Structures de données : les dictionnaires:

Nous allons maintenant étudier un autre type abstrait de données : les dictionnaires aussi appelés tableau associatif.

On retrouve une structure qui ressemble, à première vue, beaucoup à une liste. Mais au lieu d'associer chaque élément à un indice de position, dans un dictionnaire, on associe chaque élément (on parle de valeur dans un dictionnaire) à une clef, on dit qu'un dictionnaire contient des couples clef:valeur (chaque clef est associée à une valeur). Exemples de couples clef:valeur $\longrightarrow$ prenom:Kevin, nom:Durand, date-naissance:17-05-2005. prenom, nom et date sont des clefs ; Kevin, Durand et 17-05-2005 sont des valeurs.

Voici les opérations que l'on peut effectuer sur le type abstrait dictionnaire:

- ajout: on associe une nouvelle valeur à une nouvelle clef
- modif: on modifie un couple clef:valeur en remplaçant la valeur courante par une autre valeur (la clef restant identique)
- suppr: on supprime une clef (et donc la valeur qui lui est associée)
- rech: on recherche une valeur à l'aide de la clef associée à cette valeur.

Exemple: Soit le dictionnaire D composé des couples clef:valeur suivants $\longrightarrow$ prenom:Kevin, nom:Durand, date-naissance:17-05-2005. Pour chaque exemple ci-dessous on repart du dictionnaire d'origine

- ajout(D,tel:06060606); le dictionnaire D est maintenant composé des couples suivants: prenom:Kevin, nom:Durand, date-naissance:17-05-2005, tel:06060606
- modif(D,nom:Dupont); le dictionnaire D est maintenant composé des couples suivants: prenom:Kevin, nom:Dupont, date-naissance:17-05-2005
- suppr(D,date-naissance); le dictionnaire D est maintenant composé des couples suivants: prenom:Kevin, nom:Durand
- rech(D,prenom); la fonction retourne Kevin

Exercice 1: Python propose une implémentation des dictionnaires (nous avons déjà étudié cette implémentation l'année dernière: `https://pixees.fr/informatiquelycee/n_site/nsi_prem_dico.html`).
A l'aide de cette implémentation, créer le dictionnaire de l'exemple précédent et y faire les mêmes modifications.

L'implémentation des dictionnaires dans les langages de programmation peut se faire à l'aide des tables de hachage. Les tables de hachages ainsi que les fonctions de hachages sont omniprésentes en informatique. Pour avoir quelques idées sur le principe des tables de hachages, je vous recommande

le visionnage de cette vidéo: wandida: `https://www.youtube.com/watch?v=CkLctGYWFPA`

Si vous avez visionné la vidéo, vous avez déjà compris que l'algorithme de recherche dans une table de hachage a une complexité O(1) (le temps de recherche ne dépend pas du nombre d'éléments présents dans la table de hachage), alors que la complexité de l'algorithme de recherche dans un tableau non trié est O(n). Comme l'implémentation des dictionnaires s'appuie sur les tables de hachage, on peut dire que l'algorithme de recherche d'un élément dans un dictionnaire a une complexité O(1) alors que l'algorithme de recherche d'un élément dans une liste a une complexité O(n).

Exercice 2: Soit l'algorithme "x est-il présent dans le tableau t ?"

```
VARIABLE
t : tableau d'entiers
x : nombre entier
tr : booléen (VRAI ou FAUX)
i : nombre entier
DEBUT
tr ← FAUX
i ← 1
tant que i<=longueur(t) et que tr==FAUX:
  si t[i]==x:
    tr ← VRAI
  fin si
  i ← i+1
fin tant que
renvoyer la valeur de tr
FIN
```

1. Faites "tourner à la main" l'algorithme "x est-il présent dans le tableau t ?" avec t=[5,8,15,53] et x=12
2. Evaluer sa compléxité.