```
# ============================================== #
# ============================================== #
# IPT MATHS SPE                                  #
# Récursivité Partie I (Séance IV de 2017-2018) #
# Algorithmes/codes du cours                     #
# Octobre 2017                                   #
# ============================================== #
# ============================================== #

import sys
sys.setrecursionlimit(2000)

# ============== #
# ============== #
# La factorielle #
# ============== #
# ============== #

## Programme itératif

def fact_iter(n):
    if type(n) != int:
        raise TypeError('Vous devez fournir un entier !')
    elif n < 0:
        raise ValueError('Vous devez fournir un entier naturel !')
    else:
        F = 1
        for i in range(1,n+1):
            F *= i
        return(F)

## Programme récursif
# La récursivité N'EST PAS terminale !

def fact_rec(n):
    if n != 0:
        return(n*fact_rec(n-1))
    else:
        return(1)

# ===================================== #
# ===================================== #
# PGCD de deux entiers naturels         #
#                                       #
# Remarque : les codes ci-dessous ne    #
#            tiennent pas compte  de la #
#            de la situation a=0 et b=0. #
# ===================================== #
# ===================================== #

## Programme itératif

def pgcd_iter(a,b):
    while b!= 0:
        a,b = b,a%b
    return(a)

## Programme récursif
# La récursivité EST terminale !
```

```python
def pgcd_rec(a,b):
    if b == 0:
        return(a)
    else:
        return(pgcd_rec(b,a%b))

# ======================== #
# ======================== #
# Evaluation d'un polynôme #
# Méthode de HORNER        #
# ======================== #
# ======================== #


# Remarque : Le polynôme est fourni sous la forme d'une liste P où TOUS LES
# COEFFICIENTS apparaissent : P[0] est le coefficient constant P[i] est le
# coefficient de X^i.


## Programme "classique" (ce n'est pas du Horner !!!)
# Calcul basique où on évalue les monômes successivement...

def EvalPoly(P,x):
    n = len(P)
    res = 0
    for i in range(n):
        res += P[i]*x**i
    return(res)

print(EvalPoly(P,2))
print(EvalPoly(P,-1))

## Programme récursif (ce n'est pas non plus du Horner !!!)
# Avec ce programme, la liste P n'est pas altérée MAIS la récursivité N'EST
# PAS TERMINALE !
# ATTENTION : cette version, bien que récursive, NE CORRESPOND PAS à la
# méthode de Horner !

def horner_rec1(P,x):
    if len(P) == 1:
        return(P[0])
    else:
        return(P[0] + horner_rec1(P[1:],x)*x)

print(horner_rec1(P,2))
print(horner_rec1(P,-1))

## Méthode de Horner (version itérative)

def horner_iter(P,x):
    n = len(P)
    c_new = P[-1]
    for i in range(n-2,-1,-1):
        c_new = c_new*x + P[i]
    return(c_new)

print(horner_iter(P,2))
print(horner_iter(P,-1))

## Méthode de Horner (version récursive)
```

```python
# Avec ce programme, la liste P n'est pas altérée ET la récursivité est
# terminale !

def horner_rec2(P,x):
    if len(P) == 1:
        return(P[0])
    else:
        return(horner_rec2(P[:-2] + [P[-2] + P[-1]*x],x))

## Pour tester
# Le polynôme de votre choix !
P = [1,1,1,-2,3]print(horner_rec2(P,2))

print(horner_rec2(P,-1))
```