TD : structures conditionnelles

Exercice 1

Compléter la fonction `val_absolue(x)` qui prend en argument un nombre `x` et retourne sa valeur absolue.

Rappel : Si un nombre est positif alors sa valeur absolue est lui-même sinon c'est son opposé. Par exemple : $|4| = 4$ et $|-7| = 7$. La définition mathématique est :

$$|x| = \begin{cases} x \text{ si } x \geq 0 \\ -x \text{ sinon} \end{cases}$$

```
def val_absolue(x) :
    ...
    ...
    return ...
# Jeu de tests que doit vérifier votre fonction
assert val_absolue(0) == 0
assert val_absolue(-7) == 7
assert val_absolue(4) == 4
```

Remarque : La fonction valeur absolue est déjà définie en Python, il s'agit de la fonction `abs`.

Exercice 2

Compléter la fonction `maximum` qui prend en paramètres trois nombres `a, b` et `c` et renvoie la valeur du plus grand d'entre eux.

```
def maximum(a,b,c) :
    ...
    ...
    ...
    ...
    ...
    ...
```

Remarque : Python possède la fonction `max` qui retourne le maximum de tout les éléments qui lui sont donnés en argument.

Exercice 3

En python, on peut facilement échanger les valeurs de deux variables `a` et `b` en utilisant l'instruction :

```
a, b = b, a
```

Exemple :

```
>>> a = 4
>>> b = 7
>>> a, b = b, a
>>> a, b
(7,4)
```

1/ Compléter la fonction `trier` qui prend en paramètres deux nombres `a` et `b` et - si besoin - intervertit leurs valeurs afin de les renvoyer triées dans l'ordre croissant.

```
def trier(a,b) :
    ...
    ...
    ...
    return (a, b) #les valeurs de a et b ont si besoin été échangées de
     ↪  sorte que a <= b
```

2*/ Compléter la fonction `trier3` qui prend en paramètres trois nombres `a, b` et `c` et et, si besoin, intervertit leurs valeurs afin de les renvoyer triées dans l'ordre croissant.

```
def trier3(a,b,c) :
    ...
    ...
    ...
    ...
    ...
    return (a, b, c) #les valeurs de a, b, c ont si besoin été
     ↪  échangées de sorte que a <= b <= c
# Votre fonction doit vérifier le jeu de tests suivant :
assert trier3(3, 7, 1) == (1, 3, 7)
assert trier3(3, 1, 7) == (1, 3, 7)
assert trier3(7, 1, 3) == (1, 3, 7)
assert trier3(7, 3, 1) == (1, 3, 7)
assert trier3(1, 7, 3) == (1, 3, 7)
assert trier3(1, 3, 7) == (1, 3, 7)
```

Exercice 4

On dit que trois nombres entiers a, b et c strictement positifs et triés par ordre croissant constituent un triplet pythagoricien s'ils vérifient l'égalité : $a^2 + b^2 = c^2$.

1/ Programmer la fonction `triplet_pythagoricien` qui prend en paramètres trois nombres entiers strictement positifs `a, b` et `c` tels que `a <= b <= c` et renvoie `True` s'ils constituent un triplet pythagoricien (et `False` sinon).

Remarque : il est possible de compléter de façon élégante, sans utiliser `if ... : else : ...`

```
def triplet_pythagoricien(a, b, c) :
    assert a <= b and b <= c, "a, b, et c doivent être triés par ordre
     ↪  croissant"
    ...
    ...
    ...
    ...
# Votre fonction doit vérifier ce jeu de tests :
assert triplet_pythagoricien(3, 4, 5)
assert triplet_pythagoricien(36, 77, 85)
assert triplet_pythagoricien(65, 72, 97)
assert not triplet_pythagoricien(65, 72, 88)
assert not triplet_pythagoricien(10, 11, 12)
```

2/ Rajouter trois préconditions au début de la fonction afin de bien prendre en compte toutes les hypothèses sur `a, b` et `c`.

Exercice 5

Déterminer ce que fait cette fonction. Par exemple tester ce que produirait les appels :

`mystere(-5,-6)` , `mystere(-6,-5)` , `mystere(6,5)` .

```
def mystere(a,b) :
    if a < 0 :
        a = -a
    if b < 0 :
        b = -b
    if a <b :
        return a
    else :
        return b
```

Exercice 6

Le but est de compléter une fonction `prix` qui a pour paramètre `n` un nombre de photocopies N&B compris entre 1 et 500 et qui retourne le prix à payer en euros.

Le tableau suivant vous indique le prix unitaire d'une photocopie en fonction du nombre de photocopies :

	A4
de 1 à10	0,15 €
de 11 à 25	0,10 €
de 26 à 100	0,09 €
de 101 à 250	0,08 €
de 251 à 500	0,07 €

Par exemple pour 257 photocopies, il faudra payer $257 \times 0,07 = 17,99$ euros.

Les prix seront arrondis au centimes près à l'aide de la fonction `round` .

```
def prix(n) :
    assert n >0 and n<=500, "Le nombre de photocopies doit être compris
     ↪  entre 1 et 500"
    if n <= 10 :
        p = n * 0.15
    elif n > 10 and n ... :
        p = n * 0.10
    ...
    ...
    ...
    ...
    return p
```