TP 3 : Structures conditionnelles : test **if** et boucle **while**

PCSI, lycée Bertran de Born

1 L'instruction if

1.1 Le test if

Le test **if** permet d'exécuter des instructions seulement lorsqu'une condition est réalisée. La structure est la suivante :

```
if condition:
    bloc d'instructions
```

Pour vous faire la main, implémentez et exécutez le programme suivant dans le fichier temp.py en entrant en tant qu'utilisateur différentes valeurs de x (positives et négatives) :

```
x=float(input("entrez un réel x"))
if x>=0:
    print(x,"est positif")
```

1.2 Instruction if...else

L'instruction **if...else** permet d'exécuter une instruction lorsqu'une condition est réalisée et d'exécuter une autre instruction lorsque cette condition n'est pas réalisée. La structure est la suivante :

```
if condition:
    bloc d'instruction 1
else:
    bloc d'instruction 2
```

Pour vous faire la main, complétez le programme précédent avec le code suivant puis entrez en tant qu'utilisateur différentes valeurs de x (positives et négatives) :

```
x=float(input("entrez un réel x"))
if x>=0:
    print(x,"est positif")
else:
    print(x,"est strictement négatif")
    print("sa valeur absolue vaut",-x)
```

1.3 L'instruction if...elif...else

On peut avoir plusieurs conditions au sein d'une instruction **if**. On utilise alors le mot clé **elif**. C'est le cas dans le programme suivant :

```
x=int(input("Entrez un entier naturel entre 20 et 60 svp"))
if x>60:
    print("c'est trop grand")
elif x<20:
    print("c'est trop petit")
else:
    print("Merci!")
```

Notons que l'on peut avoir plusieurs conditions possibles à tester sans que ne figure le mot clé `else`. C'est le cas dans le programme suivant :

```
x=int(input("Entrez un entier naturel entre 20 et 60 svp"))
if x>60:
    print("c'est trop grand")
elif x<20:
    print("c'est trop petit")
```

1.4 Tests if imbriqués

Les tests **if** peuvent également être imbriqués les uns dans les autres. Pour comprendre le fonctionnement des tests **if** imbriqués, exécutez le programme suivant :

```
x=int(input("entrez un entier naturel svp"))
if x>=10:
    print("merci de chosir un nombre plus petit la prochaine fois")
else:
    if x==7:
        print("7 est mon chiffre préféré")
    else:
        print(str(x)+" n''est pas mon chiffre préféré")
```

Exercices avec les boucles if

Exercice 1.

Ecrire un programme qui
- demande à l'utilisateur d'entrer au clavier son âge
- indique dans la console si l'utilisateur est majeur ou mineur

Exercice 2.

Ecrire un programme commenté qui
- demande à l'utilisateur d'entrer au clavier un réel non-nul `a` et deux réels `b` et `c`
- indique dans la console si le trinôme $ax^2 + bx + c$ admet des racines réelles et, le cas échéant, affiche ces racines

Tester le programme avec plusieurs valeurs de a, b, c.

2 La boucle while

2.1 la boucle while

Le mot **while** signifie “tant que” en anglais.

Les boucles **while** permettent de réaliser la répétition d'un bloc d'instruction tant qu'une condition est réalisée. La syntaxe est la suivante :

```
while condition:
    instructions
```

Pour vous faire la main, exécutez **d'abord à la main, puis avec Python** les programmes suivants :

```
i=1
while i<7:
    i=i+2
    print(i)
```

```
i=0
u=0
while i<=3:
  print(u)
  u=2*u+2
  i=i+1
```

Sans utiliser le logiciel Python, exécuter à la main le programme suivant en précisant, dans l'ordre, toutes les instructions et tous les affichages qui ont lieu dans la console.

2.2 Utilisation des boucles while pour calculer les termes d'une suite jusqu'à un rang donné

Suite définie de manière explicite

On considère la suite (u_n) définie par $\forall n \in \mathbb{N}$, $u_n = n^2 + n + 3$.

Les deux programmes suivants permettent de calculer et d'afficher les 5 premiers termes de la suite. Le premier utilise une boucle **while** alors que le deuxième utilise une boucle **for**

```
i=0
while i<=4:
    u=i**2+i+3
    print(u)
    i = i+1
```

```
for i in range(0,5):
    u=i**2+i+3
    print(u)
```

Suite définie par récurrence

On considère la suite (u_n) définie par $u_0 = 2$ et $\forall n \in \mathbb{N}$, $u_{n+1} = 0.5 \times u_n + 5$.

Les deux programmes suivants permettent de calculer et d'afficher les 20 premiers termes de la suite. A nouveau on voit que l'on peut utiliser aussi bien une boucle **for** qu'une boucle **while** pour aboutir au résultat :

```
i=0
u=2
while i<=19:
    print(u)
    u=0.5*u+5
    i = i+1
```

```
u=2
for i in range(0,20):
    print(u)
    u=0.5*u+5
```

Question 1.

Dans le programme utilisant la boucle **while**, la condition porte-t-elle sur le rang ou sur les termes de la suite ?

Question 2.

Pour calculer (ou afficher) les termes d'une suite jusqu'à un rang déterminé, quelle type de boucle vous semble le plus adapté ?

2.3 Utilisation des boucles while pour calculer les termes d'une suite jusqu'à ce qu'une condition sur les termes de la suite soit réalisée

Etude d'un exemple

On place 10000 euros sur un compte rapportant 4% d'intérêts par an. Déterminer mathématiquement au bout de combien d'années le solde du compte aura dépassé les 13000 euros.

L'exécution du programme suivant nous permet également de répondre au problème posé sans faire aucun calcul :

```
i=0
u=10000
while u<13000:
    u=1.04*u
    i=i+1
print("Le solde aura dépassé 13000 euros au bout de "+str(i)+" années")
```

Question 3.

Aurait-on pu facilement répondre à la question en utilisant une boucle **for** ?

2.4 Le petit danger des boucles while

Voici un programme utilisant une boucle **while** :

```
i=0
u=0
while i<100:
    u=u+1
    print(u)
```

Question 4.

Que pensez-vous du programme précédent ? L'implémenter sur Python pour voir ce qui se passe dans ce cas. Puis cherchez le carré rouge !

Exercices avec les boucles while

Exercice 3.

Faire un programme Python qui calcule et affiche le plus grand nombre dont le cube est inférieur à 55000

Exercice 4.

Soit (u_n) une suite géométrique de raison 3 et de premier terme 5.

- Quelle est la limite de la suite (u_n) ?
- Faire un programme Python qui calcule à partir de quel rang tous les termes de la suite seront strictement supérieurs à 5000

Exercice 5.

Soit (u_n) une suite géométrique de premier terme 100 et de raison 0,85.

- Quelle est la limite de la suite (u_n) ?
- Faire un programme Python qui calcule à partir de quel rang tous les termes de la suite seront compris dans l'intervalle $[0; 0,01]$

Exercice 6.

Faire un programme qui demande à un utilisateur 3 nombres a,b et $u0$ positifs et qui renvoie la plus petite valeur de n pour laquelle $\sum_{k=0}^{n} u_k > 10000$ où (u_n) est la suite définie par :

$$\begin{cases} u_0 & = & u0 \\ \forall n \in \mathbb{N}, u_{n+1} & = & au_n + b \end{cases}$$

Exercices d'approfondissement

Exercice 7 (Les impôts)**.**

Le montant de l'impôt annuel est calculé de la manière suivante : le net imposable (salaire annuel brut réduit de 22 %) est imposé par tranches :

- en deça de 9700 € : 0 %
- de 9701 à 27000 € : 14 %
- de 27001 à 72000 € : 30 %
- de 72001 à 152000 € : 41 %
- au delà de 152001 € : 45 %

Le total est arrondi à l'euro inférieur.

1. Calculer à la main le montant de l'impôt pour un salaire annuel brut de 35000 € (*réponse* : 2512 €).
2. Ecrire un programme qui
   - demande à l'utilisateur d'entrer au clavier son salaire annuel brut
   - affiche dans la console le montant de l'impôt correspondant.

Exercice 8 (Devine qui c'est ?)**.**

La commande `randrange(1,N)` retourne un entier choisi « au hasard » entre 1 et N-1. Il faut l'importer depuis la bibliothèque `random`.

1. Réaliser un programme `devineKiC.py` dans lequel l'utilisateur doit deviner un entier choisi « au hasard » entre 1 et 10. La machine interroge l'utilisateur tant qu'il n'a pas trouvé le nombre mystère et le félicite cordialement lorsqu'il le devine.
2. Compléter le programme en affichant le nombre de coups joués par l'utilisateur.
3. Compléter le programme : l'ordinateur indique si le nombre mystère est plus petit ou plus grand que le nombre donné par l'utilisateur, mais on cherche à présent un nombre entre 1 et 100.
4. Même chose en demandant avant de jouer à l'utilisateur quel est le nombre maximum qu'il devra deviner.

Exercice 9 (Syracuse)**.**

1. Faire un programme en Python qui demande à l'utilisateur un entier naturel n et qui affiche les termes de la suite définie par récurrence suivante :
   - Le premier terme de la suite est le nombre demandé à l'utilisateur
   - · Si u_n est pair, $u_{n+1} = \dfrac{u_n}{2}$
     · Si u_n est impair $u_{n+1} = 3u_n + 1$
   - On s'arrête dès qu'un terme de la suite vaut 1.
2. Parmi les nombres de 2 à 100, quel est le nombre qui permet d'obtenir la suite "la plus longue"
3. Vous aussi devenez fous en vous renseignant sur le problème de Syracuse ou conjecture de Collatz