THÈME TRANSVERSAL : la programmation en python MmE TRÉHET

Vous utiliserez le langage de programmation Python pour débuter la programmation en mathématiques et en S.N.T., mais aussi pour commander des microcontrôleur en S.N.T.

## Les bases

### Un programme

Lorsqu’on utilise un moteur de recherche, un jeux vidéo, l’ordinateur ne décide pas de ce qui se passe à chacune de nos actions (clic souris, touche clavier, inclinaison manette,…), il respecte à la lettre un programme entièrement prévu à l’avance.

**Un programme est formé de lignes, dites « lignes de code ». Il indique un ensemble d’instructions (actions) destinées à être effectuées par une machine** (ordinateur).

### Le langage de programmation

Le langage compris par une machine est **le langage machine.** Ce sont des lignes constituées de 0 et de 1. Ce langage machine est binaire car chaque « case » ne peut être occupée que un « 0 » ou un « 1 ».

|  |  |
| --- | --- |
| **Langage humain** | **Langage machine** |
| 255 | 11111111 |
| 2 | 00000010 |

Pour créer un jeux vidéo, un logiciel, un site…, le programmeur (l’homme) utilise **un langage de programmation**, facilement compréhensible par l’humain. Le 1er langage de programmation que vous avez utilisé était Scratch.

Quand le programmeur veut exécuter un programme, il l’envoie à **un compilateur**, qui traduit le langage de programmation en langage machine, avant de l’envoyer à la machine. Le compilateur de Python fait parti **des interpréteurs** car il traduit le langage au fur et à mesure de son exécution.

**Bilan : Langage de programmation**

**Fichier « \*.py »**

**Python**

**Interpréteur**

**Python**

**Code binaire**

001101010011

**lignes de code écrite en**  système qui traduit **lignes de code écrite**

**langage de programmation** les lignes en Python **en langage machine**

compréhensible par l’homme en lignes de 0 et 1 compréhensible par

(surtout s’il a des bases d’anglais !!) **Vous utiliserez MuEditor** la machine

### Programmer

**« L’activité essentielle du programmeur consiste à résoudre des problèmes »**

Suivant, le langage de programmation utilisé, les instructions autorisées, et la syntaxe varient. Instructions et syntaxe doivent être respectées à la lettre sous peine de voir le programme « planter ». Il existe principalement 2 types d’erreurs :

**- les erreurs de syntaxe** : casse (minuscule/majuscule), indentation (=espace), ponctuation ( :, ., ( ), [ ], …), … L’interpréteur Python n’exécute pas un programme contenant une erreur de syntaxe. Il indique qu’il ne comprend pas la langue utilisée et l’endroit de l’erreur « SyntaxError » « Line2 ».

**- les erreurs sémantiques (logique)** : le programme ne fait pas ce que vous voulez, … L’erreur ne vient pas de la machine, mais de vous ! La machine exécute ce que vous avez écrit, et ce que vous avez écrit ne correspond pas à ce que vous vouliez !!!

## Environnement de travail

### Ce qu’on utilise

On utilise

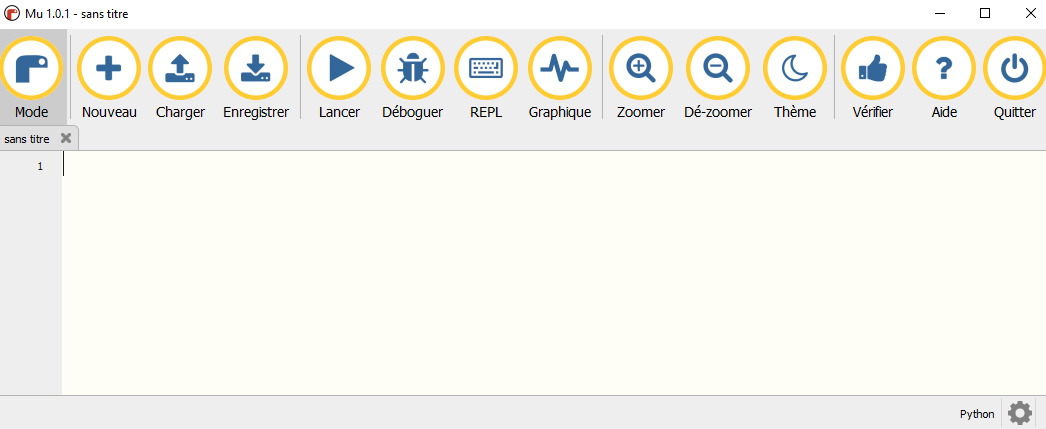




- le langage de programmation :Python3

- l’interpréteur : Mu- editor

Aller sous Appliserv/ SNT / Mu portable Lancer/ouvrir Mu-Editor : cliquer sur « Mu.launch ».



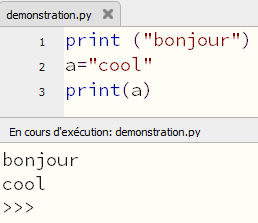
### Les essentiels du départ

#### Les icônes de base

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| nom du fichier actif |  | créer un nouveau fichier |  | Ouvrir un fichier existant |  | Enregistrer le fichier en cours | |
|  |  | | | | | |  |
| Permet de choisir le mode de fonctionnement de Mu. Pour l’instant, ce sera « Python »  Mode visible en bas, à droite de la fenêtre | Exécuter/Lancer le programme |
|  |
|  |
| Zone où on tape le programme |

#### Utiliser Mu

\* Écrire le programme ci-contre enregistré dans le fichier « demonstration » :

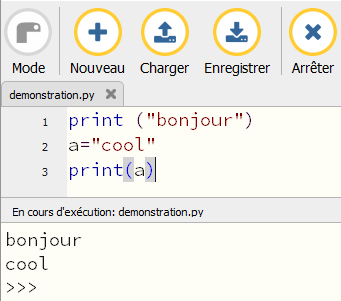


**Aide :**

\* Ouvrir un nouveau fichier

\* Enregistrer ce fichier vide sous le nom demonstration

\* Écrire le programme.



\* Exécuter le programme

**Aide :**

\* Lancer le programme.

- Une nouvelle zone, où s’exécute le programme, apparaît : c’est l’interpréteur.

- L’icône lancer se transforme en icône arrêter.

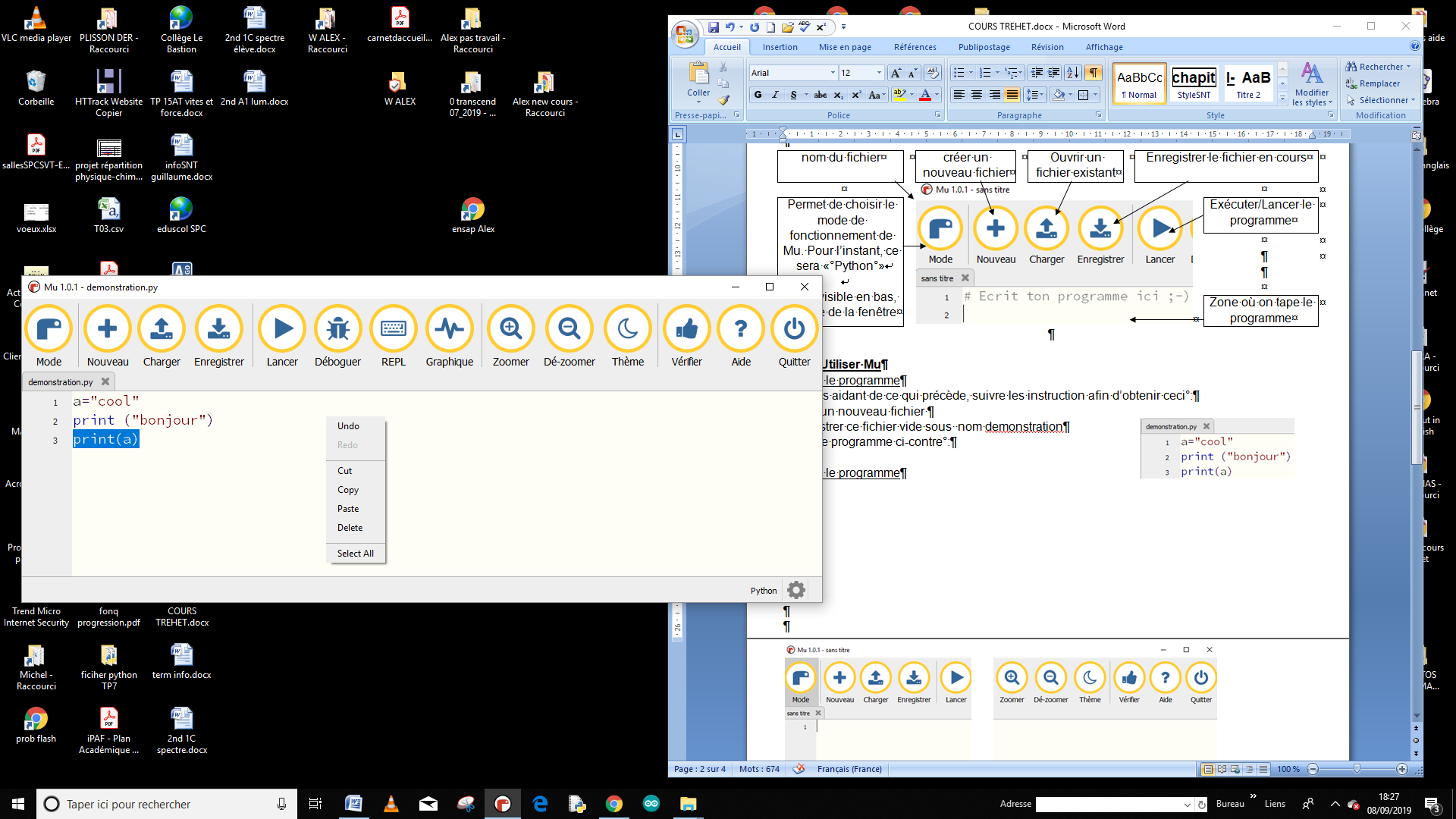
\* Quand le programme est terminé, une dernière ligne « >>> » apparaît dans l’interpréteur.

\* Vérifier que le programme s’est réalisé comme voulu.

\* Arrêter le programme. L’interpréteur se ferme. Seul le programme est visible.

#### Des outils sympathiques

\* Le « Clic droit dans la zone de programmation donne les possibilités habituelles :

**Undo** → annuler la dernière action (fonctionne plusieurs fois de suite)

**Redo**→ refaire l’action annulée

**Cut**→ couper la sélection **et** la garder en mémoire

**Copy** → copier la sélection

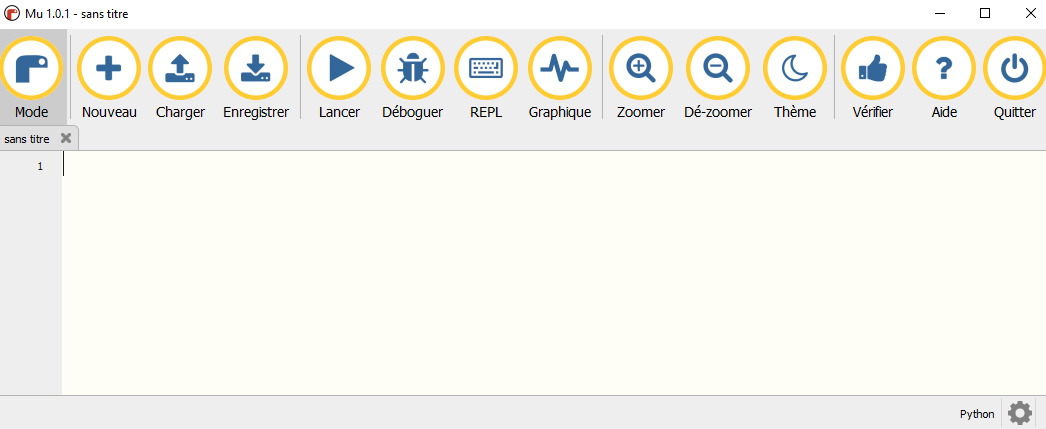
**Paste** → coller ce qui est en mémoire

**Delete** → effacer la sélection **sans** la garder en mémoire

**Select All**→ Tout sélectionner

**\* Cont+C Cont+V** **Cont+A** fonctionnent (copier/coller/sélectionner tout)

\* **enregistrer sous un nouveau nom** : double clic sur le nom du fichier au-dessus du programme.



\* la fin de la barre d’outils :

**- zoomer et Dé-zoomer** : changer la

taille de l’écriture

**- thème** : obtenir d’autres couleurs d’écriture et de fond (jour/nuit/contraste élevé)

**- vérifier** : surligne les erreurs de syntaxe sans avoir besoin de lancer le programme

**- aide :** aide en ligne

## Les bases de la programmations !

### La variable

Une variable désigne un emplacement, « une case » de stockage dans la mémoire de l’ordinateur. Cet emplacement porte un nom (lettre, mot). Exemple ci-contre : la variable s’appelle a.

Dans cette variable (emplacement), on peut stocker un objet (nombre, « mot », « phrase », …).

**Remarques :**

- La valeur d’une variable peut évoluer au cours du programme.

- une variable ne peut pas contenir d’accent

- un nombre décimal s’écrit avec un point, pas une virgule

- Un texte s’écrit toujours entre guillemets.

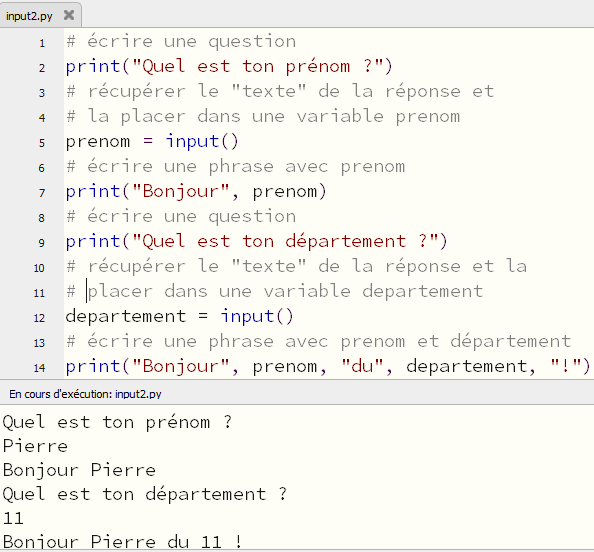
### Instruction : print( )

L’instruction **print( )** permet au programme d’écrire sur l’écran du « texte » ou le contenu d’une variable ou les deux mélangés. Le « texte » et les variables doivent être séparés par des virgules.

**#,** une ligne commençant par ce symbole **#,** n’est pas lue par le programme. Elle permet au programmeur d’expliquer une ou plusieurs lignes du programme.

### Instruction : input( )

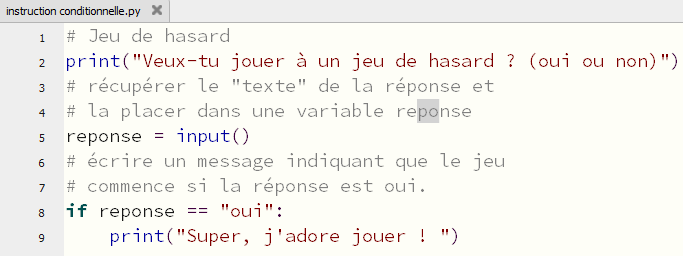
L’instruction input( ) permet de récupérer et d’utiliser les informations fournies par l’utilisateur.

L’instruction input( ) place **dans une variable « texte »** ce qui est tapé sur l’écran par l’utilisateur du programme. Ci-contre, Pierre et 11 sont du texte.

Dans l’interpréteur, il y a le texte écrit par le programme et le texte tapé par l’utilisateur.

Si on veut récupérer un nombre, il faut utiliser une autre fonction.

### Instruction conditionnelle

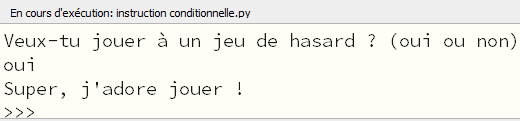
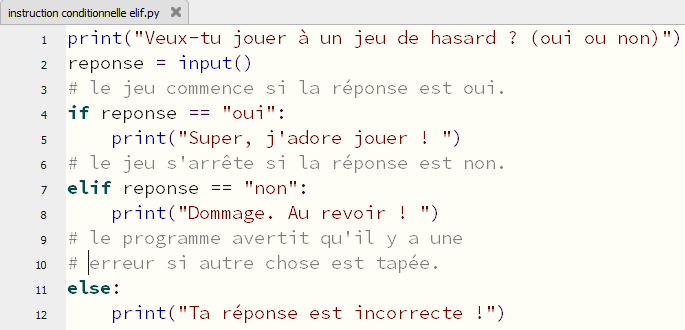
 Une instruction conditionnelle est une instruction qui ne s’exécute pas à chaque fois que le programme est lancé. Elle ne s’exécute que si une ou des conditions sont remplies.

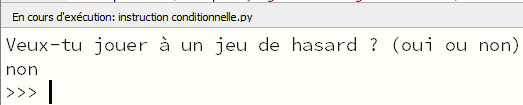
#### La structure « if condition : »

Si la condition est que deux valeurs soient égales, on teste la condition avec «  **= =** ». Voir ci-contre.

Si la condition est vérifiée, la partie du programme, qui est **indentée** (décalée de 4 espaces vers la droite) sous if, pourra s’exécuter. Sinon cette partie du programme est « sautée ».

On n’obtient donc pas toujours le même résultat à la fin du programme :



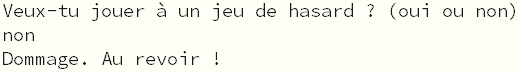
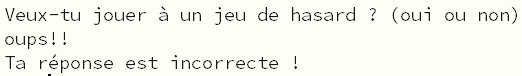


#### La structure « if condition : … elif : …»

**Elif** permet de vérifier d’autres conditions déterminées par le programmeur.

#### La structure « if condition : … elif :. else :. »

**Else** permet de traiter TOUTES les autres situations.

« L’une des compétence les plus importantes à acquérir au cours de votre apprentissage est celle qui consiste à DÉBOGUER efficacement un programme. Il s’agit d’une activité intellectuelle parfois énervante mais toujours très riche, dans laquelle il faut faire montre de beaucoup de perspicacité » et de volonté !