Dans tout ce sujet on suppose qu'on aura préalablement exécuté les instructions suivantes :

```
from math import *
import numpy as np
```

1) On définit la suite $(x_n)_{n\in\mathbb{N}}$ par $x_0 = 1$ et $\forall n \in \mathbb{N}, x_{n+1} = \sin(x_n)$.

On désire écrire une fonction `iteres(n)` dont la valeur de retour est la liste $[\mathtt{x_0}, \dots, \mathtt{x_n}]$

Les étudiants Anastragon, Bector et Coryphime proposent respectivement les codes suivants :

Code d'Anastragon :

```
def iteres (n) :
    res = np.arange(n+1)+1.
    while n > 0 :
        res = np.sin(res)
        n -= 1
    return (res)
```

Code de Bector :

```
def iteres (n) :
    res = (n+1)*[1.]
    for i in xrange (n) :
        res[i+1]=sin(res[i])
    return (res)
```

Code de Coryphime :

```
def x (n) :
    res = 1.
    while n > 0 :
        res = sin (res)
        n -= 1
    return (res)

def iteres (n) :
    res = range (n+1)
    for i in res :
        res[i] = x(i)
    return (res)
```

Parmi ces 3 codes, lequel est correct et a une complexité de l'ordre de $\mathcal{O}(n^2)$? Lequel est correct et a une complexité de l'ordre de $\mathcal{O}(n)$? Lequel est faux ?

Le code faux s'exécute–t–il ? Si oui que calcule–t–il ?

2) On se donne une matrice `m` à `p` lignes et `q` colonnes dont les éléments sont uniquement 0 et 1. On veut trouver des entiers $\mathtt{i_0} \in [\![0, \mathtt{p}]\!]$ et $\mathtt{j_0} \in [\![0, \mathtt{q}]\!]$ tels que $\mathtt{i_0} + \mathtt{j_0}$ soit maximum et $\forall \mathtt{i} \in [\![0, \mathtt{i_0}[\![, \forall \mathtt{j} \in [\![0, \mathtt{j_0}[\![, \mathtt{m_{i,j}} = 1$.

Pour simplifier on suppose la matrice non vide : $\mathtt{p} \geqslant 1$ et $\mathtt{q} \geqslant 1$ (il est inutile de le vérifier).

Les étudiantes Donadora, Euphrasie et Frédégonde proposent chacune le code suivant :

a) Quel résultat doit–on obtenir si $\mathtt{m_{0,0}} = 0$?

Les étudiantes Donadora, Euphrasie et Frédégonde proposent chacune le code suivant :

Code de Donadora :

```
def donadora (m) :
    (p,q) = (len(m) , len (m[0]))
    (i0, j0) = (0, 0)
    for i in xrange (p+1) :
        for j in xrange (q+1) :
            res = True
            for u in xrange (i) :
                for v in xrange (j) :
                    if m[u,v] == 0 : res = False
            if res and i+j > i0+j0 : (i0,j0) = (i,j)
    return (i0,j0)
```

Code d'Euphrasie :

```
def euphrasie (m) :
    (p,q) = (len(m),len(m[0]))
    (i0, j0) = (0, 0)
    while m[i0, j0] and i0 < p :
       i0 += 1
    while m[i0-1, j0] and j0 < q :
       j0 += 1
    return (i0, j0)
```

Code de Frédégonde :

```
def fredegonde (m) :
    (p,q) = (len(m),len(m[0]))
    (i0, imax, j0) = (p, p, 0)
    j = 0
    while j < q :
        i = 0
        while i < imax and m[i,j] :
            i += 1
        imax = i
        j += 1
        if i+j > i0+j0 : (i0,j0)=(i,j)
    return (i0,j0)
```

b) Parmi les codes de Donadora, d'Euphrasie et de Frédégonde, l'un est faux. Lequel ? Donner un exemple de matice pour laquelle on obtiendrait un résultat erroné.

c) Donner les complexités des deux codes restants à chaque fois sous la forme $\mathcal{O}(\mathtt{f}(\mathtt{p},\mathtt{q}))$.

d) Prouver l'algorithme de Frédégonde, en donnant un invariant de boucle précis et en précisant de façon formelle le rôle de `imax`.