



Le but du projet est de créer un programme qui permette calculer la clé de contrôle d'un code EAN13 et de tracer le code-barre correspondant lisible par une douchette.

Ce code utilisé sur tous les articles de consommation courante permet le codage de 12 chiffres, le treizième chiffre étant une clé de contrôle qu'il faudra générer (2 chiffres pour le code pays ou code système, 5 chiffres pour l'identificateur de société, 5 chiffres pour l'identificateur d'article, 1 chiffre pour la clé de contrôle)

### Calcul de la clé de contrôle :

Chiffres « n » du code à barres	4	7	1	9	5	1	2	0	0	2	8	8
Pondérations « p »	1	3	1	3	1	3	1	3	1	3	1	3
Résultats « n » × « p »	4	21	1	27	5	3	2	0	0	6	8	24

1. Pour les poids on alterne les valeurs 1 et 3.
2. On calcule ensuite la somme des résultats.
3. On calcule le reste de la division par 10 de la somme précédemment calculée :
  - si le reste de la division est égal à 0, alors la clé est 0,
  - sinon, on ôte à 10 le reste ainsi trouvé : **Clé = 10 - Reste.**

La somme vaut dans cet exemple  $4+21+1+27+5+3+2+0+0+6+8+24 = 101$ , le reste de la division par 10 est 1 ;

**La clé vaut donc  $10-1 = 9$  et le code EAN 13 complet est 471-9-5120-0288-9**

*[https://fr.wikipedia.org/wiki/EAN\\_13](https://fr.wikipedia.org/wiki/EAN_13)*

**Système d'encodage :**

Un code-barre sert à coder un nombre, il est formé de barres et d'espaces appelés « modules ». Pour la suite nous symboliserons le module barre par "1" et le module espace par "0".

- Le code-barre est formé de 95 modules.
- Les marques de début et de fin du code-barre utilisent chacun les 3 modules : "101".
- La marque de séparation placée au milieu du code utilise 5 modules : "01010".
- Les barres des séparateurs sont plus longues que les autres barres.
- 1 chiffre est codé sur 7 modules **selon l'une des 3 tables ci-dessous :**

chiffre	Table A	Table B	Table C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Le premier chiffre du code EAN n'est pas codé mais il détermine les tables à utiliser pour le codage des 6 chiffres suivants.

Valeur Chiffre 1	Table à choisir pour					
	chiffre 2	chiffre 3	chiffre 4	chiffre 5	chiffre 6	chiffre 7
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Les 6 derniers chiffres utilisent la table C.

<https://grandzebu.net/informatique/codbar/ean13.htm>

### Voici le corps principal du programme:

```
def lance():
    ean=""
    while(not verif(ean,12)):
        ean=input('Saisir les 12 premiers
chiffres :')
    nean=int(ean)
    ean13=ean+str(calculclef(ean))
    print(ean13)
    md13=chiffrage(ean13)
    print(md13)
    trace(md13)
```

### Voici le résultat attendu:

Python input

Saisir les 12 premiers chiffres :

OK Cancel



## Coups de pouce :

Utiliser des listes pour stocker les tables : « créer liste »

```
TabA=['0001101', '0011001', '0010011', '0111101', '0100011', '0110001', '0101111', '0111011', '0110111', '0001011']
TabB=['0100111', '0110011', '0011011', '0100001', '0011101', '0111001', '0000101', '0010001', '0001001', '0010111']
TabC=['1110010', '1100110', '1101100', '1000010', '1011100', '1001110', '1010000', '1000100', '1001000', '1110100']
chif1=['AAAAAA', 'AABABB', 'AABBAB', 'AABBBA', 'ABAABB', 'ABBAAB', 'ABBBAA', 'ABABAB', 'ABABBA', 'ABBABA']
```

```
from PIL import Image
from PIL import ImageDraw
def trace(module):
    HAUTEUR= 240
    LARGEUR = 95*4
    LARGEUR_BARRE = 4
    im = Image.new("RGB", (LARGEUR, HAUTEUR), "white")
    draw = ImageDraw.Draw(im)
    i=0
    for chiffre in module:
        lg=200
        if i<3 or i>91 or (i>44 and i<50): lg=240
        if chiffre == "1":
            draw.rectangle(((LARGEUR_BARRE *i,0),(LARGEUR_BARRE+LARGEUR_BARRE*i -1,lg)),fill="black",)
        elif chiffre == "0":
            draw.rectangle(((LARGEUR_BARRE *i,0),(LARGEUR_BARRE+LARGEUR_BARRE*i -1,lg)),fill="white",)
        i=i+1
    im.show()
```

```
def chiffrage(ean13):
    bord="101"
    separateur="01010"
    module=bord
    for i in range(13):
        if i==0:
            ref=int(ean13[i])
        elif i<7:
            if chif1[ref][i-1]=='A':
                module=module+TabA[int(ean13[i])]
            else:
                module=module+TabB[int(ean13[i])]
        if i==6:
            module=module+separateur
        else:
            module=module+TabC[int(ean13[i])]
    module=module+bord
    return(module)
```

```
def calculclef(ean):
    clef=0
    for i in range(12):
        if(i%2==0):
            clef=clef+int(ean[i])
        else:
            clef=clef+int(ean[i])*3
    if(clef%10==0):
        clef=0
    else :
        clef=10-clef%10
    return(clef)
```

```
def verif(ean,l):
    if len(ean)==l:
        return(True)
    else :
        return(False)
```

```
def chiffrage(ean13):
    bord="101"
    separateur="01010"
    module=bord
    for i in range(13):
        if i==0:
            ref=int(ean13[i])
        elif i<7:
            if chif1[ref][i-1]=='A':
                module=module+TabA[int(ean13[i])]
            else:
                module=module+TabB[int(ean13[i])]
        if i==6:
            module=module+separateur
        else:
            module=module+TabC[int(ean13[i])]
    module=module+bord
    return(module)
```

```
def calculclef(ean):
    clef=0
    for i in range(12):
        if(i%2==0):
            clef=clef+int(ean[i])
        else:
            clef=clef+int(ean[i])*3
    if(clef%10==0):
        clef=0
    else :
        clef=10-clef%10
    return(clef)
```

```
def verif(ean,l):
    if len(ean)==l:
        return(True)
    else :
        return(False)
```