Travaux dirigés n° 1

Instruction, variable et instruction conditionnelle

*** 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

La réalisation des TD nécessite d'installer Python 3 sur votre ordinateur. Vous trouverez ce programme sur le site `https://www.python.org`. Vous trouverez aussi des indications pour l'installation sur ce site `http://docs.python-guide.org/en/latest/`

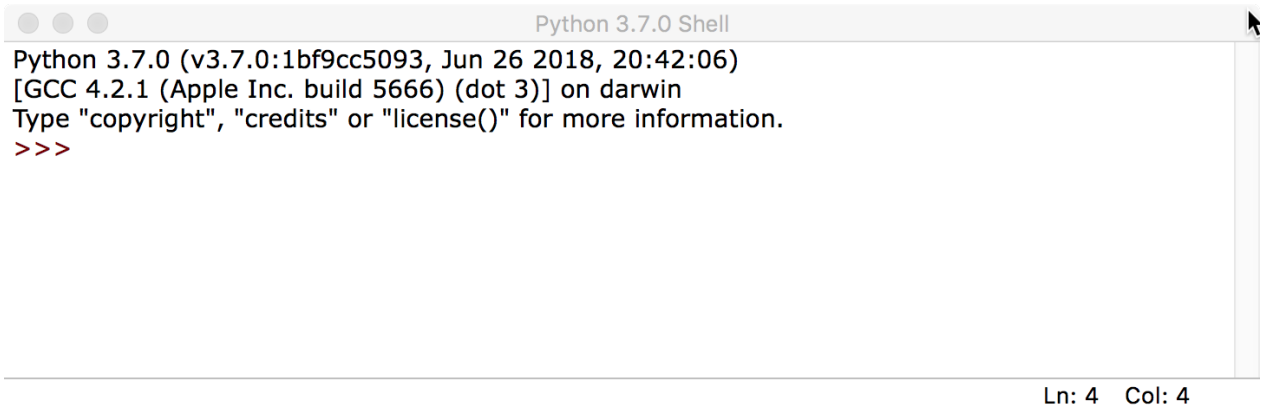
Vous trouverez par la suite des exercices obligatoires et des exercices pour approfondir. Les exercices obligatoires sont précédés par le caractère `*`.

1 Utilisation de l'interpréteur de Python

Le langage Python peut être utilisé de plusieurs manières. Pour commencer vous allez l'utiliser en mode interactif à l'aide de l'interpréteur Python, c'est à dire en « dialoguant » avec lui directement à l'aide du clavier.

Pour accéder à l'interpréteur de Python, il suffit de lancer l'application IDLE. Sous Mac OSX, IDLE se trouve dans le dossier Applications, sous-dossier Python mais vous pouvez aussi taper `idle3&` dans un terminal. Sous Windows, allez dans le menu Démarrer, trouvez le dossier Python et lancer IDLE.
L'interpréteur peut aussi être lancé depuis la ligne de commande (dans un terminal sous OSX ou Linux ou bien depuis l'invite de commande sous Windows) : il suffit de taper la commande `python3`.

Après le lancement de IDLE (ou de Python dans un terminal), vous obtenez une fenêtre similaire à celle ci-dessous. Les chevrons `>>>` indiquent que l'interpréteur est prêt à évaluer l'instruction que vous allez écrire. On définit l'instruction comme l'opération élémentaire d'un langage de programmation,



1.1 Faire des calculs

***Exercice 1.1**

Python est un langage de programmation interprété à savoir qu'il interprète les instructions et les exécute au fur et à mesure qu'il les reçoit. Aussi il est possible de l'utiliser en mode interactif comme une calculatrice. C'est ce que nous allons faire dans cette exercice. Vous allez saisir des instructions qui seront des expressions. on rappelle qu'une expression se définit comme un élément de syntaxe qui combine des littéraux, des variables, des opérateurs, et des fonctions pour en faire l'évaluation et retourner une valeur.

1) Lancez l'interpréteur Python et utilisez-le comme une calculatrice pour évaluer les expressions suivantes :

```
5+3
2 - 9
5 + 3 * 2
(5+3) * 2
20 / 3
20.0 / 3
20 // 3
20.0 // 3
20 % 3
sqrt(2)
```

L'interpréteur Python exécute l'instruction lorsque vous frappez sur la touche entrée (⏎). Faites bien attention à la valeur retournée par l'interpréteur.

2) Python respecte-t-il les priorités des opérations arithmétiques ?

Lorsqu'il y a un plus d'un opérateur, l'ordre de leur évaluation suit des règles de priorité. L'ordre peut se retenir à l'aide du mémotechnique PEMDAS. Python respecte les priorités usuelles des langages de programmation.

3) Que se passe-t-il lorsqu'on mélange des entiers et des flottants dans un même calcul ?

Lorsqu'une expression porte à la fois sur des flottants et des entiers, l'interpréteur transforme alors tous les entiers en flottants.

4) Que font les opérateurs `/` et `//` ?

L'opérateur `/` est l'opérateur de division sur les flottants.
L'opérateur `//` est l'opérateur de division entière. Appliqué sur des flottants, le résultat est la partie entière de la division renvoyée sous forme de flottant.

5) Que calcule l'opérateur `%` ?

L'opérateur `%` calcule le reste de la division entière. Appliqué sur des flottants, le résultat est renvoyée sous forme de flottant.

***Exercice 1.2**

La dernière expression de l'exercice 1.1 produit une erreur. En effet, les fonctions mathématiques ne sont pas définies dans l'interpréteur de Python, seules les opérations élémentaires le sont.

Comme beaucoup d'autres langages, Python utilise des modules (ou bibliothèques) où sont définis des ensembles de fonctions spécifiques. Ainsi avec les modules, Python peut s'étendre et s'adapter facilement à des développements spécifiques. Pour avoir accès aux fonctions définies dans un module il faut l'indiquer à l'aide de la clause `import`.

1) Tapez et exécutez les expressions suivantes et observez et analyser la réponse de l'interpréteur.

```
pi
from math import *
sqrt(2)
sqrt 2
sin(90)
pi
sin(pi/2)
sin(radians(90))
```

Respectez bien la syntaxe pour l'importation, le caractère `*` est indispensable, il indique qu'on veut importer toutes les fonctions du module.

Pour connaître les fonctions qui sont définies dans un module, il faut utiliser la commande `help("nom_du_module")`. L'interpréteur affiche alors le descriptif de toutes les fonctions qu'il contient (dans un terminal tapez la barre d'espace pour afficher la suite et `q` pour quitter l'affichage). Pour obtenir l'aide sur une fonction en particulier, tapez (après avoir importé le module dans l'environnement) `help(fonction)` sans les guillemets !

2) Essayez les commandes `help("math")` et `help(sqrt)` (après avoir exécuté `from math import *`).

Exercice 1.3

1) Évaluez les expressions suivantes dans l'interpréteur :

```
13 // 3
13 % 3
3 ** 2
2 ** 0.5
sqrt(2)
```

2) Comment savoir si un nombre est impair ?

Pour savoir si un nombre est impair il suffit de tester que le reste de sa division par 2 est égal à 1.

1.2 Utiliser des variables

Une variable en programmation sert à stocker le résultat d'une expression ou la valeur d'une constante. La variable est identifiée par un nom.
En Python, le nom d'une variable est une séquence de lettres minuscules ou majuscules et de chiffres qui doit toujours commencer par une lettre. Les caractères accentués, les cédilles, les espaces, les caractères de ponctuations et les caractères spéciaux (`&`, #, @ ...) sont interdits. Le caractère souligné est autorisé. Bien entendu le nom d'une variable ne doit pas non plus être un mot réservés du langage.
Une variable prend une valeur suite à une instruction d'affectation.

***Exercice 1.4**

Décrivez le plus complètement possible ce qui se passe lors de l'évaluation de chacune des instructions suivantes :

```
n = 7
```

On affecte l'entier 7 comme valeur à la variable n

```
rac2 = 1.414
```

On affecte le flottant `1.414` comme valeur à la variable rac2

```
x = n + rac2
```

On affecte à la variable x la valeur de l'expression située à droite du signe =, c'est à dire la somme des valeurs contenues dans les variables n et rac2.

```
print(n + rac2)
```

On demande à l'interpréteur d'afficher à l'écran la valeur de la somme des valeurs des variables n et rac2.

```
msg = "Rien a signaler"
```

On affecte la chaîne de caractères "Rien à signaler" comme valeur à la variable msg.

```
msg
```

On demande à l'interpréteur la valeur de la variable msg. Il retourne comme résultat la chaîne contenue dans msg.

```
msg2 = msg
```

On affecte à la variable msg2 la valeur de la variable msg.

```
print(msg2)
```

On demande à l'interpréteur d'afficher à l'écran la valeur de la variable msg2

```
msg3 = print(msg2)
```

On affecte à la variable `msg3` la valeur de l'expression `print(msg2)` cela à pour effet d'afficher la valeur de msg2 (c'est l'effet produit par l'appel à print).
Mais `print` étant une procédure elle ne retourne aucun résultat, ou plus exactement elle retourne la valeur spéciale `None` qui signifie `<<pas de valeur>>`.

```
msg3
```

On demande à l'interpréteur la valeur de `msg3`. Comme cette variable n'a pas de valeur, l'interpréteur ne retourne pas de valeur.

```
print(msg3)
```

On demande à l'interpréteur d'afficher à l'écran le contenu de la variable `msg3`, il affiche `None`.

Notez bien la différence entre le contenu d'une variable et l'affichage du contenu d'une variable !

***Exercice 1.5**

Testez les expressions suivantes :

```
a=10 ; b=33
a < b
```

True

```
b < a
```

False

```
a+23 = b
```

L'affectation n'est pas l'opérateur d'égalité `SyntaxError: can't assign to operator`

```
a+23 == b
```

True

```
c = b == a + 23
```

Affectation du résultat de la comparaison d'égalité dans la variable c

```
c
```

True

```
"abcd" < "abc"
```

False

```
"d" < ""
```

False

```
"4" > "123"
```

True

Attention de ne pas confondre affectation et égalité !

1) Comment s'effectue la comparaison de chaînes de caractères ?

La comparaison de chaîne de caractères s'effectue caractère par caractère de la gauche vers la droite. La comparaison est faite selon l'ordre alphanumérique ou plus exactement selon l'ordre définit par le codage ASCII des caractères dans lequel - entre autre - les chiffres précédent les lettres majuscules qui précèdent elles-même les lettres minuscules.

Attention de ne pas confondre la chaîne contenant les caractères 1, 2 et 3 avec l'entier 123 !

On peut connaître le code ASCII d'un caractère grâce à la fonction `ord`.

Exercice 1.6

1) Testez les instructions suivantes :

```
int(12)
```

convertit en entier la valeur de l'entier 12 ! ! !

```
float(12)
```

convertit en flottant la valeur de l'entier 12, le résultat est 12.0

```
int(2.71)
```

convertit en entier la valeur du réel 2.71, le résultat est 2

```
float("10.7")
```

convertit en flottant la valeur de la chaîne de caractères "10.7", le résultat est 10.7

```
int("10.7")
```

tente de convertir en entier la valeur de la chaîne de caractères "10.7". Cela provoque une erreur car la chaîne n'est pas au format d'un entier.

```
int("10") + 5
```

Le résultat est l'entier 15

2) Que font les fonctions `int` et `float` ?

Les fonctions `int` et `float` sont des fonctions de conversion de type. float("10.7") int("10.7")

Exercice 1.7

Pour chacune des expressions booléennes suivantes affectez les valeurs `True` ou `False` aux variables `a`, `b`, `c` de telle sorte que la valeur de l'expression soit `True`.

```
a or b and c
```

a=True par exemple et dans ce cas peu importe les valeur de b et c

```
a and not b or c
```

c=True par exemple et dans ce cas peu importe les valeur de a et b

```
a and (not b or a)
```

Obligatoirement a=True, peu importe b.

```
a and not(b or a)
```

Cette expression est toujours fausse quelque soit a et b.

```
(a or b) and not(a and b)
```

Si a est True, b doit être False et vice-versa.

Exercice 1.8

1) Testez les instructions suivantes :

```
r= 5
pi = 3.1415927
s = pi * r ** 2
s
type(r)
type(pi)
type(s)
type("surface")
type(sqrt)
```

2) A quoi sert la fonction `type` ?

La fonction `type` permet de connaître le type d'une expression.

1.3 Instruction conditionnelle

Une instruction conditionnelle sert à ajouter une action effectuée en fonction de l'évaluation d'une condition booléenne, à savoir vraie ou fausse. Des alternatives entre différents blocs d'instructions sont ainsi insérées dans un flux d'instructions (un programme).
Comme la structuration d'un programme Python se définie par son indentation. L'introduction d'un bloc d'instructions se termine par le caractère "deux points". Le bloc d'instructions commence à la ligne suivante la ligne d'introduction avec une indentation, par convention de 4 espaces. On n'utilise pas le caractère de tabulation sauf si l'éditeur de texte le transforme en 4 espaces. La fin du bloc se termine par le retour à l'indentation de la ligne d'introduction.
L'instruction conditionnelle utilise les mots clefs if, elif et else. La valeur de la condition est fausse si l'expression retourne une valeur, vide, 0 ou False.

***Exercice 1.9**

Lorsque vous écrivez une instruction conditionnelle, les trois points qui apparaissent quand vous passez à la ligne (en frappant la touche **Entrée** du clavier) signifient que l'instrcution que vous tapez n'est (peut-être) pas terminée. Frappez une fois de plus la touche **Entrée** pour indiquer que c'est effectivement terminé et que l'interpréteur peut maintenant exécuter l'instruction.

Attention à la syntaxe de l'instruction conditionnelle : n'oubliez pas les deux points juste après la condition avant de passer à la ligne.

1) Tapez les instructions suivantes dans l'interpréteur de Python en prenant bien soin de taper une tabulation après les 3 points affichés par l'interpréteur lorsque c'est nécessaire. Les tabulations sont indispensables en langage Python et délimitent les blocs d'instructions !

```
a = 150
if (a > 100):
    print("a depasse la centaine")
```

2) Recommencez la même instruction conditionnelle mais en affectant la valeur 20 à la variable `a`. Que se passe-t-il ?

En affectant la valeur a=20 il ne se passe rien ! L'expression d'alternative ne retourne aucune valeur puisque le cas a<=100 n'est pas prévu.

3) Saisissez maintenant la séquence d'instructions suivantes. Quelle est la différence avec la séquence précédente ?

```
a = 20
if (a > 100):
   print("a depasse la centaine")
else:
   print("a ne depasse pas la centaine")
```

L'instruction conditionnelle contient maintenant une clause si la condition n'est pas satisfaite. Quelque soit la valeur de a, un affichage est produit.

Exercice 1.10

Tapez une séquence d'instructions pour affecter des valeurs (de votre choix) à deux variables `x` et `y`, puis à l'aide d'expressions d'alternative, afficher `x est le plus grand` ou bien `y est le plus grand` ou bien `x égale y` selon les valeurs de `x` et `y`. Faites bien attention aux tabulations pour indenter les blocs d'instructions. En particulier si vous voulez passer des lignes à l'intérieur d'un bloc d'instructions vous devez quand même tapez les tabulations sur les lignes vident.

```
x = 12
y = 23

if x > y :
   print("x est le plus grand")
else :
   if y > x :
      print("y est le plus grand")
   else:
      print("x egale y")
```

Exercice 1.11

Tapez une séquence d'instructions pour affecter des valeurs à trois variables de votre choix puis, à l'aide d'alternatives, faites afficher le nom de la variable dont la valeur est la plus grande.

```
# Ecrit en python 2
a=10
b=30
c=20

if a > b :
   if a > c :
      print("a est le plus grand")
   else :
      print("c est le plus grand")
else :
   if b > c :
      print("b est le plus grand")
   else :
      print("c est le plus grand")
```

2 Premiers programmes

Dans les exercices précédents vous avez utilisé Python en mode interactif via son interpréteur. C'est pratique pour tester des instructions mais si vous voulez exécuter plusieurs fois les mêmes instructions en changeant quelques valeurs vous devez tout retaper.

Vous allez désormais taper vos instructions dans un éditeur de texte puis les conserver dans un fichier afin d'en faire un programme que vous pourrez exécuter avec Python.

Si vous utilisez IDLE, il suffit de choisir **New File** dans le menu **File** pour ouvrir un éditeur de texte dans lequel vous pouvez écrire votre programme.

***Exercice 1.12**

Saisissez le programme suivant dans un éditeur de texte, enregistrez-le dans un fichier que vous nommez `ex12.py` (on utilise l'extension .py pour les programmes écrits en Python). Ne pas saisir les numéros de ligne

```
1 # ceci est un commentaire
2 # programme exo12.py
3 # mon premier programme python
```

```
4
5 # Il faut convertir le resultat de input qui est toujours une chaine de caracteres
6 a = int(input("Entrez un entier :"))
7 # On utilise des alternatives imbriquees
8 if a < 0 :
9     print("Cet entier est negatif")
10 else :
11     if a > 0 :
12         print("Cet entier est positif")
13         if a < 100 :
14             print("et plus petit que 100")
15         else:
16             print("et superieur ou egal a 100")
17     else:
18         print("Vous avez entre 0")
```

Lorsque vous avez terminé et enregistré votre programme, exécutez-le :

- soit en demandant la commande **Run Module** du menu **Run** dans IDLE
- soit depuis une fenêtre de commande, en vous plaçant dans le répertoire où vous avez enregistré votre programme et en tapant `python ex12`.

S'il y a des erreurs à l'exécution, corrigez votre programme dans l'éditeur, sauvez-le de nouveau et ré-exécutez-le.

*Exercice 1.13

Écrivez un programme qui affiche le menu suivant, puis demande à l'utilisateur de saisir son choix (l'utilisateur entre un entier) puis affiche `OK` si le choix est valide ou bien `Choix incorrect` sinon.

```
1. Enregistrer la partie
2. Charger une partie
3. Nouvelle partie
4. Quitter
Votre choix :
```

```
# Programme 1menu.py

# Affichage du menu
print("1. Enregister la partie")
print("2. Charger la partie")
print("3. Nouvelle partie")
print("4. Quitter")

# On demande le choix de l'utilisateur
# Ne pas oublier de convertir en entier la saisie
choix = int(input("Votre choix : "))

# si le nombre saisi est entre 1 et 4 inclus on affiche OK
# sinon on affiche Choix incorrect
if choix >= 1 and choix <= 4 :
    print("OK")
else :
    print("Choix incorrect")
```

*Exercice 1.14

Écrivez un programme qui demande la saisie de trois entiers et affiche combien de ces entiers sont égaux à zéro. Pensez à utiliser des opérateurs logiques dans vos conditions.

Pour éviter d'avoir à écrire des expressions d'alternative trop compliquées on peut utiliser une variable auxiliaire pour compter le nombre de zéros.

```
# Programme 1zero.py

a = int(input("Donner un premier entier "))
b = int(input("Donner un second entier "))
c = int(input("Donner un troisieme entier "))

# nb va contenir le nombre de 0 saisis
nb = 0

if a == 0 :
   nb = nb + 1

if b == 0 :
   nb = nb + 1

if c == 0 :
   nb = nb + 1

print("Il y a", nb, " zeros")
```

Dans cette version il faut faire attention à n'oublier aucun test !

```
# Programme exo15.py

# On demande les saisies des 3 entiers, sans oublier de convertir le resultat de input
a = int(input("Donner un premier entier "))
b = int(input("Donner un second entier "))
c = int(input("Donner un troisieme entier "))

if a == 0 :
   if b == 0 :
      if c == 0 :
         print("Il y a 3 zeros")
      else :
         print("Il y a 2 zeros")
   else :
      if c == 0 :
         print("Il y a 2 zeros")
      else :
         print("Il y a 1 zero")
else :
   if b == 0 :
      if c == 0 :
         print("Il y a 2 zeros")
      else :
         print("Il y a 1 zero")
   else
      if c == 0 :
         print("Il y a 1 zeros")
      else :
         print("Il y a 0 zero")
```

***Exercice 1.15**

Écrivez un programme permettant de convertir une température donnée dans une unité (Celcius, Fahrenheit ou Kelvin) dans les autres unités.
Les unités à considérer sont Celsius (C), Fahrenheit (F) et Kelvin (K) et les formules de conversion sont :

$$C = \frac{9}{5}(F - 32) \qquad F = \frac{5}{9}C + 32$$

$$K = C + 273,15$$

Le programme demandera la valeur de la température puis l'unité dans laquelle elle est exprimée donnée sous la forme d'un caractère : C, F ou K.

Exemples d'exécution :

```
Donnez la temperature :27
Donnez l'unite C, F ou K : F
-9.0 C = 27.0 F = 264.15 K

Donnez la temperature :314
Donnez l'unite C, F ou K : K
40.85 C = 54.6944444444 F = 314.0 K
```

```
# Programme 1tempe.py
# Conversion de temperature

# On saisit la valeur de la temperature sous la forme d'un flottant
temp =  float(input("Donnez la temperature :"))
# On saisit l'unite. Comme le resultat de input est une chaine il n'y a pas besoin de
    convertir.
unite = input("Donnez l'unite C, F ou K : ")

# Si l'unite saisie n'est ni C, ni F ni K on affiche un message d'erreur
if (unite != "C") and (unite != "F") and (unite != "K") :
    print("Erreur, unite incorrecte")
else :
        # Sinon, on calcule la valeur en Celcius qu'on stocke dans une variable tempC
    # Le calcul depend de l'unite saisie
    if unite == "C" :
        tempC = temp
    elif unite == "F" :
        tempC = 9 * (temp - 32) / 5.0
    elif unite == "K" :
        tempC = temp - 273.15

    # Puis n calcule dans la variable tempF la valeur en Fahrenheit
    # a partir de la valeur en Celcius
    tempF = 5 * tempC / 9.0 + 32

    # Puis on calcule dans la variable tempK la valeur en Kelvin
    # a partir de la valeur en Celcius
    tempK = tempC + 273.15

    # On affiche la temperature dans les 3 unites
    print(tempC,"C =", tempF, "F =", tempK, "K")
```

Exercice 1.16

Écrivez un programme qui demande la saisie, sous forme de flottant, de la longueur, de la largeur et de la hauteur d'un parallélépipède rectangle et affiche son volume.

```
# programme 1volume.py

long = float(input("Donnez la longueur : "))
larg = float(input("Donner la largeur : "))
haut = float(input("Donner la hauteur : "))
print("Le volume du parallelepipede ainsi definit est ", long * larg * haut)
```

Exercice 1.17

Écrivez un programme qui demande la saisie d'un entier et qui produit l'affichage `Entier pair` ou `Entier impair` selon que l'entier saisi est pair ou impair.

```
# Programme 1pair.py

a = int(input("Donner une entier : "))
# test de la parite de a
if a % 2 == 0 :
        print("Entier pair")
else :
        print("Entier impair")
```

Exercice 1.18

Écrivez un programme qui, selon le choix de l'utilisateur, permet de calculer l'aire et le périmètre d'un carré, d'un rectangle, d'un triangle rectangle (dont on connaît les longueurs des côtés de l'angle droit), d'un parallélogramme ou d'un losange.
Exemples d'exécution :

```
1. Carre
2. Rectangle
3. Triangle rectangle
Quelle figure :3
Donner la valeur du premier cote :2
Donner la valeur du deuxieme cote :3
Aire = 3.0
Perimetre = 8.60555127546


1. Carre
2. Rectangle
3. Triangle rectangle
Quelle figure :1
Donner la valeur du cote :9
Aire =  81
Perimetre = 36
```

Afin de pouvoir calculer l'aire du triangle rectangle on a besoin de la racine carrée donc il faut importer le module math.

```
# Programme exo18.py

from math import *

# Affichage du menu
print("1. Carre")
print("2. Rectangle")
print("3. Triangle rectangle")

# Saisie du choix de l'utilisateur
figure = int(input("Quelle figure :"))

# Si la figure est un carre il faut saisir la valeur du cote
if figure == 1:
    cote = float(input("Donner la valeur du cote :"))
    print("Aire = ", cote*cote)
    print("Perimetre =", cote*4)
# sinon, si c'est un rectangle il faut saisir la longueur et la largeur
elif figure == 2:
    longueur = float(input("Donner la valeur de la longueur :"))
    largeur = float(input("Donner la valeur de la largeur :"))
    print("Aire =", longueur * largeur)
    print("Perimetre =", 2*(longueur+largeur))
# sinon, si c'est un triangle rectangle il faut saisir les valeurs des deux cotes
# qui forme l'angle droit
elif figure == 3:
    cote1 = float(input("Donner la valeur du premier cote :"))
    cote2 = float(input("Donner la valeur du deuxieme cote :"))
    print("Aire =", (cote1 * cote2)/2.0)
    print("Perimetre =", cote1 + cote2 + sqrt(cote1*cote1 + cote2*cote2))
# sinon l'utilisateur n'a pas saisit un choix valide
else:
    print("Cette figure est inconnue")
```