Triplets pythagoriciens

Eléments de correction et programme(s) codé(s) en Python

Mise en œuvre de la recherche par les élèves :

Tout d'abord, les élèves peuvent être amenés à faire des tests dans la console.

```
>>> 10**2+11**2==12**2
False
>>> 25**2+26**6==27**2
False
```

Pour plus d'efficacité, les élèves créent alors une fonction test_pythagoriciens qui prend en entrée trois entiers consécutifs donnés, qui teste si ce sont des triplets pythagoriciens et qui renvoie un booléen (les tests dans la console donnent toute leur légitimité à cet emploi des booléens).

```
def test_pythagoriciens(a,b,c):
    if a**2+b**2==c**2:
        test=True
    else :
        test=False
    return test
```

On appelle cette fonction dans la console :

```
>>> test_pythagoriciens(10,11,12)
False
```

On arrivera à une version simplifiée de l'écriture de cette fonction :

```
def test_pythagoriciens(a,b,c):
    return a**2+b**2==c**2
```

Ce script est l'occasion de travailler sur la signification du « return » qui renvoie un résultat ; dans ce cas, il renvoie le résultat d'un test donc un booléen, True ou False.

```
>>> test_pythagoriciens_2(158,159,160)
False
```

Les élèves vont chercher à automatiser la recherche.

Le script suivant ne tient pas compte du fait que a, b, c doivent être consécutifs. Les élèves s'en rendent compte lors de l'exécution.

```
for a in range(1,11):
    for b in range(1,11):
        for c in range(1,11):
            if test_pythagoriciens_2(a,b,c):
                print(a,b,c)
```

```
>>>
3 4 5
4 3 5
6 8 10
8 6 10
```

Le programme est alors modifié ainsi :

```
for a in range(1,11):
    for b in range(1,11):
        for c in range(1,11):
            if test_pythagoriciens_2(a,b,c) and b==a+1 and c==a+2:
                print(a,b,c)
```

Modification du programme afin de réduire le nombre de calculs :

```
for a in range(1,1000001):
    if a**2+(a+1)**2==(a+2)**2:
        print(a,a+1,a+2)
```

ou en définissant une fonction au préalable :

```
def test_pythagoriciens_3(a):
    return a**2+(a+1)**2==(a+2)**2

for a in range(1,101):
    if test_pythagoriciens_3(a):
        print(a,a+1,a+2)
```

La différence de vitesse d'exécution est notable entre les 2 scripts précédents ; on pourra éventuellement parler du nombre de calculs faits par l'ordinateur dans les 2 cas (complexité temporelle).

Remarque : afficher tous les triplets pythagoriciens solutions à l'aide d'une fonction
Considérons cette fonction :

```
def triplets_pythagoriciens(n):
    for a in range(1,n+1):
        if test_pythagoriciens_3:
            return a,a+1,a+2
```

La fonction écrite ci-dessus n'est pas pertinente ici ; en effet, le « return » s'arrête dès qu'un résultat est trouvé donc à 3, 4, 5 le script de la fonction s'arrête, les autres triplets d'entiers consécutifs ne sont pas testés.

Pour s'en convaincre, il suffit de tester ce script qui devrait afficher tous les triplets d'entiers, pas forcément consécutifs, satisfaisant l'égalité de Pythagore.

```
def test_faux(n):
    for a in range(1,n+1):
        for b in range(1,n+1):
            for c in range(1,n+1):
                if test_pythagoriciens_2(a,b,c):
                    return a,b,c
```

Affiche

```
>>> test_faux(10)
(3, 4, 5)
```

au lieu des résultats attendus :

```
>>>
3 4 5
4 3 5
6 8 10
8 6 10
```

car le script s'arrête dès que le return a pu renvoyer un résultat.

Pour que soient affichés et testés tous les triplets pythagoriciens, il faudrait recourir aux listes dans l'écriture de la fonction comme ci-dessous.

```
def triplets_pythagoriciens(n):
    triplets_solutions=[]
    for a in range(1,n+1):
        if test_pythagoriciens_3(a):
            triplets_solutions.append((a,a+1,a+2))
    return triplets_solutions
```

Conjecture : les élèves conjecturent que 3, 4, 5 est le seul triplet que répond à la question.
Démonstration :
On montre que la seule solution de l'équation $(a-1)^2 + a^2 = (a+1)^2$ est 4 à l'aide des identités remarquables et des équations produit nul.