# NSI 1ère – Installation de Python

Nous allons commencer cette année par installer tout ce qui nous sera nécessaire pour programmer en langage Python.

1. **L’interpréteur Python :**

Le langage Python est un langage interprété.

Un ordinateur ne comprend pas le langage humain. Tout ce qu’il comprend, c’est le langage machine. Voici un bout de programme en langage machine :

Une image contenant texte

Description générée automatiquement

Vous n’y comprenez rien ? C’est normal.

J’ai écrit ce programme et pourtant, je suis incapable de vous dire ce que fait ce bout de code. Ce que j’ai écrit ressemble à ça :

Une image contenant texte

Description générée automatiquement

Bon, d’accord, vous n’y comprenez rien non plus, mais au moins, on reconnait des mots. Et quand on connait le langage de programmation utilisé, ici du C++, on comprend le programme.

Tout ça pour dire que l’ordinateur parle le **langage machine** qui est une série de nombre (si si, je vous assure, 2F est un nombre. C’est une autre façon d’écrire ce que nous écririons sous la forme 47, mais nous allons bientôt y venir) et qu’un programmeur ne va pas s’embêter à apprendre à parler ça. Les premiers programmeurs le faisaient, et puis ils ont vite inventé des moyens d’écrire les programmes dans un langage plus compréhensible par un humain.

Un programmeur écrit donc un fichier texte appelé **code source** et il utilise un programme pour traduire le code source en **langage machine**.

En C++, le programme qui fait la traduction est un compilateur : il lit le code source et le traduit en langage machine en écrivant un fichier exécutable dans ce langage. Si le programmeur modifie son code source, il faudra le compiler à nouveau. L’ordinateur n’a besoin que de l’exécutable pour faire tourner le programme. Le C++ est un langage **compilé**. Pendant la compilation, le programme n’est pas exécuté. Chaque machine ayant son propre langage, le même programme devra être compilé pour chaque type de machine sur lequel on veut le faire tourner.

En Python, le programme qui fait la traduction est un interpréteur : il ne génère pas d’exécutable. L’interpréteur traduit le code source au fur et à mesure de l’exécution du programme. Python est un langage **interprété**. Pendant l’interprétation, le code source est traduit pour être exécuté au fur et à mesure. Il suffit qu’une machine dispose de l’interpréteur pour qu’elle puisse se faire traduire le code source dans un langage qu’elle comprend. Pas besoin de créer des exécutables différents pour chaque machine. On distribue le code source et c’est tout.

Bref, il faut que nous installions le programme qui va traduire à notre ordinateur les codes sources que nous écrirons.

Pour cela, il faut aller chercher le dernier interpréteur Python à l’adresse suivante :

<https://www.python.org/>

Puis survoler **Downloads** avec la souris et télécharger la dernière version de Python :

Une image contenant texte, capture d’écran, moniteur

Description générée automatiquement

Comme pour la plupart des programmes que l’on cherche à installer sous Windows, on obtient un installateur, c’est-à-dire un exécutable qui va placer sur notre ordinateur tous les fichiers qu’il faut là où il faut. Une fois l’installation terminée, cet installateur ne sert plus à rien et peut-être effacé.

Au début de l’installation, pensez à cocher "Add Python 3.10 to PATH" **(Super important)**



√

Vous pouvez aussi personnaliser votre installation pour choisir un autre emplacement pour les fichiers de l’interpréteur Python. Dans ce cas, évitez de décocher les cases déjà cochées pour la personnalisation.

Si tout s’est bien passé, vous aurez un nouveau dossier à l’emplacement par défaut ou à celui choisi lors de la personnalisation. Il contient tous les fichiers nécessaires à l’interpréteur Python.

Félicitations !

1. **Installation des bibliothèques :**

Lors de notre apprentissage de Python, nous verrons que, lorsqu’une portion de code doit être utilisée à différents endroits, on peut créer une **fonction**. Cela permet d’écrire cette portion de code à un seul endroit, puis on appelle la fonction chaque fois que cette portion de code doit être utilisée.

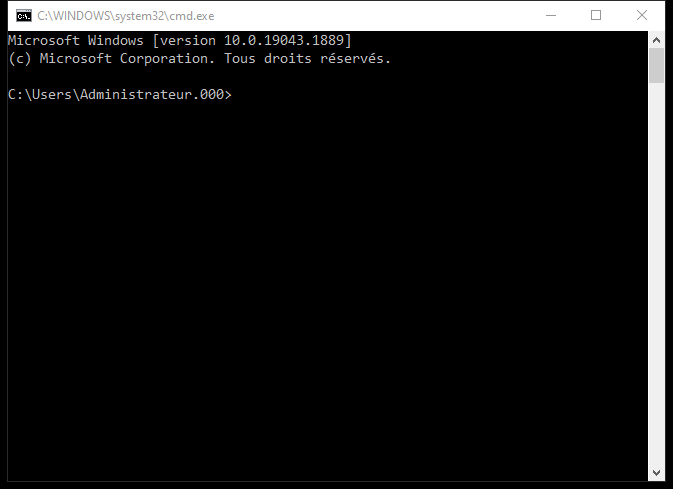
Quand on a écrit plusieurs fonctions correspondant à une thématique donnée, on peut les regrouper dans un même fichier (oui, on verra que le code source peut être réparti sur plusieurs fichiers). En Python, ce fichier s’appelle un **module**.

Un ensemble de plusieurs modules est appelé une **bibliothèque**.

Par exemple, vous pourriez écrire une fonction pour dessiner un cercle à l’écran, une autre pour dessiner un rectangle, etc… et vous allez regrouper ces fonctions dans un module pour dessiner dans une fenêtre. Et puis vous allez créer un autre module pour générer du son. Un autre pour savoir ce qui se passe au niveau de la souris ou du clavier, ce que l’on appelle les évènements qui permettent à l’utilisateur d’interagir avec le programme. Tous ces modules vont permettre de faire plein de choses et en particulier de créer des jeux. Vous allez donc les regrouper dans une bibliothèque spécialisée dans les jeux. Et vous en serez tellement fiers et cette bibliothèque sera tellement utile que vous pourrez la publier pour que d’autres puissent écrire leurs propres jeux avec votre bibliothèque. Ou bien, vous n’allez pas écrire tout ça, mais plutôt vous servir d’une bibliothèque déjà toute prête. Ça tombe bien, nous allons installer la bibliothèque pygame.

Pour installer une bibliothèque, il est possible de se rendre sur le site de cette bibliothèque, télécharger un installateur pour Windows et installer la bibliothèque. Mais dans le cas de pygame, on ne pourrait trouver qu’un installateur de la version 1.9.2 de pygame pour Python 3.2 (qui est un peu dépassé maintenant) et la version la plus récente de pygame est la 2.1.2 qui est compatible avec notre version de Python. Seulement, on n’a pas d’installateur pour cette version.

1. Le logiciel pipe

Le plus pratique est donc d’utiliser le logiciel pip. Il est fourni avec Python (sauf si vous avez décoché la case lors d’une installation personnalisée).

Ce logiciel ne fonctionne qu’en ligne de commande. Il faut donc ouvrir l’invite de commande. Le plus rapide est de taper Windows+R puis demander d’ouvrir cmd.

Vous vous souvenez du point super important lors de l’installation de Python ? On va en avoir besoin maintenant. Si Python a bien été ajouté au PATH, il n’y aura aucun problème.

1. Lister les bibliothèques

Le logiciel pip permet de gérer toutes les bibliothèques officielles Python. Pour commencer, nous allons lister celles qui sont déjà installées avec la commande pip list. Avec une installation fraiche de Python, on obtient :

Une image contenant texte

Description générée automatiquement

Si cela ne fonctionne pas, vous avez sans doute oublié de mettre Python dans le PATH. Dans ce cas, on vous informe que : 'pip' n’est pas reconnu en tant que commande interne…

Il faut soit désinstaller et réinstaller Python en prenant soin de l’ajouter au PATH, soit faire l’ajout manuellement.

1. Mettre à jour pip

On peut voir qu’il est nécessaire de mettre à jour pip (La version de Python utilisée était pourtant la plus récente en date). Il est fortement recommandé de le faire. Ainsi, nous allons utiliser la commande indiquée à l’écran : python.exe -m pip install --upgrade pip

Une image contenant texte

Description générée automatiquement

1. Installer une bibliothèque

Pip étant enfin à jour, on peut installer pygame avec la commande : pip install pygame.

**Une image contenant texte

Description générée automatiquement**

Et voilà. Bravo, vous avez installé votre première bibliothèque.

1. **Installation d’un éditeur de texte confortable :**
2. Choix de l’éditeur :

Un code source python peut être écrit en utilisant n’importe quel éditeur de texte. Donc, en théorie, le bloc-notes de Windows devrait suffire. Dans la pratique, il vaut mieux se doter d’un outil plus développé.

**Attention, un éditeur de texte n’est pas un traitement de texte. Il est hors de question d’écrire un code source avec Word. On parle ici de texte non mis en forme.**

Il existe un remplaçant du bloc-notes appelé Notepad2. Comme il n’est plus mis à jour depuis 2012, vous pouvez aussi vous tourner vers Notepad2-mod (dernière version en 2017). L’installateur de ces programmes vous permet de les installer de manière à remplacer complètement le bloc-notes d’origine de Windows. Je vous le conseille pour les fichiers txt ordinaires, les fichiers cfg ou ini. Cependant, cela peut être limité pour programmer dans la mesure ou cet outil ne permet pas d’ouvrir plusieurs codes sources dans la même fenêtre. L’exécution du code source depuis l’éditeur de texte nécessite quelques petits réglages préalables. Pour l’installer, télécharger l’installateur sur :

<https://github.com/XhmikosR/notepad2-mod/releases>

Prenez le fichier exe et exécutez le pour remplacer le bloc-notes. Même si on ne s’en servira pas pour coder en Python, cet outil reste très utile.

D’autres éditeurs de texte seront beaucoup plus adaptés. Vous avez peut-être déjà notepad++ sur votre ordinateur. On peut aussi citer des éditeurs de texte très complets tels que Atom ou Sublime Text (dont la version complète est payante, mais la version gratuite peut être suffisante). Ne les utilisant pas ou peu, je ne pourrai pas vous aider dans leur utilisation.

Moins léger que notepad++, mais moins lourd qu’Atom ou Sublime text, j’utiliserai Geany. Vous l’avez certainement déjà installé en SNT. Si ce n’est pas le cas, téléchargez l’installateur sur <https://www.geany.org/download/releases/> (prendre la version pour Windows 64 bits).

1. Coloration syntaxique :

Tous ces éditeurs disposent de la coloration syntaxique. Cela rend le code source plus coloré et donc plus agréable à l’œil, mais surtout, cela permet de repérer les fautes de frappes dans les mots clés du langage : si un mot n’est pas coloré comme il devrait, c’est qu’on l’a mal tapé.

1. Indentation et tabulations :

La plupart des langages de programmation conduisent à indenter le code source. L’indentation est l’écart par rapport à la marge.

On écrit donc parfois du code un peu plus loin de la marge.

Ou encore plus loin.

Dans des langages comme le C ou le C++, cela ne concerne que l’aspect esthétique du code source. C’est plus beau, plus aéré, et si c’est fait correctement, cela permet de mieux se repérer dans le code. Mais cela n’a aucun impact sur le fonctionnement du programme.

En revanche, en Python, l’indentation joue un rôle fondamental dans l’exécution du programme.

Il faut bien comprendre ce qu’est une tabulation ainsi que le rôle de la touche tabulation du clavier.

Une image contenant texte, équipement électronique, clavier

Description générée automatiquementLa touche tabulation est située juste à gauche de la touche A. Vous ne vous en êtes jamais servi ? Quelle erreur !

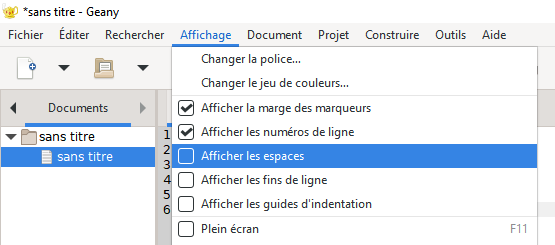
Dans la plupart des programmes, quand la fenêtre contient différentes zones de saisies, cette touche vous permet de passer de l’une à l’autre dans un certain ordre. Par exemple, quand vous devez saisir un login et un mot de passe, quand vous avez fini de taper votre login, un appui sur Tab vous fait passer dans la zone de saisie du mot de passe. Vos mains ne quittent plus le clavier pour aller sur la souris. Ça va beaucoup plus vite. Le raccourci Alt+Tab vous permet de changer la fenêtre active sans recours à la souris. En combinaison avec la touche majuscule, on circule en sens inverse. Un ordinateur dont la touche Tab est cassée perd une grande partie de son ergonomie.

Quand on utilise cette touche dans un éditeur de texte, elle rajoute un espace un peu plus large que celui obtenu avec la touche espace. Cet espace plus grand peut être constitué d’une suite d’espaces normaux (traditionnellement 4) ou d’un caractère tabulation.



En apparence, il n’y a aucune différence, mais nous verrons que pour Python, ce n’est pas du tout la même chose (et dans le fichier du code source, une tabulation occupe 1 octet alors que 4 espaces utilisent chacun 1 octet, soit 4 octets au total pour le même effet visuel).

Python va être très tatillon là-dessus. Je ne vous raconte pas le temps perdu par certains élèves pour comprendre pourquoi un code-source en apparence juste ne fonctionne pas. Heureusement, Geany dispose d’une option pour afficher les espaces blancs (un espace blanc est un espace, ou une tabulation).

En cochant la case "Afficher les espaces" du menu "Affichage", on peut voir la différence.

Une image contenant texte

Description générée automatiquementAinsi on passe de ça :

Une image contenant texte

Description générée automatiquement

A ça :

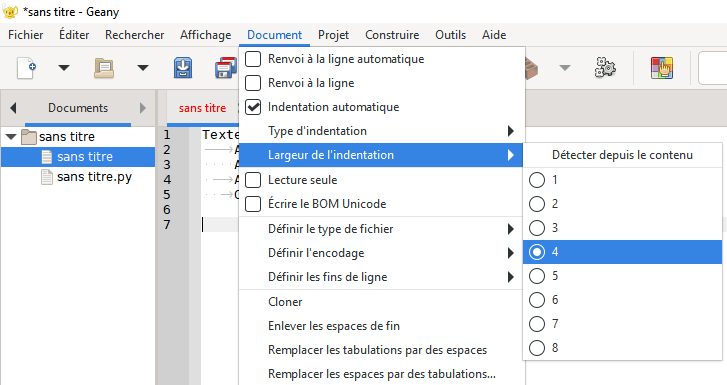
La tabulation étant "élastique", elle vient compenser des espaces classiques pour que le total soit équivalent à 4 espaces (ou à un nombre entier de fois 4 espaces). Ainsi, la ligne 4 commence par un espace suivi d’une tabulation et la ligne 5 par 2 espaces suivis d’une tabulation. Visuellement, on a l’impression qu’il y a 4 espaces au début de chaque ligne.

Par défaut, Geany est réglé pour que la touche Tab donne des tabulations : Une image contenant texte

Description générée automatiquement

C’est parfait comme ça, ne changez rien.

On peut aussi changer la largeur d’affichage d’une tabulation :



Le réglage par défaut (4) est un très bon compromis : trop et l’indentation nous entrainerait très vite trop loin de la marge, pas assez et on aurait du mal à la détecter. La majorité de ce que vous verrez dans des codes sources présentés en ligne utilise ce réglage. Il est préférable de s’y habituer.

Une autre fonctionnalité de Geany est de disposer de modèles de fichiers. Pour démarrer un projet en Python, on pourra donc aller dans Fichier -> Nouveau (selon un modèle) -> main.py

Une image contenant texte

Description générée automatiquement

Vous remarquerez que l’indentation se fait avec 4 espaces au lieu d’une tabulation.

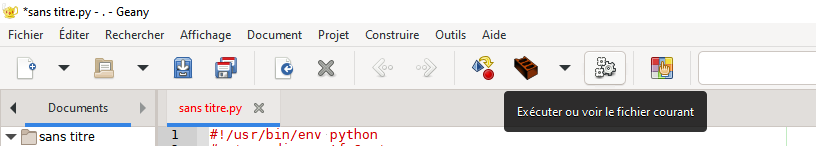
Heureusement, le menu Document permet de Remplacer les espaces par des tabulations. C’est très utile lorsque l’on veut rétablir rapidement l’indentation d’un code source. En effet, en général, si on fait un copier-coller d’une portion de code trouvée dans un tuto sur le web, on obtient une indentation avec des espaces. Et si on l’incorpore dans un fichier avec des tabulations, cela donne des erreurs. Cette fonctionnalité de Geany règle le problème en un clic.

1. Complétion automatique :

Un autre avantage en utilisant un éditeur de texte complet est la possibilité de ne pas toujours tout taper. Ainsi, après avoir tapé les 2 ou 3 premières lettres d’un mot, Geany vous propose différents mots commençants par les mêmes lettres. Avec les flèches haut et bas, on peut parcourir la liste et compléter par le mot sélectionné. On en reparlera. Geany n’est cependant pas l’éditeur le plus pratique de ce point de vue, mais il rend assez bien service.

1. Exécution du code :

Lorsque vous voulez exécuter votre code source, il suffit de cliquer sur l’icône en forme d’engrenages :



Pour ne pas avoir besoin de la souris, on peut aussi utiliser la touche F5 (sur vos claviers il faut appuyer sur Fn+F5).

Attention : pour que cela appelle l’interpréteur Python, il faut :

* Que votre fichier porte un nom se terminant par l’extension .py (de toute façon, si ce n’est pas le cas, vous n’aurez pas non plus de coloration syntaxique).
* Que Python ait bien été ajouté au PATH (dans le cas contraire, on doit indiquer à Geany où trouver l’interpréteur Python).

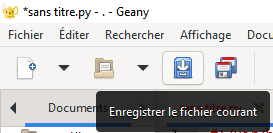
Remarque : Pour que le programme puisse être exécuté, le code source doit être enregistré. S’il s’agit d’un nouveau fichier (Sans titre.py (modèle avec coloration syntaxique) ou Sans titre (fichier vide sans coloration syntaxique)), une fenêtre pour enregistrer le fichier apparaitra.

**Pensez à bien repérer dans quel dossier vous enregistrez votre fichier et n’oubliez pas de mettre .py à la fin du nom.**

Si le fichier existait déjà, alors Geany l’enregistrera automatiquement sans rien demander. L’ancienne version sera écrasée.

1. Enregistrement des fichiers :

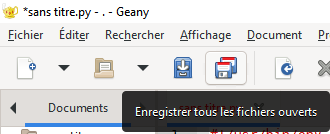
Comme on vient de le voir, Geany enregistre le code source avant son exécution. Mais il est conseillé d’enregistrer son travail régulièrement.

Le menu Fichier -> Enregistrer, l’icône Enregistrer le fichier courant ou le raccourci clavier Ctrl+S enregistrent le fichier par-dessus sa version précédente (si le fichier est nouveau, une fenêtre permet de choisir son nom et son emplacement).

Le menu Fichier -> Enregistrer sous… permet d’ouvrir la fenêtre permettant de choisir l’emplacement et le nom du fichier. Si le fichier existait déjà, le bouton Renommer permet d’effacer le fichier d’origine pour l’enregistrer à son nouvel emplacement et sous son nouveau nom. Sinon, le bouton Enregistrer permet d’enregistrer tout en conservant le fichier d’origine. Si le nouveau nom existe déjà, le fichier qui portait ce nom sera écrasé.

Geany permet de travailler sur plusieurs fichiers dans des onglets séparés. C’est pratique quand on veut avoir un projet Python réparti dans plusieurs fichiers py. Or, si on fait des modifications sur deux fichiers différents du projet et que l’on exécute le programme avec F5, seul l’onglet actif à ce moment là sera enregistré. Cela veut dire que les modifications apportées dans les autres fichiers du projet n’auront pas été enregistrées et donc l’exécution se fera avec un seul fichier comportant les modifications qu’on aura faites. Les autres fichiers seront utilisés dans l’ancienne version et on ne va pas comprendre pourquoi il y a des bugs.

Pour remédier à cela, Geany permet d’Enregistrer tous les fichiers ouverts :



On peut aussi utiliser le menu Fichier -> Tout enregistrer ou le raccourci clavier Maj+Ctrl+S. On est ainsi sûr de travailler avec les dernières versions de tous les fichiers.

On trouve le nom des fichiers ouverts dans l’en-tête de l’onglet. Geany affiche en rouge les noms des fichiers modifiés depuis leur dernier enregistrement. C’est un bon moyen de repérer ce qui n’a pas encore été enregistré.

**On ne le dira jamais assez : enregistrez souvent votre travail. N’importe quel programme, Geany compris, peut planter à tout moment. C’est rare, mais quand ça arrive et qu’on n’a pas enregistré son travail depuis une heure, c’est plus que frustrant.**

**A FAIRE POUR LA PROCHAINE FOIS :**

Nous travaillerons parfois avec des notebook Jupyter. Pour vous entrainer à installer une bibliothèque, vous installerez donc avec pip la bibliothèque notebook.

La méthode est la même que pour pygame. La seule différence, c’est que notebook a des dépendances, c’est-à-dire qu’il nécessite d’autres bibliothèques. Pour vous, cela ne change rien. Tout se fait automatiquement. Mais vous verrez beaucoup plus de choses s’afficher que lors de l’installation de pygame et cela durera un peu plus longtemps.