TP 2 informatique

BCPST 1 2017-2018

Structures et boucles conditionnelles

Points abordés

— Opérateurs logiques, comparaisons.
— Structures conditionnelles.
— Boucle.

Introduction

Un point important en informatique est la structure conditionnelle, qui consiste à vérifier si une expression est vraie ou non, et d'effectuer des instructions suivant la valeur de vérité de celle-ci. Un autre point fondamental est la notion de boucle : ceci consiste à répéter une tâche sous certaines conditions. Voici quelques exemples faisant intervenir ces deux notions dans la vie quotidienne :

1. vérification de votre titre de transport ;
2. rester dans le train tant que l'on n'est pas arrivé à sa station et sortir le cas échéant.

Le cas 1 pourrait être modéliser de la manière suivante. On code par :
— "*non*" si la personne n'a pas de titre de transport ou que le titre n'est pas valide,
— " *oui*" si le titre de transport est valide.
Dans le cas où le code est "*non*" on pourra avoir deux possibilités :

1. la personne peut payer l'amende de 30 euros,
2. la personne ne peut pas la payer.

Dans le deuxième cas, il faudra alors régler celle-ci à hauteur de 50 euros ultérieurement. Si on code respectivement par "*paye*" et "*ne paye pas*" le fait de pouvoir ou non de payer immédiatement l'amende, voici un programme *Python*modélisant ce problème :

```
print("votre titre de transport est-il valide ?  \n")
reponse=input()
if reponse=="non" :
    print("pouvez-vous payer l'amende ? \n")
    reponse=input()
    if reponse=="ne paye pas" :
        print("vous réglerez une amende de 50 euros")
    elif reponse=="paye" :
        print("vous avez payé 30 euros")
    else :
```

```
        print("vous dites n'importe quoi, accompagnez-nous au poste")
elif reponse=="oui" :
    print("merci, bonne journée \n")
else :
    print("vous dites n'importe quoi, accompagnez-nous au poste")
```

1. Essayer ce programme sur pyzo. Tester différentes saisies, pour passer par chacune des réponses possibles.
2. Pour quelles saisies successives l'interpréteur affiche "vous avez payé 30 euros" ? Le programme "passe" par un *if* si la condition est bien vérifiée. Sinon, il va à la suite.
3. Ajouter la ligne *print("le passe dézoné ça change la vie ! ")* à la fin du programme en ne laissant aucun espace. Que constatez-vous ?

Donnons une modélisation *Python* du problème du train. Si on considère que n est le nombre de stations restantes, on a par exemple,

```
print("Donner le nombre de stations que vous devez parcourir. \n")
n=int(input())
while (n>0) :
    print("il vous reste",n,"stations \n")
    n=n-1
print("vous etes arrivés \n")
```

1 Exercices

Exercice 1. Écrire un programme qui demande à l'utilisateur de saisir deux nombres et qui affiche le plus grand des deux.

Exercice 2. Écrire un programme qui demande à l'utilisateur trois entiers, qui les affiche d'abord dans l'ordre croissant puis dans l'ordre décroissant.

Exercice 3. Écrire un programme qui demande à l'utilisateur un réel et qui affiche la valeur absolue de ce nombre.

Exercice 4. Écrire un programme qui demande à l'utilisateur de saisir son âge, et qui affiche

```
vous payez le tarif jeune
```

si l'âge saisi est inférieur stricte à 18, qui affiche

```
vous payez le tarif senior
```

si l'âge saisi est supérieur à 65, et qui affiche

```
vous payez le tarif normal
```

si l'âge est compris entre 25 et 65 ans.

Exercice 5. Écrire un programme qui demande à l'utilisateur de saisir cinq réels a, b, R, x_0, y_0 et qui affiche

```
Le point (x,y) est dans le disque de centre (a,b) et rayon R,
```

si c'est le cas, et qui affiche le contraire sinon.

Exercice 6. Écrire un programme qui demande à l'utilisateur cinq réels a, b, c, x, y qui affiche

```
le point (x,y) est sur la droite d'équation ax+by+c=0
```

si (x_0, y_0) est effectivement un point de la droite d'équation $ax + by + c = 0$, qui affiche

```
le point est distant de ?? de la droite
```

où ? ? est la distance du point (x_0, y_0) à la droite d'équation $ax + by + c = 0$. On veillera à traiter les cas particuliers non mentionnés.

Exercice 7. Écrire un programme qui demande à l'utilisateur de saisir trois réels a, b, c et qui affichent les solutions de l'équation $ax^2 + bx + c = 0$. On veillera à traiter les cas particuliers.

Exercice 8. Écrire un programme qui affiche "premier" si un nombre est premier et "composé" sinon.

Exercice 9. Écrire un programme qui affiche tous les entiers de l'intervalle $[310, 477]$ qui ne sont divisibles ni par 2, ni par 3, ni par 5. Votre programme indiquera également le nombre d'entiers affichés.

Exercice 10. Écrire un programme qui demande à l'utilisateur de saisir un entier n, et qui affiche le plus grand entier k tel que $k^2 \leq n$. Par exemple, si l'utilisateur saisit 18, le programme affichera 4.

Exercice 11. Plus ou moins

1. Écrire un programme *PlusOuMoins.py* qui choisit un entier aléatoire compris entre 0 et 1000, qui demande à l'utilisateur de saisir un nombre tant qu'il n'a pas choisi le bon nombre. Par exemple, si le programme choisit aléatoirement le nombre 45, et que l'utilisateur choisit respectivement les nombres 19, 90, 45, on obtient un affichage du type :

   ```
   Saisir un nombre :
   19
   c'est plus !
   Saisir un nombre :
   90
   c'est moins !
   Saisir un nombre :
   45
   Vous avez trouvé le bon nombre.
   ```

2. Modifier votre programme pour qu'il affiche le nombre de tentatives de l'utilisateur. Dans l'exemple précédent, on aura donc :

   ```
   Vous avez trouvé le bon nombre en 3 tentatives.
   ```

Exercice 12. Écrire un programme qui demande à l'utilisateur deux entiers positifs n et p et qui affichent : le résultat de la somme

$$1 + 2^p + \cdots + n^p$$

Exercice 13. On rappelle que la suite de Fibonacci est définie ainsi :

$$\begin{array}{lcl} u_0 & = & 0 \\ u_1 & = & 1 \\ u_{n+2} & = & u_{n+1} + u_n, \forall n \geq 0. \end{array}$$

Écrire un programme demandant à l'utilisateur un entier positif et qui affiche le n^{e} nombre de Fibonacci.