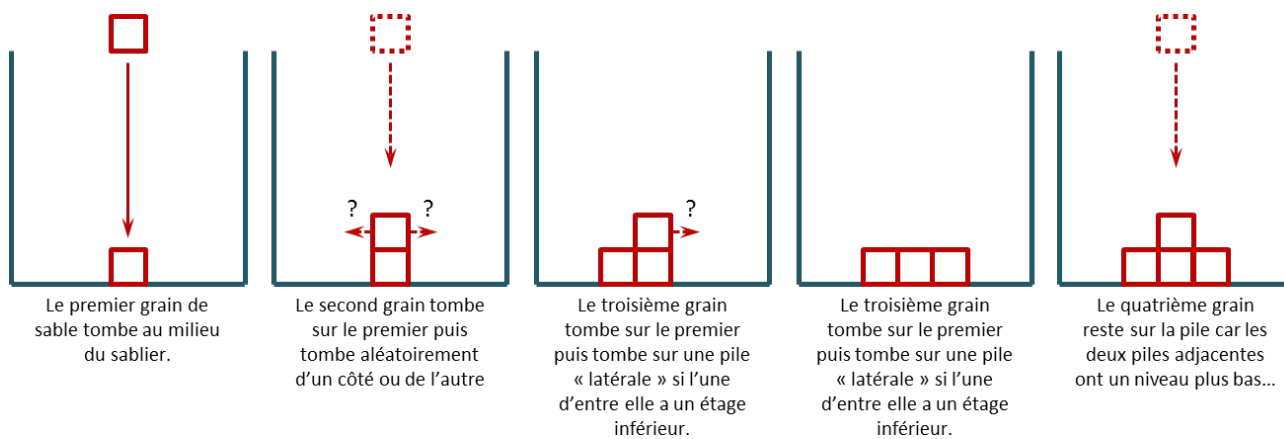


1 Présentation

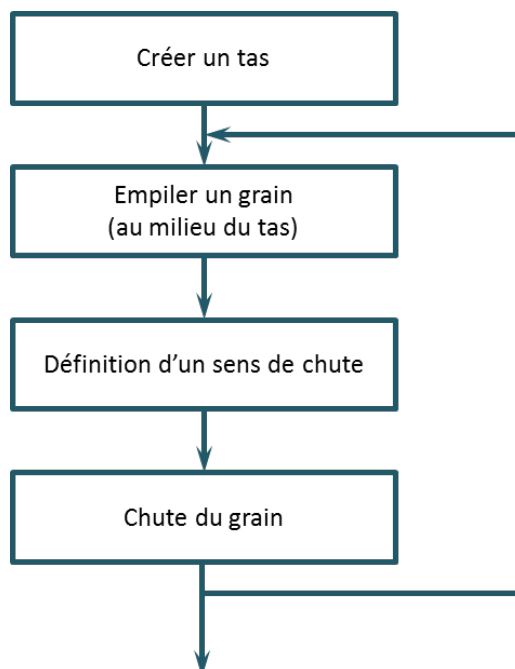
On se propose de modéliser la constitution d'un tas de sable ainsi que l'écoulement des grains dans un sablier. Afin de simplifier le problème, on se restreindra à travailler en 2 dimensions. Le tas sera modélisé par une pile de grains de sable.



Dans le cas du sablier, les grains tombent toujours sur la même pile. Le processus de constitution de la pile est le suivant :



Un algorithme très succinct présente le déroulement de la chute d'un grain de sable.



2 Création des objets

2.1 Gestion des piles de sables

Une pile de sable est modélisée par... une pile ! Cette dernière est implémentée sous forme d'une liste.

Question 1 Donner l'implémentation des fonctions élémentaires permettant de créer une pile dans Python à savoir les fonctions `creer_pile`, `empiler`, `depiler`, `est_vide`. **Pour cette question on s'autorise l'utilisation des méthodes sur les listes.**

Question 2 Donner l'implémentation de la fonction `taille_pile`, permettant de connaître la taille d'une pile. **Pour cette question seules les fonctions définies ci-dessus sont acceptées.** Donner la complexité algorithmique de la fonction implémentée

Question 3 Redéfinir la fonction `empiler` pour que le seul élément empilable soit la chaîne de caractères `"*"`. Ainsi, une pile de sable sera constituée d'une pile d'étoiles.

2.2 Gestion du tas de sable

Un tas de sable va être modélisé par une liste de piles de grains de sable.

Question 4 Implémenter la fonction `creer_tas` permettant de créer un tas de `n` piles de sables.

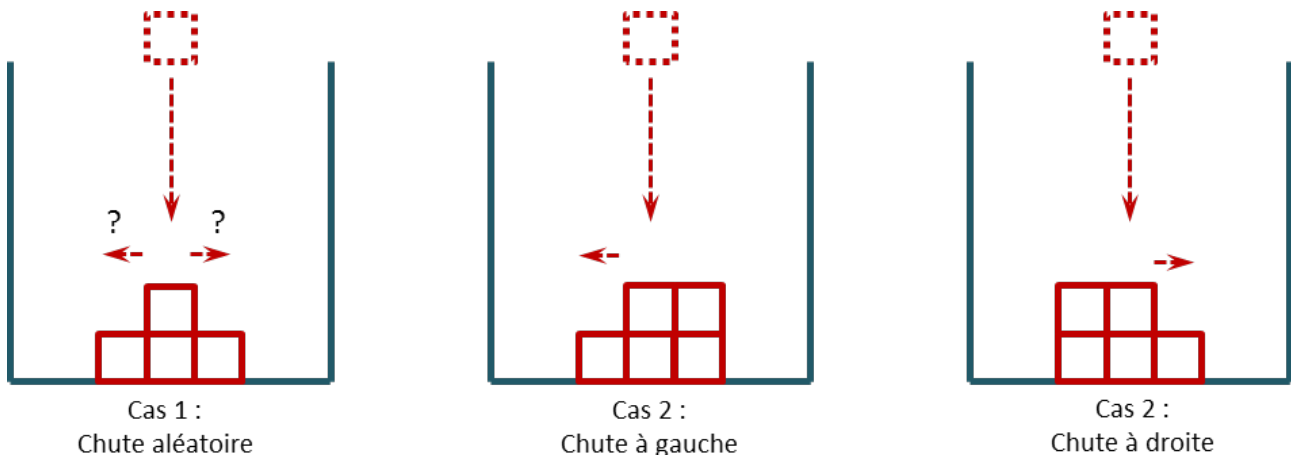
Question 5 Donner les instructions permettant de dépiler un grain de sable sur la pile `n` et d'empiler un grain de sable sur la pile `n - 1`.

3 Écoulement

On va maintenant implémenter les fonctions qui vont permettre de régir l'écoulement d'un grain de sable. On suppose que les grains tombent toujours sur la même pile.

On s'intéresse d'abord au sens d'écoulement d'un grain de sable. Pour cela, on définit une variable `sens` qui vaut 0 lorsque le grain doit s'écouler vers la gauche et qui vaut 1 lorsque le grain doit s'écouler vers la droite.

On identifie les 3 cas suivants pour déterminer le sens de chute d'un grain :



Question 6 Exprimer la condition booléenne pour laquelle un grain de sable chute à gauche.

Question 7 Exprimer la condition booléenne pour laquelle un grain de sable chute aléatoirement à gauche ou à droite.

Question 8 En réalisant un schéma donner un cas de figure pour lequel il n'y a pas d'écoulement de grain. Traduire la condition booléenne correspondante.

On considère un écoulement vers la gauche.

Question 9 Exprimer la condition booléenne permettant de savoir si un grain sur la pile `n` doit s'écouler sur la gauche. On tiendra compte du cas où le grain est sur le bord du sablier.

On appelle `cond` la condition exprimée à la question précédente. On appelle `chute` la fonction permettant de régir la chute du grain (ici vers la gauche). Les spécifications de la fonction sont les suivantes :

■ Python

```
def chute(tas, indice, sens):
    """
    Gestion d'une chute de grain de sable.
```

```
Entrées :
* tas(liste de piles) : tas de sable
* indice(int) : pile à laquelle on a déposé le dernier grain de sable
* sens(int) : 0 chute à gauche, 1 chute à droite
Sortie :
* le tas est modifié mais n'est pas retourné.
"""
```

Question 10 Implémenter la fonction `chute` permettant de gérer la chute d'un grain de sable. Cette fonction devra être récursive.

4 Affichage du tas de sable

On donne le tas suivant :

```
■ Python
>> print(tas)
[[], ['*'], ['*', '*'], ['*', '*', '*'], ['*', '*']
```

On souhaite l'afficher sous la forme suivante :

```
■ Python
-----
--*--
--*--
--*--
--*--
```

Question 11 Implémenter la fonction `affichage` permettant d'afficher un tas sous la forme définie ci-dessus.

Question 12 Analyse d'une fonction récursive... à définir.