Projet 7

ALGOINVEST&TRADE

RÉSOLVEZ DES PROBLÈMES EN UTILISANT DES ALGORITHMES EN PYTHON

Les spécifications.

- Objectif : Concevoir un algorithme permettant de trouver la meilleur combinaison d'investissement afin de maximiser les profits de nos clients.

- Coût : d'une action de l'entreprise en euros.

Actions #	Coût par action (en euros)	Bénéfice (après 2 ans)
Action-1	20	5%
Action-2	30	10%
Action-3	50	15%
Action-4	70	20%
Action-5	60	17%

- Actions : Chaque "Action" représente une action dans une entreprise différente.

- Bénéfice : (après 2 ans) Il s'agit du bénéfice réalisé par le titulaire de l'action après 2 ans d'investissement dans l'entreprise. Le bénéfice est un pourcentage du coût de l'action.

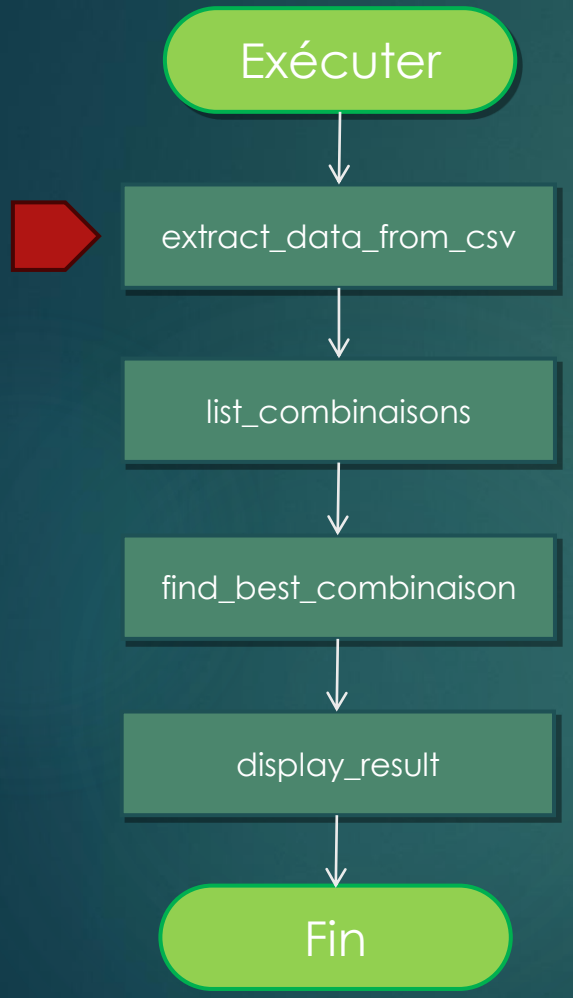
Le problème.

- Il s'agit d'un problème d'optimisation.
  - Nous voulons maximiser le bénéfice total sur 2 ans.
  - Avec les contraintes suivantes :
    - Chaque action ne peut être achetée qu'une seule fois.
    - Nous ne pouvons pas acheter une fraction d'action.
    - Nous pouvons dépenser au maximum 500 euros par client.

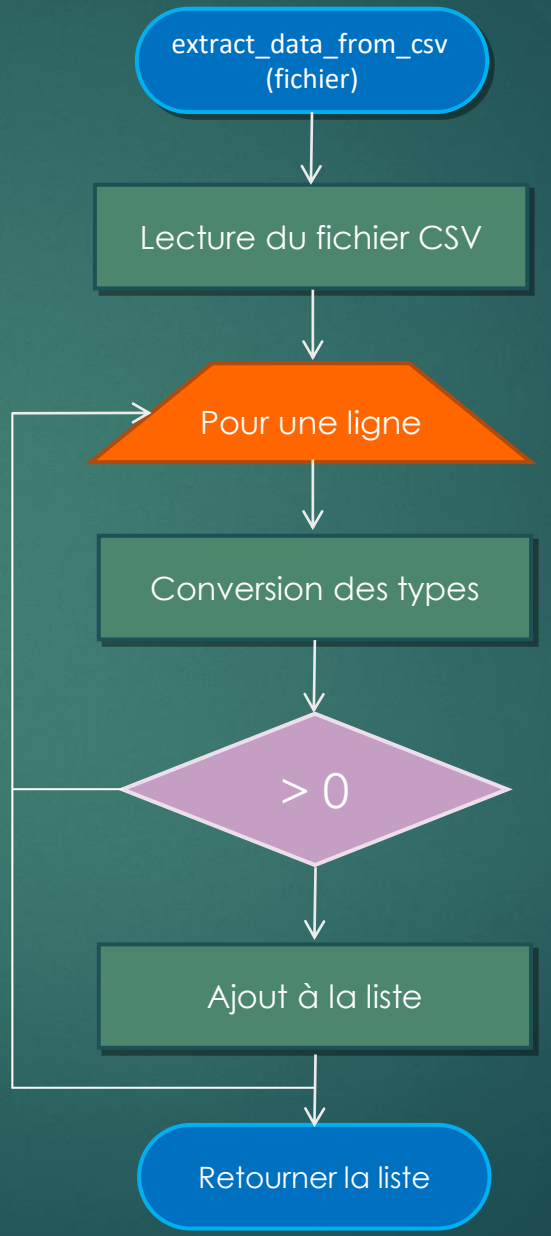
Brute force : 1er solution.

- Bruteforce.py

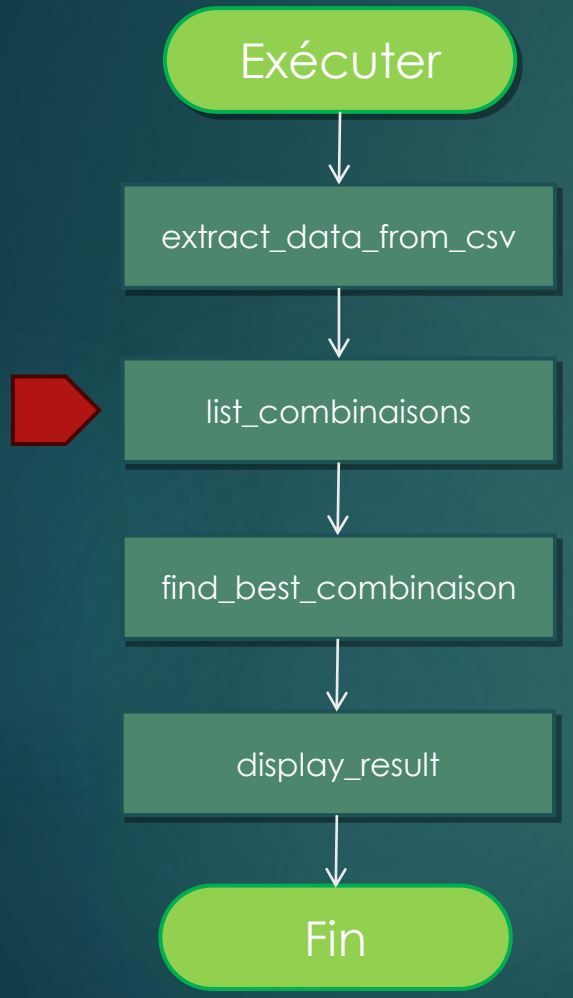
Brute force :



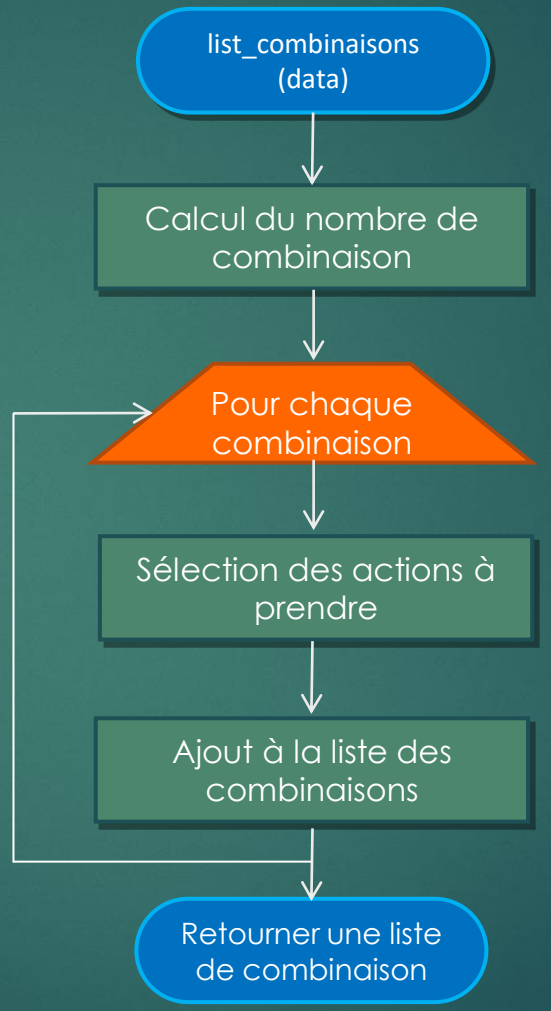
- utils.py



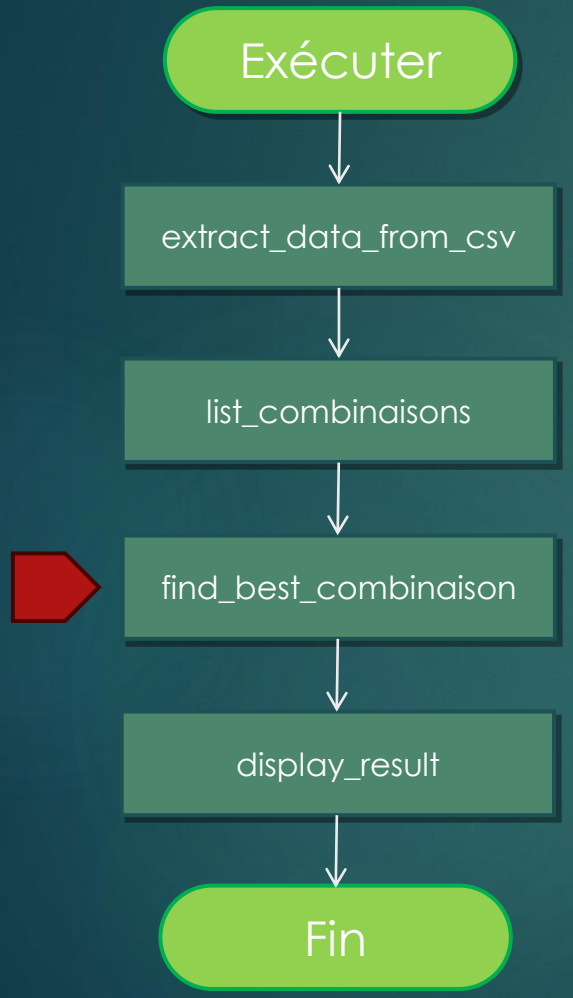
Brute force :



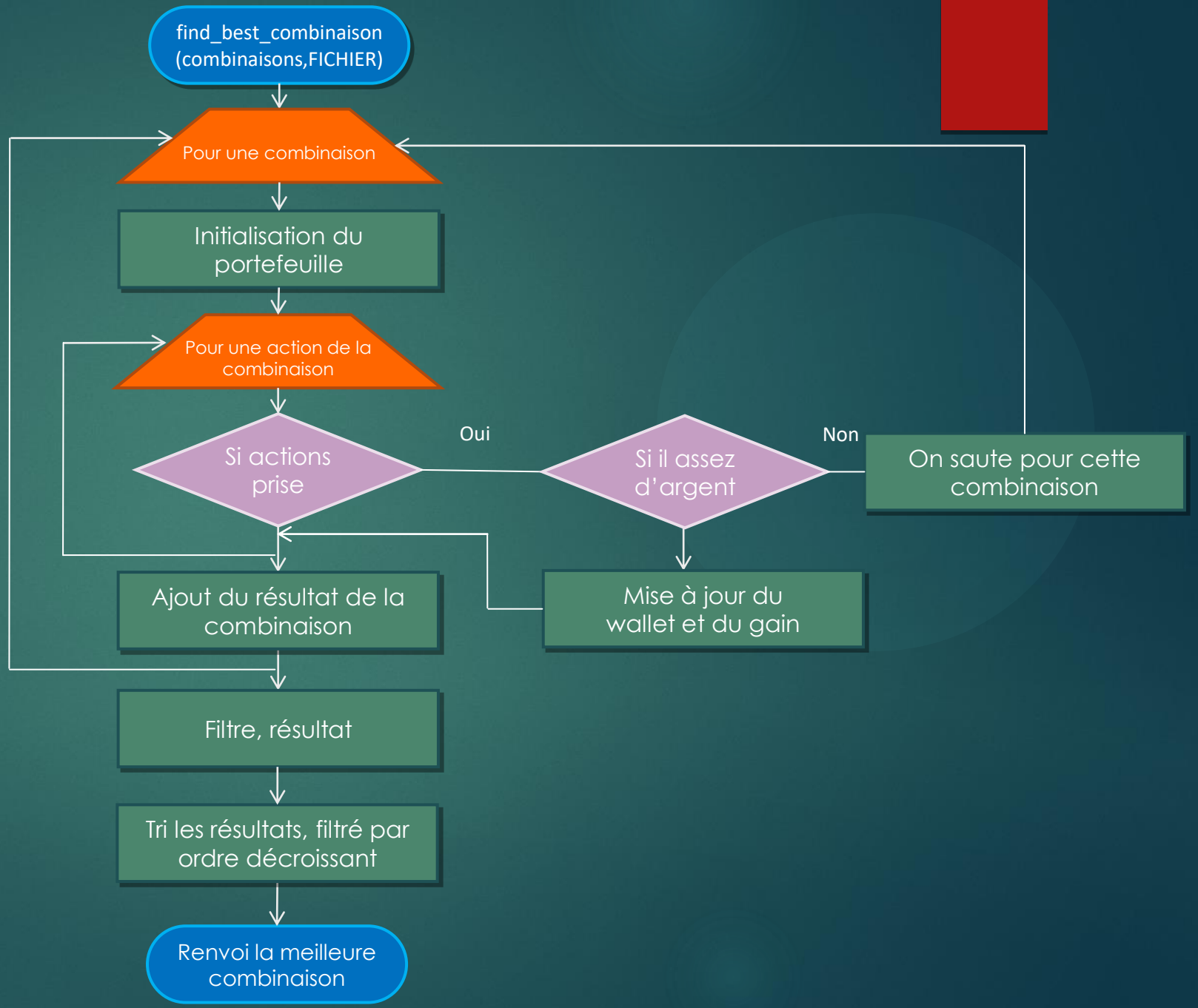
- bruteforce.py



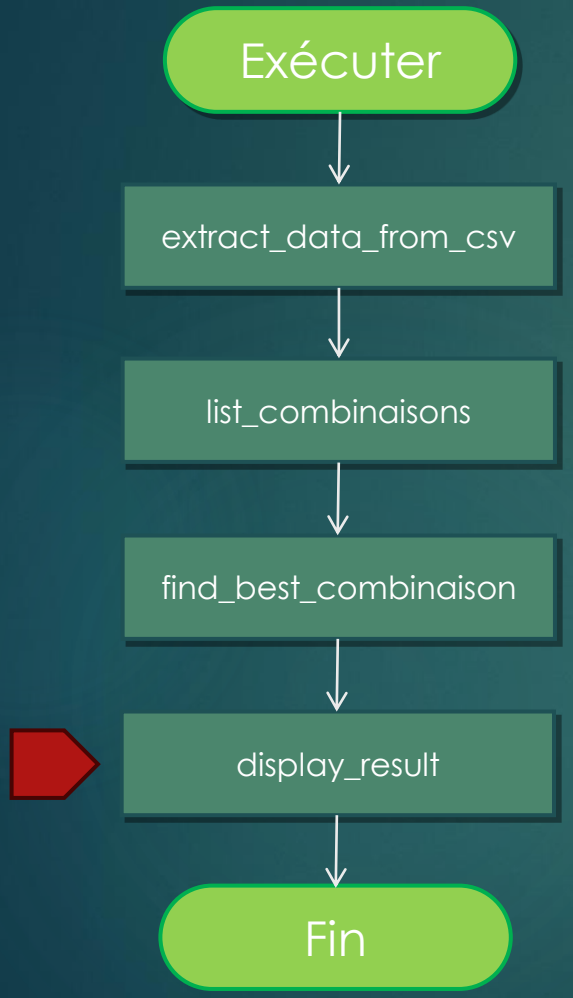
Brute force :



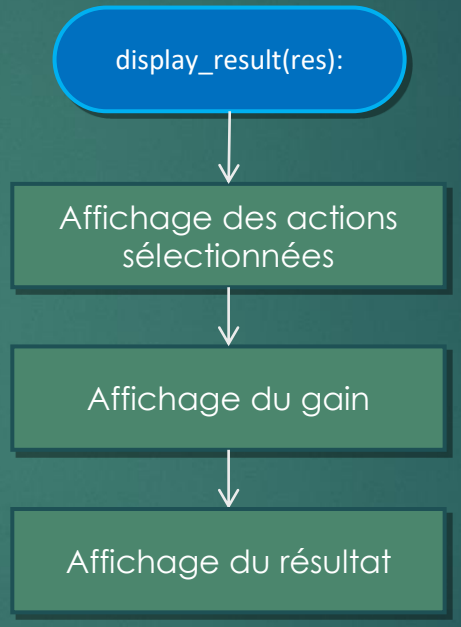
- bruteforce.py



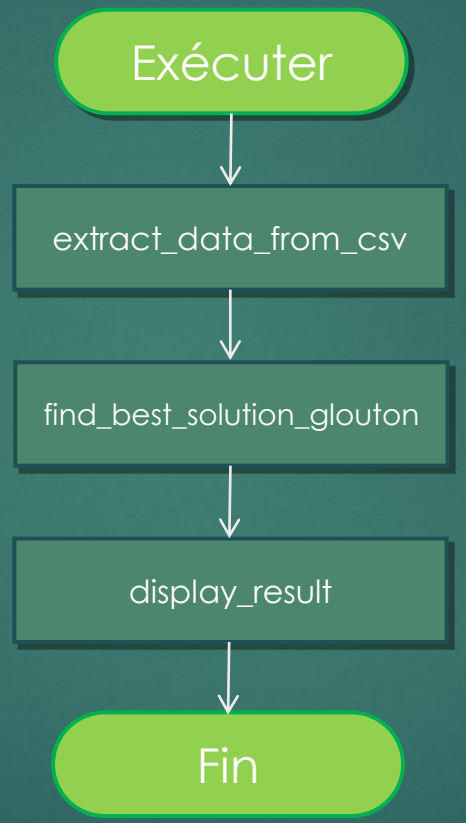
Brute force :



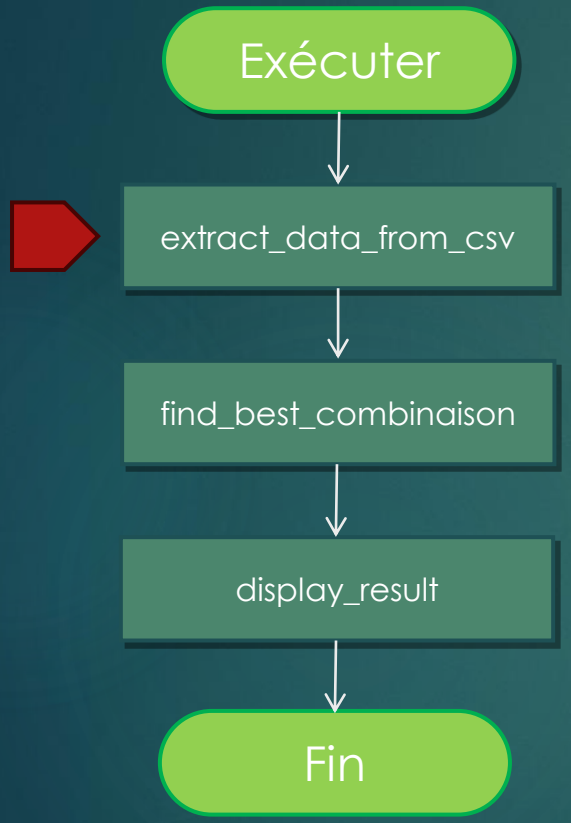
- bruteforce.py



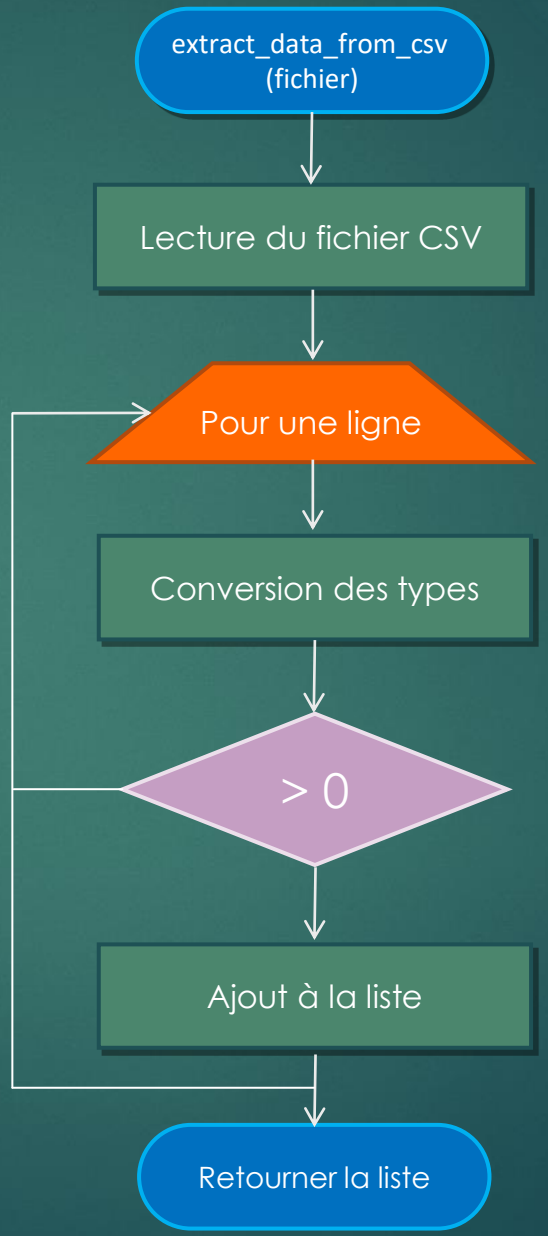
Optimized : 2eme solution.



Optimized :

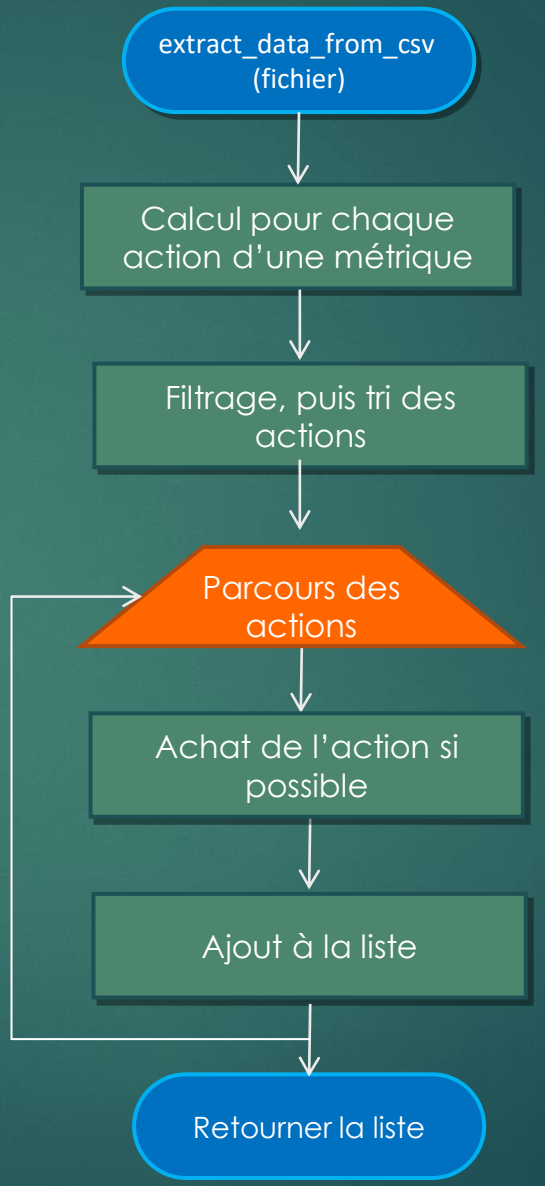


- utils.py

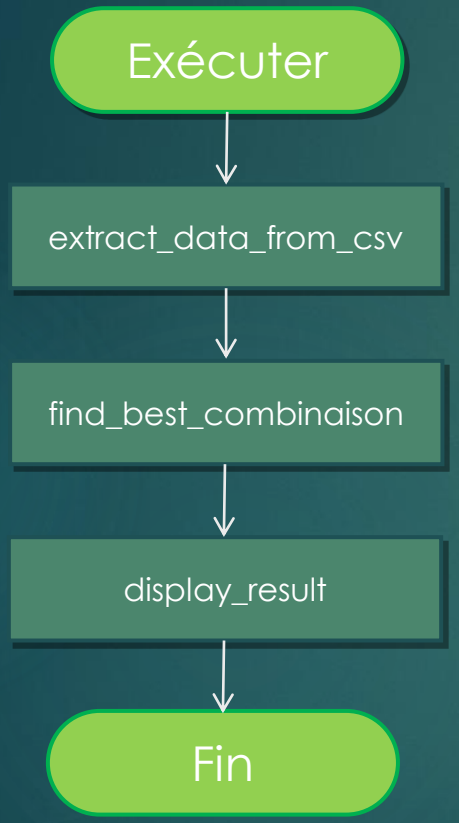


Optimized :

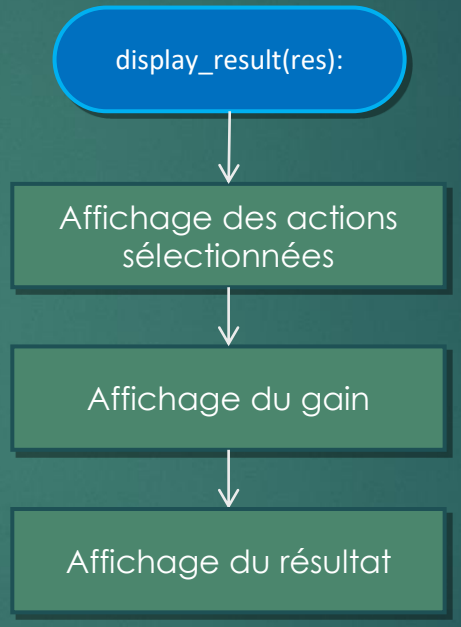
- utils.py



Optimized :

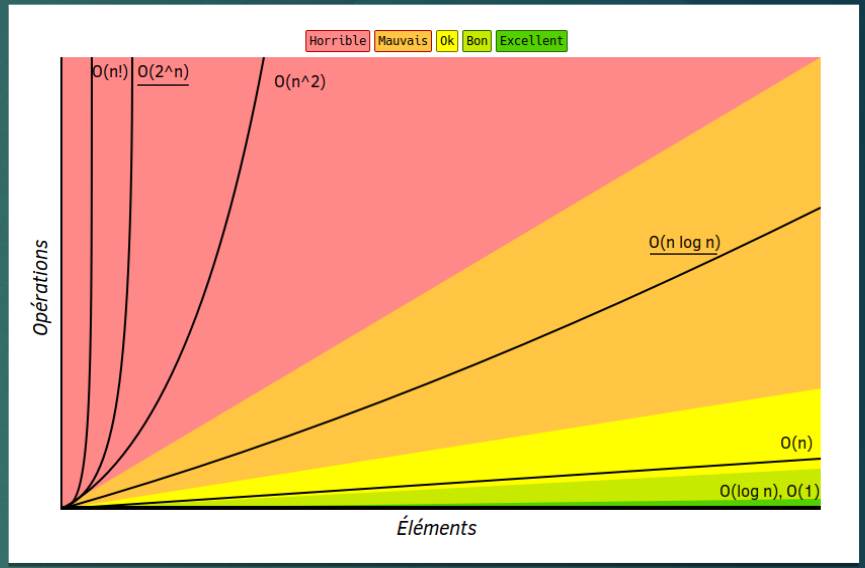


- optimized.py



Complexité Temporelle.

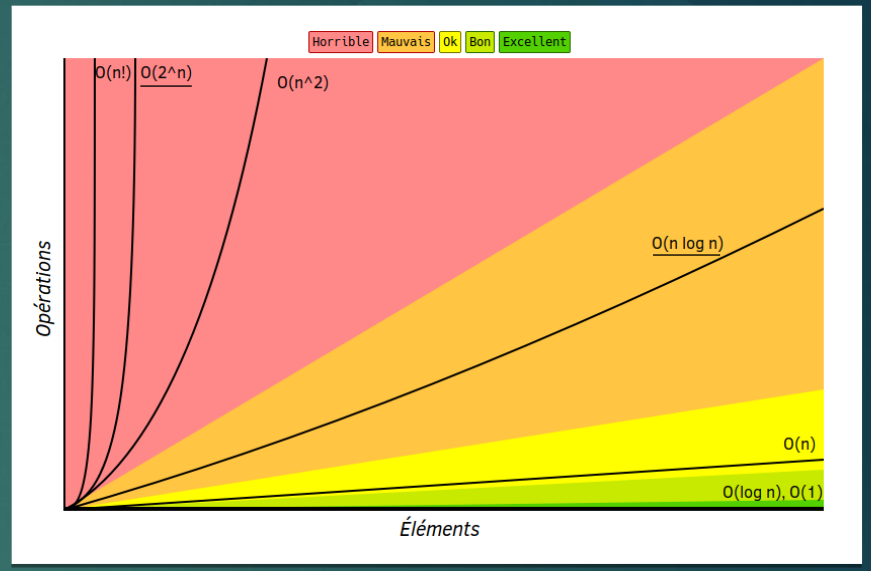
- Force brut : O(2^n).
  - L'algorithme de force brute est une approche simple qui peut être efficace pour résoudre des problèmes simples. Cependant, pour des problèmes complexes ou pour des ensembles de données de taille importante, l'algorithme de force brute peut être inefficace.
  - Par exemple : Si nous avons un ensemble de données de 10 éléments, le nombre de combinaisons possibles est de 2^10 = 1 024 (combinaison). L'algorithme de force brute devra donc essayer toutes ces combinaisons, ce qui prendra un temps proportionnel à 2^10, soit 1 024.



- Optimisé : O(n log n).
  - La complexité en O(n log n) est une complexité optimisée qui est souvent considérée comme le meilleur compromis entre performance et simplicité. Elle signifie que le temps d'exécution de l'algorithme est proportionnel au produit du nombre d'éléments et du logarithme naturel du nombre d'éléments
  - Par exemple, Si nous avons un ensemble de données de 100 éléments, le temps d'exécution d'un algorithme en O(n log n) sera proportionnel à 100 * log(100), soit environ 69 (combinaison).

Complexité Mémoire.

- Force brut : $O(2^n)$.
  - Il faut stocker l'ensemble des combinaisons à tester donc 2^n

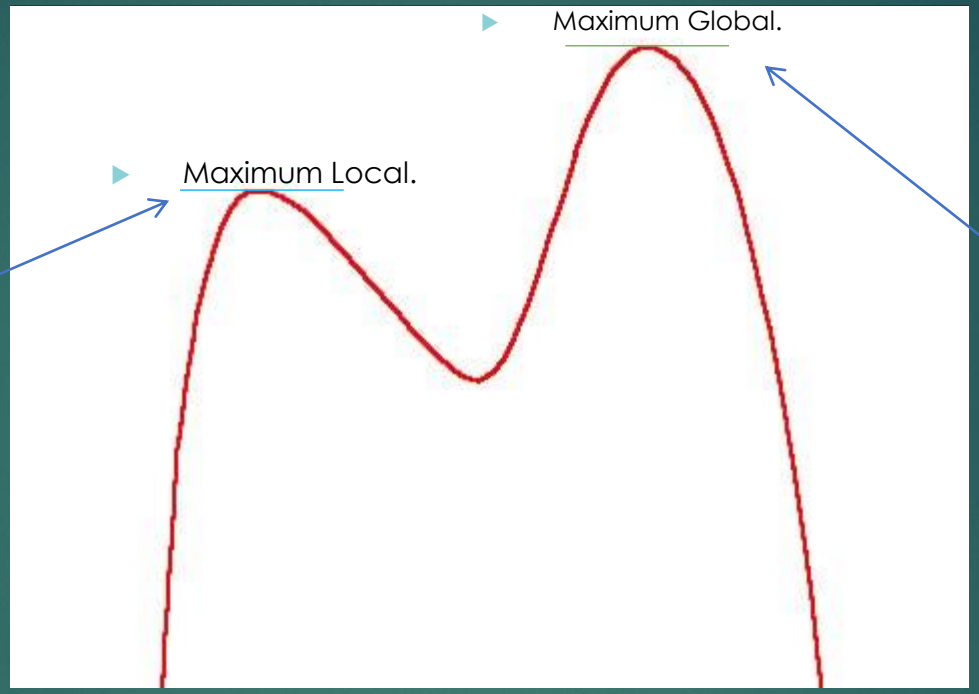


- Optimisé : O(n).
  - On stocke juste la solution que l'on construit au fur et à mesure donc au maximum n actions,

Une comparaison.

	DATASET N°1		DATASET N°2	
	SIENNA	ALGORITHME	SIENNA	ALGORITHME
COUT TOTAL (€)	498,76	499,94	489,24	499,98
BÉNÉFICE SUR 2 ANS (€)	196,61	198,51	193,78	197,77
TEMPS D'ÉXECUTION (s)	-	0.084282	-	0.062832
LISTE DES ACTIONS	SHARE-GRUT	Share-XJMO Share-KMTG Share-MTLR Share-GTQK Share-LRBZ Share-WPLI Share-GIAJ Share-GHIZ Share-IFCP Share-ZSDE Share-FKJW Share-NHWA Share-LPDM Share-QQTU Share-USSR Share-EMOV Share-LGWG Share-SKKC Share-QLMK Share-UEZB Share-CBNY Share-CGJM Share-EVUW Share-FHZN Share-MLGM	SHARE-ECAQ 3166 SHARE-IXCI 2632 SHARE-FWBE 1830 SHARE-ZOFA 2532 SHARE-PLLK 1994 SHARE-YFVZ 2255 SHARE-ANFX 3854 SHARE-PATS 2770 SHARE-NDKR 3306 SHARE-ALIY 2908 SHARE-JWGF 4869 SHARE-JGTW 3529 SHARE-FAPS 3257 SHARE-VCAX 2742 SHARE-LFXB 1483 SHARE-DWSK 2949 SHARE-XQII 1342 SHARE-ROOM 1506	Share-PATS Share-JWGF Share-ALIY Share-PLLK Share-NDKR Share-FWBE Share-LFXB Share-ZOFA Share-ANFX Share-LXZU Share-FAPS Share-XQII Share-ECAQ Share-JGTW Share-IXCI Share-DWSK Share-ROOM Share-VCXT Share-YFVZ Share-OCKK Share-JMLZ Share-DYVD

L'algorithme choisi et ses limites :



- La solution choisi est l'optimized.
- Ses limites :
  - pas forcement la meilleur solution, simplification du problème
  - Plus la taille des donnée en entrée est grande, plus la résolution est longue et exponentielle.