Enseigner Python en Seconde

Présentation du fascicule didactique

« Résoudre des problèmes avec Python — voici comment l'enseigner »

M. Hillion Dominique
Professeur de mathématiques — Lycée Schoelcher, Martinique
Enseignement de Python en Seconde — 2025-2026

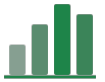
Plan de la présentation

La conviction centrale

Enseigner Python en Seconde ne devrait pas consister à faire exécuter une syntaxe —

cela devrait consister à faire résoudre des problèmes, Python étant l'outil qui les rend accessibles.

3 ans d'expérimentation · 6 séquences · résultats reproductibles d'une année sur l'autre



Six choix pédagogiques — 1/2

Choix 1 — Python comme outil de résolution

Les élèves rencontrent un problème — une balle qui rebondit, un distributeur de billets — et Python est l'outil. La syntaxe s'efface derrière le sens. Après les fonctions, les élèves n'utilisent plus input : le paradigme fonctionnel est devenu naturel.

Choix 2 — Le dispositif des paires de programmes

Six paires côte à côte : même algorithme, seule la forme change. Zéro nouveauté algorithmique. Toute l'attention sur la distinction def/return vs input/print. Dispositif conçu et expérimenté pour la première fois en 2025-2026.

Choix 3 — L'entrée systématique par le conflit cognitif

Chaque séquence commence par un déséquilibre : range(1,6) qui répète 5 fois et non 6, un TypeError sur une opération légitime. L'obstacle n'est pas contourné — il est provoqué pour être résolu.

Six choix pédagogiques — 2/2

Choix 4 — La transparence totale du contrat didactique

Ce qui est évalué a toujours été travaillé. Ce qui ne l'a pas été est absent de l'évaluation ou explicitement signalé comme bonus. Cette règle est tenue sans exception sur l'ensemble de l'année.

Choix 5 — L'architecture en spirale sur l'année entière

Les mêmes objets conceptuels réapparaissent dans des contextes enrichis : la permutation, l'algorithme de Collatz, les opérateurs // et %, le type booléen, assert, le pattern accumulateur. Ce n'est pas de la répétition — c'est une construction cumulative.

Choix 6 — L'articulation structurelle avec les mathématiques

Fonctions affines par morceaux, suites géométriques, probabilité fréquentiste, identités algébriques — les connexions sont naturelles car portées par un professeur de mathématiques qui pense les deux disciplines ensemble.

La progression annuelle — vue d'ensemble

Séq. 1

Variables, types, affectation

Devoir(s) : D1 + D2

Échange de variables · Conversion de secondes

Séq. 2

Instructions conditionnelles

Devoir(s) : D3

Fonction affine par morceaux · Tombola

Séq. 3

Fonctions Python

Devoir(s) : D4

Paires de programmes · trianrect(a,b)

Séq. 4

Boucle for

Devoir(s) : D5

Simulation du dé · Budget Oscar

Séq. 5

Boucle while

Devoir(s) : D6

Rebond de balle · DAB · Multiplication russe

Séq. 6

Chaînes de caractères

Devoir(s) : —

Code César · ord() / chr()

3 ans

6 séquences

6 devoirs

30+ exercices

6 innovations

Séquence 1 — Variables, types et affectation

1

Activité d'introduction

Scratch → Python · print(b) vs print("b") · int/float par contraste minimal · type() et booléens

Leçon

3 parties fragmentées · algorithme, affectation, types, opérations · print approfondi · input et conversion

Exercices

7 exercices : tri, rectangle, triangle, œufs, conversion, échanges · Préparation directe au Devoir 2

Contrôle

D1 : 5 exercices — polymorphie, échange · D2 : 7 exercices — input, triangle, conversion, assert, œufs

Obstacle principal : Séquentialité de l'affectation · input() renvoie toujours une chaîne · polymorphie de + et *

Séquence 2 — Instructions conditionnelles

2

Activité d'introduction

Scratch (dé) · agence immobilière : revenu ≥ 3×loyer · désindentation délibérée (Q.2d) → conflit cognitif

Leçon

Langage naturel avant syntaxe · if/else/elif · tableau de correspondance maths/Python · booléens and/or

Exercices

6 exercices : n%5, rugby (elif), inverse réel, parc attractions, fonction affine par morceaux, coureur

Contrôle

7 exercices : bagages, pièces, Collatz, tombola, lecture × 4, photocopies, œufs

Obstacle principal : Indentation structurelle · confusion = / == · elif vs if-if indépendants

Séquence 3 — Fonctions Python

3

Activité d'introduction

5 paires de programmes côte à côte : bag, syrac, billets, de, trianrect · induction avant définition

Leçon

Intérêt des fonctions · def/return · volparal · composition · exemple signe(f,x) pour avancés

Exercices

7 standard + 4 complexes · solde, vitesse, tension, sphère, cube/pyramide · cuve, location, mini4, conjecture

Contrôle

6 exercices : solde, pair(n), hyp/per, prog → d=3n², inter(x), Volpiscine + Prix

Obstacle principal : return vs print · définition vs appel · portée des variables (scope)

Séquence 4 — Boucle for

4

Activité d'introduction

range(1,6) → 5 répétitions (conflit cognitif) · list(range(…)) pour visualiser · indentation réactivée

Leçon

Syntaxe · 3 formes de range · tableaux d'accumulation (i, Somme) · for caractere in chaine

Exercices

9 exercices tous formulés comme fonctions : somme, intérêts, variantes, dé, dépréciation, pyramide, nbre_a, trace, divisibles

Contrôle

5 exercices : variantes somme, nbre_a, trace som, divisibles 3/non 9, budget Oscar

Obstacle principal : Borne exclue de range · variable compteur double rôle · initialisation hors boucle

Séquence 5 — Boucle while

5

Activité d'introduction

Rebond de balle aux 3/5 : nombre de rebonds inconnu → for impossible → while s'impose · tableau d'état

Leçon

Définition concise · syntaxe · exercice intégré (while s<50) travaillé en classe · Scratch exclu délibérément

Exercices

5 exercices : somme carrés, forage du puits, simulation dés, fonction mystère, DAB (socle garanti)

Contrôle

4 exercices : Syracuse, escargot (série harmonique), tableau d'état, DAB · Bonus : multiplication russe

Obstacle principal : Condition de continuation vs arrêt · risque de boucle infinie · affectation multiple n,i = n-20, i+1

Séquence 6 — Chaînes de caractères

6

Activité d'introduction

Aucune activité d'introduction — les élèves partent d'acquis implicites depuis le Ch.1

Leçon

Définition · opérations (+,*) · ordre lexicographique · len() · indexation par rang · for sur chaîne · méthodes

Exercices

7 exercices : initiales, chiffre_dizaine/unites, fonction essai, construction, ASCII, identite, code César

Contrôle

Contrôle à construire — compétences consolidées : boucle for sur chaîne, indexation, str(n), upper/lower

Obstacle principal : Rang démarrant à 0 · borne droite exclue (analogue range) · indices négatifs · syntaxe objet.méthode()

Exercice phare — Séquence 1 : Conversion de secondes

★ PHARE

1

Exercice phare de la Séquence 1 — Il mobilise // et % dans un contexte de modélisation concrète. Les deux valeurs testées (8567s et 2845s) sont choisies avec soin : la seconde donne un nombre d'heures nul, ce qui force la vérification du cas où un quotient vaut 0.

Énoncé

Écrire un programme qui demande une durée en secondes et affiche cette durée en heures, minutes et secondes.

Exemple : 8567 secondes → 2 h, 22 min, 47 s
Exemple : 2845 secondes → 0 h, 47 min, 25 s

Indication : utiliser // et %

Programme Python

```
total = int(input('Durée en secondes : '))
heures = total // 3600
reste = total % 3600
minutes = reste // 60
secondes = reste % 60
print(heures, 'h', minutes, 'min', secondes, 's')
```

Exercice phare — Séquence 2 : Fonction affine par morceaux

★ PHARE

2

Exercice phare de la Séquence 2 — Il demande de coder une fonction définie par morceaux, objet mathématique central du programme de Seconde. L'élève ne fait pas de l'informatique : il fait des mathématiques avec Python comme outil.

Énoncé

$f(x) = 2x+1$ si $x \leq -1$
$f(x) = -x+2$ si $-1 < x \leq 0$
$f(x) = -3x+2$ si $x > 0$

1. Compléter le programme Python.
2. Compléter le tableau de valeurs :
x : −2 | −1 | 0 | 1 | 2 | 3

Programme Python

```
x = float(input('x : '))
if x <= -1:
    y = 2*x + 1
elif x <= 0:
    y = -x + 2
else:
    y = -3*x + 2
print(round(y, 2))
```

Exercice phare — Séquence 3 : La paire trianrect(a,b)

★ PHARE

3

Exercice phare de la Séquence 3 — Cette paire condense la transition entre programme séquentiel et fonction réutilisable. Premier return multiple du corpus (triplet hyp/périmètre/aire), vérifiable avec le triplet (3,4,5).

Énoncé

Observer et expliquer les différences entre les deux versions :

Version séquentielle :

```
a = int(input('a : '))
b = int(input('b : '))
hyp = sqrt(a**2+b**2)
print(hyp, per, aire)
```

Version fonction :

```
def trianrect(a,b):
    ...
    return hyp, per, aire
```

Programme Python

```
from math import sqrt

def trianrect(a, b):
    hyp = sqrt(a**2 + b**2)
    per = a + b + hyp
    aire = a * b / 2
    return hyp, per, aire
```

Exercice phare — Séquence 4 : Simulation du lancer de dé

★ PHARE

4

Exercice phare de la Séquence 4 — Il combine boucle for, instruction conditionnelle, accumulateur de comptage et randint. La question sur la fréquence connecte directement la programmation à la probabilité fréquentiste.

Énoncé

1. Que retourne randint(1, 6) ?
2. Que teste if randint(1,6) == 6 ?
3. Exécuter avec n=1000. Même résultat ?
4. Quel cas pratique modélise cette fonction ?
5. Modifier pour retourner la fréquence du 6.

Programme Python

```
from random import randint

def de(n):
    nbre_six = 0
    for i in range(n):
        if randint(1, 6) == 6:
            nbre_six = nbre_six + 1
    return nbre_six
```

Exercice phare — Séquence 5 : Distributeur automatique de bill

★ PHARE

5

Exercice phare de la Séquence 5 — L'exercice de synthèse le plus complet de l'année. Il combine if/return, while avec affectation simultanée, et return de tuple. Un élève qui réussit maîtrise toutes les structures de l'année.

Énoncé

Un DAB propose uniquement des billets de 10€ et 20€. Il distribue le moins de billets possible.

1. À quoi sert l'instruction de la ligne 2 ?
2. Combien de billets de 10€ et 20€ pour un retrait de 330€ ?

Programme Python

```
def DAB(n):
    if n % 10 != 0:
        return 'impossible'
    else:
        i = 0
        while n >= 20:
            n, i = n - 20, i + 1
        billets_20 = i
        billets_10 = n // 10
        return billets_10, billets_20
```

Exercice phare — Séquence 6 : Le code César

★ PHARE

6

Exercice phare de la Séquence 6 — Le plus complet de l'année sur les chaînes. Il mobilise for sur chaîne, ord/chr, if/elif/else et def/return. La progression naïve → corrigée en fait un exercice de modélisation complète.

Énoncé

Jules César décalait chaque lettre de 3 rangs.
Version naïve : coder chaque caractère avec ord(c)+3.
Problèmes identifiés :
- X/Y/Z dépassent le code 90
- L'espace est codé en '#'

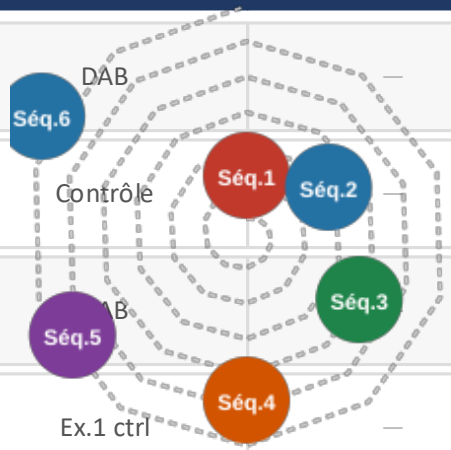
Version corrigée : compléter if/elif/else.

Programme Python

```
def code_cesar(message):
    message = message.upper()
    coder = ''
    for c in message:
        n = ord(c)
        if n == 32:       # espace
            n_code = 32
        elif n + 3 > 90: # X,Y,Z
            n_code = n + 3 - 26
        else:
            n_code = n + 3
        coder = coder + chr(n_code)
    return coder
```

L'architecture en spirale — fils rouges inter-séquences

Fil rouge	Séq.1	Séq.2	Séq.3	Séq.4	Séq.5	Séq.6
// et %	Leçon+D1	D3 (œufs)	conversion()	Tous ex.	DAB	—
Type booléen	type(7<9)	Formalisé	pair(n)	Ex.9	Contrôle	—
assert	Bonus D2	—	billets	—	AB	—
Collatz/Syracuse	—	Contrôle	syrac(n)	—	Ex.1 ctrl	—
Accumulateur	—	—	—	Tous	Tous	—
for + chaîne	—	—	—	Ex.7+ctrl	—	Formalisé
Triangle rect.	ER-Ex.3	—	trianrect()	—	—	—
Exercice des œufs	D2-Ex.7	D3-Ex.7	—	—	—	—



La philosophie de l'évaluation — les quatre principes

Socle garanti

2 à 3 exercices par contrôle ont été travaillés en classe à l'identique. Tout élève sérieux peut réussir le socle.
Exemple D6 : tableau d'état (while s<50) et DAB — tous deux corrigés en classe avant le contrôle.

Transfert proche

Les autres exercices constituent un transfert dans un contexte légèrement nouveau. La structure est connue, seuls les paramèt res changent.
Exemple D5 : divisibles par 3 et non par 9 (analogue de divisibles par 2 et non par 4, travaillé en classe).

Exercice de synthèse

Chaque contrôle se clôt sur un exercice exigeant qui révèle si les schèmes ont été réellement intériorisés.
Exemple D4 : prog(n) → $d=3n^2$ — observation, modification, conjecture, démonstration algébrique.

Transparence totale

Ce qui est évalué a toujours été travaillé. Ce qui ne l'a pas été est absent ou signalé bonus. Sans exception.
Exemple D6 : la multiplication russe (bonus) est explicitement signalée comme exercice de recherche.

Regard sur les résultats — ce que la séquence produit

Les résultats ne s'expliquent pas exercice par exercice — ils s'expliquent par la cohérence architecturale de la séquence entière.

Exercices bien réussis

Exercices travaillés en classe → restitution directe · Lecture de programmes simples · Trace d'exécution avec modèle fourni · Ex. : D5-Ex.1 (variantes de somme) repris quasi à l'identique de la feuille d'exercices

Exercices discriminants

Exercices de transfert avec complexification · Raisonnement sur le code sans machine · Ex. : D5-Ex.5 (budget Oscar) — deux accumulations dans une même boucle · D6-Ex.1 (Syracuse) — suite non monotone

La reproductibilité comme indice

Les mêmes résultats se reproduisent d'une année sur l'autre, avec des classes différentes. Cette régularité n'est pas le produit d'exercices mémorisés — c'est la trace d'une architecture pédagogique cohérente.

Le rôle formateur des exercices non évalués

Les exercices non évalués jouent un rôle irremplaçable. Ils se répartissent en trois catégories complémentaires.

Exercices traités en classe

Construisent le socle de compétences que l'évaluation mobilisera.
Ex. : DAB et tableau d'état (while s<50) pour le contrôle boucle while · paire trianrect pour le contrôle fonctions.

Exercices en découverte autonome

Permettent un transfert réel dans un contexte nouveau non guidé.
Ex. : l'escargot (série harmonique) et la suite de Syracuse pour la boucle while.

Exercices du devoir maison corrigé avant le contrôle

Préparation explicite et honnête, sans piège ni nouveauté.
Ex. : simulation des dés (DM corrigé) avant le contrôle boucle for.

Ce que l'évaluation par compétences ne voit pas

L'évaluation par compétences photographie un état terminal. Elle est aveugle à l'amont — elle ne rend pas visible le travail de construction qui a rendu la compétence possible.

Ce que cette séquence montre :

La compétence finale est le produit d'un enchaînement causal et intentionnel — pas d'exercices mémorisés.

Les élèves réussissent l'exercice 4 du contrôle boucle for (divisibles par 3 et non par 9) parce que la structure logique k%n==0 and k%m!=0 est devenue un schème implicitement acquis à travers l'exercice 9 de la feuille.

Les exercices complexes sur la composition de fonctions ne préparent pas directement tel item du contrôle : ils construisent une façon de penser — décomposer un problème en sous-fonctions.

On sait que les compétences seront transférables parce qu'on a travaillé sur des exercices conçus pour ça — pas en ciblant des items, mais en construisant la profondeur de compréhension.

La structure type d'un chapitre — quatre temps articulés

Activité d'introduction

Faire émerger le concept par l'expérimentation ou le conflit cognitif.
Le concept doit être nécessité par la situation avant d'être nommé.
On ne définit pas ce qu'on n'a pas encore eu besoin de définir.

Exercices

Diversifier les contextes d'application. Préparer le contrôle directement et indirectement.
Travailler en classe les exercices qui constitueront le socle garanti.
Le transfert ne se décrète pas — il se construit.

Leçon

Institutionnaliser ce que l'activité a fait émerger. Nommer, formaliser, systématiser.
Fournir la référence que l'élève pourra consulter.
La leçon ne découvre pas — elle officialise.

Contrôle

Évaluer avec un spectre équilibré : lecture, complétion, production, raisonnement.
Socle garanti + transfert proche + exercice de synthèse.
L'évaluation mesure ce que la séquence a construit.

L'articulation structurelle avec les mathématiques

Python n'est pas enseigné comme une discipline autonome — il est enseigné comme un prolongement des mathématiques.

Séq.1-2

Division euclidienne

// et % modélisent la décomposition d'un entier (secondes, œufs, billets). Connexion directe avec la notion de division avec reste.

Séq.2

Fonctions par morceaux

La fonction affine par morceaux — objet central du programme de Seconde — se code directement en if/elif/else.

Séq.3

Théorème de Pythagore

trianrect(a,b) encapsule une formule mathématique. return multiple = tuple de valeurs mathématiques liées.

Séq.4

Suites géométriques

La dépréciation (×0,9 par an) et les intérêts composés (×1,025) sont des suites géométriques simulées par boucle for.

Séq.5

Série harmonique

L'escargot (1 + 1/2 + 1/3 + …) révèle un comportement mathématique contre-intuitif : la série diverge, mais très lentement.

Séq.3-4

Algèbre et conjectures

prog(n) → $d=3n^2$ (D4) et n^2=somme des n premiers impairs (D5) : Python comme outil heuristique au service des maths.

Synthèse des six devoirs — cohérence de la série

★★★

N°	Notion	Contenu	Nb ex.	Points forts	Statut
D1	Variables	5 ex.	Affectation, polymorphie, échange	Lecture → simulation	1
D2	print/input	7 ex.	Types, TypeError, // %, assert	Maillon central D1→D3	2
D3	Conditionnelles	7 ex.	if/elif, or/and, %, œufs réapparat.	Piège if/if (Ex.5B)	3
D4	Fonctions	6 ex.	def/return, composition, booléens	Pivot D1-3 → D5-6	4
D5	Boucle for	5 ex.	range 3 formes, for str, accum.	Le + riche en maths	5
D6	Boucle while	5+bonus	while, DAB, Syracuse, mult. russe	Synthèse de l'année	6

Deux documents complémentaires — fascicule et mémoire

Le fascicule répond à la question : comment ça fonctionne — décision par décision, obstacle par obstacle.
Le mémoire répond à la question : pourquoi ça fonctionne — sur le plan didactique et théorique.

Le fascicule

Pour les enseignants de mathématiques

- 6 analyses didactiques détaillées
- Obstacles identifiés et traités
- Fils rouges et spirale curriculaire
- Synthèse et ajustements envisageables
- Regard sur les résultats

Le mémoire

★★★

Pour les formateurs et chercheurs

- 6 choix pédagogiques documentés
- Exercices phares avec analyse complète
- Obstacles épistémiques par devoir
- Synthèses évaluatives ★★★
- Annexes : 6 contrôles + analyses

Le mémoire — cadre didactique et philosophie

Ce mémoire documente une ingénierie pédagogique — pas l'application d'un modèle théorique, mais la formalisation d'une pratique pédagogique construite par l'expérience qui anticipait intuitivement les conditions d'acquisition.

Progression inductive systématique

On expérimente avant de définir — dans chaque séquence sans exception. La définition répond à une question que la situation a posée.

Gestion localisée de la nouveauté

Un seul élément nouveau par étape, les autres paramètres stabilisés. Réduction de la charge cognitive à chaque introduction.

Schèmes opératoires transférables

Les élèves intériorisent des façons de penser sans nécessairement pouvoir les verbaliser. Ce sont ces schèmes qui permettent le transfert.

Contrat didactique cohérent

La transparence totale — ce qui est évalué a été travaillé — est la condition de la confiance de l'élève dans la relation d'évaluation.

Reproductibilité comme indice empirique

Les mêmes résultats se reproduisent d'une année sur l'autre avec des classes différentes. Cette régularité est la trace d'une architecture cohérente.

Le mémoire — les synthèses évaluatives ★★★ de la série

Chaque devoir fait l'objet d'une synthèse évaluative selon cinq critères. L'appréciation globale situe la série dans son ensemble.

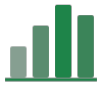
Devoir	Couverture	Progressivité	Originalité	Différenciation	Contextualisations	
D1 — Variables	★★★	★★★	★★	★★★	★★★	1
D2 — print/input	★★★	★★★	★★★	★★★	★★★	2
D3 — Conditionnelles	★★★	★★★	★★★	★★★	★★★	3
D4 — Fonctions	★★★	★★★	★★★	★★★	★★★	4
D5 — Boucle for	★★★	★★★	★★★	★★★	★★★	5
D6 — Boucle while	★★★	★★★	★★★	★★★	★★★	6

Évaluation selon cinq critères — résultats stables d'une année sur l'autre.

Le mémoire — registres sémiotiques et vocation

Registre tabulaire

Trace d'exécution, tableau d'état — présent dans 5 devoirs sur 6. Rend visible l'exécution séquentielle que l'élève ne peut pas observer directement.



Registre symbolique Python

Lecture → diagnostic d'erreur → complétion → production autonome. Le registre Python devient progressivement le registre de travail naturel.

Registre algébrique

Échange sans variable temporaire (D1) · $d=3n^2$ (D4) · n^2=somme des impairs (D5). Python comme outil heuristique au service de l'algèbre.

Registre historique et culturel

Collatz (D3,D6) · Multiplication russe — Papyrus de Rhind ~1650 av. J.-C. (D6 bonus) · Code César (Séq.6). L'algorithmique ancré dans l'histoire longue des mathématiques.

À notre connaissance, peu de ressources articulent à ce niveau l'enseignement de Python en Seconde avec une analyse didactique détaillée, une progression cohérente et un ancrage dans une pratique éprouvée.

Diffusion et perspectives

En local — Martinique

Lycée Schoelcher et établissements proches · Corps d'inspection académique · Journées pédagogiques du rectorat

Au niveau académique

Formations continues organisées par le rectorat · Présentation du fascicule comme support de formation · Échanges entre praticiens

Au niveau national

Fascicule → IREM (réseau national), Sésamath, APMEP · Mémoire → revues Petit x et Repères IREM · Colloques de didactique des mathématiques

Ce que ce corpus comble

À notre connaissance, peu de ressources articulent à ce niveau l'enseignement de Python en Seconde avec une analyse didactique détaillée, une progression cohérente et un ancrage dans une pratique éprouvée.

Ce que cette séquence démontre

Enseigner ne consiste pas à transmettre des connaissances — mais à construire les conditions dans lesquelles l'élève peut les acquérir.

La pratique a précédé la théorie

Cette séquence a été construite intuitivement,
par l'expérience. La théorie l'a rejointe.

La didactique lui donne ses mots

Schèmes opératoires, contrat didactique, transfert d'apprentissage — déjà à l'œuvre.

Ce fascicule la rend transmissible

Non pas des ressources — une ingénierie pédagogique avec ses intentions et sa logique.

M. Hillion Dominique
Lycée Schoelcher — Martinique — 2025-2026