TP 01 – Les boucles "For" (Partie 2)

I Avant-propos : range

La fonction **range** permet de créer rapidement des listes de valeurs **entières** consécutives ou régulièrement espacées.

	Explication
`range(fin)`	Renvoie la liste des entiers entre `0` et `fin-1`
`range(deb,fin)`	Renvoie la liste des entiers entre `deb` et `fin-1`
`range(deb,fin,pas)`	Renvoie la liste des entiers entre `deb` et `fin-1` avec un `pas`

1. Définir la liste [3,4,5,6,7] avec range.

Entrée [1]:

2. Définir la liste [0,1,2,3,...,100] avec range.

Entrée [2]:

3. Définir la liste [4,6,8,10,12,14,16] avec range.

Entrée [3]:

4. Définir la liste [0,3,6,9,12] avec range.

Entrée [4]:

II Présentation des boucles for

Les **boucles for** permettent de **répéter** une liste d'instruction un certain nombre de fois.

Pour tout élément d'une séquence
    Réaliser les instructions ...
Fin

```
for element in sequence:
    bloc d'instructions
```

II.1 La boucle "for . . . in Liste"

Expliquons sur un exemple.

Entrée [5]:
```
L = ["Lundi", "Mardi", "Mercredi"]
for e in L :
    print(e)
```

Valeurs de `e` successives	Instructions réalisées

Out [5]:

Exercice 1 Donner l'affichage de chaque programme après exécution.

Entrée [6]:
```
L = [3, True, "Info"]
for e in L :
    print(e)
```

Valeurs de e successives	Instructions réalisées

Out [6]:

Entrée [7]:
```
L = [1, 2, 3]
for e in L :
    print("Bonjour")
```

Valeurs de e successives	Instructions réalisées

Out [7]:

Entrée [8]:
```
L = [6, 14, 42]
for x in L :
    x = x+1
    print(x)
```

Out [8]:

Entrée [9]:
```
L = [1, 2, 3]
for e in L:
    e = e**2+1
    print(e)
```

Out [9]:

II.2 La boucle "for . . . in Chaîne de caractères"

Entrée [10]:
```
for e in "Lundi" :
    print(e)
```

Valeurs de e successives	Instructions réalisées

Out [10]:

II.3 Boucle "for . . . in range(. . .)"

Dans la partie précédente, la variable parcourait une liste. Lorsque l'on veut utiliser une variable qui prend des valeurs entières, on peut utiliser la commande `range(deb,fin)`, qui désigne l'ensemble des entiers compris entre `deb` (inclus) et `fin` (exclu).

Par exemple,

Entrée [11]:
```
for k in range(1,4) :
    print(k)
```

Valeurs de k successives	Instructions réalisées

Out [11]:

Exercice 2 Donner l'affichage de chaque programme après exécution.

Entrée [12]:
```
for k in range(2,6) :
    k = k + 1
    k = 2*k
    print(k)
```

Out [12]:

Entrée [13]:
```
for k in range(3) :
    print("Il fait chaud !")
```

Out [13]:

Entrée [14]:
```
L = ["Lundi", "Mardi", "Mercredi"]
for k in range(len(L)) :
    print(L[k])
```

Out [14]:

II.4 Boucle "for . . . in Dictionnaire"

Par défaut, appliquée à un dictionnaire, la boucle `for` porte sur les clefs de ce dictionnaire.

Entrée [15]:
```
dico = {'a':1, 'b':2, 'c':3}
for e in dico:
    print(e)
```

Out [15]:

On peut faire porter explicitement les itérations de la boucle `for` sur les clefs d'un dictionnaire grâce à la commande `keys`.

Entrée [16]:
```
dico = {'a':1, 'b':2, 'c':3}
for e in dico.keys():
    print(e)
```

Out [16]:

Si on veut plutôt faire porter les itérations de la boucle `for` sur les valeurs d'un dictionnaire, on peut utiliser la commande `values`.

Entrée [17]:
```
dico = {'a':1, 'b':2, 'c':3}
for e in dico.values():
    print(e)
```

Out [17]:

On peut aussi faire porter les itérations de la boucle `for` sur les couples (clefs, valeurs) grâce à la commande `item`.

Entrée [18]:
```
dico = {'a':1, 'b':2, 'c':3}
for e in dico.item():
    print(e)
```

Out [18]:

III Retour sur : les listes (type list)

En mathématiques, il y a essentiellement deux façons différentes d'écrire un ensemble :

- En **extension**. Cela revient à décrire un par un les éléments qu'il contient, par exemple $E = \{1, 2, 3\}$.
- En **compréhension**. Cela revient à définir l'ensemble à partir d'une condition, par exemple $E = \{x \in \mathbb{R} \mid x \geqslant 1\}$.

C'est sensiblement pareil pour générer des listes en Python.

III.1 Définition d'une liste en extension

Cela revient à écrire chaque élément de la liste...

Entrée [19]:
```
L = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

...ou à les ajouter à l'aide d'une boucle

Entrée [20]:

Out [20]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

III.2 Définition d'une liste en compréhension

La syntaxe générale pour définir une liste en compréhension est la suivante :

```
L = [ f(x) for x in TRUC]
```

où `TRUC` est soit une liste, soit une commande `range`.

Entrée [21]:
```
L = [k**2 for k in range(0,11)]
print(L)
```

Out [21]:

Entrée [22]:
```
M = [k**2 for k in range(0,11) if k !=2 if k!=3]
print(M)
```

Out [22]:

Entrée [23]:
```
N = [i+j for i in range(0,3) for j in range(0,6)]
print(N)
```

Out [23]:

Entrée [24]:
```
P =[i+j for i in range(0,3) for j in range(0,3) if i>=j]
print(P)
```

Out [24]:

IV Lecture de programme

Exercice 3 Pour tous les programmes suivantes, déterminer l'affichage du programme **sans l'ordinateur** puis vérifier à l'aide de l'ordinateur.

Entrée [25]:
```
L=[1,3,5,7]
for i in L:
    print(i**2)
```

Out [25]:

Entrée [26]:
```
for k in "Truc":
    print(k)
```

Out [26]:

Entrée [27]:
```
for n in range(4):
    n=7*n
    print(n)
```

Out [27]:

Entrée [28]:
```
L=[1,3,7]
for e in L:
    print(e)
    print(2*e)
    print("Test")
```

Out [28]:

Entrée [29]:
```
for i in range(4):
    print("Info")
print("Maths")
```

Out [29]:

Entrée [30]:
```
compteur=0
for k in range(4):
    compteur=compteur+1
    print(compteur)
print("compteur=",compteur)
```

Out [30]:

Entrée [31]:
```
compteur=0
for k in range(1,4):
    compteur=compteur+k
    print(compteur)
print("compteur=",compteur)
```

Out [31]:

Entrée [32]:
```
compteur=1
for e in [1,3,6,-1]:
    compteur=compteur+e
print(compteur)
```

Out [32]:

Entrée [33]:
```
L=[1,3,6,-1]
compteur=0
for k in range(len(L)):
    compteur=compteur+k
print(compteur)
```

Out [33]:

Entrée [34]:
```
L=["Info","Maths","Phys"]
for k in range(len(L)):
    print(k)
    print(L[k])
```

Out [34]:

Entrée [35]:
```
L=[]
for i in range(2,7):
    L.append(3*i)
    print(len(L))
print(L)
```

Out [35]:

V Correction de programme

Exercice 4

1. On souhaite afficher les lignes suivantes :
   'Oui'
   'Non'
   'Oui'
   'Non'
   'Oui'
   'Non'
   Compléter et corriger le programme suivant (en particulier, en mettant les bonnes indentations) pour qu'il affiche le résultat souhaité. Vérifier à l'aide de la console.

Entrée [36]:
```
#Version à corriger
for k in range(....)
print('Oui')
print('Non')
```

Entrée [37]:
```

```

2. On souhaite afficher les lignes suivantes :
   'Oui'
   'Oui'
   'Oui'
   'Non'
   Compléter et corriger le programme suivant (en particulier, en mettant les bonnes indentations) pour qu'il affiche le résultat souhaité. Vérifier à l'aide de la console.

Entrée [38]:
```
#Version à corriger
for k in range(....)
print('Oui')
print('Non')
```

Entrée [39]:
```

```

3. On souhaite afficher les lignes suivantes :
   'Oui'
   'Oui'
   'Oui'
   'Non'
   'Non'
   'Non'
   Compléter et corriger le programme suivant (en particulier, en mettant les bonnes indentations) pour qu'il affiche le résultat souhaité. Vérifier à l'aide de la console.

Entrée [40]:
```
#Version à corriger
for k in range(....)
print('Oui')
for k in range(....)
print('Non')
```

Entrée [41]:
```

```

VI Exercices

Exercice 5 *Les questions de cet exercice sont indépendantes.*

1. À l'aide d'une boucle `for`, afficher tous les entiers de 90 à 100.

Entrée [42]:
```

```

Out [42]:
```

```

2. À l'aide d'une boucle `for`, afficher le carré de tous les entiers de 1 à 10.

Entrée [43]:
```

```

Out [43]:
```

```

3. Ecrire un programme qui renvoie une liste contenant le résultat de toutes les multiplications de la forme `9`$\times k$ pour $k \in \{0, \dots, 10\}$. *On demande de ne pas remplir une telle liste à la main mais grâce à une boucle* `for`. *On pourra commencer par créer une liste vide, puis par la remplir au fur à et à mesure avec toutes les quantités souhaitées à l'aide d'une boucle.*

Entrée [44]:

Out [44]:

Exercice 6 (**Mode de définition d'une liste**) Dans cette exercice, on cherche trois manières différentes de définir en Python la liste suivante

```
L2 = [3,6,9,12,...,60]
```

On pourra remarquer que tous les éléments de la liste sont de la forme $3 \times k$ *avec* k *un entier allant de* 1 *à* 20.

1. Définir maintenant la liste L2 en partant d'une liste vide en utilisant une boucle `for` et des `append` successifs.

Entrée [45]:

2. Créer la liste `L2` en une instruction grâce à la définition par compréhension.

Entrée [46]:

3. Enfin, définir L2 en partant d'une liste contenant que des zéros puis modifier successivement les éléments à l'aide d'une boucle `for`.

Entrée [47]:

Exercice 7 (**Un challenge un peu bizarre... / Introduction au calcul de somme**) Dans cet exercice, on cherche à calculer la valeur de la somme `S=1+2+3+4` en utilisant une seule variable et en ne s'autorisant à faire que l'addition de deux nombres à la fois. (On suppose aussi que l'on ne sait pas faire le calcul total, ni d'éventuels calculs intermédiaires à la main...). Ainsi, on ne peut pas taper directement la commande

Entrée [48]:
```
S = 1 + 2 + 3 + 4
```

car elle fait intervenir l'addition de quatre nombres. Pour se faire, on peut introduire une variable, qui au départ vaut 0, et à la fin doit contenir la valeur finale de la somme et lui «ajouter successivement» toutes les valeurs de la somme.

Entrée [49]:
```
S = 0
S = S + 1
S = S + 2 #Maintenant S vaut 1+2
S = S + 3 #Maintenant S vaut 1+2+3
S = S + 4 #Maintenant S vaut 1+2+3+4
print(S)
```

Out [49]:
```
10
```

La plupart des instructions sont similaires. Ré-écrire ce programme de manière «compacte» à l'aide d'une boucle `for`.

Entrée [50]:

Out [50]:

Exercice 8 (**Calcul de sommes**) À l'aide de l'Exercice 7, répondre aux questions suivantes.

1. Écrire un programme qui calcule la somme des entiers de 10 à 100 (inclus). ***Vérification :*** *On vérifiera que, le programme renvoie* `5005`.

Entrée [51]:

Out [51]:

2. Écrire un programme qui calcule la somme des carrés des entiers allant de 1 à 50 (inclus). ***Vérification*** *: On vérifiera que, le programme renvoie* `42925`.

Entrée [52]:

Out [52]:

3. Ecrire un programme qui, pour un entier donné par l'utilisateur, renvoie la valeur de la somme $1+2+\cdots+n$. ***Vérification*** *: On vérifiera que pour l'entier* `6`*, le programme renvoie* `21`*.*

Entrée [53]:

Entrée [54]:

Out [54]:

4. Écrire un programme qui renvoie le produit des entiers de 1 à 10 (inclus). ***Vérification*** *: On vérifiera que pour l'entier* `6`*, le programme renvoie* `3628800`*.*

Entrée [55]:

Out [55]:

5. Écrire un programme qui calcule la somme des éléments d'une liste d'entiers. ***Vérification*** *: Pour la liste* `L=[4,-1,5,3,-5]`*, le programme renvoie* `6`*.*

Entrée [56]:

Out [56]:

Exercice 9

1. Écrire un programme (avec une boucle `for`) qui, à partir d'une liste `L1` d'entiers, renvoie une liste `L`, constituée des mêmes éléments que `L1` multipliés par 2. ***Vérification*** *: Pour la liste* `L=[1,2,-3,-4]`*, le programme renvoie la liste* `[2,4,-6,-8]`*.*

Entrée [57]:

Out [57]:

2. Écrire un programme (avec une boucle `for`) qui, à partir d'une liste `L1` d'entiers, renvoie une liste `L`, constituée des mêmes éléments que `L1` mais doublés. ***Vérification*** *: Pour la liste* `L=[1,2,-3,-4]`*, le programme renvoie la liste* `[1,1,2,2,-3,-3,-4,-4]`.

Entrée [58]:

Out [58]:

3. Écrire un programme qui, à partir de deux listes `L1` et `L2` de même longueur, renvoie une liste `L` constituée des éléments de `L1` et `L2` par alternance (un sur deux). ***Vérification*** *: Pour les listes* `L1=[1,5,3,9]` *et* `L2=[2,0,6,10]`*, le programme renvoie la liste* `[1,2,5,0,3,6,9,10]`.

Entrée [59]:

Out [59]:

Exercice 10 On place la somme de 1000 euros sur un compte d'épargne. Chaque année, les intérêts sur l'argent placé rapportent 10% (le capital est donc multiplié par 1.10 chaque année).

1. Écrire un code à l'aide d'**une boucle for** qui permet de calculer et d'afficher le capital pour les dix premières années. Vérifier que les résultats concordent avec ceux de la question précédente. ***Vérification*** *: On vérifiera que la dixième valeur obtenue vaut environ* `2593.7`.

Entrée [60]:

2. Écrire un code à l'aide d'**une boucle for** qui permet de calculer et d'afficher le capital de la dixième année uniquement. ***Vérification*** *: On vérifiera que la valeur obtenue vaut environ* `2593.7`.

Entrée [61]:

Out [61]:

3. Écrire un programme qui renvoie une liste contenant le capital des cinq premières années. ***Vérification*** *: On vérifiera que la dernière valeur de la liste obtenue est environ* `1610.5`.

Entrée [62]:

Out [62]:

Exercice 11 *Calcul de factorielle.*

1. Ecrire un programme demandant à l'utilisateur un nombre entier n et calculant `n`!. On rappelle que $0! = 1$ et que pour tout $n \geqslant 1$, $n! = 1 \times 2 \times \cdots \times (n-1) \times n$. ***Vérification :*** *On vérifiera que, pour* `n=5`*, le programme renvoie* `120`.

Entrée [63]:

2. Écrire un programme qui demande à l'utilisateur un entier `n`, et affiche le booléen `True` si votre programme renvoie la même valeur que la fonction `factorial` déjà programmée de Python.

   - Pour utiliser la fonction factorielle de Python, voici le mode d'emploi.

Entrée [64]:
```
import math
math.factorial(<entier>)
```

   - Pour faire un test d'égalité, on peut utiliser la commande `==`, qui fonctionne comme illustré sur les exemples ci-dessous.

Entrée [65]:
```
2 == 3
```

Out [65]:
```
False
```

Entrée [66]:
```
2 == 2
```

Out [66]:
```
True
```

Entrée [67]:

Exercice 12 (**Avec un compteur**) Ecrire un programme qui, pour une lettre et un mot donnés par l'utilisateur, renvoie le nombre d'occurrences de cette lettre dans le mot choisi. ***Vérification :*** *On vérifiera que pour la lettre* `a` *et le mot* `abracadabra` *le programme renvoie* `5`.

Entrée [68]:

Entrée [69]:
```
lettre = a
mot = abracadabra
```

Out [69]:

Exercice 13 (**Rectangles et triangles**)

1. Ecrire un programme qui demande à l'utilisateur un entier n et un entier p, et qui affiche à l'écran un rectangle de n lignes et p colonnes constitué de caractères « * ». ***Vérification :*** *Pour $n = 3$ et $p = 4$, le programme doit afficher :*

```
* * * *
* * * *
* * * *
```

Entrée [70]:

2. Ecrire un programme qui demande à l'utilisateur un entier n et qui affiche à l'écran un triangle rectangle isocèle de taille n constitué de caractères « * ». ***Vérification*** *: Pour $n = 3$, le programme doit afficher :*

```
*
* *
* * *
```

Entrée [71]: