Initiation à l'Algorithmique

BTS CIEL - Option Informatique et Réseaux

Co-Intervention Mathématiques - STI

Mme MASSET - M. NEGGAOUI

PÔLE D'ACTIVITÉ	COMPÉTENCE	OBJECTIF PÉDAGOGIQUE	CONNAISSANCES
Valorisation de la donnée et cybersécurité - **U6**	Coder - C08	Découvrir et appliquer les bases de l'algorithmique pour résoudre un problème simple	Langages de développement et IDE **Niveau 4**

SÉANCE 1

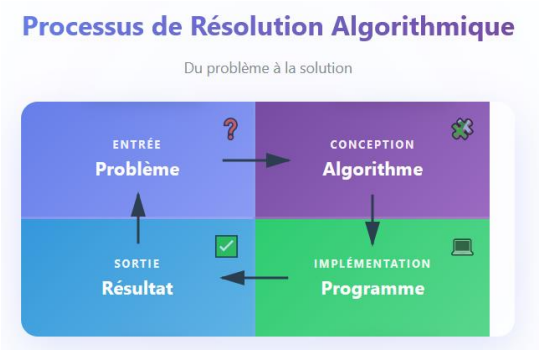
Introduction & Structures

1. Qu'est-ce qu'un algorithme?

Définition : Un algorithme est une suite d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat.

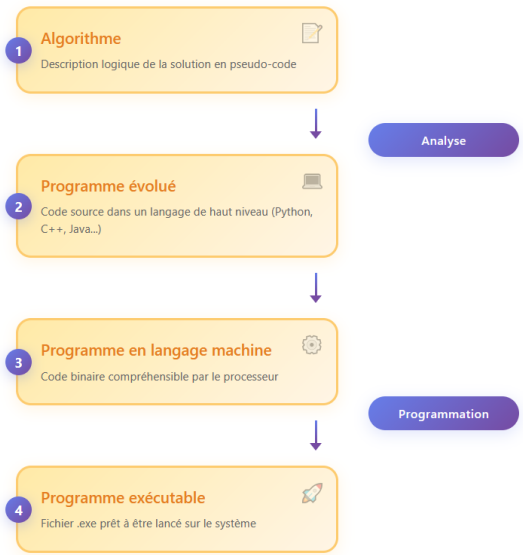
Un algorithme est un ensemble de règles ayant les 5 caractéristiques suivantes :

- Il est fini et se termine après un nombre fini d'opérations.
- Il est défini sans ambiguïté.
- S'il y a des données en entrée, leur type doit être précisé.
- Il doit avoir au moins un résultat.
- Il doit être effectif : toutes les opérations doivent pouvoir être effectuées exactement et dans un temps fini.



2. Algorithme et programmation

Étapes de mise au point d'un programme



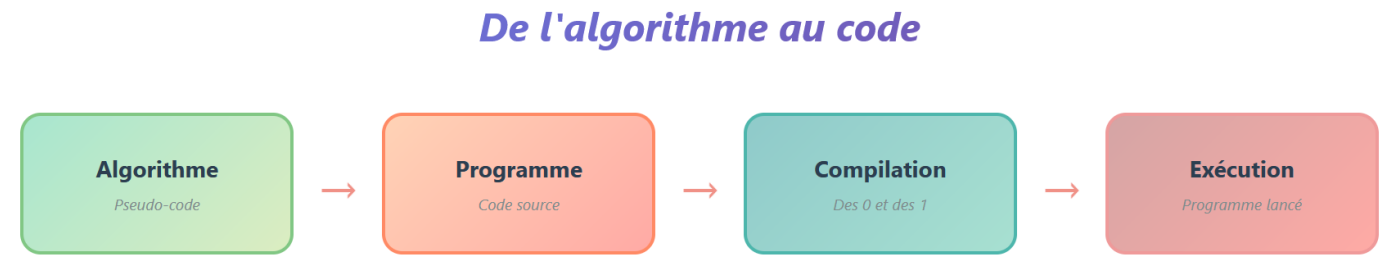
Description détaillée des étapes

1. Analyse
Elle consiste à mettre à jour la façon dont le problème sera résolu. C'est la recherche de l'algorithme de résolution. On peut écrire cet algorithme en pseudo-code.

2. Programmation évoluée
Le pseudo-code étant peu rigoureux, on convertit l'algorithme en langage de programmation évolué. On obtient alors un fichier texte ou code source.

3. Compilation
L'ordinateur ne comprend pas un langage de programmation évolué comme le C. Il convient de convertir le code source en langage machine : c'est la compilation. On utilise un logiciel appelé compilateur.

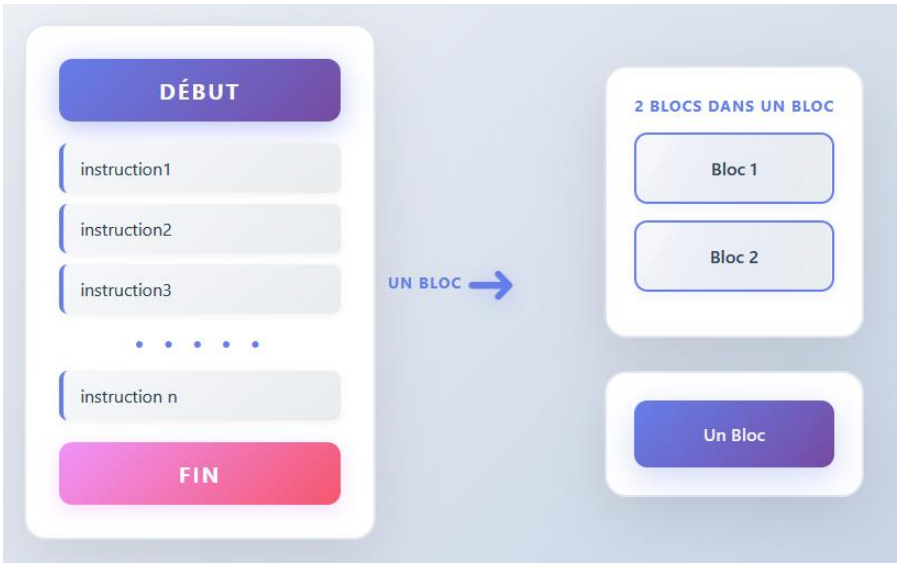
4. Édition des liens
Pour s'assurer que le programme compilé fonctionne d'un ordinateur à l'autre, on réalise l'édition des liens qui va lier le programme avec tous les éléments externes (généralement des librairies auxquelles il fait référence). On obtient un programme exécutable (fichier d'extension .exe par exemple).



3. Programmation structurée

Elle est fondée sur le bloc et l'analyse descendante.

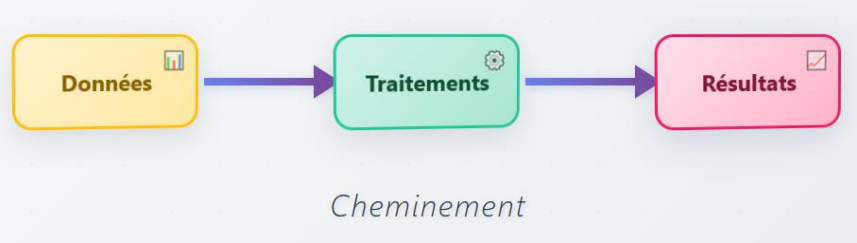
1. Notion d'instruction et de bloc
L'analyse descendante suggère qu'un algorithme complexe peut être décomposé en plusieurs sous-algorithmes plus simples.
Selon les besoins, une suite d'instructions formera un bloc délimité par les mots DEBUT et FIN. Un bloc peut en contenir d'autres.



2. Constituants d'un algorithme

Un algorithme est la décomposition d'une action complexe qui, partant de données toutes définies, permet d'obtenir un (des) résultat(s) déterminé(s).
Tout algorithme est fait :

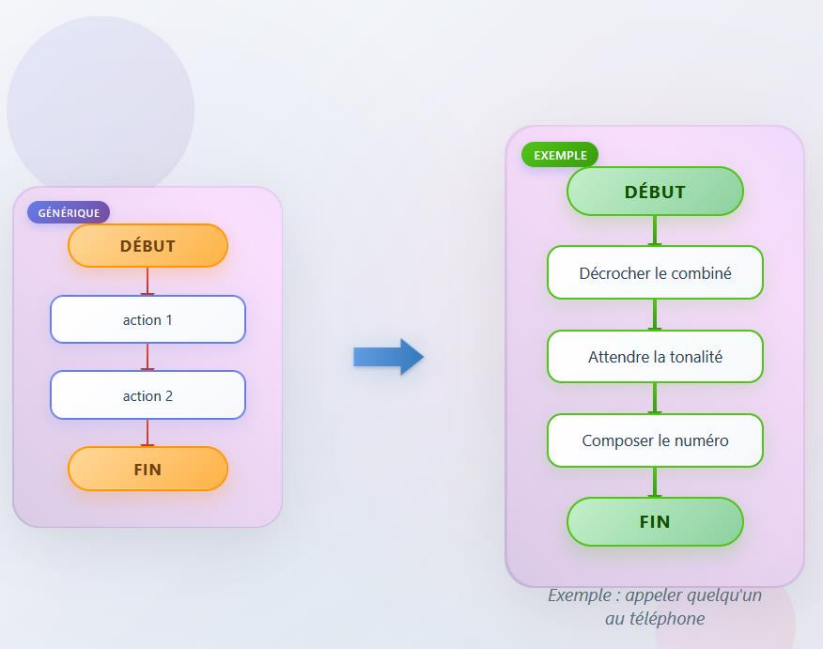
- D'enchaînements d'actions ou traitements : séquence, sélection ou répétition
- De données sur lesquelles ces actions agissent. Ces données sont des constantes ou des variables, et sont d'un certain type (entier, chaîne, etc.)
- L'algorithme produit des résultats



4. Enchaînements et données

1. Les enchaînements : séquences

L'enchaînement séquentiel consiste à exécuter des actions l'une après l'autre, du début à la fin du bloc logique qu'elles forment.

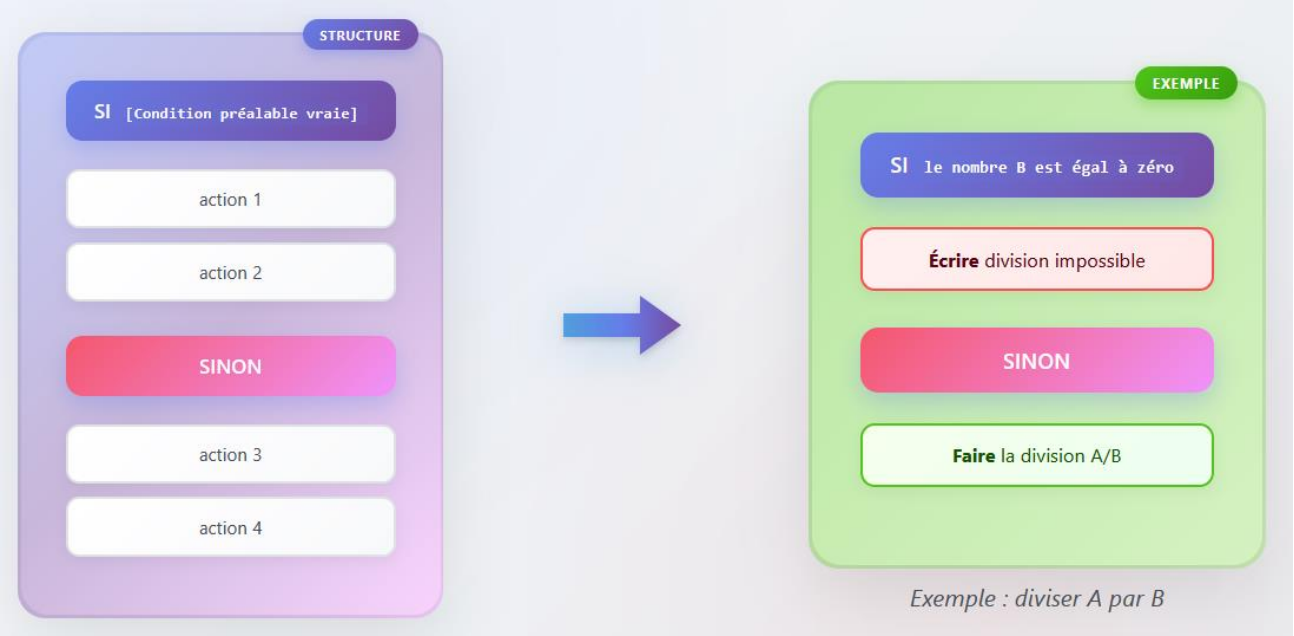


2. Les enchaînements : sélection

L'enchaînement sélectif consiste à exécuter certaines actions si une condition préalable est vérifiée, et à exécuter d'autres actions si ce n'est pas le cas.

PSEUDO-CODE

```
SI [Condition préalable vraie]
    action 1
    action 2
SINON
    action 3
    action 4
FIN-SI
```



3. Les enchaînements : répétition

L'enchaînement répétitif consiste à exécuter certaines actions aussi longtemps qu'une condition préalable est vérifiée, ou à exécuter certaines actions jusqu'à ce qu'une condition pré-établie soit vérifiée.

REPETER ... JUSQU'A

PSEUDO-CODE

```
REPETER
    Manger un peu
JUSQU'A ne plus avoir faim
```

Cette boucle s'effectue donc de **1 à plusieurs fois**.

TANT QUE

PSEUDO-CODE

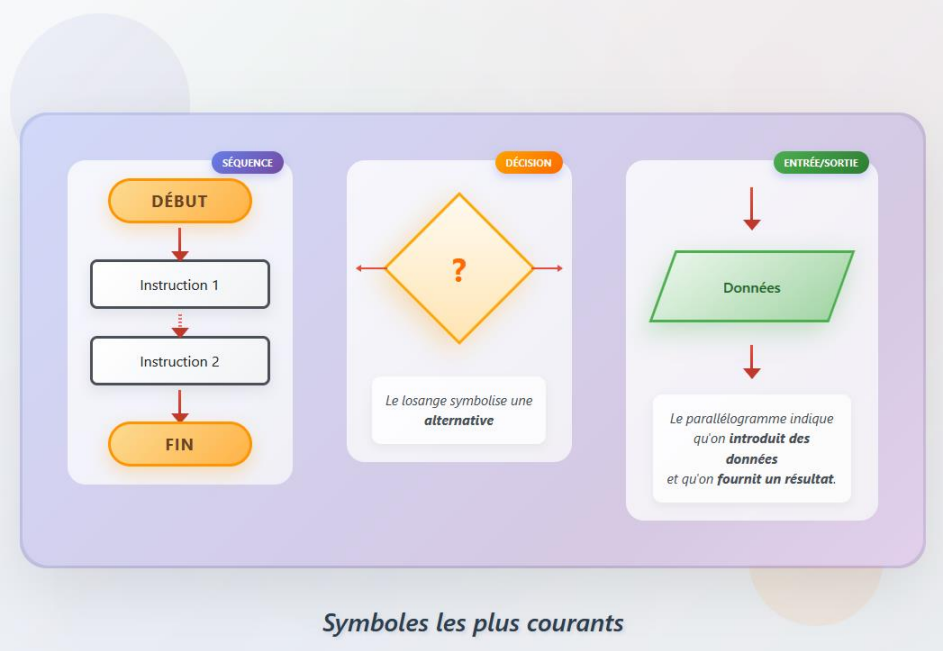
```
TANT QUE j'ai faim
    Manger un peu
FIN-TANT-QUE
```

Cette boucle s'effectue donc de **0 à plusieurs fois**.

5. Représentation d'un algorithme

1. Utiliser un organigramme

Le déroulement d'un algorithme peut être représenté par un organigramme. Ce dernier donne une impression de déroulement successif des instructions plus grande, mais ne met pas en évidence la notion de bloc.



Symboles les plus courants

Limites de l'organigramme :

- Ne met pas en évidence la notion de bloc
- Cette représentation est considérée plutôt comme un bon outil de communication et non comme un outil de conception

2. Utiliser le pseudo-code

Chaque élément d'algorithme est défini par un terme représentatif et non ambigu. Ces différents termes forment le pseudo-code.

STRUCTURE	PSEUDO-CODE

Alternative	PSEUDO-CODE
	SI <condition>
	  actions 1
	SINON
	  actions 2
	FIN-SI

Itération TANT QUE	PSEUDO-CODE TANT QUE <condition> Exécute les actions suivantes FIN-TANT-QUE
Itération REPETER	PSEUDO-CODE REPETER Les actions suivantes JUSQU'A <condition>

6. Exercices d'algorithmique

Exercice 1 : Variables et affectations

Quelles seront les valeurs des variables a, b et c après exécution des instructions suivantes :

PSEUDO-CODE

```
a ← 1     ; Explication : On affecte la valeur 1 à la variable a
b ← 5
c ← a - b
a ← 2
c ← a + b
```

Zone de réponse :

Exercice 2 : Permutation de variables

Écrire les instructions à utiliser pour permuter les valeurs de deux variables a et b.

Zone de réponse :

Exercice 3 : Analyse d'algorithme

PSEUDO-CODE

```
Algorithme test
Variables val, double : réel
Début
    val ← 4
    double ← val * 2
    Ecrire("Le double de ", val, " est ", double)
Fin
```

Quel résultat produit l'algorithme ci-dessus ?

Zone de réponse :

Exercice 4 : Calcul de moyenne

Écrire un algorithme qui, à partir de 3 notes d'un étudiant et 3 coefficients, calcule et affiche la moyenne.

Zone de réponse :

Exercice 5 : Nombre pair ou impair

Écrire un algorithme qui détermine si un entier entré par l'utilisateur est pair ou non.

Aide : L'opération modulo (mod ou %) renvoie le reste de la division euclidienne.
Exemple : 9 mod 4 = 1 car 9 = 2×4 + 1

Zone de réponse :

SÉANCE 3
Python & Codex

7. Implémentation en Python

Correspondances Pseudo-code → Python

PSEUDO-CODE	PYTHON	DESCRIPTION
a ← 5	a = 5	Affectation
Ecrire("texte")	print("texte")	Affichage
Lire(a) *Si a est un entier* *Si a est un réel*	a = input("Message: ") a = int(input("Message: ")) a = float(input("Message: "))	Saisie utilisateur Conversion en entier Conversion en réel
SI ... ALORS ... SINON ... FIN-SI	if ... : ... else: ...	Condition
TANT QUE ... ... FIN-TANT-QUE	while ... : ...	Boucle conditionnelle
POUR i DE 1 A n	for i in range(n):	Boucle bornée

Réaliser les exercices en Python en utilisant le logiciel Thonny

Exercice 1

Quelles seront les valeurs des variables a, b et c après exécution des instructions suivantes :

PYTHON

```
a = 1   # Explication : On affecte la valeur 1 à la variable a
b = 5
c = a - b
a = 2
c = a + b
print("a =", a, "b =", b, "c =", c)
```

Votre réponse :

Exercice 2 : Permutation

Écrire les instructions à utiliser pour permuter les valeurs de deux variables a et b.
À compléter :

PYTHON

```
# Exemples de valeurs pour a et b
a = 10
b = 20
```

Console (résultat attendu) :

```
>>> %Run Exercice2.py
a = 20  b = 10
>>>
```

Exercice 3

Algorithme test

Variables val, double : réel

À compléter :

```python
val = 4
```

Console (résultat attendu) :

```
>>> %Run Exercice3.py
Le double de 4 est 8
>>>
```

Exercice 4 : Calcul de moyenne

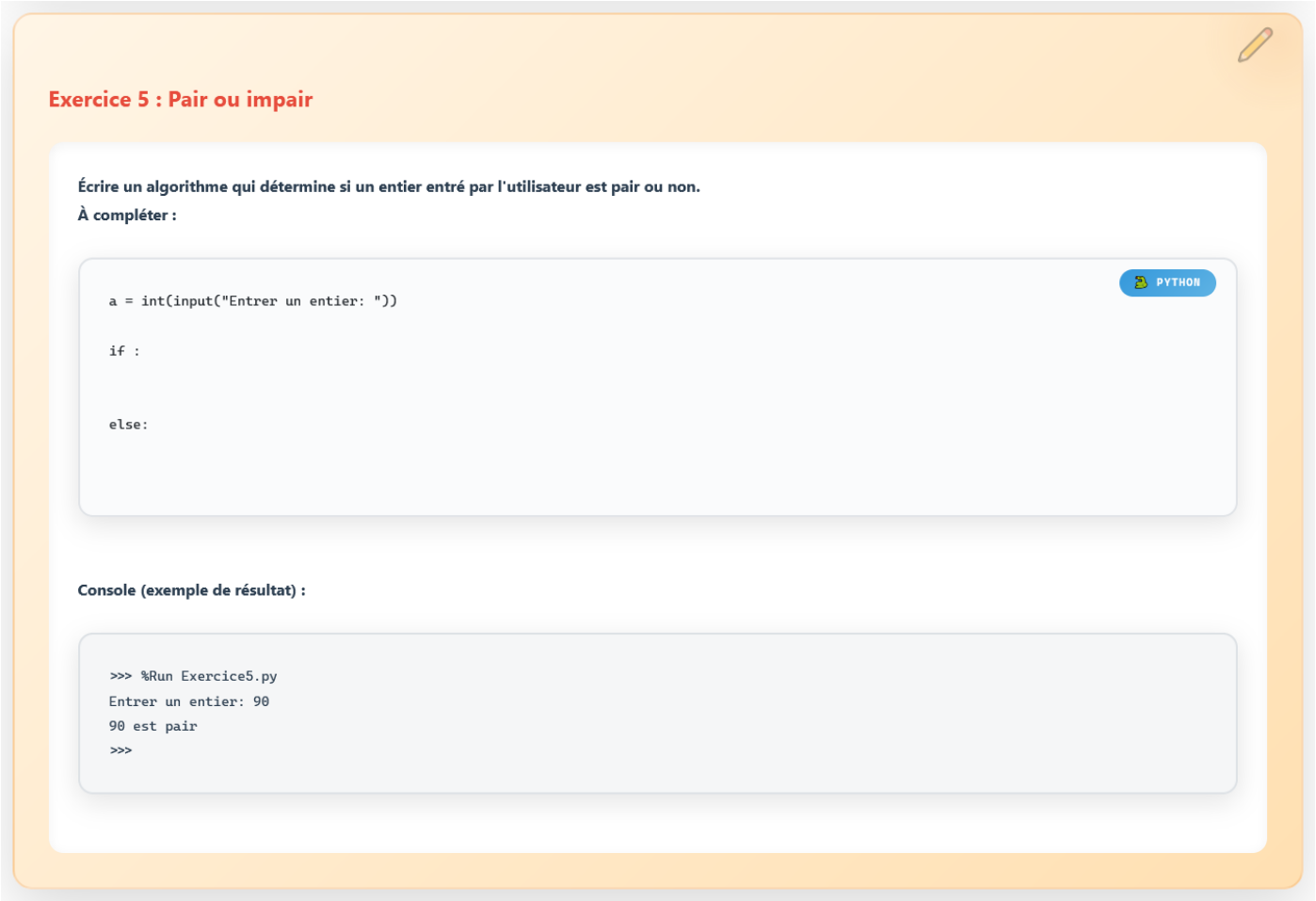
Écrire un algorithme qui à partir de 3 notes d'un étudiant et 3 coefficients calcule et affiche la moyenne.

À compléter :

```python
# On définit les valeurs des coefficients
c1 = 1
c2 = 2
c3 = 3
```

Console (exemple de résultat) :

```
>>> %Run Exercice4.py
Donner la première note : 11
Donner la deuxième note : 13.5
Donner la troisième note : 8
La moyenne est 10.33
>>>
```



Exercice 5 : Pair ou impair

Écrire un algorithme qui détermine si un entier entré par l'utilisateur est pair ou non.
À compléter :

PYTHON

```
a = int(input("Entrer un entier: "))

if :


else:
```

Console (exemple de résultat) :

```
>>> %Run Exercice5.py
Entrer un entier: 90
90 est pair
>>>
```

8. Exercices Codex - Alien

Alien (1) - Appels de fonctions
https://codex.forge.apps.education.fr/exercices/alien_1/

Alien (2) - Variables et affectations
https://codex.forge.apps.education.fr/exercices/alien2/

Alien (3) - Instructions conditionnelles
https://codex.forge.apps.education.fr/exercices/alien3/

Alien (4) - Boucles bornées
https://codex.forge.apps.education.fr/exercices/alien4/

Alien (5) - Fonctions
https://codex.forge.apps.education.fr/exercices/alien5/

Alien (7) - Boucles conditionnelles
https://codex.forge.apps.education.fr/en_travaux/alien7/