Type de document	Cours et exercices	Classe	1re	Durée	2h	Date	26/09/2022
Thème et contenu(s)	Langages et programmation – Représentation des données						
Capacités attendues	Comprendre l'utilité et les limites de l'utilisation de la console Python, être capable de créer des variables des 4 grands types primitifs : Entier, Flottant, Booléen, Chaîne de caractères						
Prérequis	Aucun						
Description	Cours et exercices, idéalement en salle informatique devant les machines						

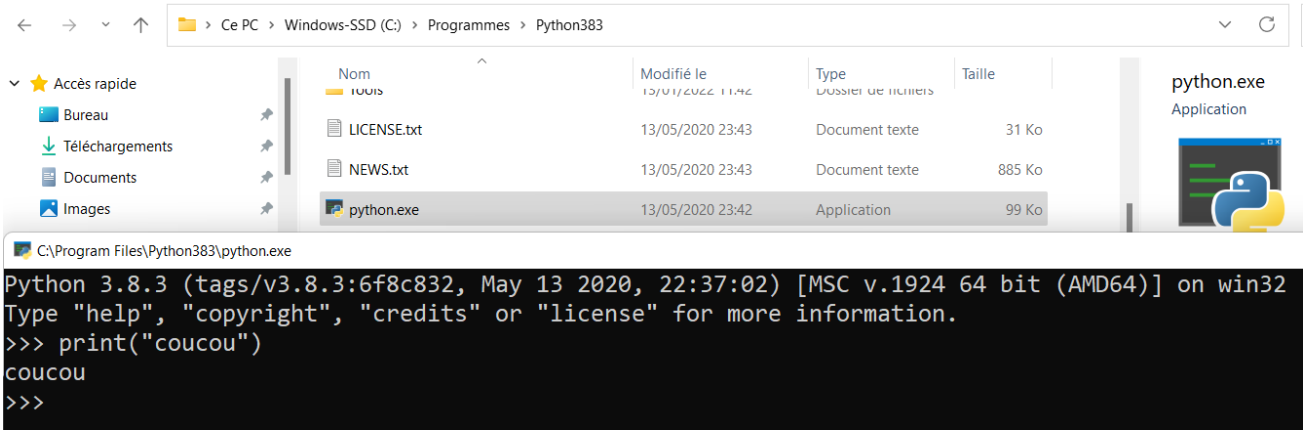
Les notions présentées ici sont la base de tous les langages de programmation. La clef de votre réussite est un entraînement régulier et poussé afin de comprendre ces concepts important. Les exemples succincts de ce document ne suffisent pas, il faut travailler de votre côté et expérimenter encore et encore, c'est aussi (voire surtout !) par l'erreur que l'on apprend la programmation.

I) La console Python

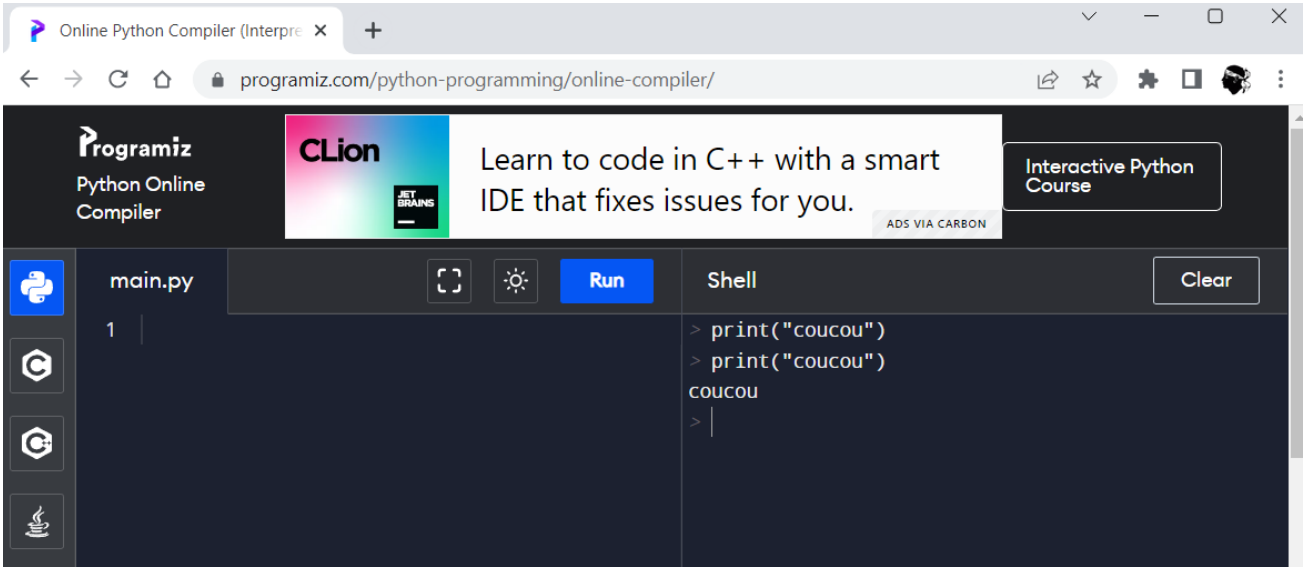
I - a) Invocation

Comme tout bon langage interprété, Python dispose d'une console (appelée parfois **terminal** ou **shell**) dans laquelle on peut donner, **une par une, des instructions à l'interpréteur**. Nous représenterons ici la console par une suite de 3 chevrons droits : >>>
On peut accéder à une console Python de 3 manières différentes (minimum) :

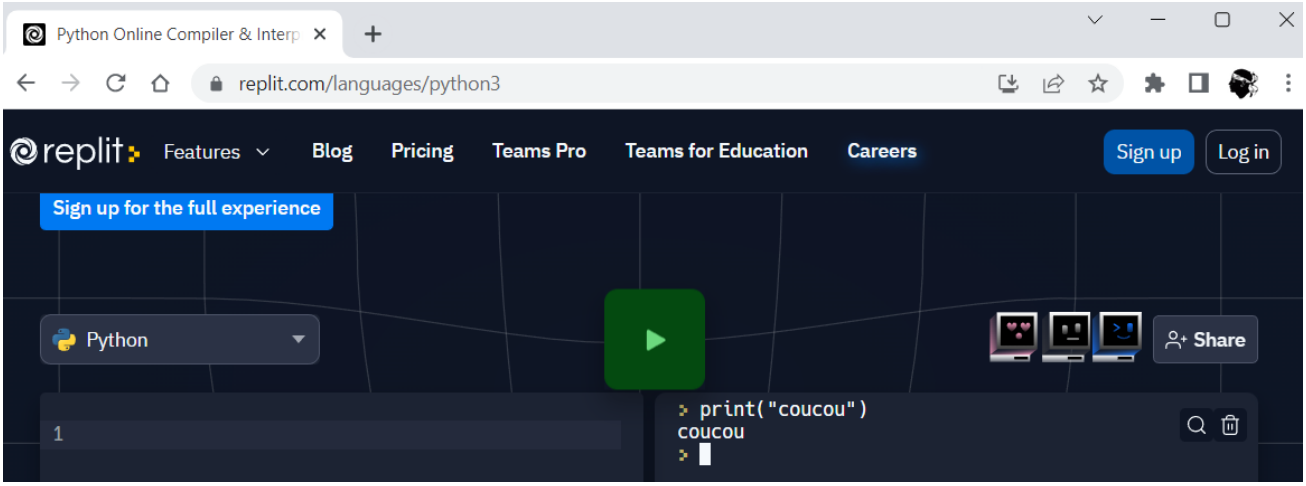
1. En utilisant la console Python directement présente sur notre machine (en supposant que Python y soit installé)
2. En utilisant une des nombreuses consoles disponibles en ligne (on utilise un navigateur Web, aucune installation n'est nécessaire)
3. En l'invoquant dans un Environnement de Développement Intégré (IDE)



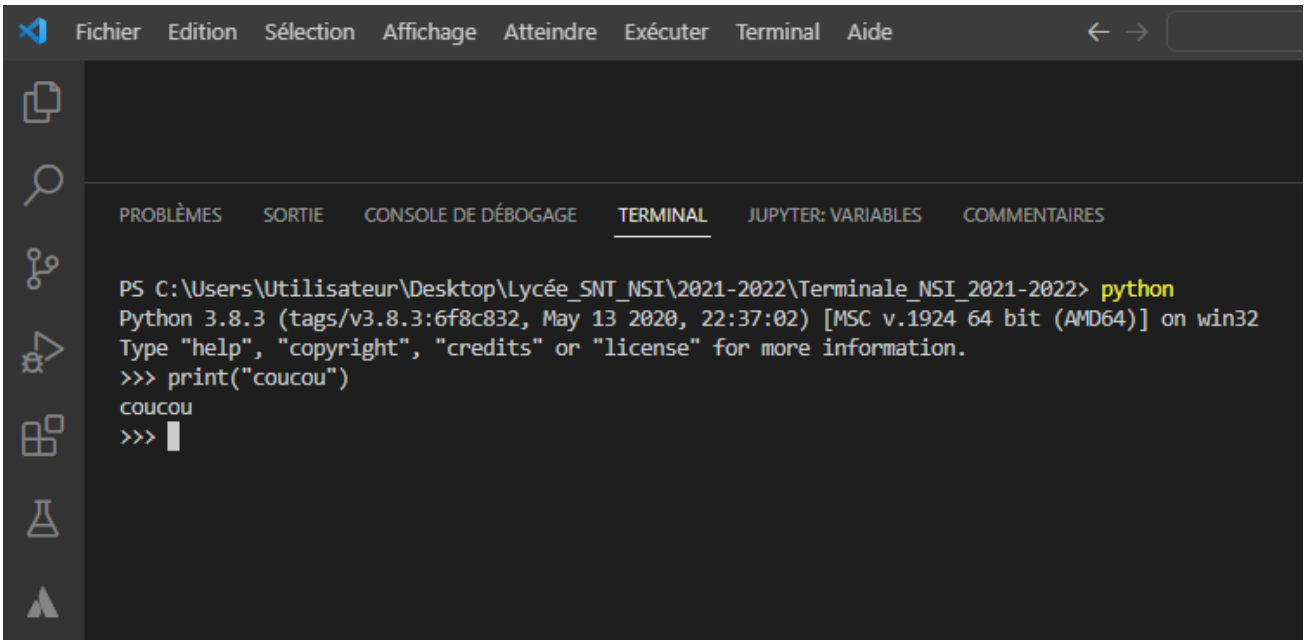
Console Python invoquée depuis notre PC Windows (1)



Console Python entièrement en ligne du site web programiz.com (2)



Console Python entièrement en ligne du site web replit.com (2)



Console Python de l'IDE VS Code (3)

I - b) Vocabulaire

La console Python peut être utilisée pour écrire n'importe quelle **expression**. Une **expression** représente un ensemble de **valeurs**, **opérateurs** et **variables**. Le rôle de tout langage de programmation est **d'évaluer des expressions**.

L'évaluation d'une expression produit un résultat

I - c) Premiers pas : mathématiques

```
>>> 2+3
5
>>> 8*6
48
>>> 2+3*9-2
27
>>> 9/2
4.5
>>> 9//2
4
>>> 9%2
1
>>> 8**2
64
```

La console Python se manipule en tapant une expression puis en appuyant sur la touche Entrée afin de demander à l'interpréteur de l'évaluer.
Nous allons tout d'abord explorer les possibilités mathématiques basiques de Python. Entrez les instructions ci-contre une par une.
Remarquez le respect de la priorité des opérateurs pour l'expression 2+3*9+2.
Que font selon-vous les opérateurs présents sur les 3 dernières lignes ?
Notez-le dans votre cours et apprenez-le !
Réponse, dans l'ordre : **Division euclidienne sans reste**, **modulo** (reste de la division entière), **puissance** (ici, 8^2)

I - d) Les opérateurs de comparaison

On peut aussi évaluer des expressions dont le résultat final n'est pas numérique mais Booléen : on appelle cela des expressions logiques.
Leur résultat est soit **True**, soit **False**. On voit ici l'analogie avec ce que nous avons dit concernant le binaire.
Bien sûr, ces expressions logiques peuvent être combinées à des expressions dont le résultat est numérique. L'idée est alors la suivante : Python évalue « tant qu'il peut évaluer » afin de produire un résultat final, c'est-à-dire qu'il va faire tous les calculs numériques nécessaires pour enfin faire sa comparaison logique.
Testez les expressions logiques ci-contre. Listez les 6 opérateurs de comparaison en Python
Réponse : **<, >, <=, >=, ==, !=**

```
>>> 3<2
False
>>> 2.5>5/2
False
>>> 8>=8
True
>>> 5<=3*2
True
>>> 2*3!=3*3
True
>>> 8**2==2**6
True
```

I - e) Les opérateurs booléens

Il est temps maintenant de mettre en pratique nos fameuses tables de vérité (qui peuvent être retrouvées en demandant à la console Python de faire des calculs). On considèrera que **False** équivaut à **0**, et que **True** équivaut à **1**.
Entrez l'extrait ci-contre dans la console, puis vérifiez « à la main » les tables de vérité complètes **or**, **and**, **nor**, **nand**, et **xor**.

```
>>> True or False
True
>>> False and True
False
>>> False or False
False
>>> True ^ True
False
>>> True ^ False
True
>>> not(True)
False
>>> not(False)
True
```

II) Notion de variable

Il n'est pas de langage de programmation qui ne permette pas à un moment donné de mémoriser une valeur qui nous intéresse. On utilse pour cela une **variable**. Le concept de variable est la fondation de quasiment tout le programme de NSI et, au-delà, la base de tout programme informatique. Sans cette notion, on ne peut pas programmer.

II - a) L'affectation

Toujours comme en mathématiques, une des choses les plus utiles en programmation est l'affectation : elle consiste à placer une valeur dans un conteneur appelé variable. La syntaxe, très simple mais très importante et à connaître par □, est la suivante :

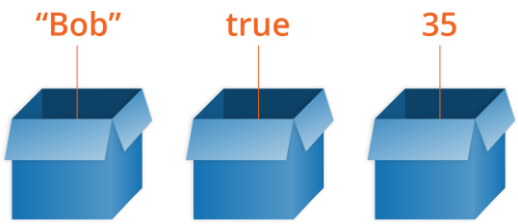
variable = valeur

Cela ne marche pas dans l'autre sens (on dit que ce n'est pas réflexif). Une variable doit être nommée correctement et respecter les trois règles suivantes :

- Toujours commencer par une lettre (de préférence minuscule, c'est une convention)
- Ne pas comporter d'espace
- Être unique

Par exemple, écrire en Python : x=3 revient à dire que l'**on place la valeur 3 dans la variable nommée x**.

Source : MDN Mozilla



https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps/Variables

En fait, la valeur que l'on affecte dans une variable peut être littérale (ne nécessitant aucun calcul) mais peut être aussi une expression, dont l'évaluation aboutit à une valeur littérale. Enfin, on peut affecter une variable (ou plutôt, sa valeur) à une autre variable.

```
>>> b=3
>>> a=b
>>> a
3
>>> b
3
```

II - b) Types de variables

Les variables peuvent être de plusieurs types, les plus courants sont :

- Entier
- Chaîne de caractères (on les écrit avec les « guillemets du 3 », appelés « double quotes »)
- Flottant (nombre à virgule, attention en informatique la virgule mathématique est un point)
- Booléen (True ou False)

II - c) Réutiliser une variable

Pour utiliser une variable ailleurs dans un calcul, il suffit de l'appeler par son nom. Bien entendu, on ne peut l'utiliser que si elle est définie, comme le montre la capture suivante :

```
>>> nsi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nsi' is not defined
>>> nsi = 14
>>> nsi
14
>>> toto
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'toto' is not defined
>>> toto = 2**2+nsi
>>> toto
18
```

II - d) Écraser une variable

Affecter une nouvelle valeur à une variable **écrase** son contenu précédent. Il n'y a plus aucune trace de ce dernier. Vérifiez cela en entrant le code ci-contre dans le shell Python.

```
>>> a=5
>>> a
5
>>> a=8
>>> a
8
```

II - e) Comment permuter deux variables ?

On se propose maintenant d'échanger le contenu de deux variables. Comment fait-on ?