Algorithme de Tris et autres

I) Rappel sur les premiers algorithmes

> ***Exercice*:**
>
> 1. Refaire l'algorithme du minimum avec une liste de nombres puis une liste de cadeaux.
> 2. Refaire l'algorithme de la recherche naïve avec une liste de nombres puis une liste de cadeaux.

II) Algorithmes de tris

II.1) Tri par insertion

Allez sur Wikipedia et regarder l'exemple : https://fr.wikipedia.org/wiki/Tri_par_insertion

Voici le pseudo code :

```
    procédure tri_insertion(tableau T)

        pour i de 1 à taille(T) - 1

            # mémoriser T[i] dans x
            x ← T[i]

            # décaler les éléments T[0]..T[i-1] qui sont plus
grands que x, en partant de T[i-1]
            j ← i
            tant que j > 0 et T[j - 1] > x
                    T[j] ← T[j - 1]
                    j ← j - 1

            # placer x dans le "trou" laissé par le décalage
            T[j] ← x
```

> ***Exercice*:**
>
> 1. Réaliser l'exemple de Wikipedia avec des cartes
> 2. Refaire l'algorithme de Tri par Insertion avec une configuration de cartes différente de l'exemple précédentes.
> 3. Faire une liste par compréhension de 30 nombres compris entre 1 et 20

> 4. Faire un script qui tri par insertion cette liste de nombres.
> 5. Faire une fonction qui prends en entrée une liste et qui renvoie cette liste triée
> 6. Appliquer cette fonction sur la liste de la question 3.

In [ ]:

II.2) Tri par selection

Voici la page wikipedia du tri par selection :
https://fr.wikipedia.org/wiki/Tri_par_s%C3%A9lection

Regarder la petite animation à droite de la page wikipedia.

```
procédure tri_selection(tableau t)
   n ← longueur(t)
   pour i de 0 à n - 2
       min ← i
       pour j de i + 1 à n - 1
           si t[j] < t[min], alors min ← j
       fin pour
       si min ≠ i, alors échanger t[i] et t[min]
   fin pour
fin procédure
```

> ***Exercice:***
>
> 1. Réaliser l'animation de Wikipedia avec des cartes
> 2. Refaire l'algorithme de Tri par Selection avec une configuration de cartes différente de l'exemple précédentes.
> 3. Faire une liste par compréhension de 30 nombres compris entre 1 et 20
> 4. Faire un script qui tri par selection cette liste de nombres.
> 5. Faire une fonction qui prends en entrée une liste et qui renvoie cette liste triée
> 6. Appliquer cette fonction sur la liste de la question 3.

In [ ]:

III) Algorithme de Recherche

III.1) Algorithme de Recherche Naïf

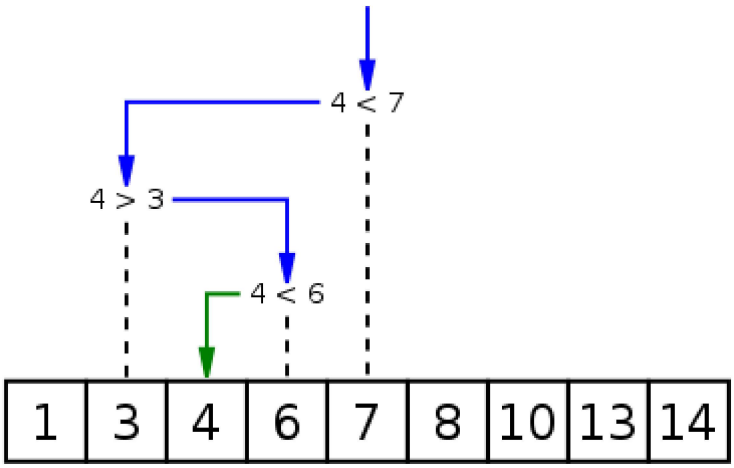
> ***Exercice:***

> Refaire une fonction de Recherche Naïve qui prend en entrée une liste de nombre et une valeur et qui renvoie `True` si la valeur est dans la liste et `False` sinon.

In [ ]:

III.2) Algorithme de recherche par dichotomie

Cet algorithme de recherche fonctionne **SEULEMENT** sur des tableaux triés (dans l'ordre croissant). La dichotomie signifie couper en deux. Donc on va chercher une valeur en coupant notre liste en 2 puis en 2 puis en 2 etc ...



Le pseudo code est le suivant (merci qui merci wikipedia!)

```
trouvé = Faux
debut = 0
fin = N #N est la taille du tableau
val = val # valeur qu'on cherche dans le tableau

Tant que trouvé != vrai et début <= fin:
      mil ← partie_entière((début + fin)/2)
      si t[mil] == val:
         trouvé ← vrai
      sinon:
         si val > t[mil]:
            début ← mil+1
         sinon:
            fin ← mil-1
```

> ***Exercice:***
>
> 1. Réaliser l'exemple de l'image avec des cartes
> 2. Refaire l'algorithme de Recherche par Dichotomie avec une configuration de cartes différente de l'exemple précédentes.

> 3. Faire une liste par compréhension de 15 nombres compris entre 1 et 30.
> 4. Faire un script qui Recherche par Dichotomie cette liste de nombres et la valeur 7.
> 5. Faire une fonction qui prends en entrée une liste triée et une valeur et qui renvoie `True` si la valeur est dans la liste et `False` sinon.
> 6. Appliquer cette fonction sur la liste de la question 3 avec la valeur 7.

II) Intermede vidéo

Petite vidéo qui illustre bien : https://www.youtube.com/watch?v=14c5N8CfnUo&ab_channel=VincentGoulet

III) Avec des classes

On va faire tourner tous nos algorithmes sur des listes d'objets. Ces objets seront des cadeaux (faut qu'on pense positif !) Ces cadeaux auront des valeurs aléatoires.

In [ ]:
```
import random as rd
class Cadeau:
    def __init__(self,numero):
        self.numero = numero
        self.valeur = rd.randint(1,1000)
    def __str__(self):
        return f"le cadeau numero {self.numero} a pour valeur {self.valeur}"
```

In [ ]:
```
liste_cadeau = [Cadeau(i) for i in range(500)]
```

In [ ]:
```
for cadeau in liste_cadeau:
    print(cadeau)
```

> ***Exercice:***
>
> Refaire les exercices précédents avec une liste de cadeaux
