AP04 BOUCLES FOR ET WHILE

I. La boucle FOR

Les boucles FOR

La boucle `FOR` est une boucle bornée. On sait à l'avance combien de fois on va la faire.

```
# En pseudo code
instruction1
Pour i allant de 0 a 6:
    instruction2
instruction3
```

- instruction1 est effectuée
- `i` est un `itérateur` qui va aller de 0 à 6, c'est à dire 7 tours
- On effectuera donc 7 fois l'instruction2
- Quand on a fini la boucle on effectue l'instruction3

La boucle FOR en python

</> Code Python

```
for i in range(0, 7):
    print(i, end=" ")
# affichage:
# 0 1 2 3 4 5 6
```

</> Code Python

```
for i in range(2, 7):
    print(i, end=" ")
# affichage:
# 2 3 4 5 6
```

</> Code Python

```
for i in range(5):
    print(i, end=" ")
# affichage:
# 0 1 2 3 4
```

Prototypage de la boucle FOR

La boucle FOR s'écrit de manière générale de la façon suivante :

</> Code Python

```
for i in range(a, b):
```

La boucle est effectuée $b-a$ fois et la variable `i` prend pour valeur tous les entiers entre $[a;b[$

Exemple : Trace de la structure FOR

</> Code Python

```
1 for i in range(2, 6):
2     print(i)
3 print('fin')
```

i	i < 6	instruction
2	True	print(2)
3	True	print(3)
4	True	print(4)
5	True	print(5)
6	False	print('fin')

Itération sur une chaine de caractères via les indices

Une chaine de caractères `str` est un itérateur, c'est à dire qu'il est possible d'itérer sur ses éléments.

```
>>> mot = 'Hello'
>>> mot[1]
'e'
```

Indice	0	1	2	3	4
Caractère	H	e	l	l	o
Indice Négatif	-5	-4	-3	-2	-1

</> Code Python

```
1 mot = 'Hello'
2 for i in range(0, len(mot)):
3     print(mot[i])
4 print('Fin')
```

i	i < len(mot)	instruction	affichage
0	True	print(mot[0])	H
1	True	print(mot[1])	e
2	True	print(mot[2])	l
3	True	print(mot[3])	l
4	True	print(mot[4])	o
5	False	print('Fin')	Fin

Itération sur une chaine de caractères via les éléments

Il est aussi possible d'itérer directement sur les éléments qui constituent cette chaine de caractère
On peut donc créer un variable nommée `lettre` par exemple qui prendra pour valeur chacun des caractères de `mot`.

Code Python

```
mot = 'Hello'
for lettre in mot:
    print(lettre)
print('Fin')
```

lettre	instruction	affichage
'H'	print(lettre)	H
'e'	print(lettre)	e
'l'	print(lettre)	l
'l'	print(lettre)	l
'o'	print(lettre)	o
None	print('Fin')	Fin

- Il n'y a pas besoin de connaitre la longueur du mot
- Il n'y plus l'utilisation d'indice
- `lettre` prend pour valeur chaque caractère de mot
- lorsque `lettre` a pris pour valeur tous les caractères de `mot`, la boucle s'arrête.

Exercice 1

Écrire une fonction permettant d'afficher tous les entiers entre 0 et 6 inclus

Exercice 2

Écrire une fonction permettant d'afficher tous les entiers entre 2 et 9 inclus

Exercice 3

Écrire une fonction permettant d'afficher toutes les lettres du mot `'abracadabra'` en itérant sur les indices

Exercice 4

Écrire une fonction permettant d'afficher toutes les lettres du mot `'abracadabra'` en itérant sur les caractères

Algorithmes de Recherche dans un itérable

Cet algorithme permet de savoir si un élément est présent dans un itérable.

- Le mot sera la variable `mot`
- La lettre que l'on cherche sera la variable `lettre_cherchee`

On peut utiliser l'algorithme suivant :

1. on parcourt `mot` à l'aide d'une boucle `FOR`
2. on observe donc chaque caractère `mot[i]`
3. si le caractère est égal à `lettre_cherchee`, alors on renvoie la valeur `True` et on arrête la boucle.
4. sinon on ne fait rien et on revient à l'étape 2 pour observer le caractère suivant.
5. si on a observé tous les caractères de `mot` sans trouver `lettre_cherchee`, on renvoie `False`

Exercice 5 : Écrire une fonction `est_present(mot, lettre_cherchee)`

Code Python

```
def est_present(mot, lettre_cherchee):
    ''' Renvoie True si lettre_cherchee est dans mot et False sinon'''




    #
```

Exercice 6

Ecrire la fonction `est_present2(mot, lettre_cherchee)` en itérant sur les indices

Exercice 7

Ecrire la fonction `est_present3(mot, lettre_cherchee)` qui renvoie l'indice de la première occurrence de lettre_cherchee dans mot ou `-1` si elle n'est pas présente

Algorithmes de comptage dans un itérable

Cet algorithme permet de compter le nombre de fois qu'un élément est présent dans un itérable.

- Le mot sera la variable `mot`
- La lettre que l'on cherche sera la variable `lettre_cherchee`

On peut utiliser l'algorithme suivant :

1. on initialise un compteur à 0
2. on parcourt `mot` à l'aide d'une boucle `FOR`
3. on observe donc chaque caractère `mot[i]`
4. si le caractère est égal à `lettre_cherchee`, alors on incrémente le compteur.
5. sinon on ne fait rien
6. on revient à l'étape 3 pour observer le caractère suivant.
7. si on a observé tous les caractères de `mot`, on renvoie le compteur

Exercice 8 : Écrire une fonction `compte(mot, lettre_cherchee)`

Code Python

```
def compte(mot, lettre_cherchee):
    ''' Renvoie le nombre de fois que lettre_cherchee est dans mot'''




    #
```

Exercice 9 : Ecrire la fonction `compte2(mot, lettre_cherchee)` qui itère sur les éléments

Code Python

```
def compte2(mot, lettre_cherchee):
    ''' Renvoie le nombre de fois que lettre_cherchee est dans mot'''




#
```

II. La boucle WHILE

Les boucles WHILE

La boucle `WHILE` est une boucle non bornée. On ne sait à l'avance combien de fois on va la faire.

```
# En pseudo code
instruction1
tant que condition1 vraie:
    instruction2
instruction3
```

- instruction1 est effectuée
- tant que la condition1 est vraie on effectue l'instruction 2
- quand la condition1 est fausse, on sort de la boucle ;
- on effectue l'instruction3

Prototypage de la boucle WHILE en python

`condition1` est un booléen qui est évalué à chaque tour de boucle. Tant que `condition1` `True`, on effectue les instructions de la boucle. Lorsque `condition1` devient `False`, on sort de la boucle.

</> Code Python

```
while condition1:
    instruction
```

III. Exercices

Exercice 10 : Dire ce qu'affichent les programmes suivants et expliquer pourquoi

</> Code Python

```
1 n = 0
2 u = 1
3 while u < 1000:
4     n = n + 1
5     u = u * 2
6 print(n)
7 print(u)
```

</> Code Python

```
1 mot = 'abracadabra'
2 c = 0
3 trouve = False
4 while c < len(mot) and not(trouve):
5     trouve = mot[c] == 'r'
6     c = c + 1
7 print(trouve)
```

Exercice 11 : Boucle FOR

1. Écrire une fonction `somme(n)` qui renvoie la somme des entiers de 1 à n inclus. Par exemple, `somme(5)` doit renvoyer $1+2+3+4+5=15$.

</> Code Python

```
def somme(n):
    ''' Renvoie la somme des entiers de 1 à n inclus'''



#
```

2. Écrire une fonction `factorielle(n)` qui renvoie la factorielle de n. Par exemple, `factorielle(5)` doit renvoyer $1\times2\times3\times4\times5=120$.

</> Code Python

```
def factorielle(n):
    ''' Renvoie la factorielle de n'''



#
```

3. Ecrire une fonction `remplace_lettre(mot, lettre_a_remplacer, nouvelle_lettre)` qui remplace toutes les occurrences de `lettre_a_remplacer` par `nouvelle_lettre` dans `mot`. Par exemple, `remplace_lettre('abracadabra', 'a', 'o')` doit renvoyer `'obrocodobro'`.

Code Python

```
def remplace_lettre(mot, lettre_a_remplacer, nouvelle_lettre):
    ''' Renvoie mot dans lequel toutes les occurrences de
    lettre_a_remplacer ont été remplacées par nouvelle_lettre'''









#
```

4. Ecrire une fonction `supprime_lettre(mot, lettre_a_supprimer)` qui supprime toutes les occurrences de `lettre_a_supprimer` dans `mot`. Par exemple, `supprime_lettre('abracadabra', 'a')` doit renvoyer `'brcdbr'`.

Code Python

```
def supprime_lettre(mot, lettre_a_supprimer):
    ''' Renvoie mot dans lequel toutes les occurrences de
    lettre_a_supprimer ont été supprimées'''









#
```

Exercice 12 : Boucle While

1. On dispose d'une feuille de papier d'épaisseur 0,1 mm. Combien de fois doit-on la plier au minimum pour que l'épaisseur dépasse la hauteur de la tour Eiffel 324 m. Écrire une fonction `nb_plis()` en Python pour résoudre ce problème.

2. Inès veut construire une pyramide à base carrée comme sur la photo. La pyramide sur la photo a 7 étages.
Inès a n billes. Combien d'étages au maximum aura sa pyramide? Écrire une fonction `nb_etages(n)` en Python pour répondre au problème.

Code Python

```
1 assert nb_etages(1000) == 14
```

3. L'ordinateur tire un nombre entier au hasard entre 0 et 100. L'utilisateur doit le trouver et pour cela propose des valeurs. L'ordinateur indique pour chaque valeur proposée si la valeur est trop petite, trop grande ou s'il a trouvé.
Écrire une fonction `jeu()` en Python pour jouer à ce jeu. En combien de coups est-on sûr de trouver?

Méthode 1 : La bibliothèque `random`

Code Python

```
import random
# pour tirer un nombre entier au hasard entre 0 et 100
nombre_aleatoire = random.randint(0, 100)
```

4. En mathématiques, la suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence généralement par les termes 0 et 1 et ses premiers termes sont 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.

   - Ecrire une fonction `fibonacci(n)` qui renvoie le terme de rang n de la suite de Fibonnaci. (le premier terme est de rang 0)

   Code Python
   ```
   assert fibonacci(0) == 0
   assert fibonacci(1) == 1
   assert fibonacci(2) == 1
   assert fibonacci(9) == 34
   ```

   - Ecrire une fonction `rang_fibonacci()` qui renvoie le rang du premier terme supérieur ou égal à 10000.

   Code Python
   ```
   assert rang_fibonacci() == 21
   ```

5. L'utilisateur donne un entier positif et le programme annonce combien de fois de suite cet entier est divisible par 2. Ecrire la fonction `divise_par_2(n)` qui renvoie ce nombre de fois. Par exemple, `divise_par_2(48)` doit renvoyer 4 car $48 = 2^4 \times 3$ et 4 est le plus grand entier tel que 48 est divisible par 2^4.

   Code Python
   ```
   assert divise_par_2(48) == 4
   assert divise_par_2(7) == 0
   ```

6. Écrire une fonction `tirage()` qui fait des tirages successifs d'entiers 0 ou 1 jusqu'à tirer 3 fois 1 consécutivement et renvoie le nombre de tirages qui ont été nécessaires.

Méthode 2 : La bibliothèque `random`

Code Python
```
import random
# pour tirer un nombre entier au hasard entre 0 et 1
nombre_aleatoire = random.randint(0, 1)
```

7. Ecrire une fonction python qui renvoie l'entier p tel que $2^p \leqslant n \leqslant 2^{p+1}$.