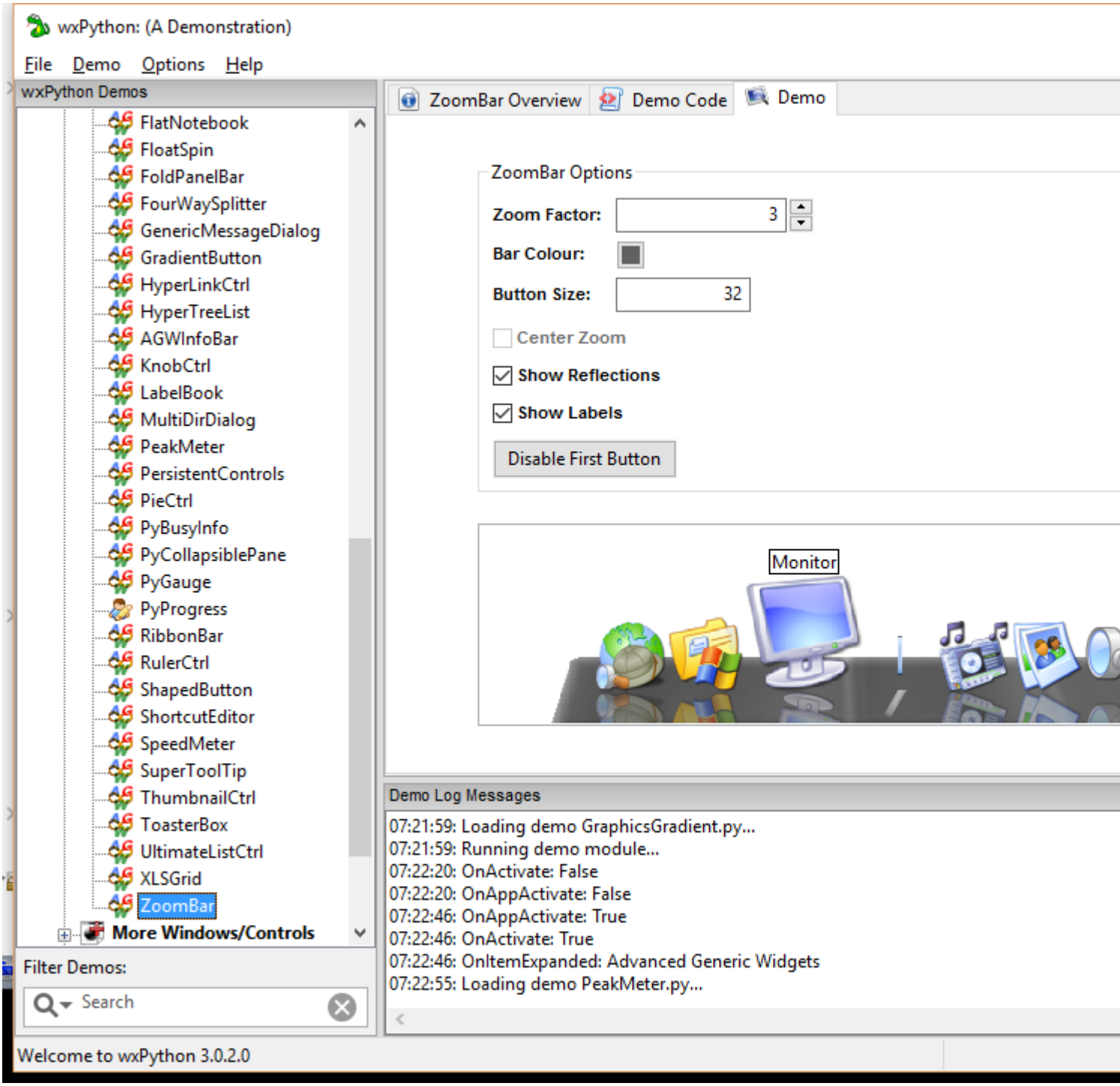
La démo wxPython avec toutes les branches fermées:



Un des ajouts récents:



Un des AGW (Advanced Generic Widgets):



Examples

Installation de wxPython Phoenix

wxPython Phoenix est la dernière version de wxPython, (actuellement *septembre 2016* sans version officielle). Il supporte à la fois Python 2 et Python 3. Vous pouvez télécharger une version instantanée (une roue Python) pour votre plate-forme et la version Python ici .

wxPython Phoenix utilise un mécanisme largement automatisé pour générer à la fois les liaisons python pour la bibliothèque wxWidgets et la documentation. La documentation de Phoenix wxPython est spécifiquement générée pour lui-même en utilisant Sphinx . Cela augmente la clarté

par rapport à la documentation C ++ de la version classique, qui inclut de nombreuses surcharges qui ne sont pas disponibles dans wxPython.

Python et **pip** doivent être installés avant l'installation de wxPython Phoenix.

Vous pouvez utiliser pip pour installer la version Phoenix de wxPython. Voici la méthode recommandée actuellement:

```
python -m pip install --no-index --find-links=http://wxpython.org/Phoenix/snapshot-builds/ --
trusted-host wxpython.org wxPython_Phoenix
```

Lorsque vous utilisez cette commande, pip installe également **wxWidgets** . Cette commande complexe de pip deviendra probablement 'pip install wxpython' lorsque Phoenix sera officiellement publié.

Note: wxPython Phoenix est actuellement en version bêta et ne possède pas tous les widgets de la version Classic.

Installation de wxPython Classic

wxPython Classic est une version **Python 2** de la bibliothèque wxPython. La génération des liaisons python nécessite un grand nombre d'interventions manuelles et la documentation est simplement la documentation de wxWidgets qui contient des annotations sur les mécanismes wxPython. Il y a normalement un délai de quelques semaines à quelques mois entre une nouvelle version de wxWidgets et la version correspondante de wxPython .

Accédez à la page de téléchargement du site Web wxPython pour voir s'il existe déjà une version de wxPython que vous pouvez télécharger pour votre plate-forme.

La dernière version de Classic est **3.0.2.0**

les fenêtres

Il existe des installateurs pour Python 2.6 et 2.7 pour les plates-formes Windows 32 bits et 64 bits sur le site Web. Il suffit de télécharger l'un d'entre eux et de les exécuter pour l'installer.

Remarque: Assurez-vous de télécharger un programme d'installation wxPython pour le bon Python que vous avez installé. Par exemple, si vous avez Python 2.7 32 bits, vous voulez un installateur 32 bits wxPython

Mac

Si vous avez OSX **10.5 ou supérieur** , vous voudrez télécharger et installer la version **Cocoa** de wxPython. La version Cocoa prend également en charge le Mac 64 bits.

Si vous avez un Mac avec une version d'OSX inférieure à **10.5** , alors vous voudrez la construction **Carbon** .

Linux

La première chose à vérifier si le gestionnaire de paquets de votre plate-forme Linux (yum, apt-get, etc.) pour voir s'il a une version de wxPython que vous pouvez installer. Malheureusement, beaucoup de paquets Linux pour wxPython sont pour la version 2.8.12.1 au lieu de 3.0.2.0. Si votre gestionnaire de paquets ne dispose pas de la dernière version, vous devrez probablement le créer vous-même.

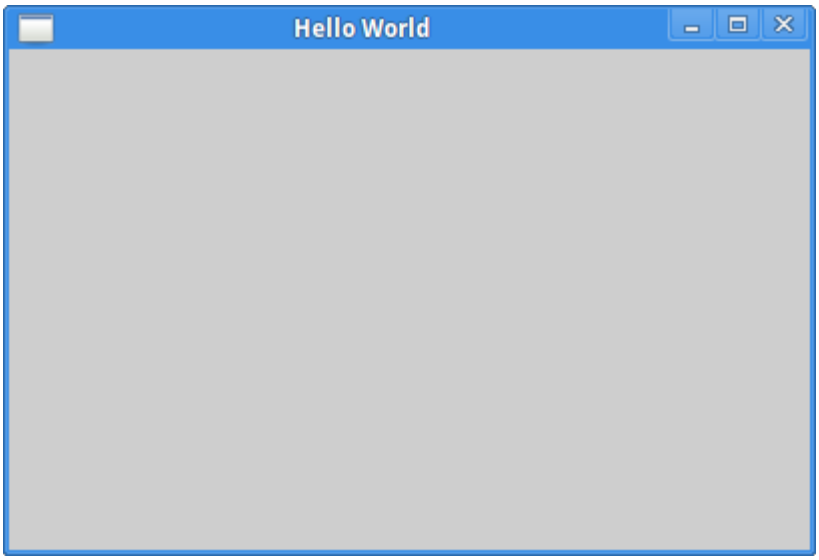
Il y a des instructions de compilation pour 3.0.2.0-Classic ici

Bonjour le monde

Un moyen simple de créer un programme **Hello World** :

```
import wx
app = wx.App(redirect=False)
frame = wx.Frame(parent=None, id=wx.ID_ANY, title='Hello World')
frame.Show()
app.MainLoop()
```

Sortie:



Un exemple plus typique serait la sous-classe **wx.Frame** :

```
import wx

class MyFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, title='Hello World')
        self.Show()

if __name__ == '__main__':
    app = wx.App(redirect=False)
    frame = MyFrame()
    app.MainLoop()
```

Cela peut également être réécrit pour utiliser le **super** de Python:

```
import wx

class MyFrame(wx.Frame):

    def __init__(self, *args, **kwargs):
        """Constructor"""
        super(MyFrame, self).__init__(*args, **kwargs)
        self.Show()

if __name__ == '__main__':
    app = wx.App(False)
    frame = MyFrame(None, title='Hello World')
    app.MainLoop()
```

Qu'est-ce qu'une série de versions de wxPython?

Le projet wxWidgets a adopté le modèle de version utilisé par le projet Linux Kernel où il existe des ensembles alternatifs de versions dans lesquels un ensemble est considéré comme "stable" et le prochain est considéré comme "développement". Pour wxWidgets, "stable" et "development" ne font pas référence à la bêtise, mais à la stabilité de l'API et à la rétrocompatibilité.

- **Stable** : pendant la durée de la série, les API existantes ne sont pas modifiées, bien que de nouvelles méthodes de classes non virtuelles puissent être ajoutées. La compatibilité binaire des librairies C ++ est maintenue en ne permettant aucune modification modifiant la taille en mémoire ou la disposition des classes et des structures. Cela peut et impose souvent des limitations sur les types d'améliorations ou de corrections de bogues pouvant être apportés dans une série stable, mais cela n'affecte vraiment que la couche C ++, car la compatibilité ascendante de Python a des connotations légèrement différentes.

- **Développement** : L'objectif principal de la série de développement des versions est d'ajouter de nouvelles fonctionnalités ou de corriger des problèmes qui n'ont pas pu être corrigés dans une série stable en raison de problèmes de compatibilité binaire, le tout pour créer la prochaine série stable. Ainsi, pendant toute la durée de la série de développement, les API peuvent être modifiées ou supprimées selon les besoins, bien que la plupart du temps, la compatibilité C ++ au niveau des sources soit maintenue via des fonctions ou des macros surchargées, etc. être incompatibilité au niveau de la source car il n'y a pas de surcharge ou de macros, et pour prendre en charge la nouvelle version de l'API, il faut parfois supprimer l'ancienne version.

En raison des problèmes de compatibilité binaire, la dernière version de développement de wxWidgets / wxPython peut souvent être moins buggée que la dernière version de la dernière version stable. Cependant, il existe un compromis selon lequel les API peuvent changer ou évoluer entre les versions de la série de développement.

Comment fonctionnent les numéros de version?

Pour les versions, wxPython utilise un numéro de version à 4 composants. Bien que cela ressemble beaucoup à la manière dont les numéros de version sont utilisés dans d'autres projets Open Source, il existe quelques différences subtiles. Donc, pour une version **ABCD,** vous pouvez en déduire:

1. **Release Series** : Les deux premiers composants du numéro de version (**AB**) représentent la série de versions, et si le composant **B** est un nombre pair, il s'agit d'une série stable. S'il s'agit d'un nombre impair, il s'agit d'une série de développement. Par exemple, les versions 2.4, 2.6 et 2.8 sont stables et l'API est plus ou moins figée dans chaque série, et 2.3, 2.5 et 2.7 sont en développement et l'API et les fonctionnalités peuvent changer ou évoluer en fonction des besoins.

De ce fait, il peut y avoir des changements assez importants entre deux séries stables (disons de 2,4 à 2,6), ce qui désactive souvent les utilisateurs car, dans d'autres projets, les modifications de cette magnitude auraient provoqué la modification du premier composant du numéro de version. Au lieu de cela, vous devriez considérer la combinaison de **AB** comme le numéro majeur de la version.

2. **Release Number** : Le troisième composant du numéro de version (C) représente l'une des versions d'une série de versions. Par exemple, 2.5.0, 2.5.1, 2.5.2, 2.5.3 ... sont toutes les versions de la série 2.5. (Et comme dans ce cas il s'agit d'une série de développement, l'API et les fonctionnalités de la version 2.5.3 ont évolué pour être différentes par rapport à la version 2.5.0).

3. Numéro de sous-version ou version de wxPython: Le quatrième composant du numéro de version (D) est utilisé pour représenter une sous-version, ou des versions incrémentielles entre les versions officielles de wxWidgets. Ces versions incluent des correctifs pour les bogues de wxWidgets que wxPython peut avoir exposés, ou des améliorations mineures importantes pour wxPython. Ce n'est pas un instantané wxWidgets arbitraire, mais plutôt une version testée du code avec des correctifs et des améliorations qui ne sont pas encore disponibles à partir de wxWidgets, à l'exception du référentiel de code source.

Source: https://wiki.wxpython.org/ReleaseSeries

Lire Démarrer avec wxpython en ligne: https://riptutorial.com/fr/wxpython/topic/6690/demarrer-avec-wxpython

Chapitre 2: Glisser déposer

Introduction

wxPython fournit plusieurs types de glisser-déposer. Vous pouvez avoir l'un des types suivants: `wx.FileDropTarget` , `wx.TextDropTarget` ou `wx.PyDropTarget` .

Les deux premiers sont assez explicites. Le dernier, wx.PyDropTarget, est juste un wrapper lâche autour de wx.DropTarget lui-même. Il ajoute quelques méthodes de commodité supplémentaires que le simple wx.DropTarget n'a pas. Nous allons commencer avec un exemple wx.FileDropTarget.

Examples

FileDropTarget

```
import wx


class MyFileDropTarget(wx.FileDropTarget):
    """"""

    def __init__(self, window):
        """Constructor"""
        wx.FileDropTarget.__init__(self)
        self.window = window

    def OnDropFiles(self, x, y, filenames):
        """
        When files are dropped, write where they were dropped and then
        the file paths themselves
        """
        self.window.SetInsertionPointEnd()
        self.window.updateText("\n%d file(s) dropped at %d,%d:\n" %
                               (len(filenames), x, y))
        for filepath in filenames:
            self.window.updateText(filepath + '\n')

        return True


class DnDPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)

        file_drop_target = MyFileDropTarget(self)
        lbl = wx.StaticText(self, label="Drag some files here:")
        self.fileTextCtrl = wx.TextCtrl(self,
                                        style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        self.fileTextCtrl.SetDropTarget(file_drop_target)
```

```
        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(lbl, 0, wx.ALL, 5)
        sizer.Add(self.fileTextCtrl, 1, wx.EXPAND|wx.ALL, 5)
        self.SetSizer(sizer)

    def SetInsertionPointEnd(self):
        """
        Put insertion point at end of text control to prevent overwriting
        """
        self.fileTextCtrl.SetInsertionPointEnd()

    def updateText(self, text):
        """
        Write text to the text control
        """
        self.fileTextCtrl.WriteText(text)


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="DnD Tutorial")
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

TextDropTarget

```
import wx


class MyTextDropTarget(wx.TextDropTarget):

    def __init__(self, textctrl):
        wx.TextDropTarget.__init__(self)
        self.textctrl = textctrl

    def OnDropText(self, x, y, text):
        self.textctrl.WriteText("(%d, %d)\n%s\n" % (x, y, text))
        return True

    def OnDragOver(self, x, y, d):
        return wx.DragCopy


class DnDPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)
```

```
        lbl = wx.StaticText(self, label="Drag some text here:")
        self.myTextCtrl = wx.TextCtrl(
            self, style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        text_dt = MyTextDropTarget(self.myTextCtrl)
        self.myTextCtrl.SetDropTarget(text_dt)

        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(self.myTextCtrl, 1, wx.EXPAND)
        self.SetSizer(sizer)

    def WriteText(self, text):
        self.text.WriteText(text)


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(
            self, parent=None, title="DnD Text Tutorial")
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

PyDropTarget

```
import  wx


class MyURLDropTarget(wx.PyDropTarget):

    def __init__(self, window):
        wx.PyDropTarget.__init__(self)
        self.window = window

        self.data = wx.URLDataObject();
        self.SetDataObject(self.data)

    def OnDragOver(self, x, y, d):
        return wx.DragLink

    def OnData(self, x, y, d):
        if not self.GetData():
            return wx.DragNone

        url = self.data.GetURL()
        self.window.AppendText(url + "\n")

        return d
```

```
class DnDPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)
        font = wx.Font(12, wx.SWISS, wx.NORMAL, wx.BOLD, False)

        # create and setup first set of widgets
        lbl = wx.StaticText(self,
                            label="Drag some URLS from your browser here:")
        lbl.SetFont(font)
        self.dropText = wx.TextCtrl(
            self, size=(200,200),
            style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        dt = MyURLDropTarget(self.dropText)
        self.dropText.SetDropTarget(dt)
        firstSizer = self.addWidgetsToSizer([lbl, self.dropText])

        # create and setup second set of widgets
        lbl = wx.StaticText(self, label="Drag this URL to your browser:")
        lbl.SetFont(font)
        self.draggableURLText = wx.TextCtrl(self,
                                            value="http://www.mousevspython.com")
        self.draggableURLText.Bind(wx.EVT_MOTION, self.OnStartDrag)
        secondSizer = self.addWidgetsToSizer([lbl, self.draggableURLText])

        # Add sizers to main sizer
        mainSizer = wx.BoxSizer(wx.VERTICAL)
        mainSizer.Add(firstSizer, 0, wx.EXPAND)
        mainSizer.Add(secondSizer, 0, wx.EXPAND)
        self.SetSizer(mainSizer)

    def addWidgetsToSizer(self, widgets):
        """
        Returns a sizer full of widgets
        """
        sizer = wx.BoxSizer(wx.HORIZONTAL)
        for widget in widgets:
            if isinstance(widget, wx.TextCtrl):
                sizer.Add(widget, 1, wx.EXPAND|wx.ALL, 5)
            else:
                sizer.Add(widget, 0, wx.ALL, 5)
        return sizer

    def OnStartDrag(self, evt):
        """"""
        if evt.Dragging():
            url = self.draggableURLText.GetValue()
            data = wx.URLDataObject()
            data.SetURL(url)

            dropSource = wx.DropSource(self.draggableURLText)
            dropSource.SetData(data)
            result = dropSource.DoDragDrop()


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
```

```
        """Constructor"""
        wx.Frame.__init__(self, parent=None,
                          title="DnD URL Tutorial", size=(800,600))
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

Lire Glisser déposer en ligne: https://riptutorial.com/fr/wxpython/topic/9709/glisser-deposer

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec wxpython	4444, Boštjan Mejak, Community, Mike Driscoll, Steve Barnes
2	Glisser déposer	Mike Driscoll