Travaux dirigés n° 2

Boucles

*** Corrigé indicatif ***

RETOUR SUR EXPÉRIENCE:
Afin d'améliorer ce support, annotez-le directement tant au niveau du sujet que de la correction fournie.

Date intervention :

Nom de l'intervenant :

Groupe de TD/TP :

Remarques générales :

Noubliez pas de me retourner votre support annoté. Merci d'avance, R. Girard

Utilisation de la boucle `while`

La boucle `while` sert à itérer (répéter) un bloc d'instructions tant qu'une condition est vérifiée. La condition est une expression booléenne, c'est-à-dire une expression qui est vraie (`True`) ou fausse (`False`).

***Exercice 2.1**

Copiez et exécutez le programme ci-dessous. Indiquez le traitement fait par ce programme. Qu'est ce qui se passe ?

```
i=1
while i<=10:
    print("3 *",i,"=",3*i)
    i=i+1
```

Ce programme affiche la table de multiplication de 3 pour les 10 premiers termes

***Exercice 2.2**

Écrivez un programme qui demande deux entiers `a` et `b` et qui affiche les nombres `i` compris entre `a` et `b` (inclus) tel que `i` n'est pas multiple de 3 et $i^2 - 4$ est multiple de 7.

```
a = int(input("Donner la valeur de a : "))
b = int(input("Donner la valeur de b : "))
# On utilise une variable i qui va prendre les valeurs des entiers compris entre a et b
# Au départ on initialise i avec la valeur de a
i = a
# Tant que i n'a pas dépassé la valeur de b
while i <= b:
    # On affiche la valeur de i uniquement si celle-ci vérifie la condition demandée
    if (i%3 != 0) and( (i*i-4) % 7 == 0):
        print(i)
    # On passe a l'entier suivant avant de boucler
    i = i+1
```

*Exercice 2.3

Écrivez un programme qui demande la saisie d'un nombre entier jusqu'à ce que le nombre saisi soit compris entre 1 et 4 (inclus).

```
# Ne pas oublier de convertir la valeur retournée par input en entier
nombre = int(input("Donner un entier [1..4] : "))
# Tant que nombre n'est pas dans l'intervalle
while not (nombre >=1 and  nombre <= 4):
    nombre = int(input("Donner un entier [1..4] : "))
```

*Exercice 2.4

Écrivez un programme qui demande en boucle à l'utilisateur d'entrer des entiers et affiche chaque entier avec son numéro d'ordre de saisie. Le programme doit s'arrêter en affichant `Merci` lorsque l'utilisateur a rentré 15 entiers ou bien s'il a saisi la valeur 0.
Exemple d'exécution :

```
Donnez un entier : 56
Entiers numéro 1 : 56
Donnez un entier : 23
Entiers numéro 2 : 23
Donnez un entier : 8
Entiers numéro 3 : 8
Donnez un entier : 0
Merci
```

Le programme doit s'arrêter lorsque l'utilisateur a rentré 15 entiers ou bien s'il a saisi la valeur 0. Traduit par une boucle while, cela veut dire que les itérations doivent être réalisées tant que la valeur saisie n'est pas 0 et que le nombre d'entiers rentré n'est pas 15.
Le test de la boucle portant sur les variables a et nbentiers il est nécessaire de les initialiser avant de commencer la boucle : on effectue une première saisie de a et on initialise nbentiers à 1 (ce qui correspond à la première saisie).
A la fin de chaque itération il ne faut pas oublier de re-saisir la valeur de a et d'incrémenter nbentiers.

```
nbentiers = 0
a = 1
while (nbentiers <15 and a!=0) :
     a = int(input("Donnez un entier : "))
     if a!=0:
        nbentiers=nbentiers+1
        print("Entiers numéro",nbentiers,":",a)
print("Merci")
```

Exercice 2.5

Écrivez un programme qui demande deux entiers `a` et `b` et qui calcule et affiche la somme des carrés des entiers impairs compris entre `a` et `b` (`a` et `b` inclus).

Par exemple, l'exécution du programme en donnant les valeurs 12 et 256 à `a` et `b` doit produire les affichages suivants :

```
Donner la valeur de a : 12
Donner la valeur de b : 256
La somme des carrés des impairs compris entre 12  et  256 vaut :  2795874
```

```python
a=int(input("Donner la valeur de a :"))
b=int(input("Donner la valeur de b : "))
# La variable somme va contenir la somme des carrés des entiers impairs
# compris entre a et b
# On initialise cette variable à 0
somme = 0
# On utilise une variable i qui va prendre les valeurs des entiers compris entre a et b
# Au départ on initialise i avec la valeur de a
i = a
# Tant que i n'a pas dépassé la valeur de b
while i <= b:
    # Si i est impair on ajoute son carré dans la somme
    if i%2 == 1:
        somme = somme + i*i
    # Et on passe a l'entier suivant avant de boucler
    i = i+1
# En sortant de la boucle while (lorsque i est devenu strictement supérieur a b)
# la variable somme contient le résultat
print("La somme des carrés des impairs compris entre", a, " et ", b, "vaut : ", somme)
```

***Exercice 2.6**

Écrivez un programme qui demande à un joueur de trouver un nombre qui a été choisit au hasard entre 1 et 100 par la machine.
Le programme doit ensuite demander en boucle au joueur de proposer un nombre et indiquer après chaque proposition si le nombre proposé est trop petit ou trop grand.
Le programme s'arrête lorsque le joueur a trouvé le nombre et indique alors combien de coups ont été nécessaires.
Pour choisir un nombre au hasard entre 1 et 100 vous utiliserez l'appel à la fonction `randint(1,100)` qui retourne un entier compris entre 1 et 100. Cette fonction nécessite d'importer le module `random` donc d'ajouter l'instruction suivante au début de votre programme : `from random import *`

Exemple d'exécution :

```
Proposer un nombre entre 1 et 100 : 50
Trop petit
Proposer un nombre entre 1 et 100 : 75
Trop grand
Proposer un nombre entre 1 et 100 : 62
Gagné en 3 coups
```

```
from random import *
# On choisit au hasard le nombre à trouver
nombre = randint(1,100)

# La variable proposition va contenir l'entier proposé par l'utilisateur
# On initialise cette variable en demandant une première saisie
proposition = int(input("Proposer un nombre entre 1 et 100 : "))

# La variable nbcoup contient le nombre d'essais effectués par l'utilisateur
# On l'initialise a 1 car on a demandé a l'utilisateur une première proposition
nbcoup = 1

# Tant que la proposition est différente du nombre à trouver
while proposition != nombre :
    # On affiche un message selon que la proposition est plus petite ou plus grande
    # que le nombre à trouver
    if proposition<nombre :
        print("Trop petit")
    else:
        print("Trop grand")
    # Puis avant de boucler on saisie une nouvelle proposition
    proposition = int(input("Proposer un nombre entre 1 et 100 : "))
    # ce qui correspond à un essai de plus
    nbcoup = nbcoup +1
# Lorsqu'on sort de la boucle c'est que la proposition est égale au nombre à trouver
print("Gagné en ", nbcoup, "coups")
```

Exercice 2.7

Écrivez un programme qui demande deux entiers a et n et qui affiche les n premiers termes de la suite définie par :

$u_0 = a$
$u_i = \frac{1}{2} \times u_{i-1}$, si u_{i-1} est pair
$u_i = 3 \times u_{i-1} + 1$, si u_{i-1} est impair

Dans la boucle `while` qu'on utilise pour calculer chaque terme, pour une valeur de i, si on veut directement traduire la formule $u_i = \frac{1}{2} \times u_{i-1}$, on peut utiliser deux variables : u_i qui contient la valeur du terme u_i et u_imoins1 qui contient la valeur du terme précédent.
Dans ce cas la boucle commence pour i=1 en ayant initialisée la variable u_imoins1 avec la valeur de a, c'est à dire u_0.
Il ne faut pas oublier, à la fin de chaque itération de remplacer la valeur de u_imoins1 par celle de u_i, après avoir affiché la valeur u_i et de passer à la valeur i+1 suivante

```
u_imoins1 = int(input("Donner la valeur de a : "))
n=int(input("Donner n :"))
i = 1
while i < n :
    if u_imoins1 % 2 == 0:
        u_i = u_imoins1 // 2
    else:
        u_i = 3 * u_imoins1 + 1
    print(u_i)
    u_imoins1 = u_i
    i = i + 1
```

Utilisation de la boucle `for`

Lorsque l'on doit traiter des données qui sont organisées en séquence comme les chaînes de caractères, le parcours de chaque élément de la séquence s'effectue par l'instruction d'itération à bornes définies `for`. L'exemple ci-dessous affiche les lettres une par une de la chaîne de caractères *bonjour*.

```
for lettre in "bonjour":
        print(lettre)
```

***Exercice 2.8**

Écrivez un programme qui demande la saisie d'une phrase (une chaîne de caractères) et qui affiche combien de voyelles contient la phrase saisie. On comptera le nombre de voyelles minuscules et majuscules non accentuées.
Indices : utilisez l'opérateur **in** *qui permet de tester si un caractère appartient à une chaîne pour tester si une lettre est une voyelle.*

Exemple d'exécution :

```
Donner une phrase : Combien cette PHRASE contient-t-elle de VOYELLES ?
La phrase :  Combien cette PHRASE contient-t-elle de VOYELLES ?
contient :  17  voyelles.
```

```
# Programme qui compte le nombre de voyelles (minuscule ou majuscule non accentuée)
# d'une chaîne de caractères saisie.
phrase = input("Donner une phrase : ? ")
# On compte le nombre de voyelles dans la variable nbvoy
nbvoy = 0
# On parcourt la séquence de caractères phrase en comptant
# et on incrémente nbvoy pour chaque voyelle rencontrée.
# Un caractère est une voyelle s'il apparaît dans la chaîne "aeiouyAEIOUY"
for c in phrase :
    if c in "aeiouyAEIOUY":
        nbvoy = nbvoy + 1
print("La phrase : ", phrase)
print("contient : ", nbvoy, " voyelles.")
```

Exercice 2.9

Écrivez un programme qui demande la saisie d'une phrase (une chaîne de caractères) et qui affiche cette phrase en remplaçant toutes les lettres qui ne sont pas des voyelles par le caractère `*`. Les voyelles doivent donc être affichées telles quelles ainsi que tous les autres caractères (ponctuation, etc ...)

Indices : pensez à utiliser les opérateurs de comparaisons sur les caractères. Une lettre est un caractère compris entre "A" et "Z" ou entre "a" et "z".
Pour que la fonction **`print`** *ne passe pas à la ligne il faut utiliser le paramètre* `end=""`

Exemple d'exécution :

```
Donner une phrase : Voici des MAJUSCULES, des chiffres (par exemple 1 23 4 8.7) et d'autres caracteres!
*oi*i *e* *A*U**U*E*, *e* **i***e* (*a* e*e***e 1 23 4 8.7) e* *'au**e* *a*a**e*e*!
```

```
1  phrase = input("Donner une phrase: ")
2
3  # On parcourt la séquence de caractères phrase :
4  # Pour chaque caractère c
5  # si c'est une voyelle on l'affiche
6  # sinon si c'est une lettre on affiche * a la place
7  # sinon on affiche le caractère c
8
9  for c in phrase :
10     # On teste si c est une voyelle
11     if c in "aeiouyAEIOUY":
12         print(c, end="")
13     else:
14         # Ca n'est pas une voyelle mais est-ce une lettre ?
15         if (c > "A" and c <= "Z") or (c >= "a" and c <= "z"):
16             print("*", end="")
17         else :
18             # Ca n'est ni une voyelle ni une lettre
19             print(c, end="")
```

***Exercice 2.10**

Écrivez un programme qui demande la saisie d'une chaîne de caractères et qui :

- vérifie que cette la chaîne entrée code un brin d'ADN, c'est à dire qu'elle n'est composée que des caractères A, C, G et T
- et si c'est le cas, construit et affiche la chaîne qui code le brin complémentaire, c'est à dire la chaîne où chaque A est remplacé par un T (et vice-versa) et chaque C par un G (et vice-versa).

Exemples d'exécution :

```
Donner une séquence de lettres qui représente un brin d'ADN : AGCTTAAXCAAGCT
Il y a au moins une erreur dans le brin donné\smallbreak

Donner une séquence de lettres qui représente un brin d'ADN : CAGTTAACCAGAT
Le brin complémentaire de :
CAGTTAACCAGAT
est :
GTCAATTGGTCTA
```

```
# programme brin ADN
print("Un brin d'ADN est une séquence de lettre composée uniquement avec des caractères
    A, C, G et T")
brin = input("Donner une séquence de lettres qui représente un brin d'ADN : ")

# Vérification que la chaîne ne contient que des ACGT
# A priori on suppose que c'est vrai. On teste les caractères
# et si l'un deux n'est pas correct c'est faux
ok=True
for base in brin:
    if not(base in "AGCT") :
        ok = False

if not ok :
    print("Il y a au moins une erreur dans le brin donné")
else :
    # On construit le brin complémentaire en parcourant
    # les caractères du brin donné en entrée.
    # Pour chaque caractère dans brin on ajoute le caractère
    # complémentaire dans brin2 qui au départ est une chaîne vide
    brin2 = ""
    for base in brin :
        if base == "A" :
            duo = "T"
        if base == "T" :
            duo = "A"
        if base == "C" :
            duo = "G"
        if base == "G" :
            duo = "C"
        brin2 = brin2 + duo
    print("Le brin complementaire de :")
    print(brin)
    print("est : ")
    print(brin2)
```