MPSI2 2016 2017 Lycée Charlemagne

Ainpho

Vendredi 28 avril

Une nombre tel que 11 937 est un premier cyclique car il est premier, mais aussi ses "cycles" 19 371, 93 711, 37 119, 71 193 *(par permutation circulaire $\overrightarrow{abcde}$)*. Donnez moi tous les premiers cycliques à deux chiffres. [1 pt.] Ecrivez un script qui cherche tous les premiers cycliques à cinq chiffres. [4 pt.]

Que fait le script Mystère [3 pt.]

```
def Mystere(L1, L2):
....for a in L1:
........if not a in L2:
............return(False)
........L2.remove(a)
....return(len(L2)==0)
```

Sans rapport avec notre devoir :

Rébus : $\overline{\frac{1}{R^7.a}}$

Multipliez haut et bas par π.

Et squale

Ainpho

Un laboratoire d'analyses médicales est doté d'une base de données qui enregistre les clients, les analyses. Si un client vient pour plusieurs analyses, il est créé une fiche pour chaque analyse. Les tables s'appellent `Client`, `Analyse`, `Tarif`, `Medecin`.

IdClient	Secu	Nom	Prénom	DateNaissance	Sexe	Adresse	CodePostal	Ville	Téléphone	Mutuelle
C124	16211...	CHOQUET	Nicolas	1962-11-20	oui	15, Villa C.Monet	75019	PARIS	07....	MGEN
C125	27308...	DUVAL	Martine	1973-08-12	F	7, rue de Crimée	75019	PARIS		sans
C126	19507...	MDAYE	Lisimba	1995-07-23	M	5, rue du Cambodge	75020	PARIS		MMUA
C127	25706...	HUANG	Sophie	1957-06-13	F	3, rue des Archives	75003	PARIS		

IdAnalyse	Date	IdInfirmier	IdClient	Type	Labo	IdMedecin	DateRetraitPossible	DateRetraitEffectif
A521	2017-03-21	I23	C124	TSH	L7	M13	2017-03-23	
A522	2017-03-21	I23	C342	THC	L8	M15	2017-03-23	2017-03-25
A523	2017-03-21	I23	C342	TSH	L7	M15	2017-03-23	2017-03-25
A524	2017-03-21	I12	C165	HCG	L9	M23	2017-03-25	

Type	Prix	TauxSecu
THC	12	80
NFS	21	85

IdMedecin	Nom	Prenom	AdresseCabinet
M23	Claude	Fabienne	rue Ginoux
M24	Kerranou	Malika	CHU

♯1 Que fait cette requête ? [1 pt.]

```
SELECT Type, COUNT(*)
FROM Analyse
WHERE Date LIKE '2017-01-%'
GROUP BY Type
```

♯2 Ecrivez une requête qui vérifie si les individus déclarés de sexe masculin ont bien un numéro de sécurité sociale qui commence par 1. [1 pt.]

♯3 Ecrivez une requête pour sortir la liste des clients *(avec téléphone et adresse complète)* qui peuvent retirer leur dossier aujourd'hui. [1 pt.]

♯4 Ecrivez une requête pour sortir la liste des clients *(avec téléphone)* qui pouvaient retirer leur dossier avant mars 2017 et ne sont toujours pas passés. [1 pt.]

♯5 Ecrivez une requête pour connaître la liste des patients *(nom, prénom)* prélevés par l'infirmier 17 la semaine dernière, classée par ordre alphabétique. [1 pt.]

♯6 Que fait

```
SELECT Analyse.Type
FROM Client JOIN Analyse ON Client.IdClient=Analyse.IdClient
GROUP BY Analyse.IdInfirmier
WHERE Client.IdClient IN
........(SELECT Client
.........FROM Analyse
.........WHERE Type = 'THC')
```

1 pt.

♯7 Ecrivez une requête qui indique le coût total des analyses effectuées hier, et le coût total de la facturation Sécurité Sociale en tenant compte des taux de remboursement. 1 pt.

♯8 Ecrivez une requête qui indique toutes les analyses *(type, nom et prénom du patient)* de l'année 2016 des patients affiliés à la MGEN. 1 pt.

♯9 Sachant que les dosages HCG sont des tests de grossesse, écrivez une requête qui vérifie si certaines de ces analyses ont été prescrites de manière absurde. 1 pt.

♯10 Donnez la liste des clients du docteur Rémi Nguyen venus dans la laboratoire en 2016, avec pour chacun la liste des types d'analyse. 1 pt.

♯11 Donnez la liste des clients qui sont venus en 2016 avec des prescriptions d'au moins deux médecins différents. 1 pt.

♯12 Que fait

```
SELECT AVG(DATEDIFF(DateRetraitEffectif, DateRetraitPossible))
FROM Analyse JOIN Client ON Client.IdClient=Analyse.IdClient
WHERE NOT(Client.Ville IN ('PARIS', 'Paris'))
```

1 pt.

♯13 Ecrivez une requête pour connaître les clients inscrits dans la base de données mais qui n'ont fait aucune analyse depuis 2014 *(inclus)*. 1 pt.

Oral Centrale — Ainpho

◇1 On considère $n \geqslant 2$ joueurs, numérotés de 1 à n, participant à un tournoi où chacun affronte tous les autres, sans égalité possible dans une rencontre. On définit la matrice $A = (a_i^k)_{i\leqslant n, k\leqslant n}$ de la manière suivante : $a_i^i = 0$, $a_i^k = 1$ si i a gagné contre k et $a_i^k = -1$ si k a gagné contre i. Ecrivez une procédure renvoyant pour n donné une matrice de tournoi aléatoire. 2 pt.

◇2 Voici une procédure *(lourde, de complexité $n!$)* de calcul de déterminant, mais il manque une sous-procédure ; écrivez la :

```
def det(M) :
....if len(M)==0 :
........return(1)
....S, signe = 0, 1
....for i in range(len(M)) :
........S+=signe*M[i][0]*det(delRowCol(M,i,0))
....signe=-signe
....return(S)
```

3 pt.

◇3 On constate pour n impair que ces déterminants sont toujours nuls. Prouvez le. 1 pt.

◇4 $J_n = (1)_{i\leqslant n, k\leqslant n}$. Calculez $\det(J_n - I_n)$. 2 pt. Soient M et N deux matrices coefficients entiers telles que $M-N$ ait tous ses coefficients pairs. Montrez que $\det(M)$ et $\det(N)$ sont deux entiers de même parité. 1 pt. Déduisez que les matrices de tournois aléatoires en taille paire sont inversibles. 1 pt.

Solution : Pierre s'est assoupi.

Correction

Ainpho

Matrices de tournois. Ainpho

On doit créer des matrices telles que $\begin{pmatrix} 0 & -1- & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 & 1 \\ 1 & -1 & 0 & 1 & 1 \\ -1 & 1 & -1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 0 \end{pmatrix}$. Comment les remplir ?

On a deux démarches. Comme dans le cours, je vais préférer la méthode dynamique où l'on fabrique les lignes une à une.

```
def Alea(n) :
....M = [] #initialisation vide
....for i in range(n) : #ligne par linge
........L = [] #on commence avec une ligne vide
........for k in range(n) : #on avance case à case
............hum hum #là, il y a du travail
........M += L #la ligne est finie, on la colle
....return(M) #la matrice est finie, on la retourne
```

Il reste à voir comment on complète les lignes, sachant que la matrice doit être antisymétique. On ne va donc créer que les termes au dessus de la diagonale ; ceux qui sont au dessous sont obtenus par passage à l'opposé.

Quand `k` est plus grand que `i` *(fin de ligne)*, on tire un nombre au hasard *(quitte à avoir créé déjà une fonction qui choisit 1 ou -1)*.

Quand `k` est égal à `i`, on met `0`.

Quand `k` est plus petit que `i` *(début de ligne)*, on recopie au signe près un terme déjà créé : `-M[k][i]`.

Donc déjà, le créateur de `-1` ou `1` :

```
def pm() : #comme plus ou moins 1
....return(2*randrange(2)-1)
```

Pui la matrice elle même, suivant le schéma indiqué :

```
def Alea(n) :
....M = []
....for i in range(n) :
........L = []
........for k in range(n) :
............if k < i :
................L.append(-M[k][i])
............if k == i :
................L.append(0)
............if k > i :
................L.append(pm())
........M.append(L)
....return(M)
```

On peut aussi effectuer une création statique. On crée d'abord une matrice avec des `0`. On balaye ensuite la moitié supérieure, on y met des `1` ou `-1`, et on complète dans le même temps la moitié inférieure

```
def Alea(n) :
....M = [[0 for k in range(n)] for i in range(n)]
....for i in range(n) :
........for k in range(i-1) :
............M[i][k] = pm()
............M[k][i] = -pm()
....return(M)
```

Evidemment, la ligne `M[k][i] = -pm()` est une erreur. Elle relance un nouveau tirage aléatoire. Il faut reprendre le précédent : `a = pm()`
`M[i][k], M[k][i] =a,`
`-a`

Que fait la procédure

```
def det(M) :
....if len(M)==0 :
........return(1)
....S, signe = 0, 1
....for i in range(len(M)) :
........S+=signe*M[i][0]*det(delRowCol(M,i,0))
....signe=-signe
....return(S)
```

Elle prend en entrée une matrice. Si la matrice est de taille nulle, son déterminant vaut 1, le programme retourne 1 *(et n'exécute pas la suite).*
Si la matrice est de taille positive, on passe à la suite.
On initialise une somme `S` à `0` *(on l'incrémentera par* `S+=...` *)*, et un clignotant à `1` *(on le fera clignoter en* `sigme=-signe` *).*
On parcourt la première colonne de la matrice avec des `M[i][0]`.
Chacun est multiplié par le déterminant d'une matrice à laquelle il manque la ligne `i` et la colonne `0`, ce sont les cofacteurs.
On reconstruit donc la formule $\det(M) = \sum_{k=0}^{n-1} (-1)^k . m_i^k . cof_i^k$.

L'énoncé indexait ses matrices de 1 *à* n *tandis que Python le fait de* 0 *à* $n-1$*. Mais comme tout est visuel ou finalement calculatoire sur les déterminants, je ne pense pas que ce soit important qu'on décalle les indices.*

Il manque donc la procédure `delRowCol`[1] qui prend trois arguments : une matrice `M` et deux indices. On doit effacer la ligne d'indice `i` et la colonne d'indice `k`.
On peut relire les lignes une à une avec un indice `ii` qui avance *(on sautera le cas ii==i)* et pour chaque ligne, on lira les termes un par un grâce à un indice `kk` *(qui évitera la valeur* `k` *).*

```
def delRowCol(M, i, k) :
....MM = []
....for ii in range(len(M)) :
........LL = []
........if ii != i :
............for kk in range(len(M[ii])) :
................if kk != k :
....................LL.append(M[ii][kk])
............MM.append(LL)
....return(MM)
```

On peut aussi utiliser la méthode `pop` qui enlève un terme sur une liste (`L.pop(k)` efface l'élément d'indice `k` de la liste `L`).

[1] delete Row (=ligne) Column (=colonne)

```
def delRowCol(M,i,k) :
....MM = [L[ :] for L in M] #crée une copie de M
....M.pop(i) #on efface une ligne
....for L in MM : #on prend les lignes une à une
........L.pop(k) #on pop chaque lignes
....return(MM)
```

Efficace, non ?

Ces matrices de tournois sont antisymétries par construction. Elles vérifient ${}^tM = -M$. On passe au déterminant : $\det({}^tM) = \det(-M)$. On simplifie avec ce qu'on sait : $\det(M) = \det({}^tM) = (-1)^n.\det(M)$.
Pour n impair, on obtient $\det(M) = -\det(M)$ d'où $\det(M) = 0$.

On retient : les matrices antisymétriques de format impair sont non inversibles (déterminant nul). Les mauvais élèves ignorent ce résultat et sont ébahis qu'on leur demande de le prouver. Les mauvais élèves oublient que ce n'est vrai qu'en format impair. Les bons élèves voient/retiennent la preuve et tiennent à la fois formule, cadre d'application et preuve...

On écrit la forme de la matrice J_n : que des 1, puis de la matrice $J_n - I_n$: que des 1 sauf la diagonale nulle. On rédige la preuve avec la matrice de taille 5 :

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix}$$ on soustrait la première ligne à chacune des autres

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix} = \begin{vmatrix} 4 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{vmatrix}$$ on somme toutes les colonnes sur la première

La matrice obtenue est triangulaire, son déterminant est le produit des termes diagonaux : $\boxed{(n-1).(-1)^{n-1}}$

On prend deux matrices à coefficients entiers de termes généraux m_i^k et n_i^k. On suppose que pour tout couple $(i,\ k)$, m_i^k et n_i^k sont de même parité *(différence paire)*.

Pour tout i et tout σ on a $m_i^{\sigma(i)} = n_i^{\sigma(i)}\ mod\ 2$. On multiplie $\prod_{i=1}^{n} m_i^{\sigma(i)} = \left(\prod_{i=1}^{n} n_i^{\sigma(i)}\right)\ mod\ 2$ *(compatibilité des congruences et de la multiplication)*. On multiplie par la signature et on somme : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} n_i^{\sigma(i)}\right)\ mod\ 2$. Les deux déterminants ont la même parité.

Façon PSI : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} (n_i^{\sigma(i)} + 2.k_i^{\sigma(i)})\right)$ *et on développe.*

Façon PC : $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right) = \left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} (n_i^{\sigma(i)} + 2.k)\right)$ *et on développe (mais on n'a pas compris que ce k dépendait de la position, confus comme vous dans les variables).*

Façon MP : *dans la formule* $\left(\sum_{\sigma\in S_n} Sgn(\sigma).\prod_{i=1}^{n} m_i^{\sigma(i)}\right)$, *il n'y a que des sommes et produits, elles sont compatibles avec les congruences modulo* 2.
A vous de choisir la rédaction que vous préférez et estimez la plus claire.

On prend une matrice de tournoi aléatoire M de taille n. Les deux matrices $J_n - I_n$ et M sont à coefficients entiers, et ont terme à terme la même parité *(1 ou −1 hors de la diagonale, et 0 sur la diagonale).*
Les deux matrices ont le même déterminant modulo 2.
On a donc $\det(M) = (n-1).(-1)^{n-1}$ *mod* 2.
Si n est pair, $(n-1).(-1)^{n-1}$ est impair.
$\det(M)$ est donc impair. Il ne peut pas être nul. La matrice M est inversible.

Requêtes SQL dans un laboratoire d'analyses. Ainpho

On procèdera souvent à des jointures de tables, en demandant que IdClient soit le même sur les deux. On pourra utiliser `FROM Client JOIN Analyse ON Client.IdClient=Analyse.IdClient` *(ou* `FROM Client JOIN Analyse WHERE Client.IdClient=Analyse.IdClient` *qui revient au même).*
L'avantage de ce type d'organisation sur les bases de données : on n'est pas obligé de re-saisir à chaque analyse les coordonnées du client ; si il change une fois de téléphone, toutes les fiches d'analyse où il intervient sont mises à jour en même temps ; comme son nom et ses coordonnées sont saisis la première fois, on évite les erreurs quand le client revient (sinon par une majuscule mise en minuscule on croit qu'il y a un autre client).

```
SELECT Type, COUNT(*)
FROM Analyse
WHERE Date LIKE '2017-01-%'
GROUP BY Type
```

On prend la table analyse, on ne se préoccupe pas des noms des clients. On ne regarde que le type de prélèvement, et on compte. Comme on groupe, on ne va pas avoir THC, THC par exemple, mais THC 2. Et on ne s'intéresse qu'aux prélévemants de janvier 2017.
On sort les différents types de prélévements en janvier 2017, et pour chacun, on compte combien de prélévements de ce type ont été effectués.
Le résultat pourra être THC 23, Serum 13, Numer 21, TSH 41, HCG 8,
Evidemment, il faudra être connaisseur pour savoir ce qui là dedans correspond à des dosages Thyroïde, Urémie, Albumine, Plaquettes,...

Ecrivez une requête qui vérifie si les individus déclarés de sexe masculin ont bien un numéro de sécurité sociale qui commence par 1.

```
SELECT Nom, Prenom, Sexe, Secu
FROM Clients
WHERE Sexe='M' AND NOT(Secu LIKE '1%')
```

Ecrivez une requête pour sortir la liste des clients (avec téléphone et adresse complète) qui peuvent retirer leur dossier aujourd'hui.

```
SELECT Client.Nom, Client.Telephone, Client.Adresse, Client.Code, Client.Ville
FROM Client JOIN Analyse
....ON Client.IdClient = Analyse.IdClient
WHERE Analyse.DateRetraitPossible='2017-04-..'
```

Ecrivez une requête pour sortir la liste des clients *(avec téléphone)* qui pouvaient retirer leur dossier avant mars 2017 et ne sont toujours pas passés.

```
SELECT Client.Nom, Client.Telephone
FROM Client JOIN Analyse
....ON Client.IdClient = Analyse.IdClient
WHERE Analyse.DateRetraitPossible<'2017-03-01' AND Analyse.DateRetraitEffectif=''
```

On teste si le champ `DateRetraitEffectif` est réduit à une chaîne vide *(ouvrez les guillemets, fermez tout de suite les guillemets).* Suivant le type de base de données, un champ vide est fait d'une chaine vide ou de `NULL`. dans ce dernier cas, on testera `Analyse.DateRetraitEffectif IS NULL`.

Ecrivez une requête pour connaître la liste des patients *(nom, prénom)* prélévés par l'infirmier 17 la

semaine dernière, classée par ordre alphabétique.

```
SELECT Client.Nom, Client.Prenom
FROM Client JOIN Analyse
....ON Client.IdClient = Analyse.IdClient
WHERE Analyse.IdInfirmier = '17' AND Analyse.Date BETWEEN '2017-..-..' AND
'2017-..-..'
ORDER BY Client.Nom
```

Que fait

```
SELECT Analyse.Type, Client.Nom
FROM Client JOIN Analyse ON Client.IdClient=Analyse.IdClient
GROUP BY Analyse.IdInfirmier
WHERE NOT(Analyse.Type='THC') AND Client.IdClient IN
........(SELECT IdClient
.........FROM Analyse
.........WHERE Type = 'THC')
```

Avec `SELECT IdClient FROM Analyse WHERE Type = 'THC'` on sort la liste de tous les patients qui ont fait au moins une fois un dosage THC *(TétraHydroCannabinol ?)*.
On regarde ensuite pour ces clients les autres analyses qu'ils ont pu faire. On affiche le type d'analyse, leur nom, et on trie en fonction de l'identité des infirmiers ayant effectué le prélèvement.

Ecrivez une requête qui indique le coût total des analyses effectuées hier, et le coût total de la facturation Sécurité Sociale en tenant compte des taux de remboursement.

```
SELECT SUM(Tarif.Prix), SUM(Tarif.Prix*Tarif.Taux/100)
FROM Analyse JOIN
....Analyse.Type = Tarif.Type
WHERE Analyse.Date='2017-04-..'
```

Ecrivez une requête qui indique toutes les analyses *(type, nom et prénom du patient)* de l'année 2016 des patients affiliés à la MGEN.

```
SELECT Analyse.Type, Client.Nom, Client.Prenom
FROM Analyse JOIN Client
....ON Analyse.IdClient = Client.IdClient
WHERE Client.Mutuelle='MGEN' AND Analyse.Date LIKE '2016%'
```

Sachant que les dosages HCG sont des tests de grossesse, écrivez une requête qui vérifie si certaines de ces analyses ont été prescrites de manière absurde.

```
SELECT Client.Nom, Analyse.IdInfirmier
FROM Analyse JOIN Client
....ON Analyse.IdClient = Client.IdClient
WHERE Client.Sexe = 'M' OR NOT (Client.DateNaissance BETWEEN ... AND ...)
....AND Analyse.Type = 'HCG'
```

Je vous laisse choisir les dates de naissance à partir desquelles et en delà desquelles il est absurde de passer un test de grossesse [2].

Donnez la liste des clients du docteur Rémi Nguyen venus dans la laboratoire en 2016, avec pour chacun la liste des types d'analyse.

[2] de 7 à 77 ans on ne fait pas que lire Tintin ?

```
SELECT Client.Nom, Analyse.Type
FROM Client JOIN Analyse JOIN
....ON Client.IdClient = Analyse.IdClient
WHERE Analyse.Date LIKE '2016%' AND Analyse.IdMedecin IN
....(SELECT IdMedecin
.....FROM Medecin
.....WHERE Nom='Nguyen' AND 'Prenom'=Rémi')
GROUP BY Client.Nom
```

Donnez la liste des clients qui sont venus en 2016 avec des prescriptions d'au moins deux médecins différents.

Pour un patient, on peut compter le nombre de médecins éyant prescrit des analyses

```
SELECT COUNT(distinct IdMedecin)
FROM Analyse
WHERE IdClient=...
```

On peut ensuite effectuer un test et obtenir la liste des patients

```
SELECT Ana1.IdClient
FROM Analyse AS Ana1
WHERE SELECT COUNT(distinct Ana2.IdMedecin)      >1
      FROM Analyse AS Ana2
      WHERE Ana2.IdClient=Ana1.IdClient
```

On ajoute ensuite la condition sur `Ana1.Date`.

```
SELECT AVG(DATEDIFF(DateRetraitEffectif, DateRetraitPossible))
FROM Analyse JOIN Client ON Client.IdClient=Analyse.IdClient
WHERE NOT(Client.Ville IN ('PARIS', 'Paris'))
```

Ecrivez une requête pour connaître les clients inscrits dans la base de données mais qui n'ont fait aucune analyse depuis 2014 *(inclus)*.

```
SELECT Nom
FROM Client
WHERE NOT IdClient IN
....(SELECT IdClient
.....FROM Analyse
.....WHERE Date > '2014-12-31')
```

Une procédure mystère. Ainpho

La procédure mystère prend deux listes `L1` et `L2` et les compare.

Elle regarde un par un les éléments de `L1` et regarde si ils sont dans une liste `L2` *(qu'on détruit peu à peu)*.

Si il n'y est pas, on sort directement.

Si il y est, on l'enlève de `L2` et on passe au suivant.

A la fin, on regarde si on a épuisé `L2`.

Si c'est le cas, c'est que les deux listes contenaient les mêmes termes, pas forcément dans le même ordre.

Nombres premiers cycliques. Ainpho

Les nombres premiers cherchés sont de la forme ab avec ba tout aussi premier.

Par exemple 17 et 71.

On s'interdit les nombres premiers commençant par 2, 4, 5, 6 et 8 car par renvsersement on obtient un nombre pair ou multiple de 5, non premier.

Bref, il reste

11	13	17	37	79
	31	71	73	97

On décompose le problème en morceaux *(principe de base en programmation)*.
On crée déjà le test de primalité basique :

```
def premier(p) :
....for div in range(2, p) :
........if p%div == 0 :
............return(False)
....return(True)
```

Je reviens sur votre erreur bête :

```
def premier(p) :
....for div in range(2, p) :
........if p%div == 0 :
............return(False)
........else :
............return(True)
```

Cette chose est une erreur grossière que je n'arrivé même pas à comprendre de votre part *(sauf à aboutir à une triste conclusion)*. En effet, dans cette procédure, dès le premier test `k=2`, vous sortez par `True` ou `False`. Vous n'aurez jamais l'occasion de faire les suivants.Ce qu'il faut faire, c'est faire ce que vous faites. Vous testez les diviseurs possibles un par un : `div=2`, `div=3`, `div=4` et ainsi de suite, a priori jusqu'à `div=p-1`.
Si un d'entre eux divise `p`, on sort tout de suite.
Sinon, on continue.
Si on a fait le tour de tous les diviseurs possibles, sans sortir par `False`, c'est que le nombre est premier, et vous sortez par `True`.
Ensuite, il faut passer de `abcde` à `eabcd`.
On divise `abcde` par 10 *(euclidienne)*, on a `abce`.
On récupère `e` *(c'est `abcde%10`)*. On le met en tête par multiplication par 10 000 : e0000+abcd=eabcd.

```
n = (n/10) + 1 000*(n%10)
```

On met en boucle cinq fois cette transformation, et on teste un par un les nombres obtenus.
Si l'un d'entre eux n'est pas premier, on sort.
Si ils sont tous premiers, on a fait le tour, on valide.

```
def SuperTest(N) : #il faut que N n'ait que 5 chiffres
....for k in range(5) : #on doit tester cinq nombres
........if not(premier(N)) : #si l'un est composé
............return(False) #on sort c'est mort
........n = (n/10)+1 000*(n%10) #on fait tourner
....return(True) #tous étaient premiers, on valide
```

Il ne reste plus qu'à utiliser cette procédure pour les entiers de 10 000 (inclus) à 1 000 000 (exclu).

```
for N in range(10000, 100000) :
....if SuperTest(N) :
........print(N)
```

On va ainsi croiser

11 939	19 391	19 937	37 199	39 119

Si on veut travailler sur des nombres dont le nombre de chiffres n'est pas imposé, il faut ajouter dans `SuperTest` quelques lignes :

```
NN = N
c = 0
puiss = 1
while NN != 0 :
....NN=NN/10
....c += 1
....puiss *=10
```

et faire une boucle pour `k` de `0` à `c-1`.