SE FAMILIARISER AVEC LE LANGAGE PYTHON…

Après avoir réalisé le premier exercice de « traduction », n'hésitez pas à piocher selon vos goûts dans cette liste d'exercices…

- Calcul de PGCD et PPCM

Traduisez en Python l'algorithme suivant :

```
Algorithme pgcdPpcm

# cet algorithme permet de calculer le PGCD et le PPCM de deux entiers
# naturels non nuls entrés au clavier

variables   a, b, aa, bb, reste, pgcd, ppcm : entiers naturels

début

      # lecture des données
  Entrer ( a, b )

      # initialisations
  aa ← a
  bb ← b
  reste ← aa mod bb

      # boucle de calcul
  tantque ( reste ≠ 0 )
      aa ← bb
      bb ← reste
      reste ← aa mod bb
  fin_tantque

      # calcul du pgcd et du ppcm
  pgcd ← bb
  ppcm ← ( a * b ) div pgcd

      # affichage résultat
  Afficher ( pgcd, ppcm )

fin
```

```
# pgcdPpcm

# ce script permet de calculer le PGCD et le PPCM de deux entiers
# naturels non nuls entrés au clavier

# lecture des données
a = int(input("a ? "))
b = int(input("b ? "))

# initialisations
aa = a
bb = b
reste = a % b

# boucle de calcul
while (reste != 0):
    aa = bb
    bb = reste
    reste = aa % bb

# calcul du pgcd et du ppcm
pgcd = bb
ppcm = ( a * b ) // pgcd

# affichage résultat
print("PGCD :",pgcd,"PPCM :",ppcm)
```

- Calcul d'image

Écrire un script Python qui calcule l'image d'un nombre x par une fonction du type $f(x) = ax^2 + bx + c$ (les valeurs de a, b et c seront demandées à l'utilisateur).

```
# imageFonction

# Ce script calcule l'image d'un nombre x par une fonction du type
# f(x) = ax2 + bx + c (a, b et c donnés)

# lecture données
a = int(input("a = "))
b = int(input("b = "))
c = int(input("c = "))
x = int(input("x = "))

# calcul
res = a * x * x + b * x + c

# affichage résultat
print( res )
```

- Longueur d'un segment

Écrire un script Python permettant de calculer la longueur d'un segment donné par les coordonnées de ses deux extrémités (qui seront demandées à l'utilisateur).

```
# longueurSegment

# Ce script calcule la longueur d'un segment AB
# (XA, YA, XB, YB donnés)

import math

# lecture données
XA = float(input("XA : "))
YA = float(input("YA : "))
XB = float(input("XB : "))
YB = float(input("YB : "))

# calcul de la longueur
longueur = math.sqrt ( ((XB-XA) * (XB-XA)) + ((YB-YA) * (YB-YA)) )

# affichage résultat
print("longueur du segment : ", longueur)
```

- Somme de deux fractions

Écrire un script Python permettant de calculer le numérateur et le dénominateur d'une somme de deux fractions entières (on ne demande pas de trouver la fraction résultat sous forme irréductible !).

```
# sommeDeDeuxFractions

# Ce script calcule le numérateur et le dénominateur d'une somme
# de deux fractions entières.

# lecture données
numA = int(input("numérateur A : "))
denomA = int(input("dénominateur A : "))
numB = int(input("numérateur B : "))
denomB = int(input("dénominateur B : "))

# calcul d'un dénominateur commun
denomSomme = denomA * denomB

# calcul du numérateur
numSomme = (numA * denomB) + (numB * denomA)

# affichage résultat
print("la somme vaut",numSomme,"/",denomSomme)
```

- Équation de droite donnée par deux points

Écrire un script Python permettant de déterminer une équation de la droite passant par deux points donnés.

```
# équationDroite
```

```
# Ce script détermine l'équation d'une droite donnée par deux de
# ses points

# lecture données
XA = float(input("XA : "))
YA = float(input("YA : "))
XB = float(input("XB : "))
YB = float(input("YB : "))

# cas particulier : XA = XB
if ( XA == XB ):
    print( "équation de la droite : x = ", XA )

# cas général
else:
    # calcul du coefficient directeur
    coef = ( YA - YB ) / ( XA - XB )

    # calcul de l'ordonnée à l'origine
    cste = YA - coef * XA

    # affichage résultat
    print ( "équation de la droite : y = ", coef, "x + ", cste )
```

- Équation de droite parallèle

Soient trois points A, B et C ; écrire un script Python permettant de déterminer une équation de la droite passant par A et parallèle à la droite (BC).

```
# équationDroiteParallèle

# Ce script détermine l'équation d'une droite passant par A et
# parallèle à un segment BC

# lecture données
XA = float(input("XA : "))
YA = float(input("YA : "))
XB = float(input("XB : "))
YB = float(input("YB : "))
XC = float(input("XC : "))
YC = float(input("YC : "))

# cas particulier 1 : XB = XC
if ( XB == XC ):
    print( "équation de la droite : x = ", XA )

# cas particulier 2 : YB = YC
elif ( YA == YB ):
    print( "équation de la droite : y = ", YA )

# cas général
else:
    # calcul du coefficient directeur
    coef = ( YB - YC ) / ( XB - XC )

    # calcul de l'ordonnée à l'origine
    cste = YB - coef * XB

    # affichage résultat
    print ( "équation de la droite : y = ", coef, "x + ", cste )
```

- Intersection de deux intervalles

Écrire un algorithme permettant de calculer l'intersection de deux intervalles [a, b] et [c, d].

```
# intersectionIntervalles

# Ce script permet de calculer l'intersection de deux intervalles
# d'entiers de la forme [a, b] et [c, d].

# lecture données
a = float(input("Entrer a : "))
b = float(input("Entrer b : "))
c = float(input("Entrer c : "))
d = float(input("Entrer d : "))

# calcul du maximum des bornes inférieures
if (a < c):
```

```
    max = c
else:
    max = a

# calcul du minimum des bornes supérieures
if (b < d):
    min = b
else:
    min = d

# affichage du résultat
if (max <= min):
    print ("[", max,";",min,"]")
else:
    print ("Intersection vide.")
```

- Calcul de la n-ième valeur d'une suite

Écrire un script Python permettant de calculer la n-ième valeur d'une suite de la forme $u_n = a.u_{n-1} + b$, $u_0 = c$ (les valeurs de a, b et c seront demandées à l'utilisateur).

```
# niemeValeurSuite

# Ce script permet de calculer la n-ième valeur d'une suite de la
# forme un = aun-1 + b, u0 = c

# lecture des données
a = int(input("a ? "))
b = int(input("b ? "))
c = int(input("c ? "))
n = int(input("n ? "))

# initialisations
un = c

# boucle de calcul
for i in range(1,n+1):
    un = a * un + b

# affichage résultat
print("la",n,"ième valeur de la suite est",un)
```

- Minimum de trois nombres

Écrire un script Python permettant d'afficher le plus petit de trois nombres entrés au clavier.

```
# minimumTroisNombres

# Ce script permet d'afficher le plus petit de trois nombres
# entrés au clavier.

# lecture données
a = int(input("a ? "))
b = int(input("b ? "))
c = int(input("c ? "))

# comparaisons
if (a < b):
    if (a < c):
        min = a
    else:
        min = c
elif (b < c):
    min = b
else:
    min = c

# Affichage minimum
print("Le plus petit de ces trois entiers est",min)
```

- Calcul de factorielle

Écrire un script Python permettant de calculer la factorielle d'un entier naturel entré au clavier (n'hésitez pas à tester votre script pour des « grands nombres »…).

```
# factorielle
```

```
# Ce script permet de calculer la factorielle d'un entier naturel
# entré au clavier

# lecture des données
n = int(input("n ? "))

# initialisation
fact = 1

# boucle de calcul
for i in range(2,n+1):
    fact = fact * i

# affichage du résultat
print("La factorielle de",n,"vaut",fact)
```

- Afficher les diviseurs d'un entier

Écrire un script Python permettant d'afficher les diviseurs d'un entier naturel par ordre croissant.

```
# afficheDiviseurs

# Ce script permet d'afficher les diviseurs d'un entier naturel
# par ordre croissant

# lecture des données
n = int(input("n ? "))

# cas où n est nul
if (n == 0):
    print("Tous les entiers sont diviseurs de 0")

# cas général
else:
    # boucle de parcours, si diviseur divise n, on l'affiche
    for diviseur in range(1,(n // 2) + 1):
        if (n % diviseur == 0):
            print(diviseur)

    # dernier diviseur : n
    print(n)
```

- Approximation de Pi

Écrire un script Python permettant de calculer une approximation de π à partir de la formule de Leibniz (on demandera le nombre d'étapes de calcul) :

$$\pi \approx 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots\right)$$

```
# approximationDePI_1

# Ce script calcule une approximation de PI

# lecture données
nbEtapes = int(input("Entrer le nombre d'étapes : "))

# initialisations
approxPi = 1
signe = -1

# boucle de calcul
for i in range (1,nbEtapes+1):

    # on ajoute le i-ième terme
    approxPi = approxPi + signe / (2*i+1)

    # on change de signe pour le tour suivant
    signe = -signe

# affichage du résultat
approxPi = 4 * approxPi
print (approxPi)
```

- Nombre premier

Écrire un script Python permettant de déterminer si un entier naturel entré au clavier est premier.

```
# nombrePremier

# Ce script détermine si l'entier N est premier ou non

# lecture de N
N = int(input("N ? "))

# initialisations
racineDeN = int(math.sqrt(N))
diviseur = 2

# tant qu'on n'a pas trouvé de divisieur, on avance...
while ((N % diviseur != 0) and (diviseur <= racineDeN)):
    diviseur = diviseur + 1

# si diviseur est allé au-delà de racineDeN, N est premier
if (diviseur > racineDeN):
    print ("Le nombre", N, "est premier.")
else:
    print ("Le nombre", N, "est divisible par", diviseur)
```

- Liste des 100 premiers nombres premiers (sic)

Écrire un script Python permettant d'afficher la liste des 100 premiers nombres premiers.

```
# 100nombresPremiers

# ce script affiche la liste des 100 premiers nombres premiers
import math

# initialisations
compteur = 0
n = 1

# boucle principale
while (compteur < 100):

    # initialisations
    racineDeN = int(math.sqrt(n))
    diviseur = 2

    # tant qu'on n'a pas trouvé de divisieur, on avance...
    while ((n % diviseur != 0) and (diviseur <= racineDeN)):
        diviseur = diviseur + 1

    # si diviseur est allé au-delà de racineDeN, N est premier
    if (diviseur > racineDeN):
        print(n)
        compteur = compteur + 1

    # on passe à l'entier suivant
    n = n + 1
```

- Calcul du n-ième nombre de Fibonacci

Écrire un script Python permettant de calculer le nombre de Fibonacci F(n), sachant que F(0) = 0, F(1) = 1, et F(n) = F(n-1) + F(n-2).

```
# fibonacci

# Ce script permet de calculer le nombre de Fibonacci F(n)

# lecture des données
n = int(input("n ? "))

# initialisations
fibo = 1
fiboprecedent = 0

# test si cas simple ( n = 0 )
if (n == 0):
    print("Le ",n,"ième nombre de fibonacci vaut",0)

else:
    # boucle de calcul pour le cas général
```

```
    for i in range(1,n+1):
        fibo = fibo + fiboprecedent
        fiboprecedent = fibo - fiboprecedent

    # affichage résultat
    print("Le ",n,"ième nombre de fibonacci vaut",fibo)
```

- Nombres à trois chiffres

Écrire un script Python permettant d'afficher par ordre croissant tous les nombres à 3 chiffres dont la somme des chiffres est multiple de 5.

```
# nombresTroisChiffresMultiple5

# ce script affiche la liste des nombres à trois chiffres dont la somme
# des chiffres est multiple de 5

# boucle principale
for n in range(100,1000):

    # on récupère les chiffres
    unite = n % 10
    dizaine = ((n - unite) // 10) % 10
    centaine = (n - 10 * dizaine - unite) // 100

    # on affiche n si la somme des chiffres est multiple de 5
    if ((centaine + dizaine + unite) % 5 == 0):
        print(n)
```