Quelques modules arithmétiques

```
#PGCD
def pgcd(a, b):
   if a == 0 :
      return b

   return pgcd(b % a, a)
 ---------------------------------------------------------------------------
#PPCM
def ppcm(a, b):
   return (a*b) // pgcd(a, b)
-----------------------------------------------------------------------------
#Premier
def isPrime(n):
 if (n <= 1):
  return False
 for i in range(2, n):
  if (n % i == 0):
   return False;
 return True;
-----------------------------------------------------------------------------
#Combinaison
def combinaison(n, r):
   return (factorial(n) / (factorial(r) * factorial(n - r)))
-----------------------------------------------------------------------------
#Arrangement
def arrangement(n, r):
   return (factorial(n) / factorial(n - r))
-----------------------------------------------------------------------------
#Fonction puissance
def puissance(x, n):
   resultat = 1
   for i in range(n):
      resultat *= x
   return resultat
-----------------------------------------------------------------------------
#Fonction factorielle
def factoriel(n):
   if n == 0:
      return 1
   else:
      return n * factoriel(n-1)
-----------------------------------------------------------------------------
 #Fonction conversion base 10 base 2
def conversion_base_10_base_2(n):
   if n > 1:
      conversion_base_10_base_2(n // 2)
   print(n % 2, end = '')
-----------------------------------------------------------------------------
#Conversion base 10 en base 8
def convert_base10_to_base8(n):
   return oct(n).replace("0o","")
```

```
n = int(input("Entrez un nombre en base 10 : "))
print("Le nombre en base 8 est :",convert_base10_to_base8(n))
-----------------------------------------------------------------------------------------
#conversion entre bases 10 en base 2
def conv_dec_bin(n):
   """conversion entre base 10 et base 2"""
   bina=''
   while(n !=0):
      if n%2==0 :
         bina='0'+bina
      else:
         bina='1'+bina
      n=n//2
   return bina
-------------------------------------------------------------------------------------------------
#conversion entre bases 10 et base 16
def conv_dec_hex(n):
   """conversion entre base 10 et base 16"""
   hexa = "0123456789ABCDEF"
   if n // 16 == 0:
      return hexa[n % 16]
   else:
      return conv_dec_hex(n // 16) + hexa[n % 16]
-------------------une autre solution ----------------------------------------
def base10_to_base16(n):
   """Convert a base 10 number to base 16"""
   if n < 0:
      return '-' + base10_to_base16(-n)
   (d, m) = divmod(n, 16)
   if d > 0:
      return base10_to_base16(d) + "0123456789ABCDEF"[m]
   else:
      return "0123456789ABCDEF"[m]
-------------------------------------------------------------------------------
# conversion base 16 en base 2
def hex_to_binary(hex_string):
   binary_string = ""
   for char in hex_string:
      binary_string += bin(int(char, 16))[2:].zfill(4)
   return binary_string
-------------------------------------------------------------------------------
# conversion base 2 en base 16
def binary_to_hex(binary_string):
   hex_string = ""
   for i in range(0, len(binary_string), 4):
      hex_string += hex(int(binary_string[i:i+4], 2))[2:]
   return hex_string
-------------------------------------------------------------------------------
# conversion base 10 en base b quelconque :
def dec2base(n, b):
   if n == 0:
      return [0]
   digits = []
```

```
    while n:
        digits.append(int(n % b))
        n //= b
    return digits[::-1]
-----------------------------------------------------------------------------------------
# divisibilité par 11 (méthode somme)
def divisiblePar11(n):
    if (n == 0):
        return True
    odd_digit_sum = 0
    even_digit_sum = 0

    while (n != 0):
        odd_digit_sum += n % 10
        n = int(n / 10)

        if (n != 0):
            even_digit_sum += n % 10
            n = int(n / 10)

    return ((odd_digit_sum - even_digit_sum) % 11 == 0)
-----------------------------------------------------------------------------------
#divisibilité par 17 :
def divisible_par_17(nombre):
  if nombre == 0:
    return True
  elif nombre < 0:
    return divisible_par_17(-nombre)
  else:
    return divisible_par_17(nombre - 17)
-----------------------------------------------------------------------------------------
#conversion d'un réel en base 2
```