Langage de programmation (2/3)

Chapitre3

I- Booléens

Définition 1 :

Un **booléen** est une variable qui peut avoir deux états possibles, généralement vrai ou faux, en langage Python **True** ou **False**. Les booléens sont très utiles pour des tests.

Propriété 1 :

Les **tests** en Python :

Voilà quelques opérateurs :

- Pour tester **l'égalité** deux éléments : a==b renvoie True si a et b sont égaux, False sinon.
- Pour tester la **différence** deux éléments : a!=b renvoie True si a est différent de b et False sinon.
- Pour tester la **comparaison** deux éléments : a < b renvoie True si a < b et False sinon.

Remarque :

Pour tester la comparaison et l'égalité ensemble de deux éléments : a <= b renvoie True si a≤b, False sinon.

Exercice 1 :

Exécuter le code suivant :

```
a = 3
print("Est-il vrai que a=3 ?")
print(a == 3)
type( a == 3)
```

1. Quel rôle différent ont les lignes 1 et 3 du code précédent ?
2. Comment s'écrit en langage Python le type booléen ?

Remarques :

Attention à ne pas confondre :

- a=3 qui stocke la valeur 3 dans la variable a.
- a==3 qui teste si a est égal à l'entier 3 ou pas.

Définition 2 :

Deux opérateurs logiques sont aussi très utiles : et et ou, soit en langage Python **and** et **or** ; ces opérateurs lient deux affirmations booléennes.

Propriété 2 :

- Une affirmation condition1 and condition2 est vraie uniquement lorsque les deux conditions sont vraies.
- Une affirmation condition1 or condition2 est fausse uniquement lorsque les deux conditions sont fausses.

Exercice 2 :

1. Proposer une fonction est_chiffre qui prend en paramètre un nombre entier et qui renvoie un booléen : True si ce nombre entier est compris entre 0 et 9, False dans les autres cas.
2. Rajouter une précondition pour tester que la saisie est bien un nombre entier.

Exercice 3 :

Deviner avant de lancer les codes suivants quel sera le booléen résultat. Ensuite, vérifiez en exécutant chaque code.

1.
```
b = 56
print(b != 5 and b !=6)
```

2.
```
b = 56
print(b <= 5 or b >= 6)
```

Propriété 3 :

- Dans un and, si la première condition est FAUSSE, la deuxième n'est même pas évaluée : le tout est FAUX.
- Dans un or, si la première condition est VRAIE, la deuxième n'est même pas évaluée : le tout est VRAI.

Exercice 4 :

1. Exécuter la fonction suivante avec n=4 puis n=5 :

```
def de_l_or(n):
    return (n < 5) or (erreur > zut)
```

Expliquer ce qu'il se passe.

2. Que se passe-t-il si l'ordre des deux conditions n < 5 et (erreur > zut) est inversée ?

Propriété 4 :

- Il est possible de tester l'appartenance d'un élément dans un 'ensemble' grâce au mot-clé in.
- Il est possible de tester la non appartenance d'un élément dans un 'ensemble' grâce à not in.

Exercice 5 :

1. Deviner avant d'exécuter le code ce que va afficher le script suivant :
```
mot = "AbcDE"
print('a' in mot and 'f' not in mot)
```
2. Vérifier en exécutant le code.
3. Deviner avant d'exécuter le code ce que va afficher le script suivant :
```
mot = "AbcDE"
print('a' in mot or 'C' not in mot)
```

4. Vérifier en exécutant le code.

Bilan 1 :

- Un booléen est un type de variable n'ayant que deux valeurs possibles True et False,
- On teste une égalité avec ==,
- On teste une différence avec !=,
- On teste une comparaison avec <=, >=, < ou <,
- On teste l'appartenance avec in,
- On teste la non-appartenance avec not in,
- On peut lier deux affirmations booléennes avec les opérateurs and et or.

II- if ....then....else

1) Structure alternative

En pseudo-code (avec de nombreuses variantes pour identifier les blocs d'instructions), la structure conditionnelle s'écrit :

```
1 if condition then
2       bloc 1
3 else
4       bloc 2
```

En langage Python, la structure conditionnelle s'écrit :

```
if condition :
    instruction(s) à effectuer dans la cas où la condition est remplie
else :
    instruction(s) à effectuer dans la cas contraire
```

Remarques :

- Le bloc else n'est pas obligatoire
- Vous remarquerez le symbole : très important en Python qui marque le début d'un bloc.
- C'est l'**indentation** (=décalage) qui délimite le bloc d'instructions.
- Aucune condition n'est à préciser après un else car tous les cas contraires à la condition du if sont ici pris en compte directement.

Exemple 1 :

```
a=float(input("Entrer un nombre réel : "))
if a>=0 :
    print("Vous avez entré un nombre positif ou nul :",a)
else :
    print("Vous avez entré un nombre négatif :",a)
```

Exercice 6 :

Qu'affiche le programme de l'exemple dans chacun des cas suivants :

1. Avec a=8 ?
2. Avec a=-6 ?
3. Avec a=0 ?
4. Avec a="positif" ?

Propriété 5 :

Il est possible d'imbriquer plusieurs instructions conditionnelles ensemble.

Exemple 2 :

On peut modifier l'exemple précédent ainsi en trois cas distincts par imbrication de deux instructions conditionnelles :

```
a=float(input("Entrer un nombre réel : "))
if a>0:
    print("Vous avez entré un nombre strictement positif",a)
else: # ici, on est dans le cas où a<= 0
    if a==0:
        print("Vous avez entré un nombre nul",a)
    else: # ici, on est dans le cas où a<=0 et a!=0 donc où a<0
        print("Vous avez entré un nombre négatif",a)
```

2) Structure avec elif

Plutôt que d'imbriquer plusieurs instructions conditionnelles lorsqu'il y a plus de deux cas, il est possible d'utiliser la structure elif , qui est la contraction de else if.

```
if condition1 :
    instruction(s)
elif condition2 :
    instruction(s)
elif condition3 :
    instruction(s)
else :
    instruction(s)
```

Exemple 3 :

Il est possible de réécrire l'exemple précédent en remplaçant l'imbrication en utilisant elif comme montré ci-dessous :

```
a=float(input("Entrer un nombre : "))
if a>0 :
    print("Vous avez entré un nombre strictement positif.")
elif a==0 :
    print("Vous avez entré un nombre nul.")
else :
    print("Vous avez entré un nombre strictement négatif.")
```

Remarques :

- Il est nécessaire de préciser une condition après elif (comme après un if),
- Ne pas oublier de finir par un else pour prendre en compte tous les cas non pris en compte par le if et les éventuels elif.

Exercice 7 :

Un club sportif formant des jeunes de moins de 14 ans vous a demandé de créer un programme pour informer les familles de la catégorie à laquelle appartiendra leur enfant en fonction de leur âge.

Le club sportif vous a donné ces informations :

- "Poussin" de 6 à 7 ans,
- "Pupille" de 8 à 9 ans,
- "Minime" de 10 à 11 ans,
- "Cadet" après 12 ans.

1. Écrire une fonction categorie qui :
    - Demande l'âge d'un enfant à l'utilisateur,
    - Informe l'utilisateur de la catégorie de l'enfant,
2. Rajouter des préconditions afin que d'assurer le bon fonctionnement de cette fonction.

Exercice 8 :

Voici réécrit à l'aide d'une fonction l'exemple précédent. Au lieu d'afficher un long texte, elle renvoie une chaîne de caractères explicitant le signe.

```
def signe_saisie() -> str:
    a=float(input("Entrer un nombre : "))
    if a>0 :
        return "strictement positif"
    elif a==0 :
        return "nul"
    return "strictement négatif"
```

1. Sans exécuter le script ci-dessus repérer quel est le principal changement dans la structure conditionnelle avec l'exemple initial. Puis deviner s'il convient dans chacun des cas.
2. Exécuter le script et tester-le avec au moins un des différents cas possibles.
3. Pourquoi ce script fonctionne-t-il malgré la suppression du else ?

Exercice 9 :

1. Réaliser une fonction main.
   Cette fonction, sans argument, affiche dans un menu trois actions différentes, nommées Traitement A, Traitement B, Traitement C et une dernière qui permet de sortir du programme. Ensuite, le choix de l'utilisateur est enfin affiché à l'écran.

Exemple de trace d'exécution ; cas où l'utilisateur tape 2 :

```
Menu
Traitement A, tapez 1
Traitement B, tapez 2
Traitement C, tapez 3
Sortir, tapez 4
Votre choix : 2
Votre choix : Traitement B
```

Exercice 10 :

1. Proposer une fonction, nommée mini2 qui prend comme arguments deux nombres entiers a et b et qui renvoie le minimum de ces deux nombres.
2. Proposer une fonction, nommée mini4, qui :
    - Prend comme arguments quatre nombres entiers a, b, c et d,
    - Fait appel plusieurs fois à la fonction mini2 pour finalement trouver le minimum des quatre nombres,
    - Renvoie le minimum trouvé.

Exercice 11 :

Une année n est bissextile, c'est-à-dire qu'elle dure 366 jours, que dans l'un des deux cas suivants :

- si l'année est divisible par 4 et non divisible par 100 ;
- si l'année est divisible par 400.

1. Proposer une fonction est_bissextile qui prend en argument une année de type nombre entier et qui renvoie un booléen précisant si l'année est bissextile ou non.

Indication : le code a % b renvoie le reste de la division euclidienne de a par b.
Exemple : 15 % 6 renvoie 3 car 15=6×2+3.

2. Utiliser judicieusement and et or afin d'obtenir un script utilisant seulement un if.

Bilan 2 :

La structure d'une instruction conditionnelle pour une alternative est :

```
if condition 1:
    Traitement des instructions du bloc 1
else:
    Traitement des instructions du bloc 2
```

On peut généraliser à plus de deux cas avec le mot-clé elif :

```
if condition 1:
    Traitement des instructions du bloc 1
elif condition 2:
    Traitement des instructions du bloc 2
else:
    Traitement des instructions du bloc 3
```

3) Ecriture condensée

Propriété 6 :

Il est possible d'écrire sous forme condensée sur une ligne certaines instructions conditionnelles.

Pour cela, au lieu d'avoir :

```
if condition 1:
    Traitement des instructions du bloc 1
else:
    Traitement des instructions du bloc 2
```

on réécrit sous la forme suivante l'instruction :

```
Traitement des instructions du bloc 1 if condition 1 else Traitement des instructions du bloc 2
```

Exemple 4 :

Voici une fonction qui renvoie le maximum de deux entiers a et b grâce à une instruction conditionnelle :

```
def maxi(a:int, b:int) -> int:
    if a > b :
        return a
    else:
        return b
```

On peut réécrire en une ligne l'instruction conditionnelle ; ainsi cette fonction devient :

```
def maxi(a:int, b:int) -> int:
    return a if (a > b) else b
```

Exercice 12 :

Réécrire la fonction mini2 de l'exercice déjà traité en écrivant sur une seule ligne l'instruction conditionnelle.

III- Exercices de renforcements

1) Sur les booléens

Exercice 13 :

1. Proposer une fonction eau_liquide qui prend en paramètre un nombre entier tempe (qui modélise la température de l'eau) et qui renvoie un booléen : True si ce nombre entier est compris entre 0 et 100, False dans les autres cas.
2. Rajouter une précondition pour tester que la saisie est bien un nombre entier.

Exercice 14 :

Deviner avant de lancer les codes suivants quel sera le booléen résultat. Ensuite, vérifier en exécutant chaque code.

1.
```
a = 6
print(a >= 5 and a < 6)
```

2.
```
b = -1
print(b <= 0 or b >= 5)
```

Exercice 15 :

1. Deviner avant d'exécuter le code ce que va afficher le script suivant ?

```
mot = "Nous sommes heureux et heureuses de faire NSI !"
print('a' not in mot and 'i' in mot)
```

2. Vérifier en exécutant le code.

3. Deviner avant d'exécuter le code ce que va afficher le script suivant ?

```
mot = "Quelle chance que de pouvoir aller au lycée !"
print('N' not in mot or 'é' in mot)
```

4. Vérifier en exécutant le code.

2) Sur if else voire elif

Exercice 16 :

Proposer un algorithme qui :

- Demande à l'utilisateur de saisir un nombre,
- Affiche si ce nombre est plus grand ou égal à 10 ou s'il est strictement plus petit.

Exercice 17 :

Un théâtre pratique trois types de tarifs suivant le nombre de pièces regardées.

- Pour au plus de 2 pièces vues, la personne doit payer 15€ la séance,
- Entre 3 et 5 pièces vues, la personne doit payer 12€ la séance,
- À partir de 6 séances, la personne doit payer 10€ la séance.

Écrire un programme qui :

- Demande le nombre de pièces que la personne veut voir durant la saison théâtrale,
- Affiche le prix à payer.

Exercice 18 :

1. Proposer une fonction etat_eau qui prend en paramètre un nombre entier tempe (qui modélise la température de l'eau en °C) et qui renvoie une chaîne de caractère : "solide" si ce nombre entier est négatif, "liquide" si la température est comprise entre 0 et 100 °C et "gazeux" si la température dépasse 100 °C.
2. Est-il possible d'écrire cette fonction etat_eau sans utiliser un else ni trois if ?
3. Rajouter deux préconditions pour tester que la saisie est bien un nombre entier et que la température est possible, c'est-à-dire supérieure ou égale à -273°C.

Exercice 19 :

Vous êtes en charge de la programation d'un distributeur automatique. Vous avez d'abord à gérer l'affichage du choix du consoomateur.

1. Réaliser une fonction choix.
   Cette fonction, sans argument, affiche dans un menu les quatre sélections possibles différentes, nommées Boissons, Gâteaux, Fruits, Légumes et une dernière qui permet de sortir du programme.
   Ensuite, la sélection effectuée par l'utilisateur est enfin affiché à l'écran.

Exemple de trace d'exécution ; cas où l'utilisateur tape 3 :

```
Menu
    Boissons, tapez 1
    Gâteaux, tapez 2
    Fruits, tapez 3
    Légumes, tapez 4
    Quitter, tapez 5 ou autre chose
    Votre choix : 3
    Votre choix : Fruits
```

2. Tester la fonction choix en n'oubliant aucun cas possible.

Exercice 20 :

On suppose que vous disposez déjà des deux fonctions mini2 et mini4 programmées dans un exercice précédent.

Proposer une fonction, nommée mini4, qui :

- prend comme arguments huit nombres entiers, nommés de a à h,
- fait appel aux fonctions mini2 et mini4 pour finalement trouver le minimum des huit nombres,
- renvoie le minimum trouvé.

Exercice 21 :

Écrire un programme qui :

- Demande l'âge d'une personne,
- Affiche si la personne est majeure où mineure.

Exercice 22 :

Un cinéma pratique trois types de tarifs pour deux personnes.

- si les deux personnes sont mineures, elles payent 7€ chacune,
- si l'une seulement est mineure, elles payent un tarif de groupe de 15€,
- si les deux personnes sont majeures, elles payent 18 € en tout.

Écrire un programme qui :

- Demande l'âge de chacune des personnes,
- Affiche le prix à payer.