Séance du 23.01. - Logique

mardi 16 janvier 2024 09:56

Exercice 24.
Montrer que pour tout entier $n \geq 1$, $4^n - 1$ est multiple de 3.

Montrons que : $4^n - 1 \equiv 0 \ [3]$

$4^n - 1 \equiv 1^n - 1 \ [3]$ car $4 \equiv 1 \ [3]$

$\equiv 1 - 1 \equiv 0 \ [3]$

donc $4^n - 1$ est un multiple de 3, pour tout $n \in \mathbb{N}$.

Autre rédaction possible :

On a : $4 \equiv 1 \ [3]$

d'où $4^n \equiv 1^n \ [3]$

$4^n - 1 \equiv 1^n - 1 \equiv 1 - 1 \equiv 0 \ [n]$

- Montrons que pour tout $n \in \mathbb{N}$, $50^n - 1$ est multiple de 7

$50 \equiv 1 \ [7]$ car $50 = 7 \times 7 + 1$

$50^n \equiv 1^n \ [7]$

$50^n - 1 \equiv 1^n - 1 \equiv 1 - 1 \equiv 0 \ [7]$

Pour tout $n \in \mathbb{N}$, $50^n - 1$ est un multiple de 7.

Exercice 25.
Sur les billets de banque en euros figure un nombre de 11 chiffres décimaux précédé d'une lettre. En remplaçant cette lettre par son rang dans l'alphabet (de 1 à 26), on obtient ainsi un nombre de 12 ou 13 chiffres.

Le billet ne peut être authentique que si le reste de ce nombre dans la division euclidienne par 9 est 8.

a) Les billets portant le numéro U57794585675 et S00212913867 peuvent-ils être authentiques ?

b) Un billet authentique porte le numéro T2303557409x (il y a une tache sur le dernier chiffre). Que vaut x ?

c) Le nombre figurant sur un billet authentique est 16122340243 mais la lettre est effacée. Quelle peut-être cette lettre ?

a) • U57794585675 → 2157794585675 $\equiv 8 \ [9]$

Ce billet est authentique

• S00212913867 → 1900212913867 $\equiv 4 \ [9]$

Ce billet est un faux billet.

b) T2303557409x → 2023035574090x

On a $2023035574090 \equiv 4 \ [9]$

c) 16122340243

On a $2023035574090 \equiv 4 \ [9]$

c) 16122340243

Application Python :

Codage ASCII. Il existe des fonctions en Python qui permettent de passer des caractères aux numéros correspondants :

- ord(caractere) : Donne le code ASCII du caractère.
- chr(numero) : Donne le caractère correspondant au code ASCII.

Ecrire une procédure valid(N) qui affiche l'authenticité du billet associé au numéro N.

Procédure pour vérifier la validité d'un billet

```
def valid(N):
    lettre = N[0]
    nombre_a_coller = ord(lettre) - 64
    nombre_en_caractere = str(nombre_a_coller)
    Numero = int(nombre_en_caractere + N[1:12])
    reste = Numero % 9
    if reste == 8 :
        print("Billet authentique")
    else :
        print("Faux billet")
```

Procédure qui détermine la lettre manquante

```
def chercher_lettre(N):
    i = 65
    continuer = True
    while continuer and i < 91 :
        nb = i - 64
        nombre = str(nb) + N
        nombre = int(nombre)
        if nombre % 9 == 8:
            print("La lettre est : ",chr(i))
            print("Le numbéro du billet est : ",chr(i) + N)
            continuer = False
        i = i + 1
```

Chapitre 4

Logique propositionnelle

1 Notion de proposition

Définition : proposition

Une proposition est un énoncé qui a un sens et pour lequel on peut dire avec certitude qu'il est vrai ou faux. On dit qu'on peut lui associer une *valeur de vérité*. Cette valeur peut se noter VRAI ou FAUX mais on peut aussi choisir de la noter 0 (pour faux) ou 1 (pour vrai).

Exemples

- « $2 + 2 = 5$ » est une proposition fausse.
- « $2 + 2 \equiv 1 \ [3]$ » est une proposition vraie.

2 Connecteurs logiques

Définition : négation d'une proposition

L'opérateur de négation se note avec une barre ¯. C'est un opérateur *unaire*, c'est à dire qu'il s'applique à *une* proposition.
Il est défini par la table de vérité suivante :

P	$\overline{P}$
0	1
1	0

(not en Python)

$\overline{P}$ se lit « non P ».

Définition : conjonction de deux propositions

L'opérateur de *conjonction* correspond au **and** de PYTHON, au **And** de VISUAL BASIC, au **&&** de C++.
Il se note ∧, c'est un opérateur *binaire* car il s'applique à deux propositions.
Il est défini par la table de vérité suivante :

P	Q	P ∧ Q
0	0	0
0	1	0
1	0	0
1	1	1

P ∧ Q se lit « P et Q » et n'est vrai que si P est vrai et Q aussi.

Définition : disjonction de deux propositions

L'opérateur de *disjonction* correspond au **or** de PYTHON, au **Or** de VISUAL BASIC, au **||** de C++. Il se note ∨, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ∨ Q
0	0	0
0	1	1
1	0	1
1	1	1

P ∨ Q se lit « P ou Q » et est vrai dès que P est vrai ou Q est vrai. (au moins 1 vrai)

Définition : équivalence de deux propositions

Il se note ⇔, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ⇔ Q
0	0	1
0	1	0
1	0	0
1	1	1

P ⇔ Q se lit « P équivaut à Q » et n'est vrai que si P et Q ont la même valeur de vérité.

Définition : implication

Il se note ⇒, c'est un opérateur *binaire*. Il est défini par la table de vérité suivante :

P	Q	P ⇒ Q
0	0	1
0	1	1
1	0	0
1	1	1

P ⇒ Q se lit « P implique Q ».

- Quand P est fausse, P ⇒ Q est vraie : « le faux implique n'importe quoi ».
- Quand P est vraie, P ⇒ Q n'est vraie que si Q est aussi vraie : « le vrai n'implique que le vrai ».

Exercice 54

On note P et Q les affirmations suivantes :
P = « Paul aime le foot »
Q = « Paul aime les maths »
Représenter les affirmations suivantes sous forme symbolique en utilisant P, Q et des connecteurs logiques.

- A = « Paul aime le foot mais pas les maths »
- B = « Paul n'aime ni le foot, ni les maths »
- C = « Paul aime le foot ou il aime les maths et pas le foot »
- D = « Paul aime les maths et le foot ou il aime les maths mais pas le foot »

- $A = P \wedge \bar{Q}$
- $B = \bar{P} \wedge \bar{Q}$
- $C = P \vee (\bar{P} \wedge Q)$
- $D = (P \wedge Q) \vee (\bar{P} \wedge Q)$