Conjecture de Goldbach

La **conjecture de Goldbach** est une hypothèse mathématique qui n'a jamais été démontrée… ni infirmée. Elle stipule que tout entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers.
Ce problème mathématique est connu depuis le XVIIIème siècle et n'a toujours pas été résolu !
On souhaite vérifier cette conjecture en programmant une fonction `goldbach(n)` permettant d'afficher, pour tout entier n, sa décomposition en (somme de) nombres premiers.

Etablir une liste des nombres premiers

Notre programme va tester les différentes combinaisons de somme de nombres premiers : il faut donc que ces nombres premiers soient stockés quelque part.
Il convient donc, dans un premier temps, de réaliser un programme `liste(n)` qui va stocker les nombres premiers allant de 2 à n. Si ce n'est pas déjà fait, rendez-vous sur l'activité Nombres premiers [1], on pourra réutiliser les fonctions déjà programmées.

Un programme pour tester des combinaisons

Plusieurs solutions sont bien sûr possibles. L'idée proposée ici est de créer une liste de nombres premiers allant de 2 jusqu'à n. Puis, de parcourir cette liste en testant les différentes sommes possibles entre ces nombres.

1. La conjecture de Goldbach ne s'applique qu'à certains types de nombres. Quelles sont les conditions à poser dans notre programme pour être sûr qu'il ne réalise la vérification que dans des cas légitimes ?

   On pourra utiliser deux conditions pour écarter deux types de cas, les entiers inférieurs à 3, et les nombres impairs :

   ```
   if n<3:
     return "C'est inferieur a trois"
   elif n%2!=0:
     return "C'est impair"
   ```

2. Pour la suite du programme, on propose de passer en revue les éléments de la liste des nombres premiers. On pourra utiliser, par exemple, deux variables `i` et `j` qui permettront de désigner tour à tour les éléments dans la liste. On rappelle que, par exemple, `liste(n)[2]` désigne l'élément d'indice 2, donc le troisième, de la liste retournée avec la fonction `liste(n)`. On pourra afficher les couples `i,j` solutions avec la fonction `print()`.

1. `www.numworks.com/fr/professeurs/activites-pedagogiques/seconde/nombres-premiers`

Dans le cas le plus simple, on pourra afficher tous les couples, avec toutes les permutations :

```
for i in range(len(liste(n))):
  for j in range(len(liste(n))):
    if liste(n)[i]+liste(n)[j]==n:
      print(liste(n)[i],liste(n)[j])
```

On pourra ensuite, par une légère modification, chercher à éviter les permutations en excluant pour `j` les valeurs déjà prises par la variable `i` :

```
  for j in range(i,len(liste(n))):
```

On peut aussi envisager, pour que l'écriture soit simplifiée, une variable `premiers=liste(n)` au début du programme.

3. Comment pourrait-on compter le nombre de solutions en modifiant légèrement ce programme ?

On peut ajouter une variable `solution`, initialisée à 0 en début de programme et incrémentée de 1 à chaque fois que la calculatrice affiche un couple solution.
On pourrait ensuite ajouter un `return solution` en fin de programme. On propose ici le script complet :

```
from math import *
def nombre_premier(n):
  if n>1:
    for k in range(2,sqrt(n)+1):
      if k<n and n%k==0:  #need k<n if n=2
        return False
    return True
def liste(n):
  return [k for k in range(1,n+1) if nombre_premier(k)==True]
def goldbach(n):
  #solution=0
  if n<3:
    return "C'est inferieur a trois"
  elif n%2!=0:
    return "C'est impair"
  else:
    for i in range(len(liste(n))):
      for j in range(i,len(liste(n))):
        if liste(n)[i]+liste(n)[j]==n:
          #solution+=1
          print(liste(n)[i],liste(n)[j])
  #return solution
```