Préparation des écrits

- Commentez vos programmes !

  Expliquez l'objectif de chaque étape

  Introduisez les noms utilisés

  Proposez des invariants de boucle

- Choisissez des noms clairs pertinents

  réserver `i`/`j` aux entiers de boucle

  éviter a, b, c... sauf s'ils apparaissent dans l'énoncé

  respecter les notations du sujet

- Décomposez en sous-fonctions si c'est utile

- Soignez l'indentation

Rigueur de rédaction

- Pas de notations mathématiques !

  $\frac{x}{2} \to$ `x/2`, $2n \to$ `2*n`, $i \geqslant 0 \to$ `i>=0`, $x^2 \to$ `x**2`

- Respecter scrupuleusement la syntaxe

  `[ 0, 1 ]` et non `[ 0 ; 1 ]`

  n'oubliez pas les :

  n'oubliez pas les parenthèses pour `range`, `len`, etc.

- Attention aux capitales (**`if`** et non `If`, **`True`** et non `true`)

Règles concernant les fonctions

- Respecter scrupuleusement les signatures données par le sujet

- Les arguments ne doit pas être changés sans avertissement sauf si c'est demandé

  Au besoin, commencer par les copier (`lst2 = lst` n'est pas une copie)

  Sinon, *préciser qu'ils sont modifiés et en tenir compte ensuite*

- On ne dispose que des arguments et des valeurs globales **définies par le sujet**

  S'il y a un n dans le sujet non fourni, l'extraire d'un argument

Erreurs courantes

- Vérifiez soigneusement les bornes des `range`
- Cherchez les situations où il pourrait y avoir débordement
- $\frac{a}{bc}$ s'écrit « `a/(b*c)` » et non « `a/b*c` »
- La négation de « `c1<eps and c2<eps` » est « `c1>=eps or c2>=eps` »

Du bon usage des booléens

Éviter	Préférer
`if XXX == True :`	`if XXX :`
`if XXX == False :`	`if not XXX :`
`if (X in L) == False :`	`if X not in L :`

`if cond :`
`    x = x`
`else :`
`    x = y`
	`if not cond :`
	`    x = y`

`if cond :`
`    return True`
`else :`
`    return False`
	`return cond`

Recommandations de style

Si une opération est à effectuer au début de chaque itération

mettre le code... au début de l'itération !

Éviter	Préférer

`x = 0`
`for ... in ... :`
`    ...`
`    ...`
`    x = 0`
	`for ... in ... :`
	`    x = 0`
	`    ...`
	`    ...`

Recommandations de style

Dans les expressions numériques, le code s'adapte à ce que l'on veut dire :

Éviter	Préférer
`int(x/10)`	`x//10`
`x - 10*(x//10)`	`x%10`
`x//1`	`int(x)`

Recommandations pour l'efficacité des programmes

On ne multiplie pas inutilement les appels à une fonction

Éviter	Préférer

`if f(x) > m :`
`    m = f(x)`
	`res = f(x)`
	`if res > m :`
	`    m = res`

`L[0] = L[0] + f(x)[0]`
`L[1] = L[1] + f(x)[1]`
	`res = f(x)`
	`L[0] = L[0] + res[0]`
	`L[0] = L[0] + res[1]`

Recommandations pour l'efficacité des programmes

On profite de ce qui a été calculé à l'étape i pour l'étape $i+1$

Par exemple, si (u_n) est une suite définie par $u_{n+1} = f(u_n)$

Et `U(i)` une fonction retournant (u_n) en temps linéaire en i

Éviter	Préférer

`i = 0`
`while U(i) < 100 :`
`    i = i + 1`
	`i, u = 0, U(0)`
	`while u < 100 :`
	`    i, u = i+1, f(u)`

(en $O(n^2)$)	(en $O(n)$)

Stuctures de contrôle autorisées

- **`if`** `...` : `...` est *toujours* autorisé

  avec **`else`** et **`elif`** si besoin

- **`for`** `...` **`in`** `...` : `...` est *toujours* autorisé

  le sujet peut imposer de n'itérer que sur des `range(...)`

  on évitera **`break`** *si possible*, et encore plus **`continue`**

  on proscrit totalement **`else`** pour un **`for`**/**`while`**

- **`while`** `...` : `...` est *toujours* autorisé

  avec mêmes restrictions que pour un **`for`**

Modules

- Il faut savoir charger un module

- Pour les fonctions de modules, on vous donnera une documentation

- L'utilisation de modules non cités par le sujet est proscrite

  Exception éventuellement possible pour `math`

  `x**0.5` remplace `math.sqrt(x)`

  $\lfloor \frac{a}{b} \rfloor$ pour $a, b \in \mathbb{Z}^2$ s'écrit `a // b`

  `int(x)` peut remplacer `math.floor(x)`

Listes

Un sujet peut fortement limiter ce qui est autorisé sur des listes

- `len(L)` est toujours autorisé
- `L[i]` peut être restreint à $i \geqslant 0$

  on remplacera `L[-k]` par `L[len(L)-k]`
- `L.append(elem)` et `L.pop()` sont souvent permis

  attention, `L.pop(i)` peut ne pas l'être !

  on évitera `L += [ elem ]`
- `elem in L` est fréquemment interdit

Listes

- `+` et `*` sur des listes peuvent être proscrits (XENS)
- `for elem in L:` peut parfois l'être
- Les slices peuvent ne pas être autorisés
- Les compréhensions peuvent ne pas être autorisées
- `L1 == L2` peut ne pas être autorisé
- `max(L)`, `sum(L)`, `L.insert(i, elem)`, `L.extend(iterable)`, `L.count(elem)`, `L.index(elem)`, `L.remove(elem)` sont très souvent à réécrire