Programmation sous SIG

Python et ArcMap

Objectifs :

— connaître les différentes manières d'utiliser Python dans ArcMap
— être capable d'écrire un script de géotraitement simple
— connaître l'organisation générale de la bibliothèque ArcPy

Préalables

Au cours de ce TD, nous allons explorer les différentes manières d'utiliser Python avec ArcMap, l'application bureautique traditionnelle du système ArcGIS, en essayant de mettre en avant les avantages et limites de chacune des possibilités. Avant d'aborder la partie programmation, nous commencerons par un petit rappel sur l'utilisation des outils de géotraitement d'ArcMap : l'ArcToolbox et ModelBuilder.

Notre cas d'étude sera le suivant : nous disposons de données sur le positionnement des stations de métro parisiennes. Les données en question sont constituées d'un fichier csv [1] par ligne de métro avec les coordonnées de chaque station de la ligne. Nous souhaitons charger et mettre en forme ces données dans un SIG. Dans un autre TD, nous nous servirons des données traitées pour réaliser une application de calcul d'itinéraires.

Au cours de ce TD, nous nous contraindrons à utiliser le plus possible les outils de l'ArcToolbox. La finalité étant de programmer dans ArcMap, nous verrons en effet que l'ArcToolbox est liée à Python.

ArcMap est toujours basé sur la version 2.7 de Python, différente de celle utilisée pour les cours de Pyhton à l'ENSG.
Les différences ne sont pas énormes, mais il est prudent de s'en souvenir en cas de message d'erreur étrange.

1. Comma-separated values : représentation de données tabulaires sous forme de valeurs séparées par des virgules

1 Introduction aux géotraitements avec ArcMap

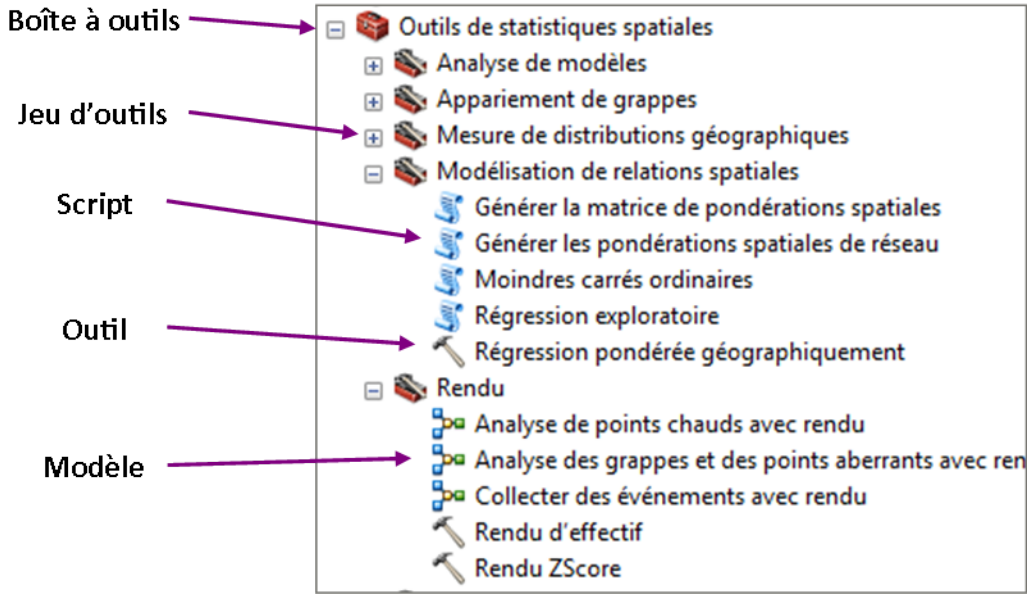
1.1 L'ArcToolbox

1.1.1 Découverte de l'ArcToolbox

⇒ Créez un nouveau document ArcMap (*TD3.mxd*)

⇒ Dans la barre d'outils Standard, cliquez sur **ArcToolbox**.

ArcToolbox est une fenêtre ancrable qui peut être placée n'importe où dans l'application. Elle contient plusieurs boîtes à outils qui sont composées d'outil, de scripts et/ou de modèles. Une boîte à outils peut également être organisée en plusieurs jeux d'outils.



⇒ Dans la boîte à outils **Outils de conversion**, combien d'outils comporte le jeu d'outils contenant l'outil **Classe d'entités vers classe d'entités** ?

Réponse .....
.....

1.1.2 Utilisation de l'ArcToolbox pour créer une géodatabase

Afin d'organiser les données que nous allons créer, nous souhaitons utiliser une géodatabase fichier. Pour la créer, nous aimerions utiliser l'ArcToolbox mais nous ne savons pas si un tel outil existe, ni comment le localiser dans l'arborescence des Toolbox.

Esri propose deux types de géodatabase pour des utilisations bureautiques :

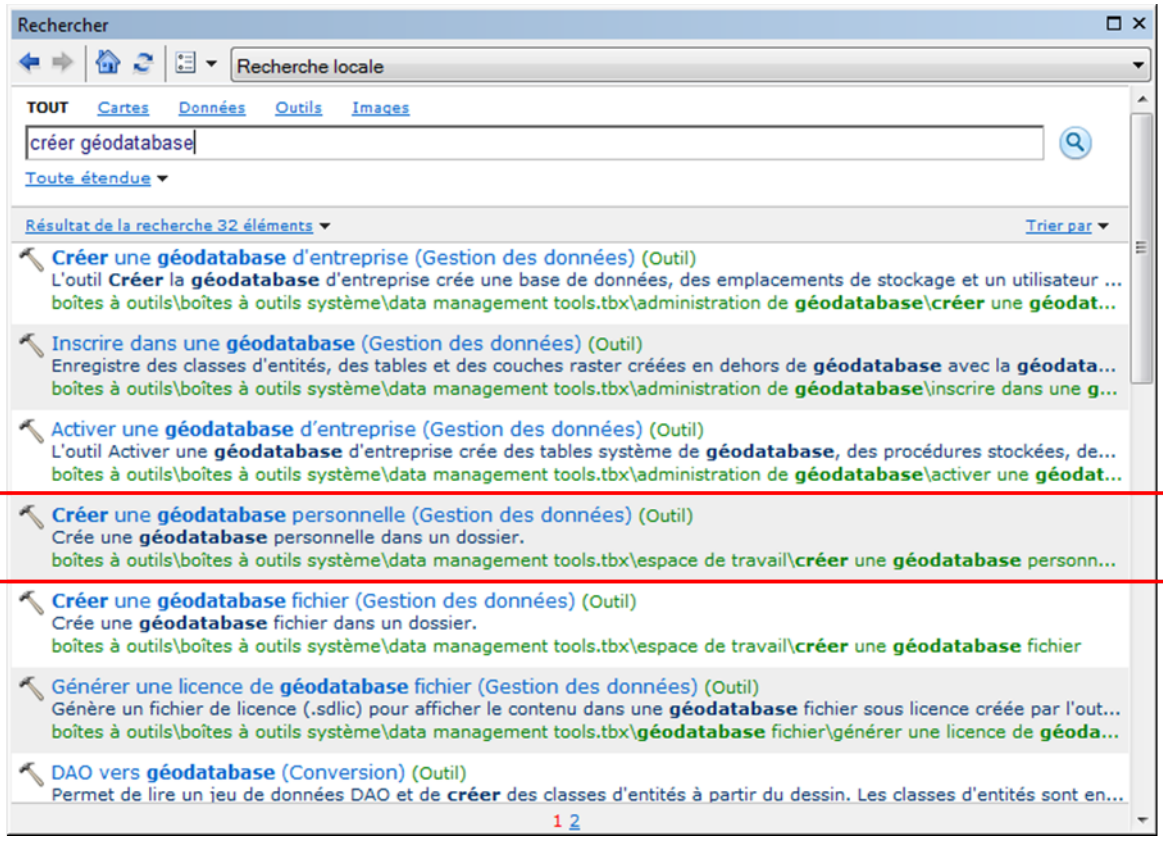
- la géodatabase personnelle : basée sur une base Access, sa taille est limitée à 4Go (taille max d'un fichier sur un ordinateur Windows) et ne peut être utilisée par plus de 5 utilisateurs en simultané ;
- la géodatabase fichier : utilise plusieurs fichiers pour stocker l'intégralité de la base, ce qui lui permet de ne pas être limitée à 4Go. La limite d'utilisateurs simultanés est de 20 et ses performances sont par ailleurs souvent bien meilleures.

Sauf cas exceptionnels, nous utiliserons toujours des géodatabases fichier.

⇒ Activez l'outil de recherche en cliquant sur le bouton dans la barre d'outils **Standard**.

⇒ Effectuez une recherche à l'aide des mots clé *géodatabase* et *créer*.

Un listing alphabétique des résultats disponibles s'affiche.



Le lien en vert vous permet de localiser l'outil dans l'ArcToolbox.

⇒ Retrouvez l'outil pour créer une géodatabase dans l'ArcToolbox et exécutez-le.

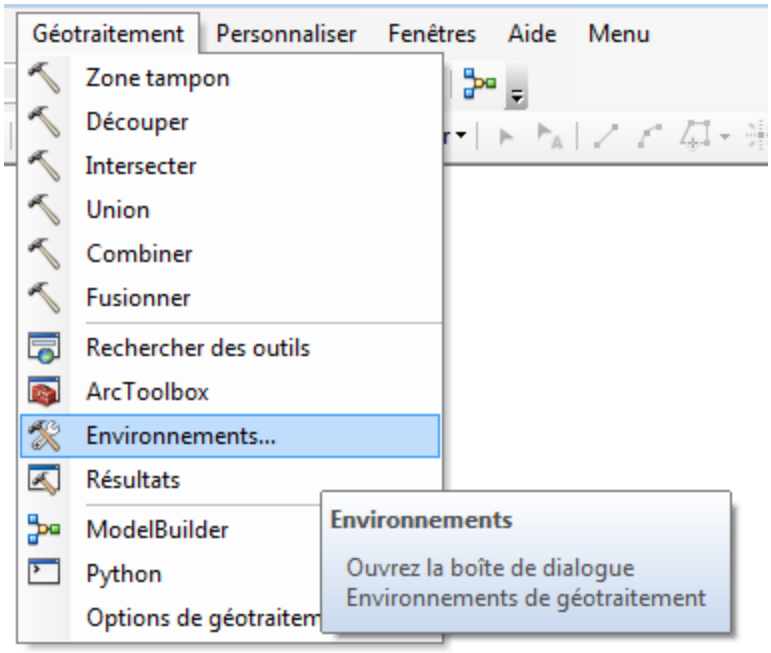
⇒ Choisissez *TD2.gdb* dans *D :\ProgSIG\TD2*, puis cliquez sur **OK**.

1.1.3 Définition de l'environnement de travail

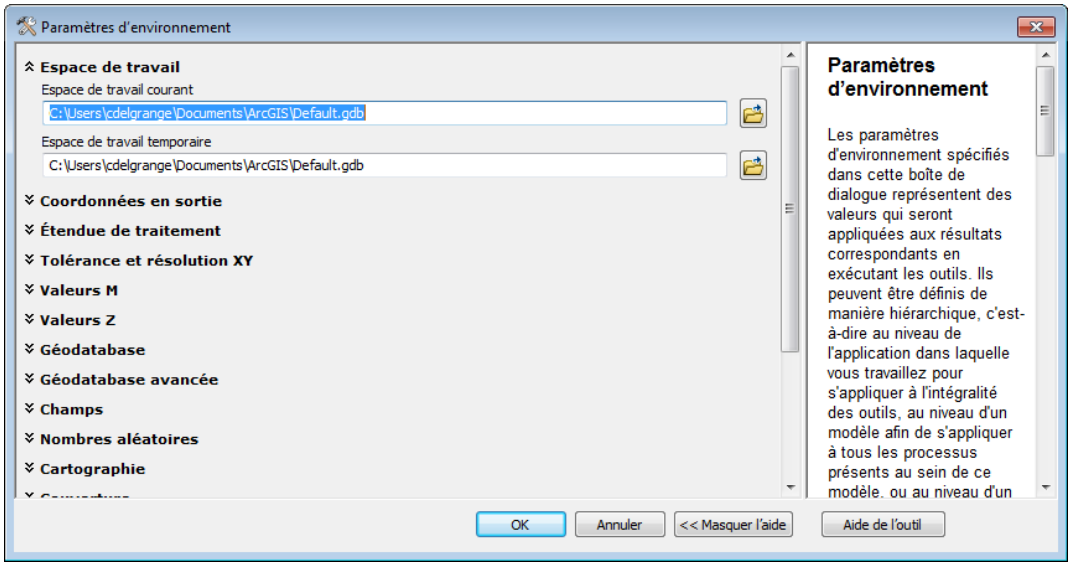
Un certain nombre de paramètres influent sur les résultats des outils de géotraitement sans pour autant être visible dans les boîtes de dialogue des outils. Il s'agit des paramètres d'environnement.

Nous pouvons par exemple spécifier un espace de travail par défaut, une étendue, un système de projection particulier pour les données en sortie de traitement, etc. Les valeurs choisies s'appliquent à tous les outils de géotraitement.

⇒ Dans le menu **Géotraitements**, sélectionnez l'item **Environnements...**.



⇒ Développez **Espace de travail** et pour **Espace de travail courant** choisissez le chemin de la géodatabase personnelle que vous avez créée précédemment.



L'espace de travail courant peut être un répertoire, une géodatabase ou un jeu de classes d'entités. Ce sera par défaut le chemin utilisé pour les données en entrée/sortie.

⇒ Pour **Espace de travail temporaire**, choisissez : *D :\ProgSIG\TD2\tmp*

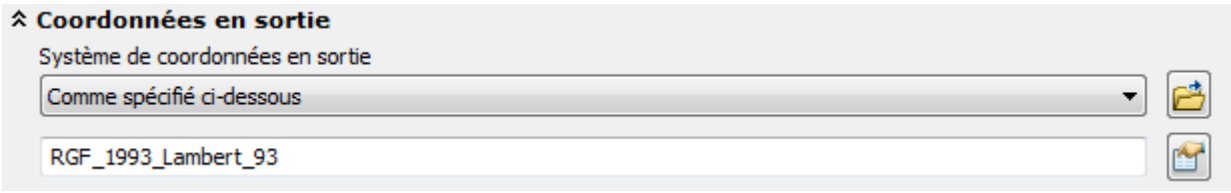
L'espace de travail temporaire sera utilisé par les outils pour lesquels le chemin d'accès des données en sortie et/ou le nom de la sortie par défaut ne sera pas modifié.

Nous allons également préciser le système de coordonnées utilisé pour les données en sortie des outils.

⇒ Dépliez *Coordonnées en sortie* et dans la liste déroulante sélectionnez **Comme spécifié ci-dessous**.

⇒ A la droite de la ligne qui s'active en-dessous cliquez sur le bouton représentant une main avec une feuille.

⇒ Recherchez et sélectionnez la projection *RGF_1993_Lambert_93*.



⇒ Cliquez sur **OK** pour fermer la fenêtre *Paramètres d'environnement*.

1.1.4 Utilisation de géotraitements

Les données à notre disposition pour modéliser le réseau de métro sont constituées d'un fichier texte par ligne de métro avec les coordonnées x et y, ainsi que le nom, de chacune des stations de la ligne.

Afin de mettre en place un processus valable pour toutes les lignes, nous commençons par nous concentrer

sur la ligne 1 uniquement.

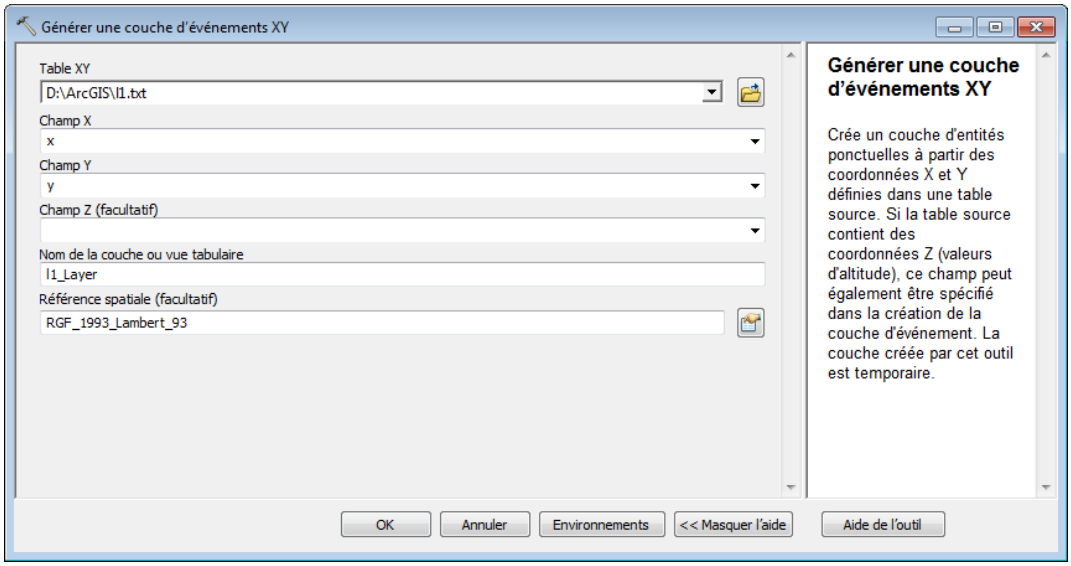
⇒ Après avoir récupéré les données et pris connaissance du formatage des fichiers, proposez une méthodologie pour leur exploitation dans ArcMap.

Réponse .....
.....

⇒ Dans **Outils de gestion de données** > **Couches et vues tabulaires**, cliquez sur **Générer une couche d'événements XY**

Cet outil crée une couche d'entités ponctuelles à partir de coordonnées X et Y définies dans une table. Le fichier fichier texte des stations fait ici office de table.

⇒ Configurez les champs de la manière suivante et validez.



Afin d'enregistrer physiquement la couche d'entités dans notre géodatabase, nous utiliserons l'outil **Classe d'entités vers classe d'entités**.

⇒ Retrouvez, paramétrez et utilisez cet outil pour créer la classe d'entités *Stations_ligne1* dans la géodatabase.

Pour finir, il nous reste à relier les stations de la ligne dans une polyligne.

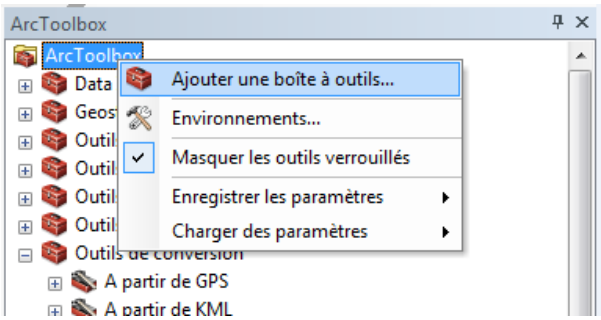
⇒ Utilisez le script **Points vers lignes** pour réaliser cette opération et créer dans la géodatabase une classe d'entités *Ligne1*.

A ce stade, votre document ArcMap doit contenir deux couches : les ponctuels des stations de la ligne 1 et le linéaire de la ligne.

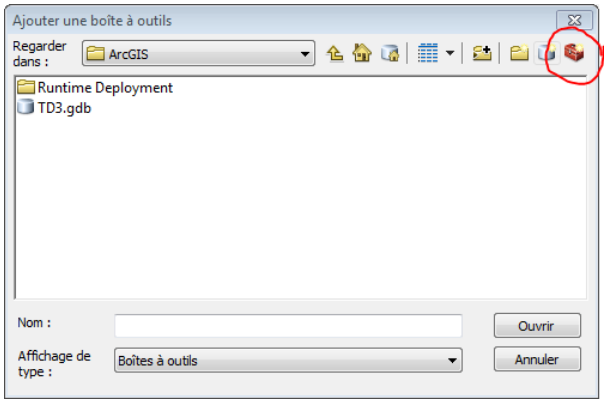
1.1.5 Création d'une boîte à outils personnelle

Nous avons donc réalisé toutes les étapes de chargement des données. Sans être excessivement long, le processus serait assez longs à appliquer aux 14 lignes de métro. La première optimisation que nous allons essayer de mettre en place consistera à rassembler tous les outils utilisés dans une unique boîte à outils.

⇒ Effectuez un clic droit sur l'ArcTolbox > **Ajouter une boîte à outils...** :



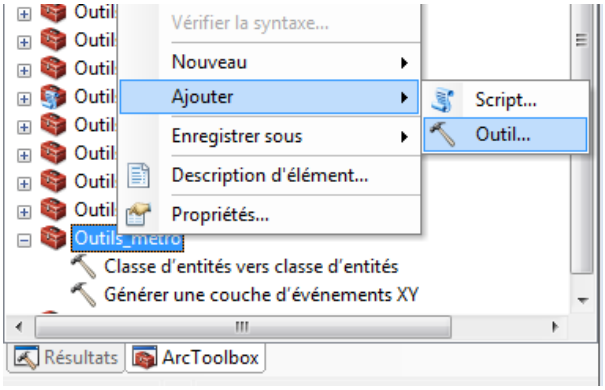
⇒ Dans le fenêtre qui s'ouvre, cliquez sur **Nouvelle boîte à outils** :



⇒ Nommez votre boîte à outils *Outils_ metro.tbx*.

En validant, la nouvelle boîte à outils apparaît dans l'ArcToolbox. Il reste à y ajouter les outils. La procédure sera légèrement différent selon qu'il s'agisse d'un outil "simple", d'un script ou encore d'un modèle.

⇒ Clic droit sur la nouvelle boîte à outils > **Ajouter** > **Outil...**



⇒ Sélectionnez les outils **Générer une couche d'événements XY** et **Classe d'entités vers classe d'entités**.

⇒ Clic droit sur la nouvelle boîte à outils > **Ajouter** > **Script...**

⇒ Pour le script *Points vers lignes*, retrouvez le dans l'ArcToolbox et copiez-coller le dans votre boîte personnelle.

La nouvelle boîte à outils est maintenant configurée. En l'utilisant, vous pouvez ajouter à votre document les stations et lignes de la ligne 2.

⇒ Le résultat est-il satisfaisant ? Que pourrions nous faire de mieux ?

Réponse .....
.....

1.2 ModelBuilder pour enchaîner des géotraitements

1.2.1 Créer un nouveau modèle

ModelBuilder est un environnement permettant d'enchaîner de manière graphique des géotraitements. Nous l'utiliserons dans cette partie pour automatiser le processus de création des géométries des lignes de métro.

⇒ Dans ArcToolbox, faites un clic-droit sur *Outils_metro* dans la boîte à outils et cliquez sur **Nouveau** > **Modèle**.

⇒ Dans le menu **Modèle**, cliquez sur **Propriétés du modèle**.

⇒ Allez dans l'onglet **Général**.

⇒ Dans **Nom**, tapez *CreationLineaireMetro*.

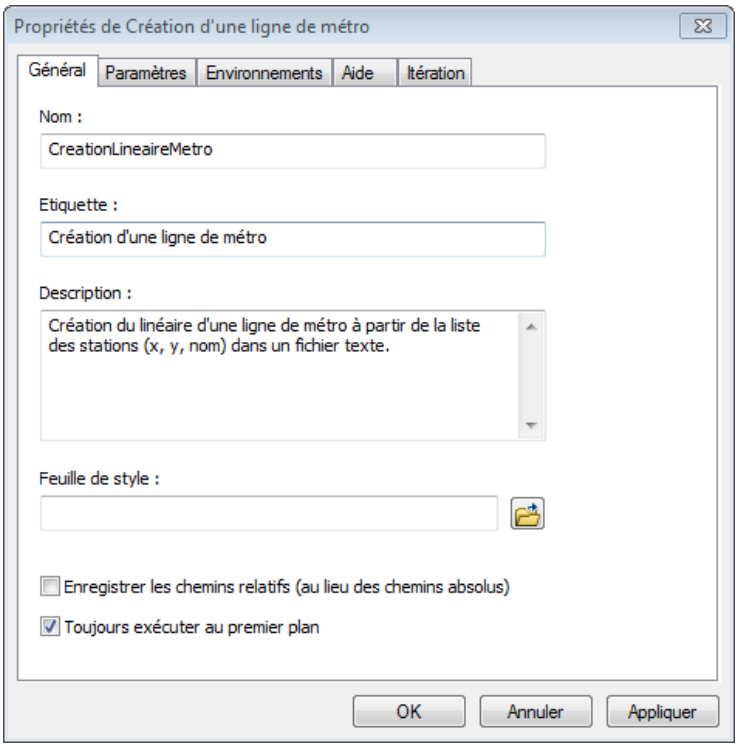
Le nom définit comment l'outil sera appelé en programmation. Il existe certaines conventions à respecter dans le nom des modèles : pas d'espace ni de point par exemple.

⇒ Pour **Etiquette**, tapez *Création d'une ligne de métro*.

L'étiquette est le nom sous lequel apparaîtra votre modèle dans votre boîte à outils. Dans l'étiquette, vous pouvez utiliser les espaces et les points pour décrire de manière claire ce que fait le modèle.

Vous pouvez également ajouter une description plus détaillée.

⇒ Une fois ces champs paramétrés, cliquez sur **OK**.



1.2.2 Ajouter un outil au modèle

⇒ Dans l'arborescence de l'ArcToolbox, retrouvez l'outil **Générer une couche d'événements XY** et faîtes-le glisser dans la fenêtre du modèle.

Dans un Modelbuilder, les éléments en blanc indiquent que vous avez besoin d'apporter des informations complémentaires à cet outil pour que le modèle puisse l'exécuter. Ici, il s'agit simplement de spécifier les données en entrée et d'indiquer un nom pour la couche en sortie de l'outil.

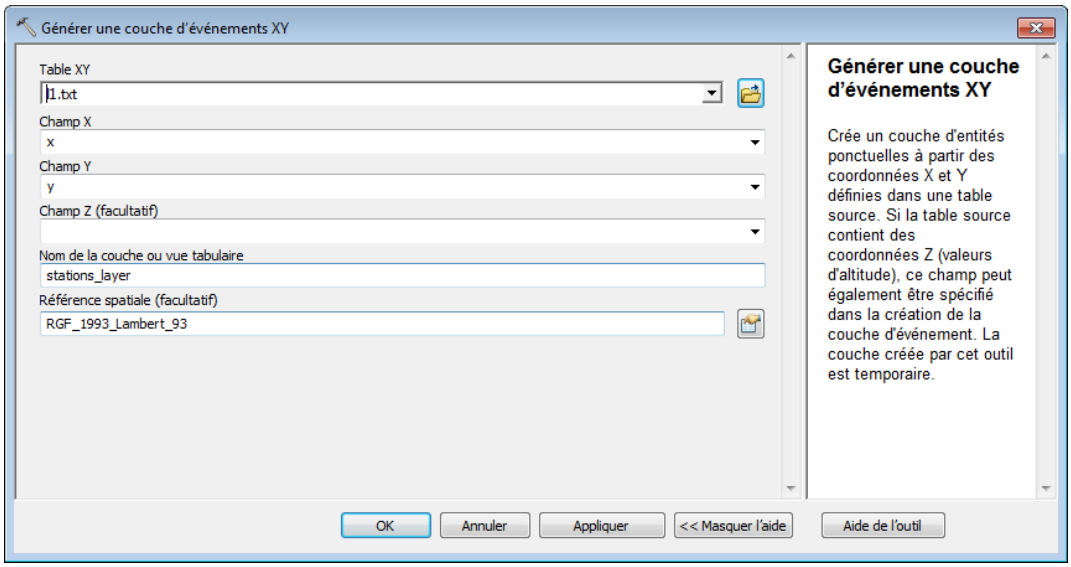
⇒ Dans le modèle, double-cliquez alors sur l'outil **Générer une couche d'événements XY**.

La boîte de dialogue qui apparaît est la même que celle que vous auriez obtenu si vous aviez ouvert l'outil **Générer une couche d'événements XY** depuis l'ArcToolbox.

⇒ Dans **Table XY**, sélectionnez pour l'instant le fichier *l3.txt*.

⇒ Pour **Champ X** et **Champ Y** saisissez respectivement x et y.

⇒ Pour **Nom de la couche ou vue tabulaire** en sortie saisissez *stations_l3_layer*.



⇒ Validez

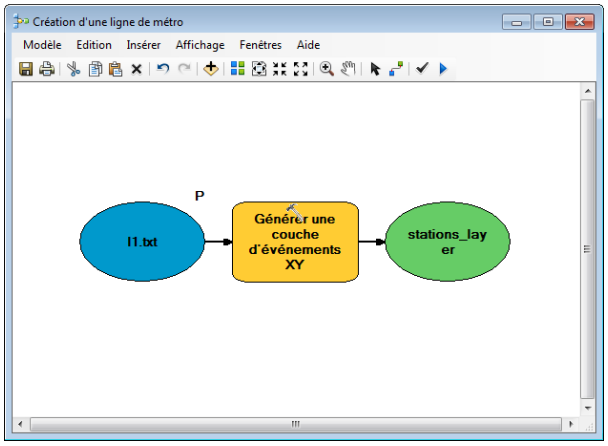
Dans le modèle, l'outil **Générer une couche d'événements XY** et les données en entrée/sortie sont maintenant colorés car renseignés. Il est donc prêt à être exécuté.

1.2.3 Passer des éléments en paramètre

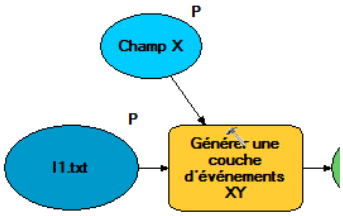
La **Table XY** en entrée est en fait un paramètre du modèle : le fichier doit pouvoir être changé à chaque lancement du modèle.

⇒ Effectuez un clic droit sur l'élément *l1.txt* du modèle > **Paramètre du modèle**.

Un petit P apparaît en haut à droite de l'élément.



Notre modèle suppose que les fichiers de départ présentent toujours la même structure, et en particulier que les champs X et Y se nomment toujours *x* et *y*, ce qui pourrait ne pas être le cas.
De même que pour les données, il est possible de créer des modèles génériques laissant à l'utilisateur la possibilité de saisir les valeurs des paramètres.
Pour cela, faites par exemple un clic-droit sur l'outil **Générer une couche d'événements XY**. Dans le menu contextuel, lancez le menu **Générer une variable** > **Paramètre de départ** > **Champ X**.
Un élément/paramètre (bleu ciel) vient s'intégrer au modèle.
Vous pouvez double-cliquer sur cet élément pour définir une valeur par défaut et/ou effectuer un clic-droit pour le passer en paramètre du modèle.

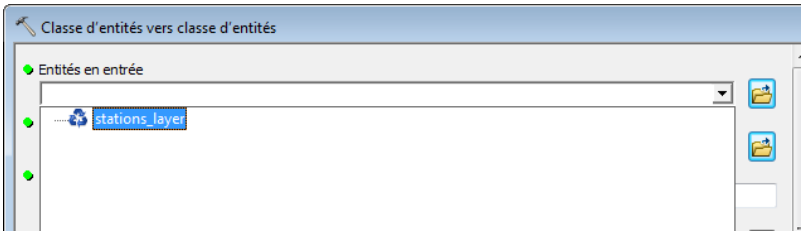


La même opération peut être effectuée pour le champ y.

1.2.4 Enchaîner plusieurs outils

⇒ En utilisant les mêmes méthodes, ajoutez l'outil **Classe d'entités vers classe d'entités** au modèle et paramétrez-le pour que la donnée en entrée soit la couche *stations_layer* du modèle.

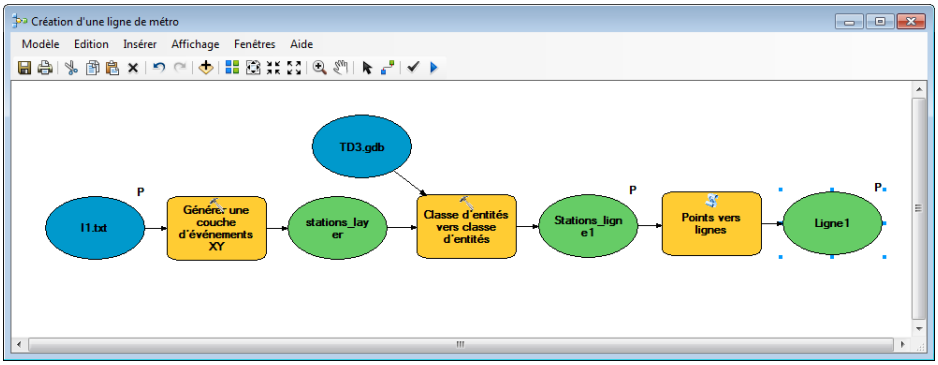
⇒ Appelez le classe d'entités en sortie *Stations_ligne1*.



La liste déroulante du modèle détecte automatiquement les données présentes dans votre modèle. Ces données sont marquées d'un icône spécifique (triangle bleu).
Un icône différent (parallélogramme jaune) apparaîtra pour dénoter les données présentes dans la table des matières d'ArcMap (si ArcMap est ouvert et que des données y sont chargées).
Bien entendu, vous pouvez aussi sélectionner des données qui ne sont ni dans votre modèle, ni dans la table des matières de votre session ArcMap.

⇒ Ajoutez enfin au modèle le script **Points vers lignes** en passant la sortie en paramètre du modèle (pour pouvoir modifier son nom).

⇒ Enregistrez votre modèle.



1.2.5 Exécuter le modèle

⇒ Exécutez le modèle via le menu **Modèle** > **Exécuter**.

⇒ Vérifiez que la classe d'entités *Ligne3* a bien été créée.

Dans le modèle, les outils ainsi que les données dérivées sont affectés d'un ombrage qui indique que les processus ont bien été exécutés. Vous devez Valider votre modèle entier (dans le menu Modèle) si vous voulez voir disparaître l'ombrage pour suivre le cheminement la prochaine fois que vous exécuterez votre modèle.

⇒ Refermez votre modèle.

⇒ Exécutez-le sur plusieurs lignes en double-cliquant dessus dans l'ArcToolbox et en modifiant les paramètres.

⇒ Le résultat est-il satisfaisant ?

Réponse .....
.....

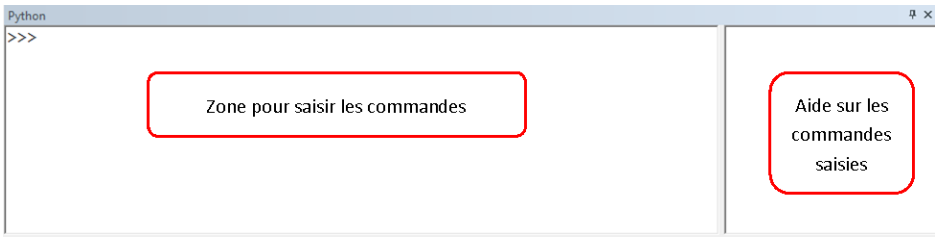
2 Les utilisations de Python avec ArcMap

2.1 La console Python

Dans ce paragraphe, nous allons nous initier à la syntaxe des commandes Python lancées depuis la console Python d'ArcMap. Nous nous servirons de ces acquis pour revenir ensuite sur notre problématique de calcul d'itinéraires dans le métro.

⇒ Dans la barre d'outils Standard, cliquez sur le bouton **Fenêtre Python**.

Vous taperez vos lignes de commandes pour accéder aux outils dans la partie de gauche. Dans la partie de droite s'affichera l'aide des commandes que vous aurez tapées.



Dans un premier temps, uniquement pour nous exercer, nous allons appliquer un buffer de 20m à la ligne de métro.

⇒ Dans la fenêtre Python, tapez `arcpy.buf`

Une liste déroulante apparaît avec plusieurs commandes commençant par *buf*.

⇒ Sélectionnez `Buffer_analysis`.

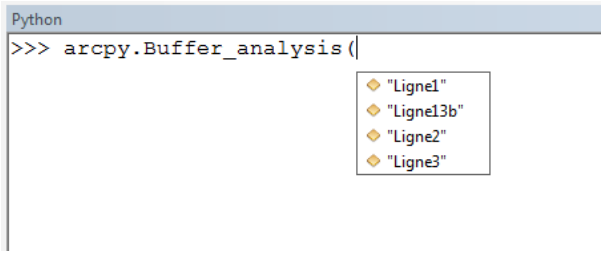
⇒ Ouvrez une parenthèse : `arcpy.Buffer_analysis(`

Un descriptif présente tous les arguments requis pour la syntaxe Buffer_analysis dans l'aide, à droite de la fenêtre. L'argument courant est surligné en orange.

```
Usage: Buffer_analysis (in_features , out_feature_class ,
    buffer_distance_or_field  {line_side} , {line_end_type} , {dissolve_field
    ...})
```

Tous les argument sont séparés par une virgule et les indiquent que les arguments concernés sont optionnels.

Sous la commande vous pouvez voir aussi une liste déroulante qui répertorie automatiquement toutes les classes d'entités chargées dans ArcMap et pouvant être utilisées dans la commande pour l'argument surligné.



⇒ Renseignez les arguments de la manière suivante :

- `<In_features>` : sélectionnez "Ligne1" dans la liste déroulante et appuyez sur **Tabulation**
- `<Out_feature_class>` : "Ligne1_buff"
- `<Buffer_distance_or_field>` : 50
- `ine_sideline_side` : "FULL" ou "" car il s'agit de la valeur par défaut
- `{line_end_type}` : "" (valeur par défaut)

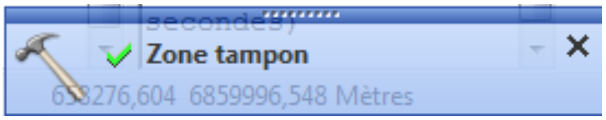
— {dissolve_option} : "ALL"
— {dissolve_field;dissolve_field...} : ""

Au final, la commande doit être la suivante :

```
arcpy.Buffer_analysis("Ligne1", "Ligne1_buff", 50, "FULL", "", "ALL", "")
```

⇒ Appuyez sur **Entrée**.

A la fin de l'exécution de la commande, un cadre s'affiche en bas à droite de l'écran vous indiquant la réussite du traitement (ou l'échec en cas de problème).



La couche *Ligne1_buff* est ajoutée au document.

Vous venez d'exécuter votre premier géotraitement avec Python. La méthode est la même pour exécuter n'importe quel autre géotraitement disponible dans ArcMap.

⇒ Quels peuvent être les intérêts à utiliser la console Python pour exécuter des commandes ?

Réponse .....
.....

ArcGIS embarque sont propre environnement Python :

```
>>> import sys
>>> sys.path
C:\Python27\ArcGIS10.5
```

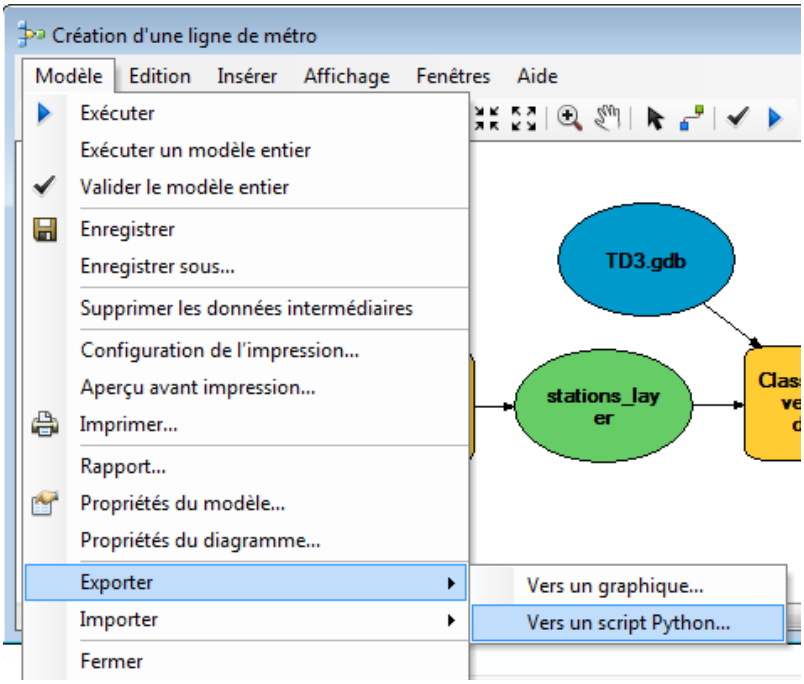
Généralement, cela ne pose pas de souci particulier, mais il conviendra tout de même d'être vigilant lors de l'installation de bibliothèque annexes (dans quelle version de Python s'installent-elles ?) ou pour utiliser les outils Python d'ArcGIS en dehors d'ArcMap (version de Python utilisée ?).

2.2 De ModelBuilder à Python

Pour dépasser les limites du modèle établit lors du paragraphe précédent, nous allons voir qu'un ModelBuilder peut s'exporter en Nous cherchons ici à générer un script Python correspondant au ModelBuilder que nous avons réaliser précédemment pour pouvoir l'améliorer et corriger les problèmes soulevés.

⇒ Ouvrer la fenêtre de modification de votre modèle *Création d'une ligne de métro*

⇒ Allez dans le menu **Modèle** > **Exporter** > **Vers un script Python...**.



⇒ Dans votre répertoire de travail, sauvegardez un fichier *creation_ligne_metro.py*.

⇒ Ouvrez ce script un environnement de développement Python[2].

Nous retrouvons une structure que nous retrouverons dans les tous les scripts de géotraitement pour ArcGIS :

1. Une information sur le système d'encodage utilisé (important en Python 2.7)
2. Une en-tête décrivant le contenu du script
3. Le chargement d'un module `arcpy`
4. Des déclarations de variables
5. Des appels de fonctions `arcpy` (`MakeXYEventLayer_management`, `FeatureClassToFeatureClass_conversion`, etc.)

2. PyCharm, PyScripter, Idle, Notepad++... Nous ne vous imposons pas un éditeur particulier, m

```
1  # -*- coding: utf-8 -*-
   # ---------------------------------------------------------------------------
   # creation_ligne_metro.py
   # Created on: 2016-08-19 16:01:44.00000
2  #   (generated by ArcGIS/ModelBuilder)
   # Usage: creation_ligne_metro <l1_txt> <Ligne1>
   # Description:
   # Création du linéaire d'une ligne de métro à partir de la liste des stations (x, y, nom) d
   # ---------------------------------------------------------------------------

3  # Import arcpy module
   import arcpy

   # Script arguments
   l1_txt = arcpy.GetParameterAsText(0)
   if l1_txt == '#' or not l1_txt:
       l1_txt = "D:\\ArcGIS\\l1.txt" # provide a default value if unspecified

4  Ligne1 = arcpy.GetParameterAsText(1)
   if Ligne1 == '#' or not Ligne1:
       Ligne1 = "D:\\ArcGIS\\TD3.gdb\\Ligne1" # provide a default value if unspecified

   # Local variables:
   stations_layer = "stations_layer"
   TD3_gdb = "D:\\ArcGIS\\TD3.gdb"
   Stations_ligne1 = "D:\\ArcGIS\\TD3.gdb\\Stations_ligne1"

   # Process: Générer une couche d'événements XY
   arcpy.MakeXYEventLayer_management(l1_txt, "x", "y", stations_layer, "PROJCS['RGF_1993_Lambe

5  # Process: Classe d'entités vers classe d'entités
   arcpy.FeatureClassToFeatureClass_conversion(stations_layer, TD3_gdb, "Stations_ligne1", "",

   # Process: Points vers lignes
   arcpy.PointsToLine_management(Stations_ligne1, Ligne1, "", "", "NO_CLOSE")
```

Dans la console Python d'ArcMap, nous n'avons pas eu besoin d'importer le module `arcpy` pour appeler ses fonctions.
En effet, `arcpy` est chargé par défaut dans la console. Partout ailleurs, il sera nécessaire de l'importer pour pouvoir l'utiliser.

⇒ Exécutez le script.

⇒ Que se passe-t-il ?

Réponse .....
.....

⇒ Essayez d'améliorer le script de sorte que le numéro de ligne et le répertoire où se trouvent les données deviennent des paramètres.

⇒ Lorsque le script s'exécute correctement, retournez dans ArcMap pour visualiser le résultat.

Vous pouvez constater qu'il n'est pas nécessaire de conserver ArcMap ouvert pour que le script fournisse des résultats corrects.

Générer des scripts Python via un ModelBuilder permet généralement de faire gagner du temps pour

sortir un script Python quasi fonctionnel, en réduisant la phase de recherche des fonctions ArcPy et de leur paramétrage.

Toutefois, les scripts ainsi générés sont rarement d'une propreté exemplaire. Un "nettoyage" est bien souvent nécessaire avant de pouvoir les passer en production.

2.3 De Python à une boîte à outils

⇒ Dans votre boîte à outils *Outils_metro*, ajouter le script *creation_ligne_metro.py* (clic droit sur la boîte à outils > **Ajouter** > **Script...**).

Une fenêtre nous demande des informations sur le script que nous voulons ajouter.

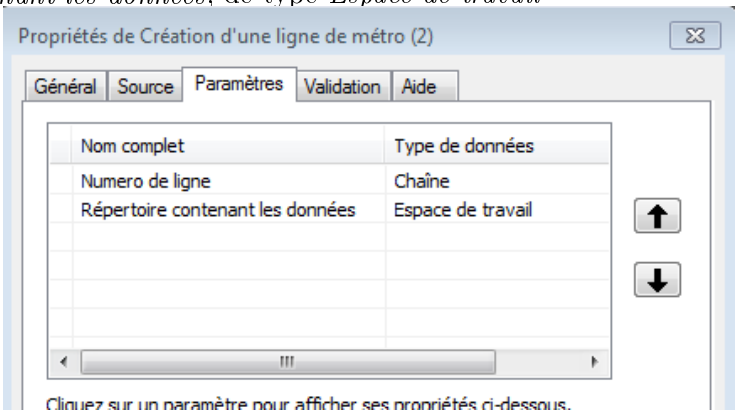
⇒ Renseignez les champs de la manière suivante :
- Nom : CreationLigneMetro
- Etiquette : Création d'une ligne de métro

⇒ Passez à l'étape suivante.

Il nous faut maintenant localiser le script sur le disque dur.

⇒ Faites pointer l'outil vers votre script *creation_ligne_metro.py*.

⇒ Passez à l'étape suivante pour définir les paramètres du script :
- *Numéro de ligne*, de type *Chaîne*
- *Répertoire contenant les données*, de type *Espace de travail*



⇒ Validez

⇒ Exécutez le script qui vient d'être ajouté à l'ArcToolbox.

A ce stade, nous avons un outil fonctionnel permettant de générer les classes d'entités pour chacune des lignes de métro. Nous avons vu que cet outil peut s'exécuter de différentes manières qui font tous appels aux mêmes fonctions d'ArcMap :
- dans un ModelBuilder (voir 1.2)
- dans un script autonome (voir 2.2)
- dans un script ajouté à l'ArcToolBox (voir 2.3)

Nous avons également vu que chaque Toolbox est équivalente à une fonction Python que nous savons

exécuter.

2.4 Extension Python pour ArcMap

La dernière solution que nous allons étudier consiste à ajouter une extension dans ArcMap. L'extension sera mieux intégrée à l'interface d'ArcMap puisqu'elle pourra prendre la forme d'un menu ou d'une barre d'outils et pourra écouter certains événements (ouverture d'une session d'édition, fermeture du document, etc.).

La diffusion des extensions est également aisée puisque nous produirons un addin ArcMap comme ceux que nous avons utilisé lors du premier TD.

2.4.1 Création de l'extension

⇒ Rendez-vous sur sur le site d'Esri[3] pour télécharger l'assistant de création d'extensions Python (*ArcGIS Python Add-In Wizard*).

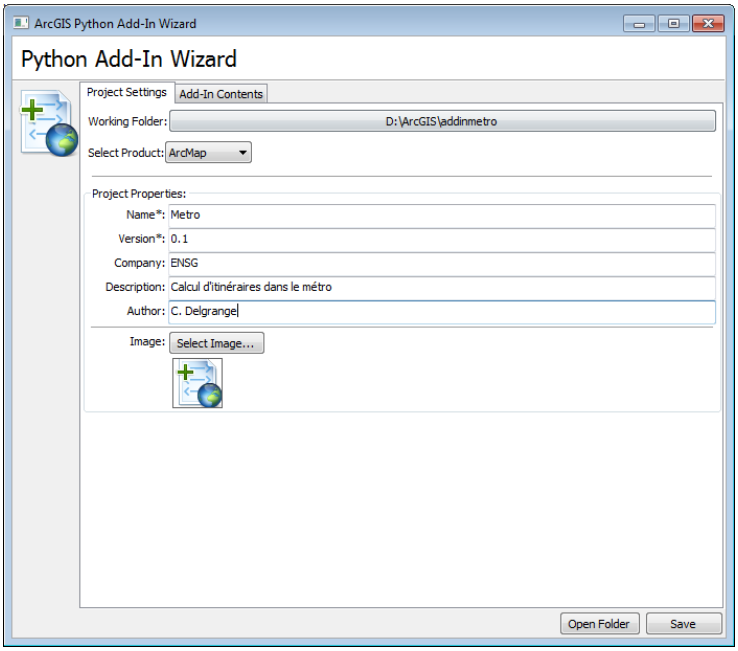
⇒ Décompresser l'archive téléchargée.

L'exécutable se trouve alors dans *addin_assistant\bin\addin_assistant.exe*.

⇒ Lancez l'assistant.

⇒ Choisissez le répertoire où sera enregistré le complément.

⇒ Dans l'onglet **Project Settings**, précisez quelques propriétés (nom, version, description, auteur, entreprise) :



3. http://www.arcgis.com/home/item.html?id=5f3aefe77f6b4f61ad3e4c62f30bff3b

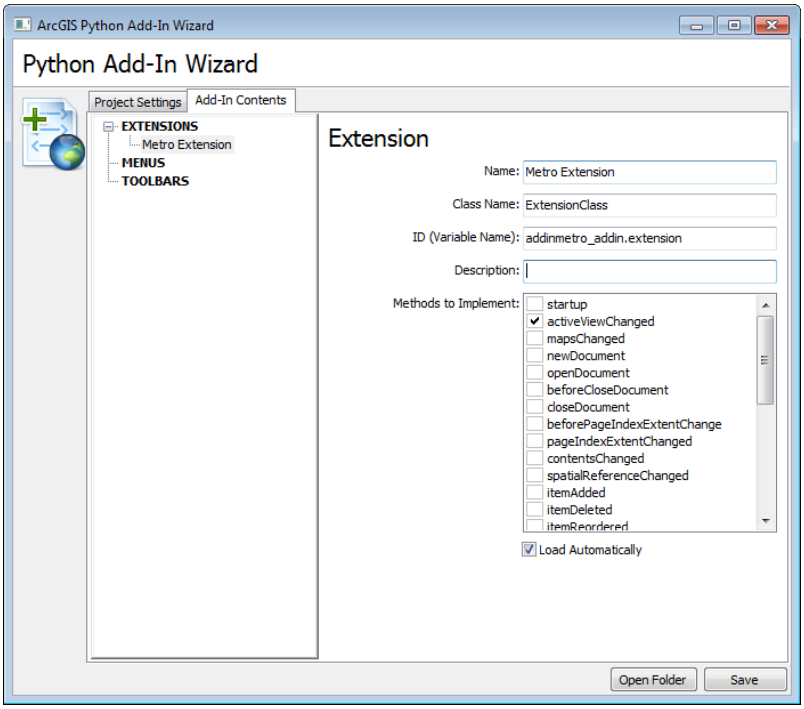
⇒ Allez ensuite dans l'onglet **Add-in content**.

Cet onglet permet d'élaborer le contenu de l'addin. Différents types de composants sont disponibles :

- les extensions qui permettent de gérer des événements liés à l'application ;
- les menus, et sous-menu et boutons associés ;
- les barres d'outils et outils associés (boutons, combo box, etc.).

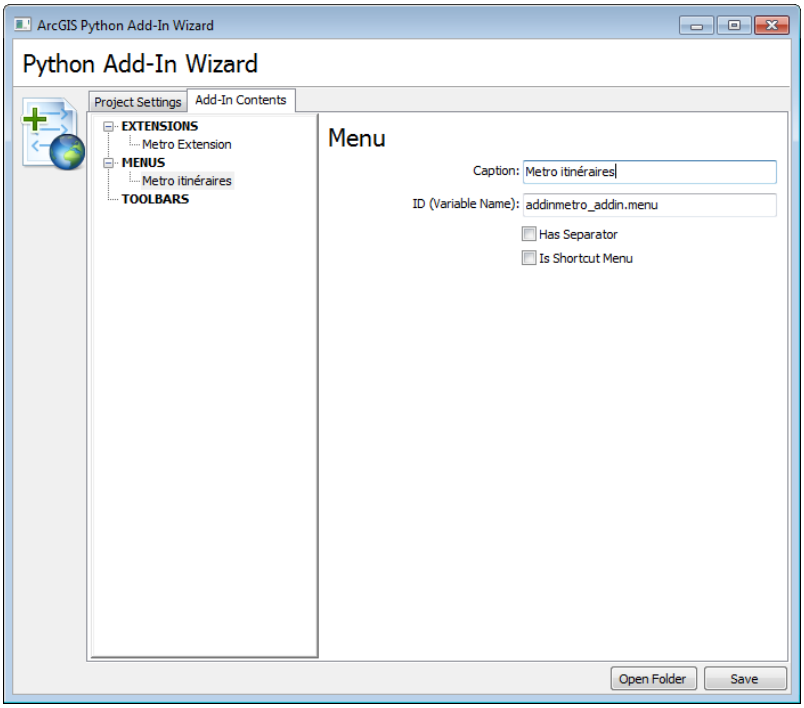
⇒ Effectuez un clic droit sur **EXTENSIONS** pour ajouter une extension.

⇒ Paramétrez l'extension comme sur l'image ci-dessous :



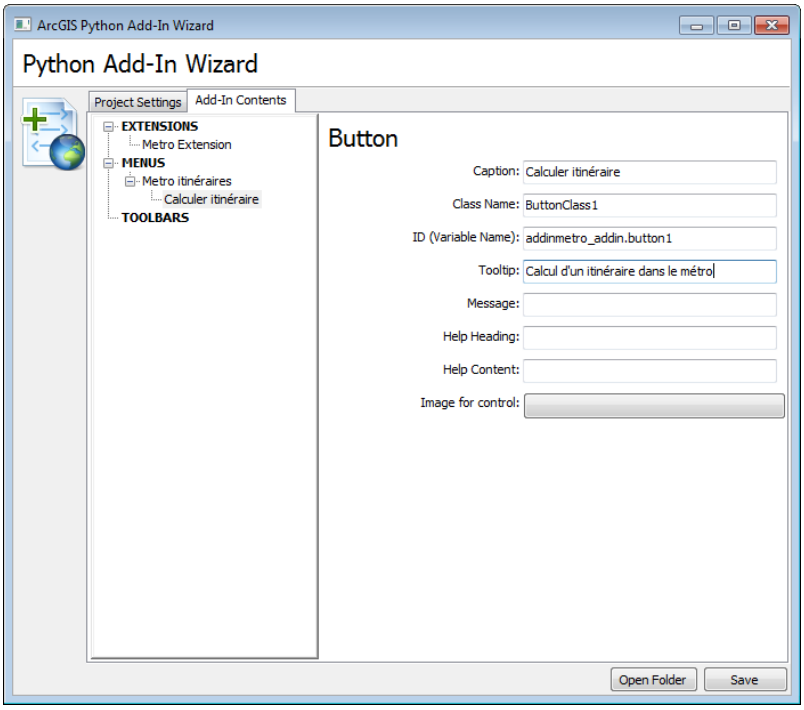
⇒ Effectuez ensuite un clic droit sur **MENUS** pour ajouter un nouveau menu.

⇒ Nommez ce menu *Métro itinéraire*



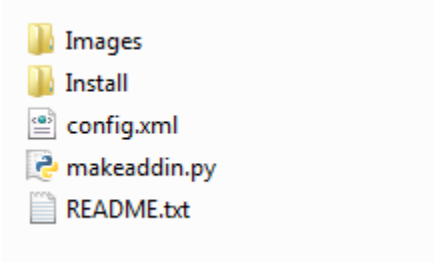
⇒ Par clic droit sur le menu *Métro itinéraire*, ajoutez un bouton.

⇒ Configurez le bouton de la manière suivante :



⇒ Cliquez sur *Save* pour enregistrer la configuration.

Vous pouvez alors refermer l'assistant et ouvrir le répertoire où vous avez enregistré votre extension.



Un addin Python pour ArcMap contiendra toujours les éléments suivants :

- un fichier `config.xml` qui contient la description de la structure de l'addin (ce que nous avons paramétré à l'aide de l'assistant) ;
- un répertoire `Install` contenant un fichier `addinmetro_addin.py` où se trouve la logique métier de l'addin ;
- un fichier `makeaddin.py` permettant de générer à partir des éléments précédent un fichier *.esriaddin* reconnu par ArcMap.

2.4.2 Modification de la structure de l'extension

L'assistant de création de complément Python est très pratique pour démarrer la configuration d'un projet. Il n'est cependant pas impossible de s'en passer. Dans ce paragraphe, nous ajouterons un bouton au menu *Métro itinéraire* précédement créé.

⇒ Ouvrez le fichier *config.xml* dans un éditeur de texte.

⇒ Après avoir observé la structure du fichier xml, ajoutez un nouveau bouton au menu *Métro itinéraire*

Ce bouton aura les caractéristiques suivantes :

- `caption` = Aide
- `class` = ButtonClass2
- `id` = addinmetro_addin.button2

L'attribut `class="ButtonClass2"` permet de lier le bouton à une classe Python qui est appelé lorsque l'utilisateur interagit avec l'interface. Il est donc nécessaire d'ajouter cette classe au fichier contenant la logique métier.

⇒ Ouvrez le fichier *addinmetro_addin.py*.

⇒ En vous inspirant de la classe `ButtonClass1`, créez une classe pour le bouton d'aide.

⇒ Quelle fonction devrait être appelée lors d'un clic sur un bouton ?

Réponse .....
.....

⇒ Dans cette fonction, ajoutez l'instruction suivante :

```
pythonaddins.MessageBox("Aide de l'extension Metro...", "Aide")
```

pythonaddins est un module qui apporte des fonctionnalités pour la gestion de l'interface utilisateur dans des compléments Python (il n'est utilisable que dans le cadre de compléments Python). La documentation en ligne [a] nous renseigne sur les fonctions utilisables de ce module.
La fonction `MessageBox(message, titre)` permet d'ouvrir une boîte de dialogue simple.

a. https://desktop.arcgis.com/fr/desktop/latest/guide-books/python-addins/the-pythonaddins-module.htm

Générer une première fois l'addin pour voir s'il est conforme à nos attentes.

⇒ Exécutez le fichier *makeaddin.py* (avec Python 2.7).

⇒ Installez ensuite l'addin de manière classique (voir TD1 de programmation sous SIG).

⇒ Ouvrez ArcMap pour tester votre addin.

2.4.3 Ajout de fonctionnalités

Avant de commencer le développement de la fonctionnalité de calcul d'itinéraire, nous avons besoin de nous assurer que deux stations (et uniquement deux) sont bien sélectionnées dans ArcMap.

⇒ Complétez la méthode `onClick()` du bouton *Calculer itinéraire* pour qu'elle réalise les opérations suivantes :

- s'il n'existe pas encore de classe d'entités *Stations* regroupant toutes les stations :
  - copie de l'ensemble des stations dans une unique classe d'entités *Stations* et suppression des doublons ;
  - ajout de la classe d'entités au document ;
- sinon, décompte du nombre de stations sélectionnées dans la couche :
  - si deux stations sont sélectionnées : lancer le calcul de l'itinéraire (ne rien faire pour l'instant) ;
  - sinon, afficher un message d'erreur.

Vous veillerez à informer l'utilisateur des actions effectuées.

Pour tester l'existence d'une classe d'entités, vous utiliserez la fonction `arcpy.Exists(object)`. Les ArcToolbox suivantes seront également utiles :

- Supprimer les doublons
- Combiner
- Compter
- Générer une couche