Remarque :

Dans tout le TP, il est interdit d'utiliser les fonctions classiques des listes comme ***len***, ***copy***… Il est interdit d'accéder à une valeur de la liste avec [].

On ne peut qu'utiliser les fonctions de base définies à l'exercice 1. Attention, lorsque l'on a une liste p listes, écrire ***q = p*** donne un second nom à la liste p, ce qui induit que toute modification sur p ou q s'effectue sur la même liste ! Une fonction qui ne renvoie aucun résultat renvoie None, et non « None ».

Exercice 1 : fonctions de base

Créer les fonctions suivantes :

creer_pile()	Fonction renvoyant une pile vide
est_vide()	Fonction renvoyant le résultat d'un test en booléen (True ou False)
empiler(p,x)	Fonction empilant l'élément x dans la pile p
depiler(p)	Fonction dépilant la pile p et renvoyant l'élément dépilé

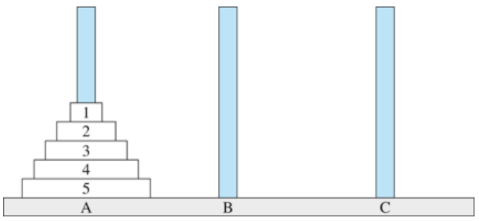
Exercice 2 : fonctions avancées

En n'utilisant que les 4 fonctions de base et les fonctions nouvellement créées, créer les fonctions suivantes, dans l'ordre ! (Aucune fonction ne devra avoir besoin d'une fonction demandée après) :

sommet(p)	Fonction renvoyant le sommet de la pile p sans enlever ce terme de p si la pile est non vide – Sinon, retourne None
echange_sommet(p)	Fonction qui échange le sommet et le terme juste en dessous dans la pile p
taille(p)	Fonction renvoyant la taille de la pile p sans pour autant que p soit vide à la fin (on pourra passer par une seconde pile)
dupliquer(p)	Fonction qui duplique une pile p sans utiliser l'égalité entre 2 piles qui lie deux noms au même objet – On a donc à la sortie la pile initiale et une nouvelle piles, indépendantes
vider(p)	Fonction vidant la pile p
inverser(p)	Fonction inversant la pile p
empiler_piles(p,q)	Fonction qui ajoute les éléments d'une pile q à la pile p. Une fois exécutée, la pile q doit être inchangée
cycler(p)	Fonction qui transforme la pile p en plaçant son sommet tout en bas et tous les autres termes étant remontés d'un cran

Exercice 3 : Tours de Hanoï

L'objectif de ce casse-tête est de réussir à reproduire sur le barreau C la tour placée sur le barreau A. Il repose donc sur un principe de piles. En particulier, à chaque déplacement, il n'est autorisé d'empiler sur une pile qu'un nombre plus petit que le sommet de la pile destinataire.



Importez la fonction random en incluant à votre code : from random import*

En se basant sur les fonctions créées aux exercices précédents, ***créer les fonctions suivantes*** :

creation_piles(n)	Fonction créant la ***pile A*** contenant n étages dont le dernier est l'étage 1 ainsi que les ***piles vides B et C*** et enfin la liste de listes ***Piles*** contenant les 3 piles A, B et C dans cet ordre. Seule la liste « Piles » sera retournée
pile_est_vide(p)	Fonction renvoyant un Booléen ***True ou False***, répondant à la question : la pile p est-elle vide ?
choix_depiler(Piles)	Fonction prenant comme argument la liste de piles « ***Piles*** » et retournant un numéro entre 0 et 2 correspondant à la pile à dépiler choisie aléatoirement parmi les piles A, B, C non vides sous la forme d'un entier valant 0 (pile A), 1 (pile B), 2 (pile C).

choix_empiler(Piles,sommet_a_deplacer,provenance)
Fonction prenant comme argument la liste de piles « Piles », la pile de provenance du sommet à rempiler et la valeur de ce sommet et qui retourne un entier entre 0 et 2 correspondant à la pile de destination, déterminé aléatoirement parmi les 3 piles A, B et C. Il ne pourra être empilé sur sa pile de provenance, et devra être plus petit que le sommet de la pile de destination s'il existe

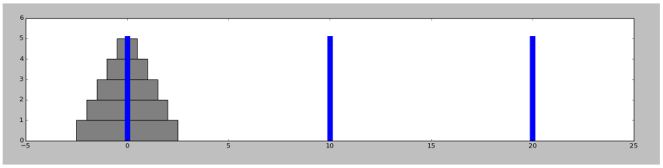
resolution(Piles)	Fonction prenant en argument la liste de piles « ***Piles*** » et solvant le problème des tours de Hanoï sur le principe du hasard décrit pour les fonctions précédentes Cette fonction retournera un compteur calculant le nombre de déplacements réalisés
solution(n)	Fonction créant la liste de piles « Piles » pour n étages, solvant le problème et retournant le compteur associé

Résoudre le problème pour n=4 et montrer qu'il faut en moyenne 530 mouvements pour trouver la solution aléatoirement.

Maintenant que votre programme fonctionne, vous pouvez copiez et collez le contenu du fichier ***Affichage.py*** au début de votre fichier, puis ajoutez à votre code au départ après avoir créé les listes initiales et après chaque mouvement le code : ***f_animation(Piles)***

Vous pouvez faire un essai en tapant : ***f_animation([[5,4,3,2,1],[],[]])***

Vous obtiendrez :



Remarque : en affichant à chaque déplacement l'état des piles dans la console en faisant ***print(Piles)***, et lorsque vous aurez trouvé une solution simple, vous pourrez l'essayer ici :

http://jeux.prise2tete.fr/tour-de-hanoi/tour-de-hanoi.php

Nous verrons bientôt que ce problème se traite par récursivité et que le nombre minimum de déplacements vaut : $2^n - 1$

Mettre en place la fonction suivante :

meilleure_solution(n)	Fonction faisant tourner la résolution pour n étages jusqu'à ce que le résultat présente le minimum de déplacements On pourra implémenter le stockage des compteurs successifs et l'affichage de leur moyenne