Type de document	Cours et exercices	Classe	1re	Durée	2h	Date	11/10/2022
Thème et contenu(s)	Langages et programmation – Représentation des données						
Capacités attendues	Bases de Python : notion d'itérable, utilisation dans les boucles						
Prérequis	Connaître les 4 types de base : Boolean, Integer, Float, String						
Description	Cours et exercices, idéalement en salle informatique devant les machines						

I) Les itérables

Un **itérable est un objet Python** dont on peut parcourir les valeurs (par exemple à l'aide d'une boucle). Il existe en Python beaucoup d'itérables, dont par exemple la **chaîne de caractères** (String). Cela signifie qu'on peut parcourir une chaîne **caractère par caractère**.

II) Boucles

Le but d'une **boucle** est d'effectuer un certain nombre de fois (déterminé ou indéterminé) une/plusieurs instructions. Dans le cas que nous allons voir, nous utilisons des boucles pour **itérer sur des itérables**.

II - a) La boucle **for**

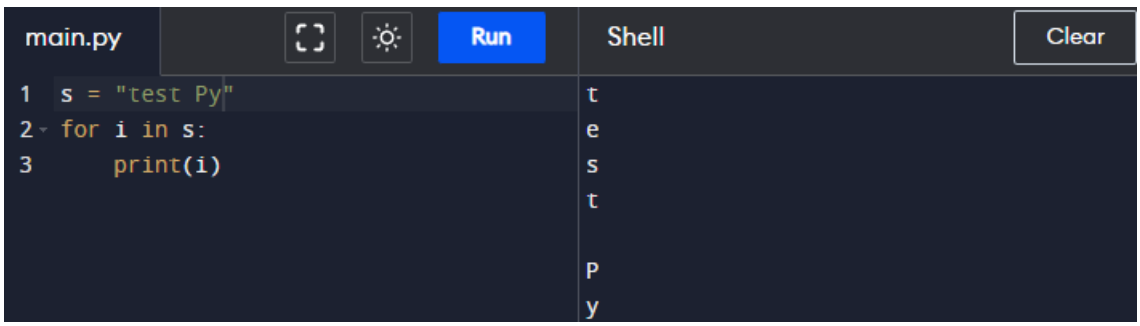
La boucle **for** s'appelle grâce à l'instruction Python…`for`. Sa syntaxe est la suivante :

```
for <variable> in <itérable> :
        <instruction_1>
        <instruction_2>
        …
        <instruction_n>
```

Remarques importantes :

- Comme pour le if , ne pas oublier les « : » et respecter l'identation
- Toujours comme pour le if, faire attention à ce qu'on veut mettre dans le for (qui sera exécuté plusieurs fois)
- On peut nommer la variable située après le for comme on veut : elle n'a pas besoin d'être déclarée avant ça

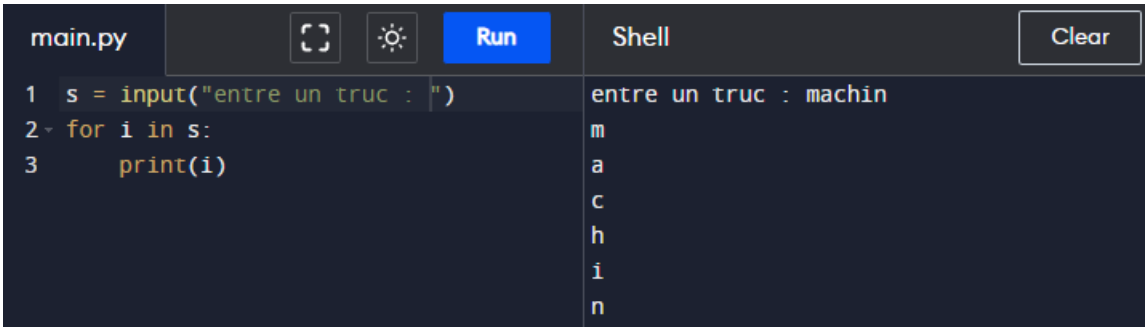
II - b) Exemple basique



Il faut bien comprendre ce que fait la variable `i` : elle est créée sans qu'on lui affecte aucune valeur, et **va valoir successivement tous les éléments de l'itérable** (donc successivement toutes les lettres de la chaîne). C'est un des rares cas où on n'utilise pas la syntaxe `variable = valeur`.

Il faut **toujours** choisir une variable qui n'a pas été utilisée auparavant par notre programme (si `i` est prise, utiliser `j`…etc.)

II - c) Exemple avec une chaîne entrée par l'utilisateur

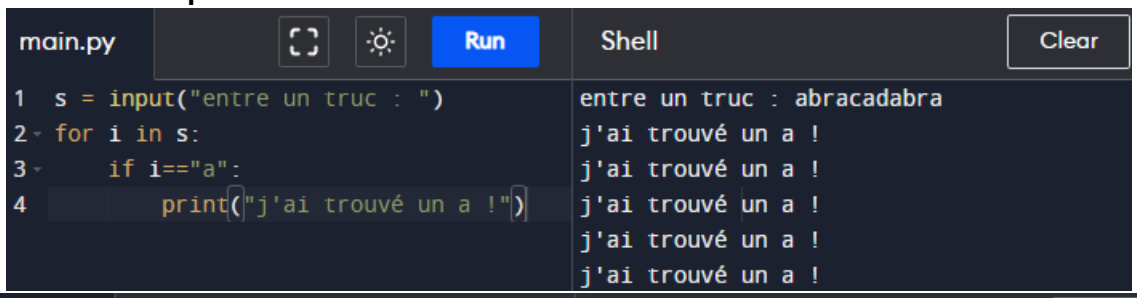


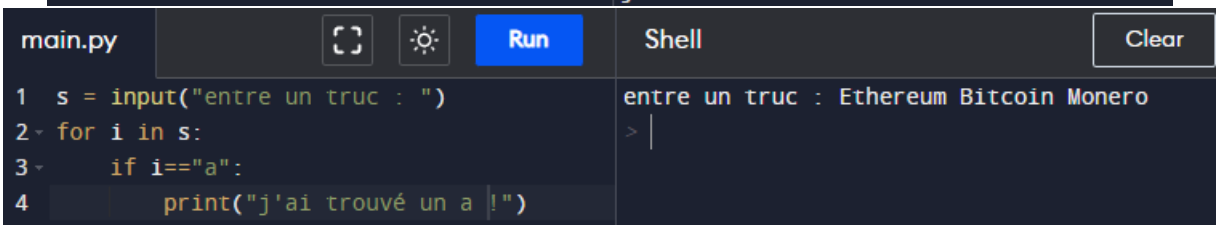
III) Exercices

III - a) Exercice 1

Consigne : Demander à l'utilisateur d'entrer une chaîne de caractères puis afficher « j'ai trouvé un a » dès qu'un a est détecté. Ne rien faire sinon.

Correction et exemple d'utilisation :

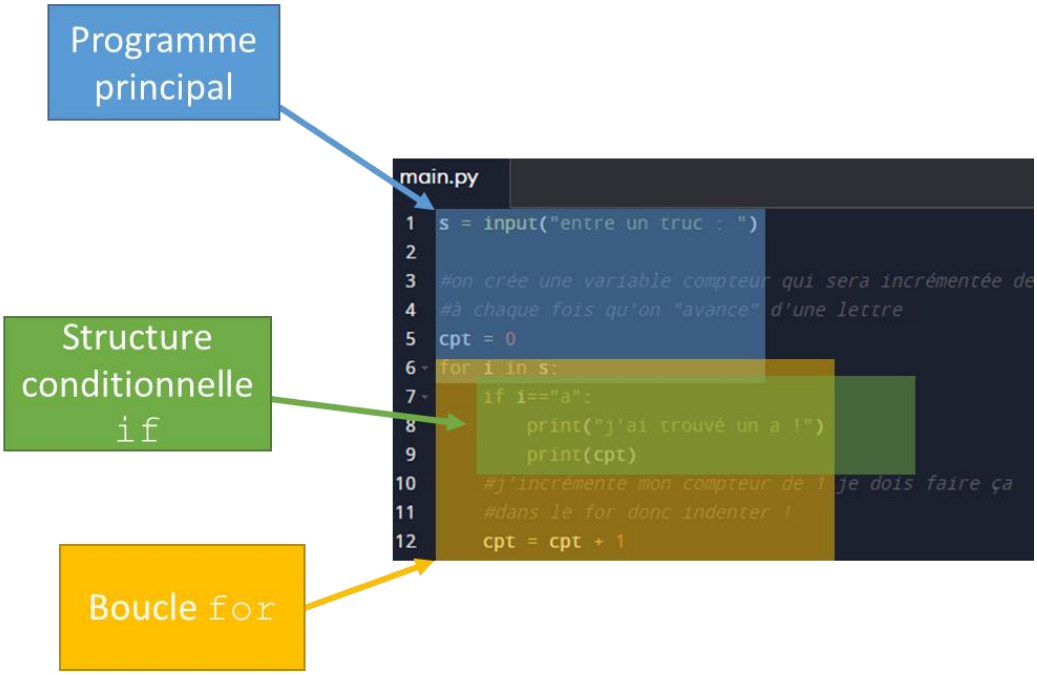




III - b) Exercice 2

Consigne : Demander à l'utilisateur d'entrer une chaîne de caractères puis afficher « j'ai trouvé un a à la position XX» dès qu'un a est détecté, XX désignant l'emplacement de la lettre « a » (on démarre de zéro). Par exemple, dans la chaîne « abracadabra », on trouve un « a » à la position 0,3,5,7,10.

Correction et exemple d'utilisation :



Remarques : la difficulté ici est de savoir comment « compter ». On a vu qu'une variable pouvait contenir une valeur, ou une expression. Ici, on va réaffecter à cpt son ancienne valeur + 1. Si on ne faisait pas ça « dans le for », cpt resterait à zéro…
On a un cas ici ou un `if` se trouve dans un `for`. L'indentation doit quand même être effectuée sous peine d'obtenir des comportements bizarres, voire des erreurs…Le schéma peut donc vous aider.

III - c) Exercice 3

Consigne : Demander à l'utilisateur d'entrer une chaîne de caractères puis afficher combien de fois la lettre « a » se trouve dedans.
Correction et exemple d'utilisation :

main.py — Run — Shell

```
1  s = input("entre un truc : ")
2  #on crée une variable compteur qui sera incrémentée de 1
3  #à chaque fois qu'on "avance" d'une lettre
4  cpt = 0
5  for i in s:
6      if i=="a":
7          cpt = cpt + 1
8  #syntaxe du print qui mélange des variables et du texte écrit à l'avance
9  #c'est juste une manière de faire plus joli lorsque ça s'affiche dans la console
10 print(cpt, "a se trouvent dans cette chaîne")
```

```
entre un truc : abracadabra
5 a se trouvent dans cette chaîne
>
```

Ici, le compteur ne s'incrémente que si on est « rentré » dans le `if`, en d'autres termes il ne s'incrémente que si on a trouvé un « a ». Une fois la boucle finie, le programme reprend son exécution normale et la ligne 10 est calculée.

III - d) Exercice 3

Consigne : Demander à l'utilisateur d'entrer une chaîne de caractères, puis d'entrer une lettre, et enfin afficher combien de fois la lettre demandée se trouve dans la chaîne.
Correction et exemple d'utilisation :

main.py — Run — Shell

```
1 s = input("entre une chaîne : ")
2 l = input("entre une lettre : ")
3 cpt = 0
4 for i in s:
5     if i==l:
6         cpt = cpt + 1
7 print("la lettre",l,"se trouve",cpt,"fois dans la chaîne")
```

```
entre une chaîne : abracadabra
entre une lettre : b
la lettre b se trouve 2 fois dans la chaîne
>
```

Ne vous laissez pas impressionner par le `print()`, il ne fait que mélanger des variables et du texte défini à l'avance, le tout séparé par des virgules.

IV) Créer un itérable avec `range()`

Pour plusieurs raisons, on peut vouloir itérer un certain nombre de fois dans un **itérable composé de nombres** et pas de caractères. On utilise pour cela la fonction `range()`, qui crée une suite de nombres sous forme d'itérable. On peut soit utiliser directement cet itérable dans une boucle `for`, soit le stocker et l'utiliser plus tard.

Attention ! L'itérable généré démarre à zéro… `Range(n)` crée un itérable allant de `0` à `n-1`
Par exemple, `range(4)` va créer un itérable composé des valeurs `(0,1,2,3)`

Voici un exemple des deux cas :

main.py — Run — Shell

```
print("directement dans le for :")
for i in range(5):
    print(i)

print("d'abord mis dans une variable et utilisé ensuite :")
r = range(5)
for i in r:
    print(i)
```

```
directement dans le for :
0
1
2
3
4
d'abord mis dans une variable et utilisé ensuite :
0
1
2
3
4
```

On peut utiliser un itérable directement dans une boucle for : sous forme de string, de range…

Exercice :

- Utilisez une boucle pour afficher le double de chaque nombre entre 0 et 49
- Idem pour les nombres de 0 à 100
- Même chose mais pour tous les nombres entre 0 et un nombre entré par l'utilisateur
- Affichez toutes les puissances de 2 entre 0 et 8 inclus
- Utilisez une boucle pour afficher tous les nombres impairs entre 0 et 51 inclus
- Comptez de 2358 à 2407 (point de cours : la fonction range() prend un paramètre par défaut, si on lui en donne 2 le 1er est le début/départ le 2nd la fin/l'arrivée).

V) `range()` en utilisation avancée

Utilisation avancée de `range()` : `range(n,m,p)` avec n étant la valeur de départ, m la valeur d'arrivée, p le pas. La valeur d'arrivée est systématiquement exclue. Voici 4 exemples :

```
for i in range(0,5):
    print(i)
print("**********")
for i in range(0,10,2):
    print(i)
print("**********")
for i in range(5,0,-1):
    print(i)
print("**********")
for i in range(85,78,-3):
    print(i)
```

```
0
1
2
3
4
**********
0
2
4
6
8
**********
5
4
3
2
1
**********
85
82
79
```