Lycée Loges

Chapitre 7
Récursivité TP

Simon Dauguet
simon.dauguet@gmail.com

29 janvier 2026

Exercice 1 :
Un palindrome est un mot qui se lit de la même manière à l'endroit comme à l'envers (en omettant les espaces et autres signes de ponctuation). Par exemple, "elle", "kayak", "Élu par cette crapule" et "Engage le jeu que je le gagne" sont des palindromes tandis que "il", "papa" ou "Seth a été à Sète" n'en sont pas.

Pour la suite, on suppose qu'un mot ne contient que des lettres minuscules, sans accent, ni espace ou marque de ponctuation.

1. Écrire une fonction itérative `palindromeI(mot: str) -> bool`, qui vérifie si `mot` est un palindrome.
2. Un mot vide est-il un palindrome ? un mot contenant un seul caractère ?
3. Supposons que nous ne puissions comparer qu'un couple de lettres, lesquelles vérifieriez-vous pour savoir si le mot peut être un palindrome ? Comment se ramener à ce cas en avançant dans la comparaison du mot ?
4. Écrire une fonction récursive `palindromeR(mot: str) -> bool`, qui vérifie si `mot` est un palindrome.

Exercice 2 :
On souhaite tracé le triangle de Sierpinski en partant d'un triangle équilatéral.

Voici les résultats attendus :

TSierpinski(n)	n=1	n=2	n=3
résultat			

1. À partir des points $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$ sommets du triangle parent ABC, quelles sont les coordonnées des points sommets des trois triangles fils ?
2. Pour créer récursivement les triangles de Sierpinski, on commence par créer une fonction permettant de dessiner un triangle. Puis, on construit la fonction récursive qui créée les différents calques avec tous les triangles. Et enfin, la fonction qui permet d'afficher tous les calques créés par récursivité. Compléter la fonction récursive d'appel principal
`TSierpinski(n:int, A:tuple=(0,0), B:tuple=(1,0), C:tuple=(1/2,mt.sqrt(3)/2)) -> None`, et l'appel de la fonction `Sierpinski(n:int)->None` permet alors d'afficher les triangles de Sierpinski de profondeur n.

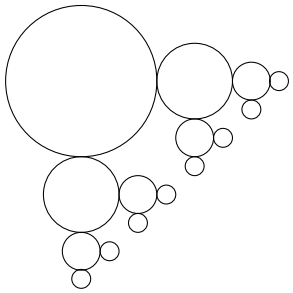
```python
def triangle(A:tuple, B:tuple, C:tuple) -> None : # trace le triangle ABC
    plt.fill( [ A[0], B[0], C[0] ], [ A[1], B[1], C[1] ], 'k')

def TSierpinski(n:int, A:tuple=(0,0), B:tuple=(1,0), C:tuple=(1/2,mt.sqrt(3)/2)) -> None:
par défaut, triangle équilatéral de côté 1
    if n == 1:
        ...
    else:
        ... # plusieurs lignes

def Sierpinski(n:int) -> None :
    plt.figure()
    triangle_sierpinski(n)
    plt.show()
```

Exercice 3 :

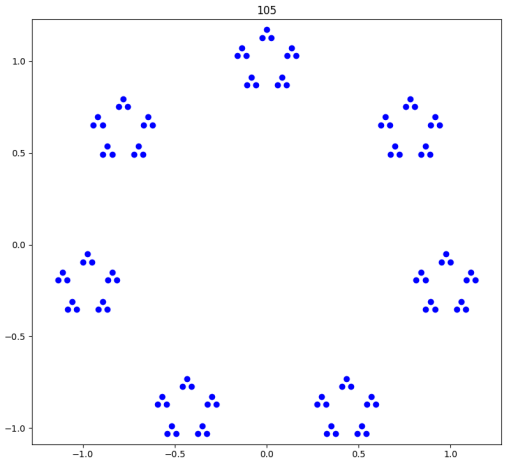
On souhaite obtenir la figure ci-contre de manière récursive (ici profondeur 4).



La figure est formée d'un cercle et de deux copies de ce cercle ayant subies une réduction d'un facteur 2, ces deux petits cercles étant tangents extérieurement au cercle initial et tels que les lignes des centres sont parallèles aux axes du repère. Ces deux petits cercles deviennent à leur tour "cercle initial" pour poursuivre la figure.
Commencer par créer une fonction `cercle(coord:tuple, r:float) -> None` qui créé le calque du cercle de centre `coord` et de rayon `r`. Puis créer les fonctions répondant au problème dont l'une construira les calques des cercles par récursivité, en s'inspirant de la méthode proposée dans l'exercice précédent.

Exercice 4 (Décomposition en produit de nombres premiers) :

Le but de cet exercice est de fournir le moyen de fournir le graphe de la décomposition en produit de nombres premiers d'un entier.



1. Écrire une fonction `pgpd(n:int) -> int` qui renvoie le plus grand diviseur premier de l'entier `n`.
2. Écrire une fonction `cercle(n:int) -> tuple` qui renvoie le couple de deux listes correspondants respectivement aux abscisses et ordonnées de `n` points régulièrement répartis sur le cercle trigonométrique.
3. Écrire récursive une fonction `diagrame(n:int) -> tuple` renvoyant les listes X_n et Y_n respectivement des abscisses et ordonnées des points du diagramme de la factorisation de n :

- Si n est premier, les points du diagramme sont en cercle
- Sinon, en notant d le plus grand diviseurs premier de n, X_n est la liste des $x_d + \frac{x}{d}$ où x_d parcourent X_d et x parcourt $X_{\frac{n}{d}}$.
- De même pour Y_n.

4. Enfin, écrire une procédure `afficheDiv(n:int) -> None` qui affiche le diagramme de décomposition d'un entier n. On mettra en titre du graphe l'entier n.

   Remarque : Attention à ce que les cercles soient des cercles.

Exercice 5 :

On souhaite dessiner le flocon de Von Koch.

Voici les résultats attendus où chaque simili triangle est un triangle équilatéral sans la base :

flocon(n)	n=0	n=1	n=2	n=3
résultat				

En vous inspirant des exos précédent, proposer une fonction `VonKoch(n:int) -> None`, qui trace les résultats affichés sur un seul segment, puis une autre version `VonKochEntier(n:int) -> None` qui trace le flocon de Von Koch "en entier" (sur les trois côtés).