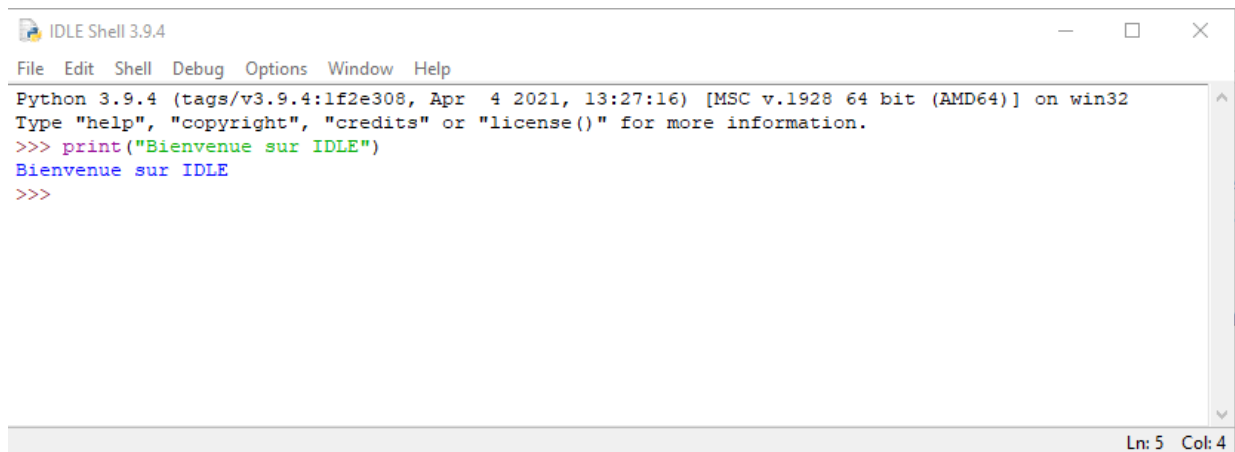

Éléments de base du langage Python

1 Un éditeur et un interpréteur

1.1 IDLE Python

Le langage python utilise un **interpréteur** pour exécuter les programmes. C'est lui qui s'ouvre lorsqu'on lance python dans la liste des programmes sous le nom *IDLE Python* ou alors depuis un terminal par la commande *python*.



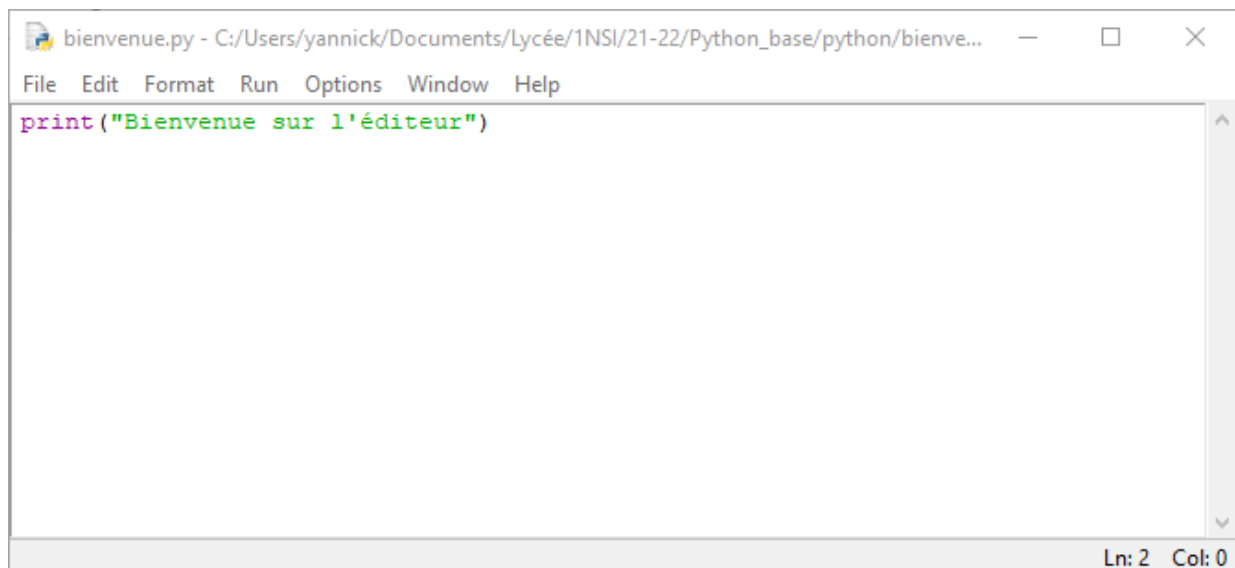
```
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 4 2021, 13:27:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Bienvenue sur IDLE")
Bienvenue sur IDLE
>>>
```

Remarque

On reconnaît facilement l'**interpréteur** car la ligne commence toujours par >>>.

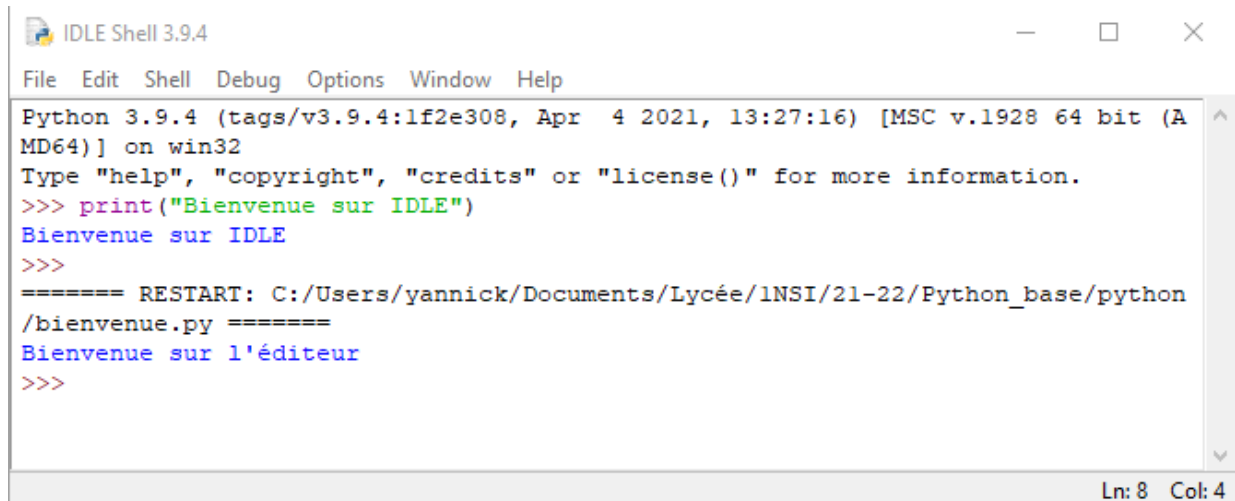
Pour écrire les programmes sur plusieurs lignes, on utilise un **éditeur**. Il faut un **éditeur** qui reconnaît le langage python pour que la syntaxe soit colorisée. L'interpréteur dispose d'un menu qui permet d'ouvrir l'éditeur par le menu File -> New File.

Une nouvelle fenêtre apparaît! C'est l'éditeur et c'est dans celle-ci qu'on saisira nos programmes.



```
print("Bienvenue sur l'éditeur")
```

Il seront lancés par un appui sur la touche F5 après avoir enregistré son fichier dans son espace de travail.

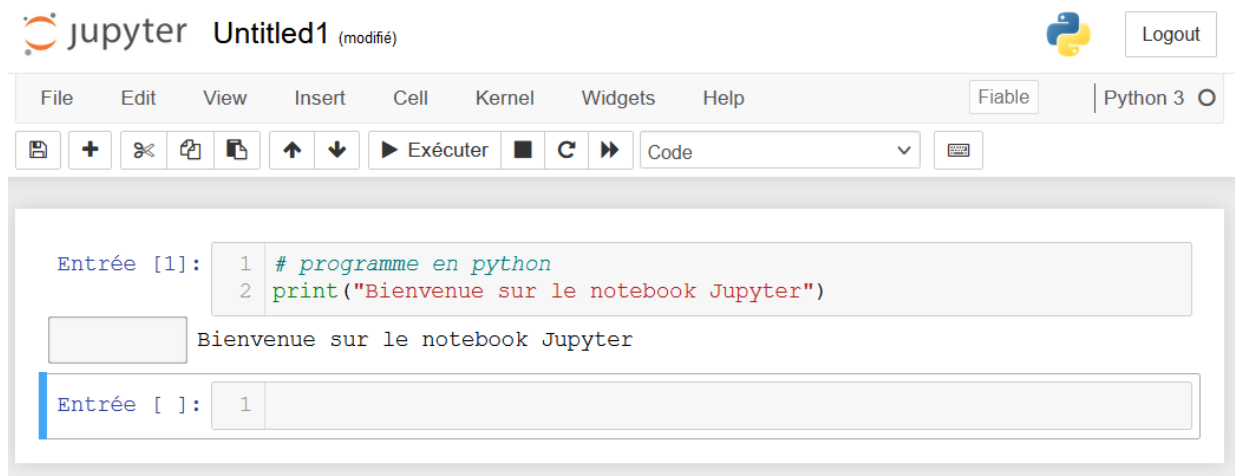


```
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 4 2021, 13:27:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Bienvenue sur IDLE")
Bienvenue sur IDLE
>>>
===== RESTART: C:/Users/yannick/Documents/Lycée/1NSI/21-22/Python_base/python /bienvenue.py =====
Bienvenue sur l'éditeur
>>>
```

1.2 Notebook JUPYTER

Un **notebook** est une application web qui permet avec un navigateur (firefox, chrome, etc) de faire de la programmation en python et aussi avec d'autres langages.

La page web propose des cellules pour écrire ses programmes en python.



- On peut l'utiliser comme client seul qui suppose une installation sur le PC.
- On peut l'utiliser sur un serveur. Son principal atout est la centralisation des travaux sur une seule machine qui est accessible dans tout le lycée sur n'importe quel poste.

Un **notebook** est accessible au lycée depuis un navigateur (firefox, chrome, etc.) en saisissant l'adresse <http://dumont.jupyter.lyc14.ac-caen.fr:31416>

L'authentification sur le serveur se fait avec le **login** constitué du **nom** et **prénom** accolés, sans espaces, ni majuscules, ni caractères accentués.

Le mot de passe est votre date anniversaire au format *JJMMAAAA*.

Il y a aussi un notebook jupyter sur l'ENT. Il est à utiliser plutôt à la maison pour afficher les notebooks.

2 La base du langage Python

2.1 Variable en python

Une variable est un espace de stockage, un espace mémoire qui va recevoir une valeur. Cette valeur est associée au nom de la variable qui sera choisi.

La variable peut être déclarée (nommée) par une seule lettre, un mot de plusieurs lettres ou une combinaison alphanumérique mais sans commencer par un chiffre.

Remarque

Un nom donné à une variable doit être facile à comprendre, savoir la grandeur qu'elle représente.

Exemple

- Une variable qui représente un nombre : **i**, **n**, **x**, **y**, etc
- Une variable qui représente une chaîne de caractères : **mot**, **nom**, **texte**, etc
- Deux variables qui représentent des grandeurs similaires peuvent avoir des noms proches mais qui se différencient par un chiffre placé à la fin : **mot1** et **mot2** ou **x_1** et **x_2**, etc

2.2 Affectation d'une variable en python

Donner une valeur à une variable est une **affectation**. En **python**, l'affectation se fait avec le signe mathématique égal **=**.

La signification est donc très différente de celle rencontrée en mathématique !

En Python :

$x = x + 2$ signifie que la variable x est augmentée de 2 en valeur. Donc si la variable x vaut 3, après l'affectation, la variable x vaut 5. Elle est augmentée de 2.

En maths :

$x = x + 2$ est une égalité dans laquelle le nombre x a la même valeur, qu'il soit à gauche ou à droite du signe égal. Dans ce cas précis, il n'existe pas de nombre vérifiant l'égalité, donc il n'y a pas de solution.

2.3 Affectation simple et multiple

Il existe plusieurs façons d'affecter des valeurs à des variables.

1. Affectation directe simple :

- **a=4**
- **b=5**
- **c=6**

2. Affectation indirecte simple :

- **a=3**
- **b=4**
- **c=a+b**

3. Affectation en ligne ou multiple :

- **a,b,c=3,4,6** ici **a=3**, **b=4** et **c=6**
- **a=b=c=1** ici **a=1**, **b=1** et **c=1**

3 Saisir et afficher des données dans une variable

3.1 Saisir avec input

Pour saisir au clavier la valeur d’une variable, on utilise la fonction **input**.

Exemple

```
input("a=")
```

La chaîne “a=” est le **prompt** c’est à dire le message affiché pour indiquer la saisie à effectuer.

Attention, ainsi écrit, aucune valeur n’est mémorisée dans une variable ! Pour mémoriser la valeur dans une variable, il faut une affectation à une variable.

Exemple

```
a=input("a=")
```

Remarque

C’est une chaîne de caractères qui est saisie. Si on veut saisir un nombre entier ou flottant, il faudra transformer la chaîne “nombre” en nombre avec les fonctions **int** ou **float**.

```
a=input("a=")
a=int(a)
```

ou en une seule instruction

```
a=int(input("a="))
```

3.2 Afficher avec print

Pour afficher la valeur d’une variable, on utilise simplement la fonction **print**.

Exemple

```
# La valeur contenue dans la variable a est affiché.
print("La variable a vaut :", a)
```

4 Types d’une variable en python

1. Les variables ont un **type** différent selon la valeur qui lui est donnée. Au moment de l’affectation, la valeur est stockée en mémoire et python lui attribue un type.
2. Une variable peut représenter un nombre entier ou à virgule, un booléen, une chaîne de caractères ou un groupe de valeurs rassemblées en une liste ou un tableau.
3. En python, le type d’une variable peut changer selon le contexte. Une variable peut être un nombre puis devenir une chaîne de caractères ou inversement.
4. La fonction “type” est une fonction en python qui permet de connaître le **type** d’une variable.

4.1 Le type numérique

En python, une variable peut désigner un nombre **entier** ou un nombre **flottant** (nombre réel en mathématiques).

Avec les variables de type numérique, on peut utiliser des opérateurs mathématiques comme la somme (+), la différence (-), le produit (*) et le quotient (/).

Selon l'opérateur, le résultat peut avoir un type différent des valeurs avant le calcul. Par exemple, le quotient de 2 nombres entiers peut donner un résultat réel, donc un flottant.

En plus des 4 opérations mathématiques, il existe en Python des opérateurs mathématiques tels que:

Opérateur	Opération mathématique
x//y	quotient entier de la division de x par y
x%y	reste entier de la division de x par y
abs(x)	valeur absolue de x
int(x)	partie entière de x
float(x)	conversion de x en flottant
x**y	x à la puissance y

4.2 Le type booléen

Une variable de type **booléen** prend 2 valeurs possibles : **True** (vrai) ou **False** (faux).

Ces deux valeurs, True ou False sont aussi notées 1 pour **True** et 0 pour **False**.

Les opérateurs pour les variables de type booléen sont le “ET” logique, le “OU” logique et la négation “NON”.

En python, ces opérateurs sont “AND”, “OR” et “NOT”.

On redonne la table de vérité de ces opérations:

x	y	not(x)	x or y	x and y
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

4.3 Le type string ou chaîne de caractères

Une variable de type **string** est une variable dont la valeur est constitué de lettres, chiffres, espaces et de tout caractère alphanumérique et de symboles.

L'affectation d'une chaîne de caractère à une variable se fait en notant la chaîne entre guillemets doubles “*chaîne de caractères*” ou simples ‘*chaîne de caractères*’.

Des variables de type string peuvent être réunies par **concaténation**. L'opérateur de concaténation est le +.

Exemple

1. Déclaration et affectation des variables txt1 et txt2 de type string:

- `txt1 = "La vie est belle"`
- `txt2 = 'mais cruelle parfois!'`

2. Concaténation des contenus des variables dans une variable `txt`: `txt = txt1+txt2`

la variable `txt` aura pour valeur la chaîne : "La vie est belle mais cruelle parfois!"

5 Structure conditionnelle

5.1 Les tests

Un test compare des valeurs et renvoie un booléen : **True** ou **False**

Les opérateurs de comparaison sont:

- `==` pour comparer deux valeurs égales : `a == 3`,
- `!=` pour comparer deux valeurs non égales.
- `<`, `>`, `<=` et `>=` pour comparer comme en maths : `<`, `>`, `<=` et `>=`.
- Les opérateurs NOT, AND et OR peuvent être utilisés dans les tests.

Exemple

Soit la variable `a = 5` :

- `a < 3` est **False**
- `a! = 0` est **True**
- `a > 0 and a <= 5` est **True**
- `a < 0 or a! = 5` est **False**
- `not(a == 5)` **False**

5.2 Structure conditionnelle SI

On peut soumettre l'exécution d'une instruction à une condition.

La structure est de la forme:

```
if test:
    instructions
```

Le test renvoie une valeur booléenne.

- Si la valeur du test est **True**, les instructions sont exécutées;
- Si la valeur du test est **False**, aucune instruction est exécutée.

Exemple

```
if a>0:
    print("Le nombre saisi est positif")
```

5.3 Structure conditionnelle SI ... SINON

Lorsque le test de la condition **if** est à **False**, il est possible d'exécuter d'autres instructions. Elles sont introduites par le mot **else**.

La structure devient :

```
if test:
    instructions
else:
    autres instructions
```

Exemple

```
if a>0:
    print("Le nombre saisi est positif")
else:
    print("Le nombre saisi est négatif")
```

5.4 Structure conditionnelle SI ... SINON SI ... SINON

La commande `else` **ne contient pas de test**. Il est possible de soumettre un nouveau test avec la commande `elif`. La structure est alors de la forme:

```
if test 1:
    instructions
elif test 2:
    autres instructions
elif test 3:
    encore des instructions
else:
    les dernières instructions
```

Exemple

```
if a>0:
    print("Le nombre saisi est positif")
elif a<0:
    print("Le nombre saisi est négatif")
else:
    print("Le nombre saisi est nul")
```