Arithmétique : divisibilité et congruences
Problème de recherche : la clé des codes ISBN

Un code ISBN-13 est une suite de 13 chiffres $d_1 d_2 \dots d_{13}$. On calcule la somme pondérée

$$S = d_1 + 3d_2 + d_3 + 3d_4 + \cdots + 3d_{12} + d_{13}$$

(poids 1 pour les rangs impairs, 3 pour les rangs pairs). Le code est **valide** si $S \equiv 0 \pmod{10}$; le dernier chiffre d_{13}, appelé clé, est choisi pour que ce soit le cas.

La mission

1. Écrire une fonction `valide(code)` qui prend une liste de 13 chiffres et renvoie `True` si le code est valide. Vérifier que 978-2-07-036822-8 est valide.
2. Un chiffre du code 978-2-07-0□6822-8 est illisible. Le retrouver.
3. Démontrer qu'une erreur portant sur un seul chiffre est toujours détectée. On pourra remarquer que $3 \times 7 \equiv 1 \pmod{10}$.
4. On échange deux chiffres voisins distincts a et b. Démontrer que l'erreur passe inaperçue si et seulement si $a - b = 5$ ou $b - a = 5$. Quelle proportion des échanges de deux chiffres voisins distincts est détectée ?

Pour aller plus loin

Avec Python, lister les échanges de deux chiffres voisins distincts du code 978-2-07-036822-8 qui ne sont pas détectés. Trouver un code valide commençant par 978-2-07 pour lequel un tel échange n'est pas détecté.

Coups de pouce

Bloqué ? Des indices progressifs sont disponibles en ligne. N'en ouvrir qu'un à la fois, et seulement après avoir cherché !

Éléments de correction

1. Tester un code

```
def valide(code):
    S = 0
    for i in range(13):
        if i % 2 == 0:      # rangs 1, 3, ..., 13
            S = S + code[i]
        else:
            S = S + 3 * code[i]
    return S % 10 == 0

print(valide([9,7,8,2,0,7,0,3,6,8,2,2,8]))
```

On obtient `True` : $S = 9+21+8+6+0+21+0+9+6+24+2+6+8 = 120 \equiv 0 \pmod{10}$.

2. Le chiffre illisible

Le chiffre manquant x est le 8e, de poids 3. La somme des autres termes vaut 111 : il faut $111 + 3x \equiv 0 \pmod{10}$, soit $3x \equiv -1 \equiv 9 \pmod{10}$. En multipliant par 7 : $21x \equiv 63$, soit $x \equiv 3 \pmod{10}$. Le chiffre est **3** (on retrouve le code de la question 1). On peut aussi tester les dix valeurs avec la fonction `valide`.

3. Une erreur sur un chiffre

Si le chiffre d, de poids $p \in \{1;3\}$, est remplacé par d', la somme devient $S' = S + p(d' - d)$. L'erreur passe inaperçue si $S' \equiv 0$, c'est-à-dire $p(d' - d) \equiv 0 \pmod{10}$. Si $p = 3$, on multiplie par 7 : $21(d' - d) \equiv 0$, donc $d' - d \equiv 0 \pmod{10}$. Dans les deux cas, 10 divise $d' - d$; comme $-9 \leqslant d' - d \leqslant 9$, on a $d' = d$. Une erreur sur un chiffre est **toujours détectée**.

4. Échange de deux chiffres voisins

Deux rangs voisins ont des poids 1 et 3. Si a (poids 1) et b (poids 3) sont échangés, la somme passe de $a + 3b$ à $b + 3a$ pour ces deux termes : elle varie de $2(a - b)$ (et de $2(b - a)$ dans l'autre cas). L'erreur passe inaperçue si et seulement si 10 divise $2(a - b)$, c'est-à-dire 5 divise $a - b$. Comme $a \neq b$ et $-9 \leqslant a - b \leqslant 9$: $a - b = \pm 5$.

Il y a $10 \times 9 = 90$ couples $(a;b)$ de chiffres distincts ; 10 vérifient $|a - b| = 5$: $(0;5)$, $(1;6)$, $(2;7)$, $(3;8)$, $(4;9)$ et les couples inverses. Donc $\frac{80}{90} \approx 89\,\%$ des échanges sont détectés.

Pour aller plus loin

```
c = [9,7,8,2,0,7,0,3,6,8,2,2,8]
n = 0
for i in range(12):
    if c[i] != c[i+1]:
        e = c[:]                    # copie
        e[i], e[i+1] = e[i+1], e[i]
        if valide(e):
            n = n + 1
```

```
print(n)
```

Le programme affiche 0 : les chiffres voisins de ce code ne diffèrent jamais de 5, tous les échanges sont détectés.
Pour piéger le contrôle, il faut deux chiffres voisins dont la différence vaut 5. Par exemple, 978-2-07-050822-8 est valide ($S = 120$) ; en échangeant le 0 et le 5 voisins, on obtient 978-2-07-500822-8, encore valide ($S = 110$).
Enrichissement : l'ancien ISBN-10 utilisait des poids $10, 9, \ldots, 1$ et des calculs modulo 11, qui détectent aussi tous les échanges de chiffres voisins.

Ce qu'on a utilisé du chapitre

— Congruences modulo 10 ; compatibilité avec l'addition et la multiplication.
— Inverse modulo 10 : $3 \times 7 \equiv 1 \pmod{10}$.