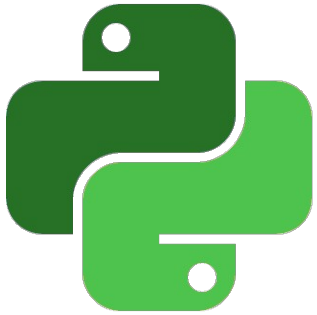


Modules Python

09-Interfaces graphiques : WX

- Introduction.
- Les bases.
- Menus & Barres d'outils.
- Mise en page.
- Gestion des événements.
- Boites de dialogue standards.
- Drag & Drop.
- Composants avancés.
- Dessin.
- Personnalisation & Divers.

Simon CHOLLET - simon.chollet@ens.fr

Introduction

- Généralités.
- Modules internes.

Généralités

- **wx** : Framework multi-plateformes pour développer des interfaces graphiques.
- Elle est basée sur wxWidgets programmée en C++.
- **5** modules :
  - **Windows** : Fenêtres, boites de dialogue, frames, etc.
  - **GDI** : **G**raphics **D**evice **I**nterface, utilisé pour dessiner.
  - **Core** : Objets de base, parents des autres.
  - **Misc** : Logging, joysticks, config. applications, etc.
  - **Controls** : Boutons, toolbar, etc.

Classes internes

- **Widgets de base** :

  Composants graphiques de base.

  Non appelable directement.

- **Top level Widgets** :

  Composants graphiques de haut niveau.

Classes internes

- **Conteneurs** :

  Composants graphiques pour contenir d'autres composants.

  - wx.ScrolledWindow
  - wx.Panel
  - wx.SplitterWindow
  - wx.Notebook

- **Widgets dynamiques** :

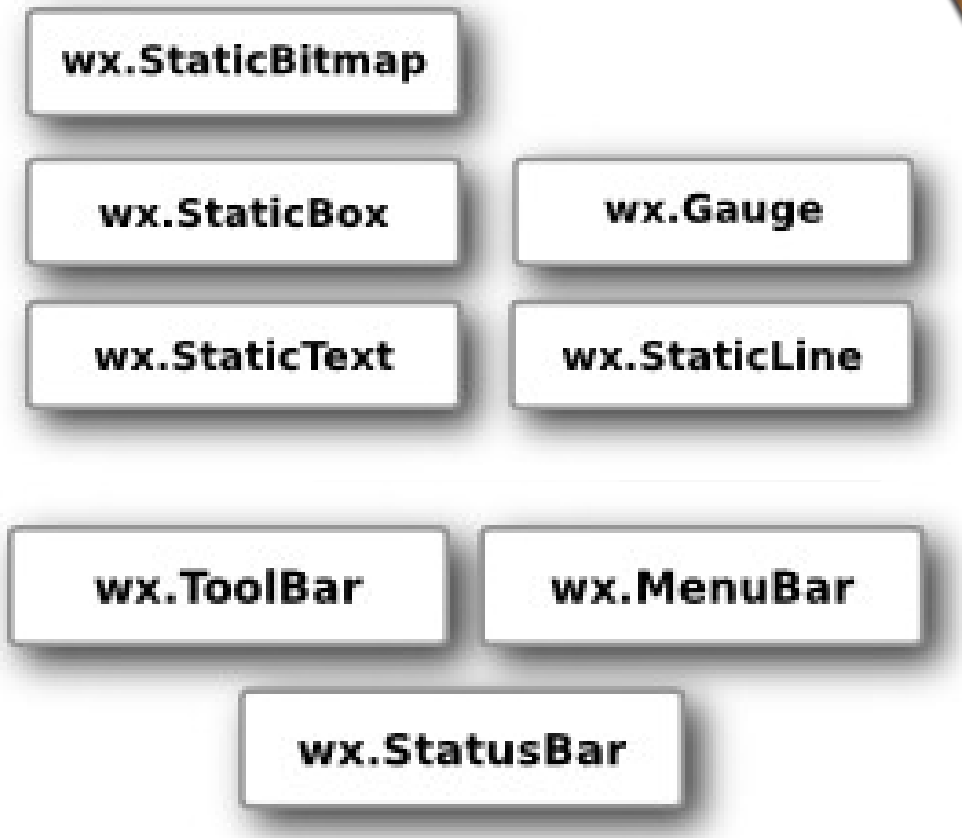
  Composants graphiques éditables par l'utilisateur.

  - wx.ToggleButton
  - wx.CheckBox
  - wx.TextCtrl
  - wx.SpinCtrl
  - wx.ComboBox
  - wx.BitmapButton
  - wx.Slider
  - wx.Choice
  - wx.RadioButton
  - wx.Button
  - wx.ScrollBar
  - wx.Grid
  - wx.RadioBox
  - wx.SpinButton
  - wx.ListBox

Classes internes

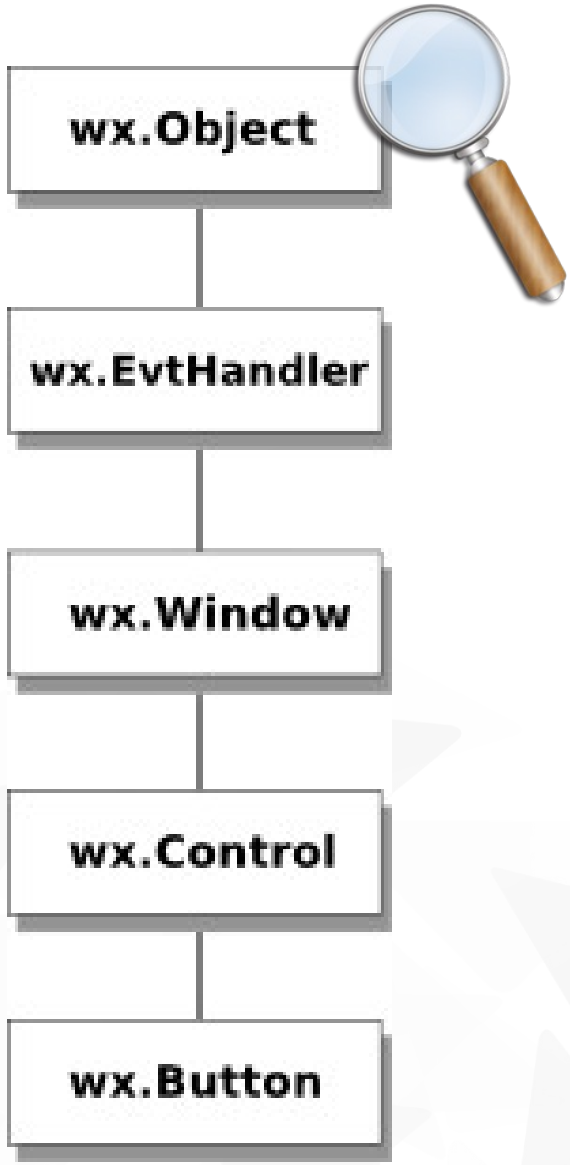
- **Widgets statiques** :
  Composants graphiques pour afficher des informations.

- **Autres Widgets** :
  Les inclassables …



Classes internes

- **Héritage** :
  Tous les composants → classe mère : « `wx.Object` ».
- Dépend des types de composants.
- **Exemple** : « `wx.Button` ».



Les bases

- Fenêtre simple.
- Composants standards.

Fenêtre simple

- **App()** : Créer l'**application principale** (unique).
- **`Frame()`** : **Créer** une fenêtre.
- **`Show()`** : **Afficher** la fenêtre.
- **`MainLoop()`** : **Lancer** l'application.
- **`Move()`** / **`MoveXY()`** : **Positionner** une fenêtre.
- **`SetPosition()`** : **Positionner** une fenêtre.
- **`SetDimensions()`** : **Positionner** et **dimensionner** une fenêtre.
- **`Center()`** : **Centrer** la fenêtre sur l'écran.
- **`GetClassName()`** : Récupérer la **classe** d'un objet.
- **`GetValue()`** : Récupérer la **valeur** d'un objet.

Composants standards

- **`Button()`** : Bouton **classique**.
- **`ToggleButton()`** : Bouton à **deux états**.
- **`RadioButton()`** : Bouton **radio**.
- **`StaticLine()`** : **Ligne** horizontale ou verticale.
- **`StaticText()`** : **Etiquette** de texte.
- **`StaticBox()`** : **Boite** encadrée.
- **`ComboBox()`** : Boite de liste de choix.
- **`CheckBox()`** : **Case** à cocher.
- **`StatusBar()`** : Barre de **statut**.
- **`Gauge()`** : Barre de **progression**.
- **`Slider()`** : Potentiomètre linéaire.
- **`SpinCtrl()`** : « **Roulette** » numérique.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_1.py` .. `code_wx_6.py` ».
- **Modifier** des paramètres et tester les effets.

Les Menus

- Menus.
- Barre de statut.
- Menu contextuel.

Les menus et sous-menus

- **`MenuBar()`** : Créer une **barre de menu**.
- **`Menu()`** : Créer un **menu**.
- **`MenuItem()`** : Créer un **élément** de menu.
- **`Append()`** : **Ajouter** des entrées à un élément « menu ».
- **`Bind()`** : **Associer** une méthode à un élément.
- **`SetMenuBar()`** : **Associer** la barre de menu à la fenêtre.

Les menus et sous-menus

- **`MenuBar()`** : Créer une **barre de menu**.
- **`Menu()`** : Créer un **menu**.
- **`MenuItem()`** : Créer un **élément** de menu.
- **`Append()`** : **Ajouter** des entrées à un élément « menu ».
- **`Bind()`** : **Associer** une méthode à un élément.
- **`SetMenuBar()`** : **Associer** la barre de menu à la fenêtre.
- **`AppendSeparator()`** : Ajouter un **séparateur**.
- Possible :
  - De régler des **raccourcis** clavier.
  - De régler des **icônes** sur les actions.
- Il existe des identifiants « classiques » :

  `ID_NEW`, **`ID_OPEN`**, **`ID_SAVE`**, etc.

Les barres d'Outils – Statut

- **`ITEM_CHECK`** : **Case à cocher** dans un menu.
- **`ITEM_RADIO`** : **Bouton radio** dans un menu.
- **`CreateToolBar()`** : Créer une **barre d'outils**.
- **`AddLabelTool()`** : **Ajouter** un élément à une barre d'outils.
- **`Realize()`** : **Dessiner** la barre d'outils.
- **`CreateStatusBar()`** : Créer une barre de **statut**.

Menu contextuel

- **`PopupMenu()`** : Créer un **menu contextuel**.
- Principe :
  - Créer une **classe** qui hérite de « **`wx.Menu`** ».
  - Rajouter des éléments.
  - **Associer** des actions aux éléments.
  - Créer une action dans l'application principale qui appelle ce menu contextuel.

Finalement ...

- **Beaucoup** de choses existent !
- Possible d'ajouter des éléments par d'**autres méthodes** plus « sélectives » : `AppendCheckItem()`, etc.
- J'ai choisi de prendre toujours la **même méthode** « `Append()` » qui rend le code moins lisible, mais les modifications plus rapides.
- A vous de sentir **comment coder** vos fenêtres.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « **`code_wx_7.py`** .. **`code_wx_12.py`** ».
- **Modifier** des paramètres et tester les effets.

Mise en page

- Mise en page absolue.
- Arrangements vertical / horizontal.
- Arrangements en grille.

Mise en page absolue

- On **choisit** la taille et la position de chaque composant.
- Ces dimensions **n'évoluent pas** si redimensionnement.
- Les fenêtres n'auront **pas la même forme** sur d'autres OS.
- Le **changement de taille** de polices pose problème.
- Placement **figé** → si modification, refonte totale nécessaire.
- **`Panel()`** : Créer un **conteneur** de composants.

Arrangements verticaux / horizontaux

- **Aucun** : Inclut dans la Frame, elle-même.
- **BoxSizer()** : **Plusieurs** composants → **colonne** / **ligne**.
- **Add()** : **Ajouter** un composant dans le conteneur.
  - **EXPAND** : Occupe **toute** la place disponible.
  - **ALIGN_LEFT** : Aligner à **gauche**.
  - **ALIGN_RIGHT** : Aligner à **droite**.
  - **ALIGN_TOP** : Aligner en **haut**.
  - **ALIGN_BOTTOM** : Aligner en **bas**.
  - **ALIGN_CENTER** : **Centrer**.
  - **ALIGN_CENTER_VERTICAL** : Centrer en **vertical**.
  - **ALIGN_CENTER_HORIZONTAL** : Centrer en **horizontal**.
  - **ALL** : Toutes les bordures.

Positionnement sur grille

- **`GridSizer()`** : **Positionner** sur une **grille**.
- **Toutes** les cellules ont la **même taille**.
- **`Add()`** : **Ajouter** un composant dans le conteneur.
- **`AddMany()`** : Ajouter **plusieurs** composants dans le conteneur.

Positionnement sur grille

- **`FlexGridSizer()`** : **Positionner** sur une **grille**.
- Les cellules peuvent avoir des dimensions variables.
- **`Add()`** : **Ajouter** un composant dans le conteneur.
- **`AddMany()`** : Ajouter **plusieurs** composants dans le conteneur.
- **`AddGrowableCol()`** : Réglage **colonne** dimensionnable.
- **`AddGrowableRow()`** : Réglage **ligne** dimensionnable.

Positionnement sur grille

- **`GridBagSizer()`** : **Positionner** sur une **grille**.
- Conteneur le plus difficile à utiliser.
- Les composants peuvent être placés sur **plusieurs lignes** ou **colonnes**.
- **`Add()`** : **Ajouter** un composant dans le conteneur.
- **`AddMany()`** : Ajouter **plusieurs** composants dans le conteneur.
- **`AddGrowableCol()`** : Réglage **colonne** dimensionnable.
- **`AddGrowableRow()`** : Réglage **ligne** dimensionnable.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_13.py` .. `code_wx_20.py` ».
- **Modifier** des paramètres et tester les effets.

Gestion des événements

- Evénements.
- Identifiants.

Evénements

- L'application principale possède un **gestionnaire** de **messages** : **boucle** d'écoute, de dispersion des messages et événements.
- Les **événements** peuvent (ou pas) appeler une fonction.
- Exemples d'événements **standards** :
  - **`EVT_MOVE`** : L'objet a changé de **position**.
  - **`EVT_SIZE`** : L'objet a été **redimensionné**.
  - **`EVT_CLOSE`** : La fenêtre a été **fermée**.
  - Etc.
- **`Bind()`** : Méthode pour **associer** action ↔ événement.
- **`Veto()`** : **Arrêt** du traitement de message.
- **`Skip()`** : **Continuer** le traitement de message.
- **`GetId()`** : Récupérer l'**identifiant** de contrôle associé.
- **`GetKeyCode()`** : Récupérer la **touche** appuyée.
- **`GetEventObject()`** : Récupérer l'**objet** qui a créé l'événement.

Identifiants

- **Entier** « unique » permettant d'identifier les objets graphiques.
- **Participe** à la **gestion** des messages et événements.
- **3 types** :
  - Créés automatiquement par le **système** (< 0).
  - Identifiants **standards** (> 0).
  - Créés par l'**utilisateur** (> 0) : méthode `NewId()`.
- **Recommander** d'utiliser des identifiants **système**, dits « **standards** », ils viennent avec ressources graphiques / comportements sur différents OS.
- Par **exemples** :
  - `ID_ANY` : Affectation par le **système**.
  - `ID_DELETE` : **Supprimer** un élément.
  - `ID_CANCEL` : **Annuler** une action.
  - `ID_SAVE` : **Sauvegarder**.
  - `ID_EXIT` : **Quitter** une application.
  - Etc.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « **`code_wx_21.py`** .. **`code_wx_28.py`** ».
- **Modifier** des paramètres et tester les effets.

Boites de dialogue standard

- Boite de dialogue simple.
- Boite A propos …

Boite de dialogue standard

- **`MessageBox()`** : **Afficher** une boite de dialogue **simple**.
- Des types de **boutons pré-existent** :
  - **`OK`** : Bouton OK.
  - **`CANCEL`** : Bouton Annuler.
  - **`YES_NO`** : Boutons Oui & Non.
  - **`YES(NO)_DEFAULT`** : Bouton Oui(Non) sélectionné.
- Des types d'**icônes pré-existent** :
  - **`ICON_EXCLAMATION`** : Avertissement.
  - **`ICON_ERROR`** / **`ICON_HAND`** : Erreur.
  - **`ICON_INFORMATION`** : Information.
  - **`ICON_QUESTION`** : Question.

Boite de dialogue « A propos »

- Une classe **simple** avec les contrôles **pré-existants**.
- **`AboutDialogInfo()`** : **Constructeur** de la boite.
- **`SetName()`** : Régler le nom de l'**auteur**.
- **`SetVersion()`** : Régler la **version**.
- **`SetDescription()`** : Régler la **description** du programme.
- **`SetCopyright()`** : Régler le **Copyright** du programme.
- **`SetWebSite()`** : Régler l'**URL** du programme.
- **`SetLicence()`** : Régler le texte de **licence** du programme.
- **`AddDeveloppeur()`** : Régler le nom du **développeur**.
- **`AboutBox()`** : **Créer** la boite de dialogue elle-même.
- Etc.

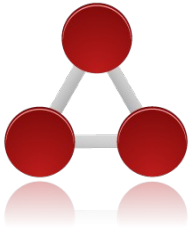
Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_29.py` .. `code_wx_31.py` ».
- **Modifier** des paramètres et tester les effets.

Drag & Drop

- Drag & Drop Texte.
- Drag & Drop Fichier.

Drag&Drop



- **`TextDropTarget()`** : Pour du **texte**.
- **`FileDropTarget()`** : Pour un **fichier**.
- **Plus de détails** dans les exemples fournis.
- Il existe de **nombreux exemples** autour de cette thématique !

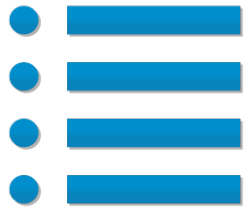
Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_32.py` .. `code_wx_34.py` ».
- **Modifier** des paramètres et tester les effets.

Composants graphiques « avancés »

- Zone de liste.
- Zone HTML.
- Aide en ligne.

Zone de liste



- **`ListCtrl()`** : Créer une **zone de liste**.
- **Plusieurs** styles possibles :
  - **`LC_LIST`**, **`LC_REPORT`**, **`LC_VIRTUAL`**, **`LC_ICON`**
  - **`LC_SMALL_ICON`**, **`LC_ALIGN_LEFT`**, **`LC_EDIT_LABELS`**
  - **`LC_NO_HEADER`**, **`LC_SORT_ASCENDING`**, **`LC_SORT_DESCENDING`**
  - **`LC_HRULES`**, **`LC_VRULES`**
- Possibilités d'avoir des classes **améliorées** :
  - **`ColumnSorterMixin()`** : **Trier** les colonnes dans une vue.
  - **`ListCtrlAutoWidthMixin()`** : **Dimensionner** les colonnes.
  - **`ListCtrlSelectionManagerMix()`** : **Sélection** non limitée.
  - **`TextEditMixin()`** : Possible d'**éditer** du contenu.
  - **`CheckListCtrlMixin()`** : Chaque ligne peut être **sélectionnée**.

Zone d'affichage HTML

- **`HtmlWindow()`** : Créer un contrôle d'affichage **HTML**.
- **`SetBackgroundColour()`** : Régler la **couleur** fond.
- **`SetStandardFonts()`** : Régler les **polices**.
- **`SetPage()`** : Remplir avec du **contenu** HTML.
- **`GenStaticText()`** : Classe de **base** d'un texte statique.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_35.py` .. `code_wx_44.py` ».
- **Modifier** des paramètres et tester les effets.

Dessin

- Généralités.
- Evénement de dessin de fenêtre.
- Dessin de points.
- Dessin en couleurs.
- Types de ligne.
- Types de remplissage.

Généralités

- **Device Context** : Pour dessiner sur les **fenêtres**.
  - **`PostScriptDC()`** : Sur un fichier de type **Postscript**.
  - **`MemoryDC()`** : Sur une **image**.
  - **`PrinterDC()`** : Sur une **imprimante** (*Windows uniquement*).
  - **`ScreenDC()`** : Sur l'**écran**.
  - **`ClientDC()`** : Sur la partie **cliente** de la fenêtre.
  - **`PaintDC()`** : Identique à **`ClientDC()`**, sauf **méthodes**.
  - **`WindowDC()`** : Sur la fenêtre **entière** (*Windows uniquement*).
- **`DrawLine()`** : Trace une **ligne**.
- **`DrawRectangle()`** : Trace un **rectangle**.
- **`SetPen()`** : Régler un **crayon** pour tracer.
- **`SetBrush()`** : Régler un **pinceau** pour remplir.

Dessin ...

- **`DrawPoint()`** : Tracer des **points**.
- **`DrawEllipse()`** : Tracer des **ellipses**.
- **`DrawRoundedRectangle()`** : Tracer **rectangle** arrondi.
- **`DrawPolygon()`** : Tracer des **polygones**.
- **`DrawArc()`** : Tracer **arc** de cercle.
- **`DrawSpline()`** : Tracer des **lignes** arrondies.
- **`DrawCircle()`** : Tracer des **cercles**.
- **`SetCap()`** : Régler le style de **fin** de la ligne.
- **`SetJoin()`** : Régler le style de **raccord** de lignes.
- **`GradientFillLinear()`** : Régler **gradient** de remplissage.
- **`BrushFromBitmap()`** : Régler l'**image** de fond.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_46.py` .. `code_wx_62.py` ».
- **Modifier** des paramètres et tester les effets.

Personnalisation & Divers

- Configuration.
- Fond de bouton.
- Gestes souris.
- Puzzle.
- Tetris.
- Conclusions.

Personnalisation …

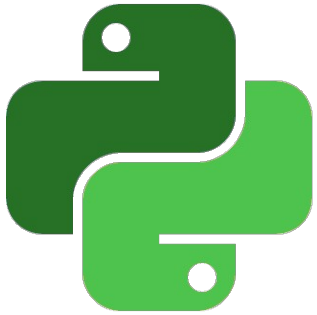
- **`Config()`** : Gérer un fichier de **configuration**.
- **`ReadInt()`** : **Lire** un entier dans le fichier.
- **`WriteInt()`** : **Ecrire** un entier dans le fichier.
- **`GenButton()`** : Créer un **bouton**.
- **`MouseGestures()`** : **Gérer** des gestes souris.

Exercices ...

- **Tester** les différents fichiers d'exemples :
  « `code_wx_63.py` .. `code_wx_67.py` ».
- **Modifier** des paramètres et tester les effets.

Conclusions

- **Beaucoup** de choses disponibles :
  - En terme de **composants graphiques**.
  - D'**exemples** de contrôles graphiques.
- Librairie graphique **complète**.
- Pas d'éditeur d'interfaces graphiques.
- Demande de construire les interfaces « à la main ».
- Nous en verrons une autre : Qt.



Modules Python

Merci ...

- … de votre attention.
- … de ne pas hésiter à poser des questions.
- … de rester motivé pour la suite.