PyCosmic_2a.py

```
#!/usr/bin/env python3.6
# -*- coding: utf-8 -*-
# Série première. En Général Python
# Origine : Cosmic 9876543210123456789
# MusicAtoumic PyCosmic_2a Ce 7 novembre 2017
# Collection Python Compulse

""" Version public"""

import time

t = time.time()

Premiers = [1]  # Liste Les Nombres Premiers
Fermiers = [1]  # Liste Des nombres communs
Ramieres = [1]  # Dictionnaire Les limites communes
minR = [0]  # Section positionnement Mini
maxR = [0]  # Section positionnement Maxi

Presigne = [0]  # Porte Fonction
Presigne[0] = 1  # :appelant::CoComm::Racine:
maxFerme = [0]  # Donnée :max(Fermiers):
maxFerme[0] = 1  # :CoMulti::Racine::CoComm:

Lectyp1 = [0]  # Lecteur type 1
Lectyp1[0] = 1
Lectyp5 = [0]  # Lecteur type 5
Lectyp5[0] = 1

# Origine Cosmic
Cosmic = 876524154632551
snScript = 1  # := 1: Affiche les communs
sQuare = int(Cosmic ** .5)

# Zéro issue
limit = 1
if Cosmic not in (0, 1, 2, 3, 5):

    # Coaxe de base
    Coaxes = [2, 3, 5]
    for i in range(7):
        if i in Coaxes and not Cosmic % i:
            Premiers.append(i)
            Fermiers.append(i)
            Presigne[0] = i  # Première signature


    def Racine(cine):
        """Espace Racine(cine):
        Comparaison avec la racine première commune
        Composition des préalables nombres premiers"""

        RacPre = []  # maxFerme[0] = max(Fermiers)
        for pre in Premiers:
            sec = cine // pre
            if not cine % sec and sec not in Premiers:
                # Cas: Premier * Premier = Cosmic
                cine = sec
```

```python

        # maxFerme[0] = max(Fermiers)
        if Premiers:
            dvMaxfer = Cosmic // maxFerme[0]
            ipMaster = dvMaxfer // cine
            if ipMaster not in Fermiers:
                if ipMaster >= sQuare and max(Premiers) > 1:
                    ipMaster = ipMaster // Premiers[1]
                RacPre.append(cine)
                RacPre.append(ipMaster)

        for Rac in RacPre:
            cine = Rac
            for pre in Premiers:
                if not cine % pre and pre > 1:
                    break
            else:
                if cine <= sQuare:
                    Premiers.append(cine)
                    Premiers.sort()
                    Fermiers.append(cine)
                    Presigne[0] = cine  # Première signature


    def CoComm(coco):
        """Espace CoComm(coco):
        Lecteurs à intervalles donnés
        Cueilleurs aux moments donnés"""

        if len(Premiers) > 1 and coco == 1:
            coco = max(Premiers)
            co6 = coco % 6
            if co6 == 5:
                coco -= 4

        def RegType(Rc):
            """Calibrage RegType(Rc):
            Calage proche type (1&5)"""
            car1 = Rc
            if car1 % 6 == 1:
                car5 = car1 - 2
            else:
                differe = (car1 % 6) - 1
                car1 -= differe
                car5 = car1 - 2
            Lectyp1[0] = car1
            Lectyp5[0] = car5

        # Réglages (1&5)
        if coco < 7:
            Lectyp1[0] = 7
            Lectyp5[0] = 11
        else:
            RegType(coco)

        # Intervalle :CaxeP: maxPremiers(sup(square))
        if len(Premiers) > 1:
            CaxeP = int((Cosmic // max(Premiers)) ** .5)
            mR = maxFerme[0] * max(Ramieres)
            if mR >= sQuare and len(Ramieres) > len(Premiers):
                maxR[0] = Ramieres[-2]
```

```
                RegType(maxR[0])
            else:
                minR[0] = mR
        else:
            CaxeP = sQuare

        Presigne[0] = 0

        # Lecture Temps six
        while not Presigne[0]:
            if not Cosmic % Lectyp1[0] and Lectyp1[0] not in Premiers:
                if Lectyp1[0] not in Fermiers:
                    Racine(Lectyp1[0])
                    break
            elif not Cosmic % Lectyp5[0] and Lectyp5[0] not in
Premiers:
                if Lectyp5[0] not in Fermiers:
                    Racine(Lectyp5[0])
                    break
            Lectyp1[0] += 6
            Lectyp5[0] += 6

            if Lectyp1[0] > CaxeP:
                break


    def CoMulti():
        """Espace CoMulti():
        Accumulation des nouveaux communs
        Formation des nouvelles données"""

        for p1 in Premiers[1:]:
            for f1 in Fermiers:
                produit = p1 * f1
                # Condition commune
                while not Cosmic % produit:
                    if produit not in Fermiers:
                        Fermiers.append(produit)
                        Fermiers.sort()
                    break
                else:
                    if produit not in Ramieres:
                        pfp = [f1, produit]
                        for pf in pfp:
                            if pf not in Ramieres and Cosmic % pf:
                                Ramieres.append(pf)
        maxFerme[0] = max(Fermiers)
        Ramieres.sort()


    # Espace appelant
    while Presigne[0]:

        if max(Fermiers) <= Cosmic:
            CoMulti()

        # Fermiers(ipMax) _____________
        # print('#  F Fermiers:', Fermiers)
        if max(Fermiers) >= sQuare:
            # (':Fermiers=Cosmic: Communs Complets')
            if max(Fermiers) == Cosmic:
```

```
                break

        elif len(Fermiers) > 1:
            dvFer = max(Fermiers) * Premiers[1]
            if dvFer >= sQuare:
                ipFer = Cosmic // dvFer
                if ipFer in Premiers:
                    break

        # Limites ____________________
        # print('#  L Premiers:', limit)
        if 1 < len(Premiers) < 2:
            Lpr = Premiers[1]
        else:
            Lpr = limit

        if limit != 0:
            CoComm(Lpr)

# ########## #   ZERO   ISSUE   # ############################ #

else:  # Zéro issue
    Fermiers = []
    Premiers = [0]
    if not Cosmic:
        Fermiers.append(Cosmic)
        Fermiers.append(Cosmic)
    else:
        Fermiers.append(Cosmic)

# ############################################################ #

fer = len(Fermiers)
print('\n Premiers Libres Communs(', fer, ')ex \n', 'PL', Premiers)
if snScript:
    Libre = {}
    if not Cosmic:  # Zéro issue
        Libre[Cosmic] = Fermiers[1]
    else:
        for ff in Fermiers:
            if ff <= sQuare:
                f = ff
                Libre[f] = Cosmic // ff
            else:
                f = Cosmic // ff
                Libre[f] = ff
    for ip, dv in sorted(Libre.items()):
        print('{} * {} typ {}*{}'
              .format(ip, dv, ip % 6, dv % 6))

print('\n PyCosmic_2a.py En: \n', time.time() - t)

```