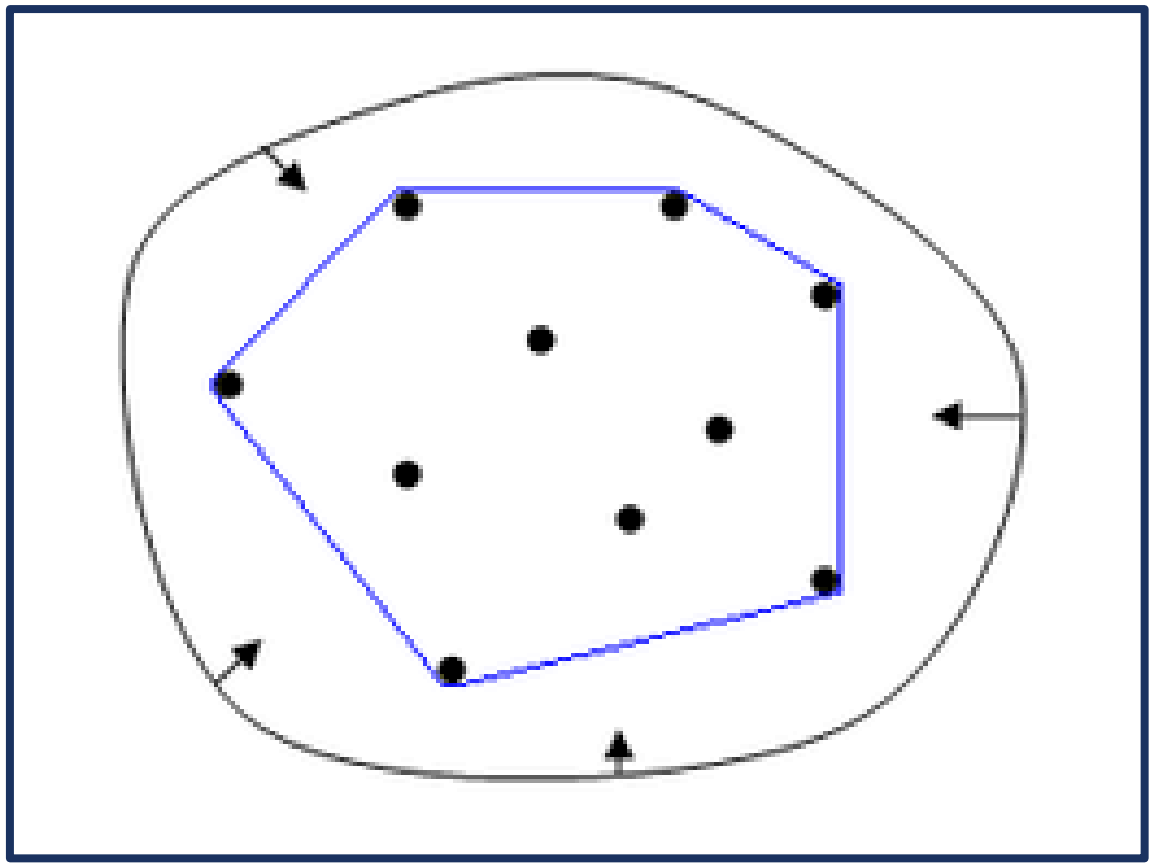
ENVELOPPE CONVEXE

MATHS-INFO

SOMMAIRE

INTRODUCTION



- Ensemble convexe le plus petit parmi ceux qui le contiennent
- Peut être comparée à la région limitée par un élastique qui englobe tous les points qu'on relâche jusqu'à ce qu'il se contracte au maximum.

OBJECTIFS

Créer un nuages de points aléatoirement

Trouver l'enveloppe convexe

Trois méthodes de calcul:

Graham

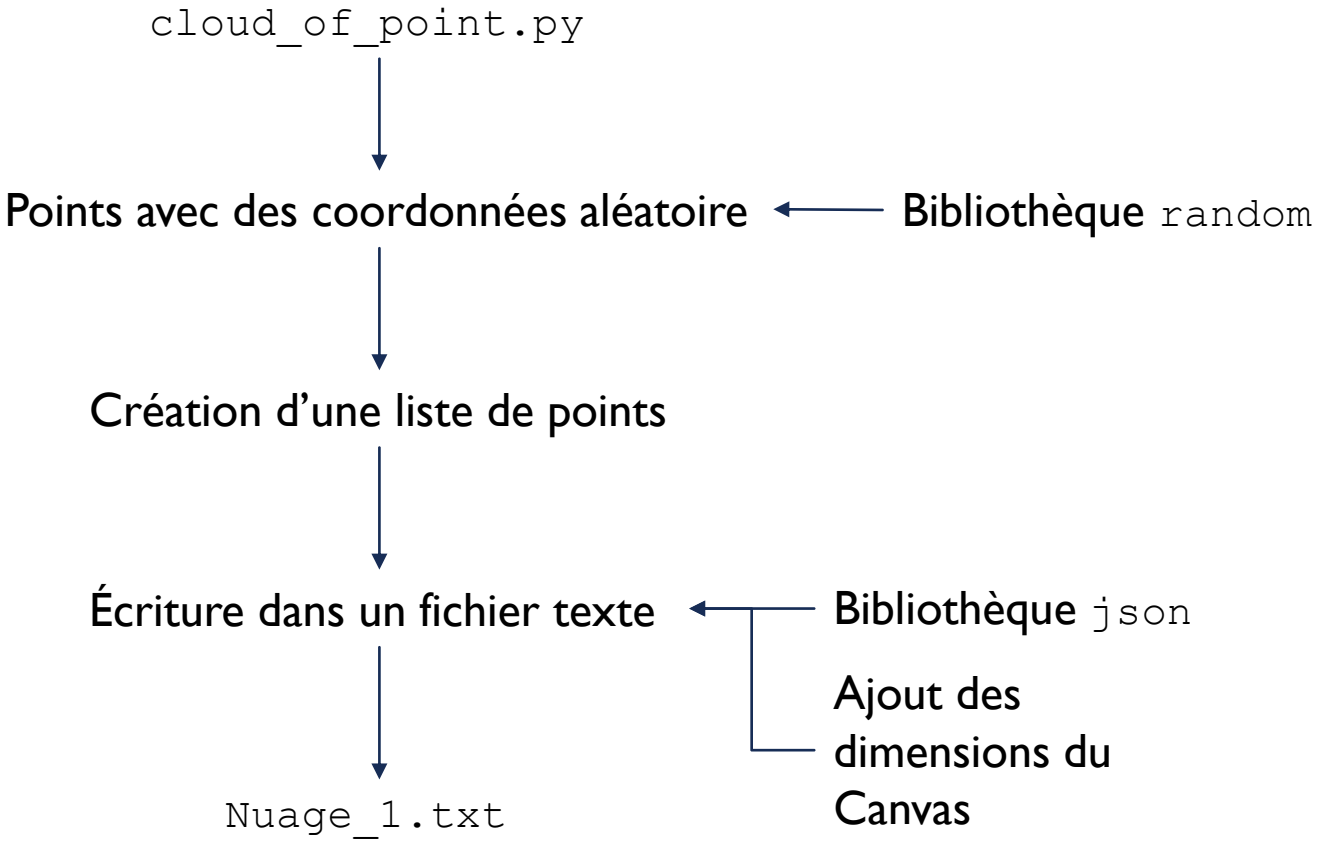
Jarvis

Quickhull

LECTURE ET ÉCRITURE DES NUAGES

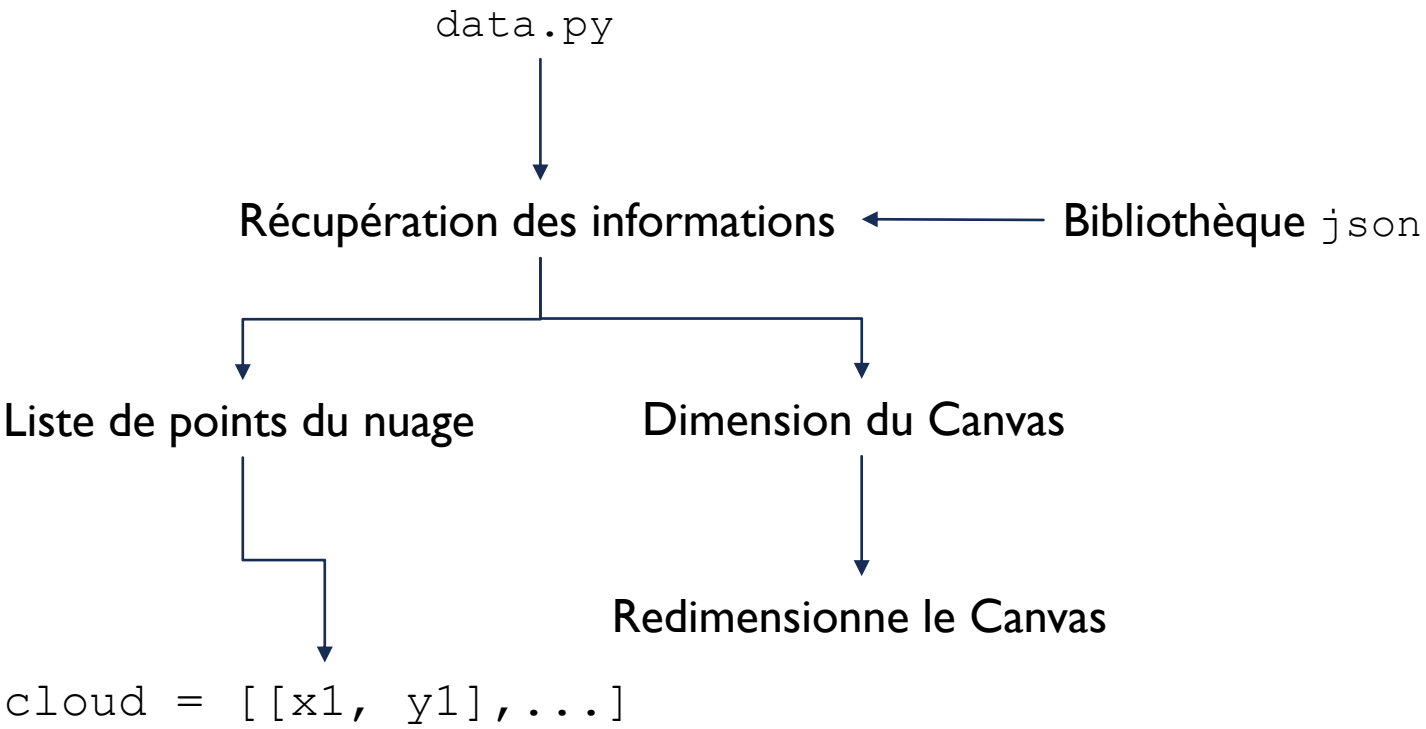
ANAÏS GALLERAND ; EDOUARD GAUTIER ; ANTOINE ORVAIN

ÉCRITURE




```
[[[219, 379], [44, 104],
[82, 235], [437, 445], [98,
116], [33, 276], [199,
319], [111, 56], [66, 95],
[242, 53], [297, 322],
[101, 90], [390, 278],
[439, 305], [435, 21],
[123, 75], [248, 319],
[408, 20], [246, 145],
[472, 182], [114, 248],
[156, 316], [206, 168],
[275, 395], [399, 449]],
[500, 500]]
```

LECTURE


data.py
Récupération des informations
Bibliothèque json
Liste de points du nuage
Dimension du Canvas
Redimensionne le Canvas
cloud = [[x1, y1],...]

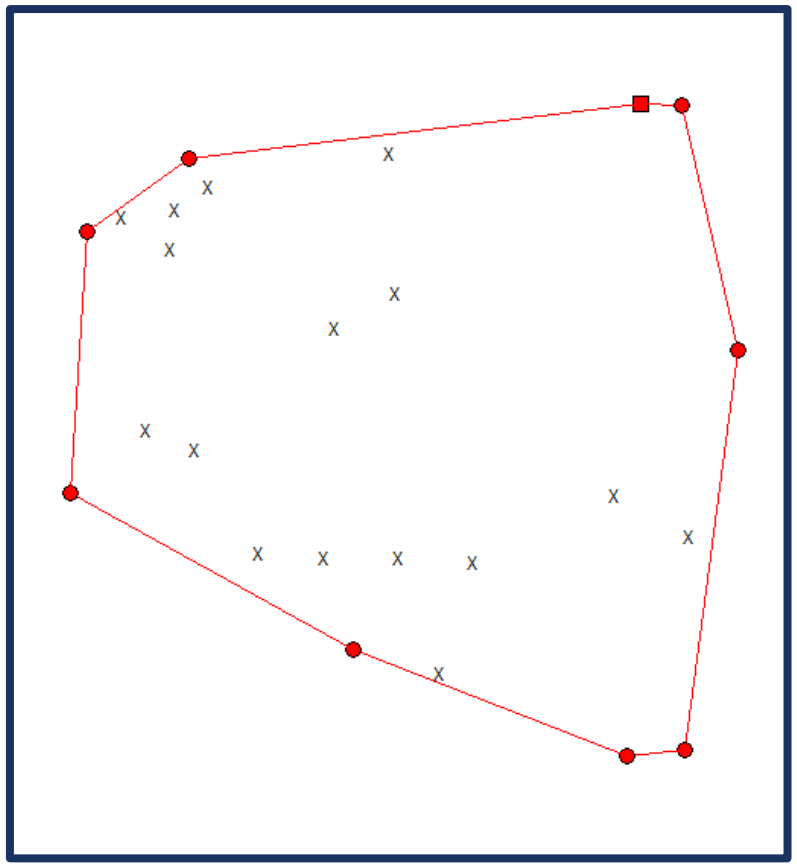
MÉTHODE DE GRAHAM

RONALD LEWIS GRAHAM

Wikipédia

- Mathématicien
- Né le 31 octobre 1935 en Californie
- « Un des principaux architectes du développement rapide des mathématiques discrètes ces dernières années à l'échelle mondiale »
- A travaillé sur :
  - Théorie de l'ordonnancement
  - La géométrie algorithmique
  - La théorie de Ramsey et les suites quasi-aléatoires
  - Inventeur de la méthode de Graham pour résoudre le problème de l'enveloppe convexe

MÉTHODE DE GRAHAM



- Algorithme pour le calcul de l'enveloppe convexe d'un ensemble de points dans le plan
- Publication de l'algorithme original en 1973
- Création d’une classe Graham
- Notions mathématiques abordées : méthode de la tangente, produit vectoriel

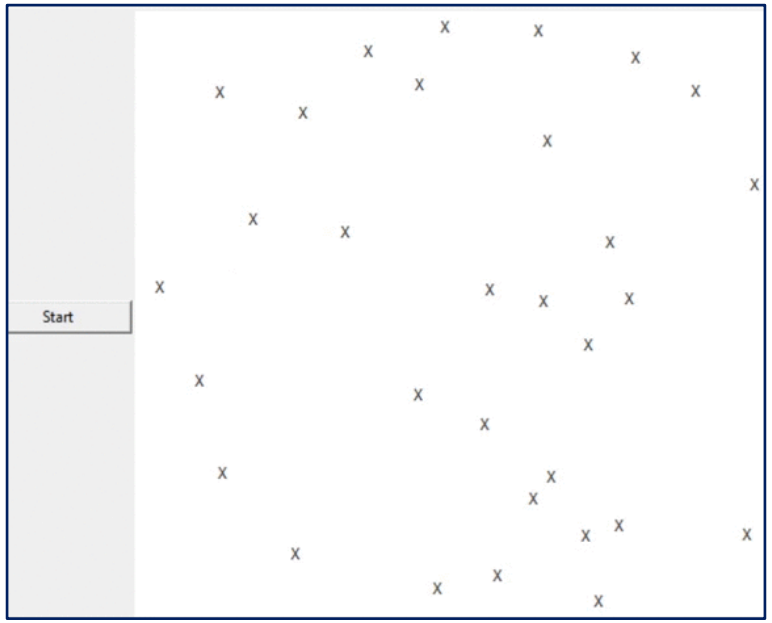
RECHERCHE DE L'ORIGINE RELATIVE

`self.cloud = [[x1, y1],...]` → `def find_origin(self)` → `self.origin = [x,y]`

Représentation du nuage de points sous la forme d'une liste de points, Eux-mêmes une liste de leurs coordonnées

Recherche du point avec la plus petite ordonnée et abscisse

Retourne une liste qui représente le point d'origine



TRI DE LA LISTE

```
self.cloud = [[x1, y1],...]
```

```
self.cloud = [[x2, y2], [x9, y9], …]
```

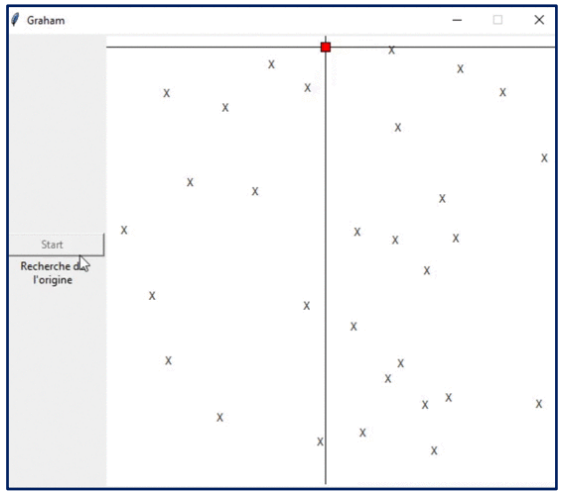
Liste de points triée dans l'ordre croissant des angles (points numérotés dans l'animation)

Méthode du tri fusion

```
def
sorting_fusion(self, list_fusion: list)
```

```
def angle(self, point: list)
```

Tri selon le critère de l'angle trigonométrique
Méthode de la tangente



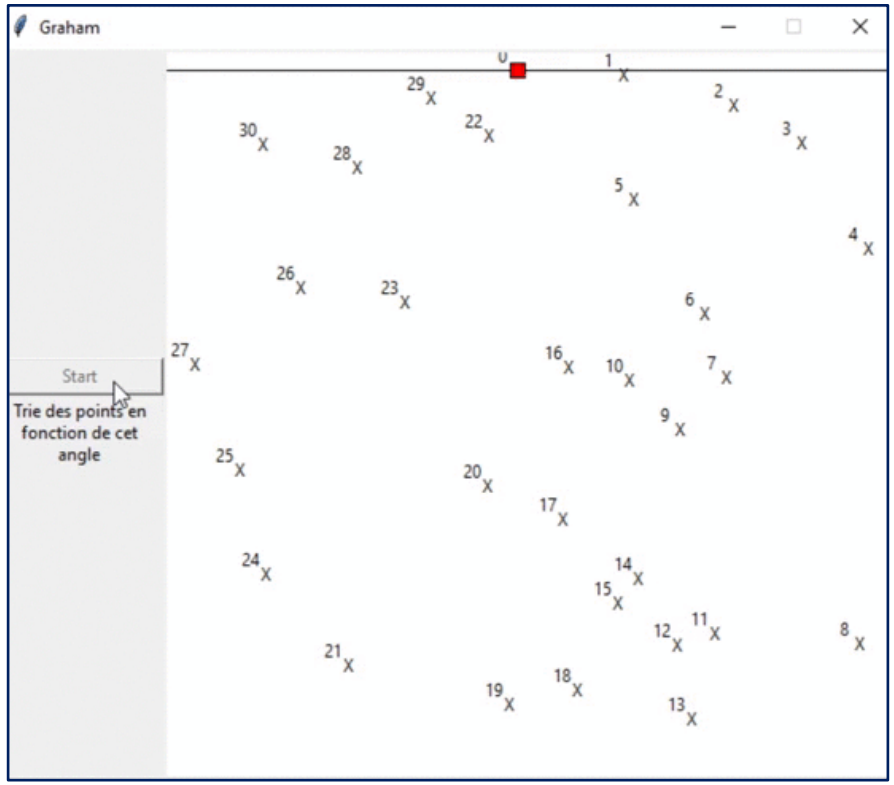
RECHERCHE DE L'ENVELOPPE CONVEXE

Segment formé des deux derniers points de l'enveloppe

On regarde quel point se trouve le plus à gauche du segment

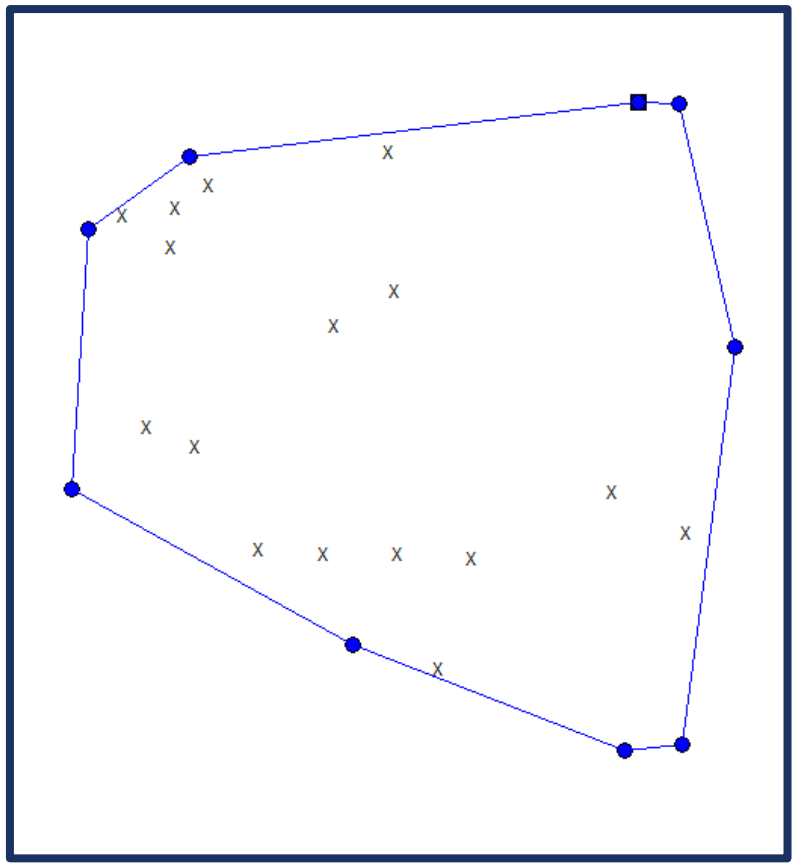
```
def vectorial_product(self,
point_a: list,
point_b: list,
point_c: list)
```

Le point le plus à gauche fait partie de l'enveloppe



MÉTHODE DE JARVIS

MÉTHODE DE JARVIS



- Algorithme pour calculer l'enveloppe convexe d'un ensemble fini de points
- Création d'une classe Jarvis
- Notions mathématiques abordées : méthode de la tangente, produit vectoriel

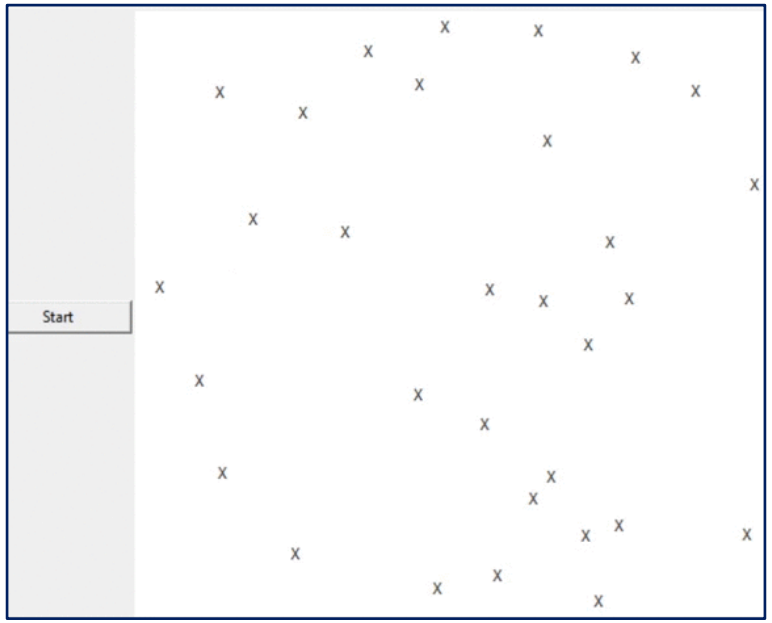
RECHERCHE DE L'ORIGINE RELATIVE

`self.cloud = [[x1, y1],...]` → `def find_origin(self)` → `self.origin = [x,y]`

Représentation du nuage de points sous la forme d'une liste de points, Eux-mêmes une liste de leurs coordonnées

Recherche du point avec la plus petite ordonnée et abscisse

Retourne une liste qui représente le point d'origine



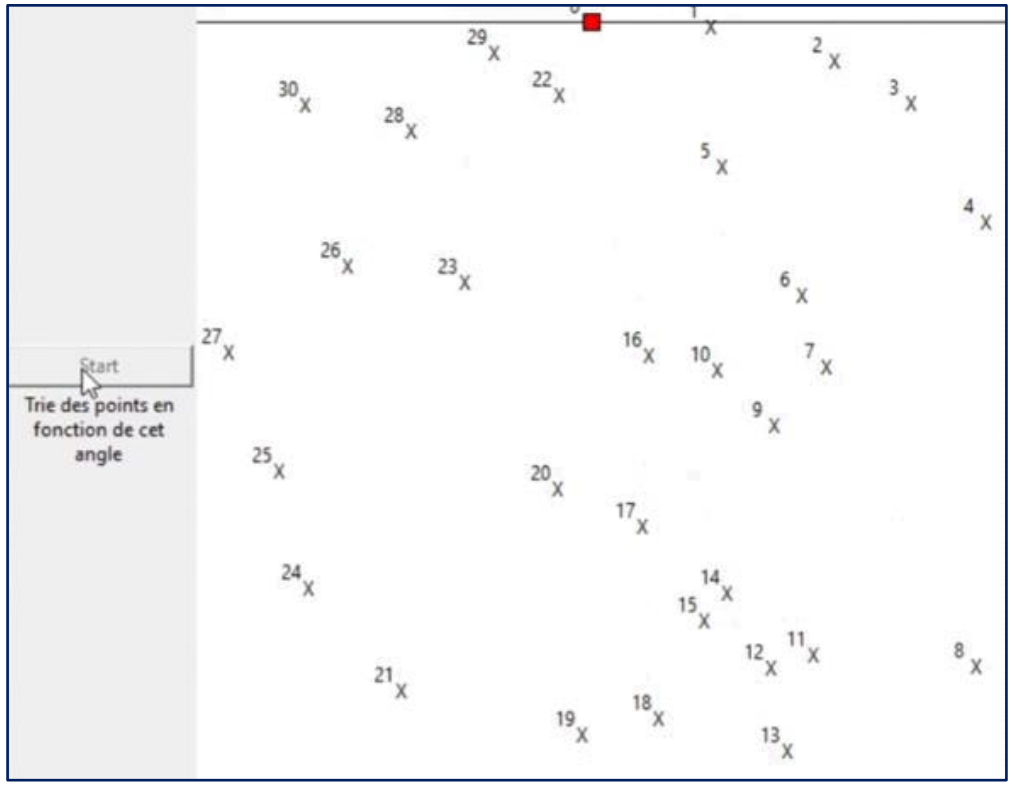
RECHERCHE DE L'ENVELOPPE CONVEXE

Segment formé des deux derniers points de l'enveloppe

Recherche du point avec le plus grand angle

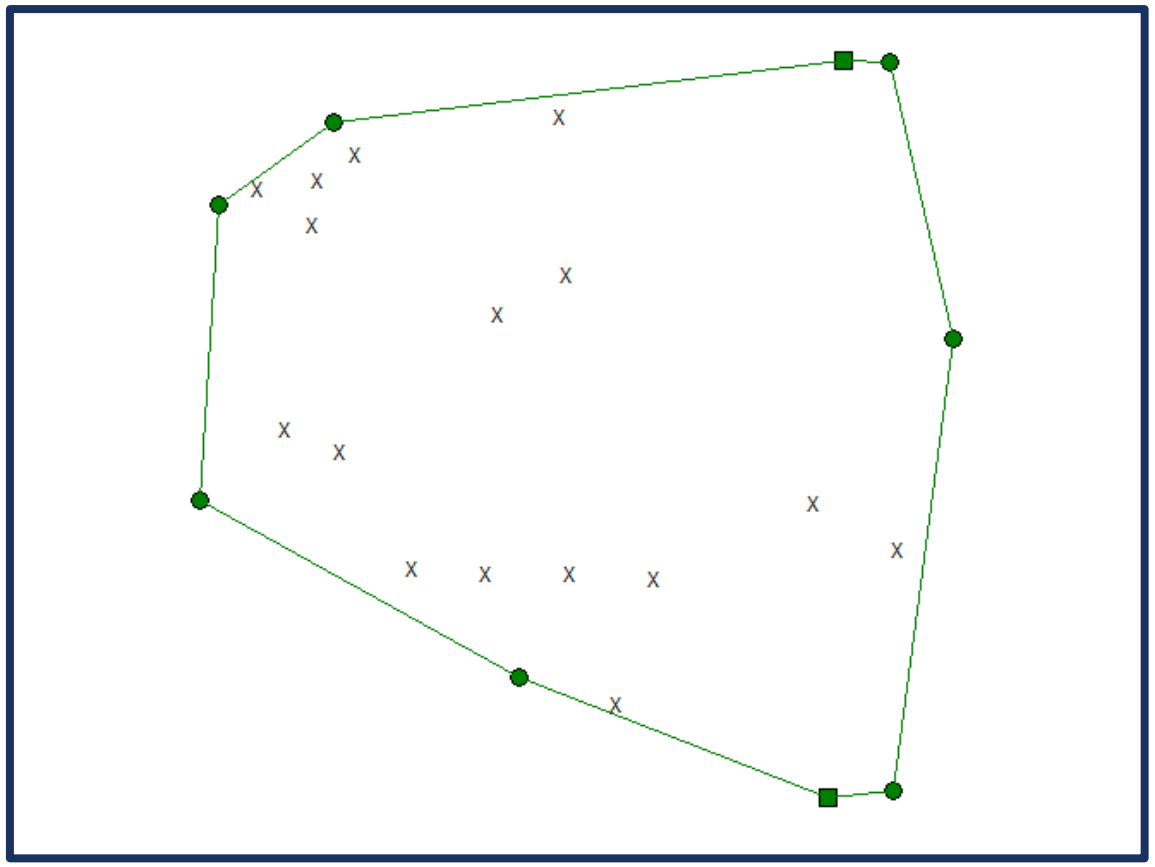
```
def find_next_point(self,
previous_point_1: list,
previous_point_2: list)
```

Le point le plus à gauche fait partie de l'enveloppe



MÉTHODE DE QUICKHULL

MÉTHODE DE QUICKHULL



- Algorithme pour calculer l'enveloppe convexe d'un ensemble fini de points
- Algorithme du type diviser pour régner
- Création d'une classe Quickhull
- Notions mathématiques abordées : méthode de la tangente, produit vectoriel

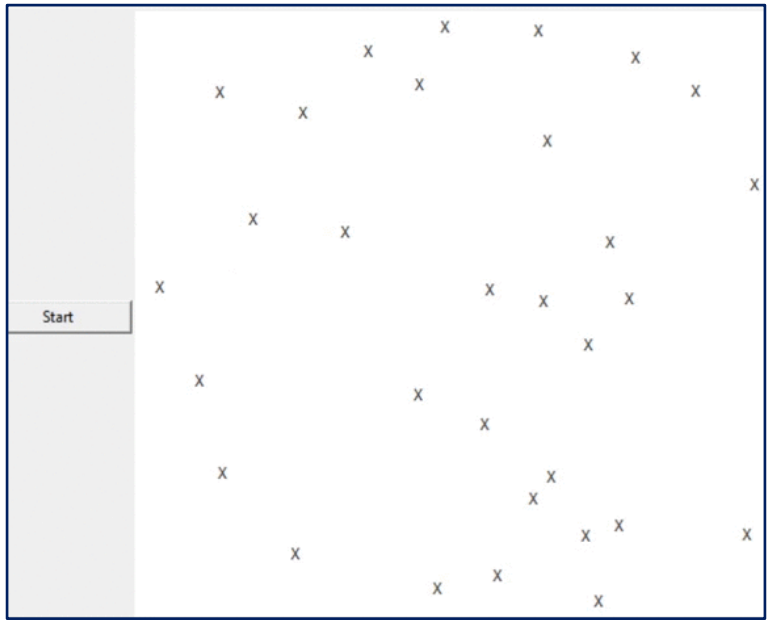
RECHERCHE DE DES ORIGINES RELATIVES

```
self.cloud = [[x1, y1],...]
```

Représentation du nuage de points sous la forme d'une liste de points, Eux-mêmes une liste de leurs coordonnées

```
def find_origin(self)
```

Recherche du point avec la plus petite ordonnée et abscisse et du point avec la plus grande ordonnée et abscisse




```
self.origin_min = [x,y]
self.origin_max = [x,y]
```

Retourne deux listes qui représentent les points d'origine

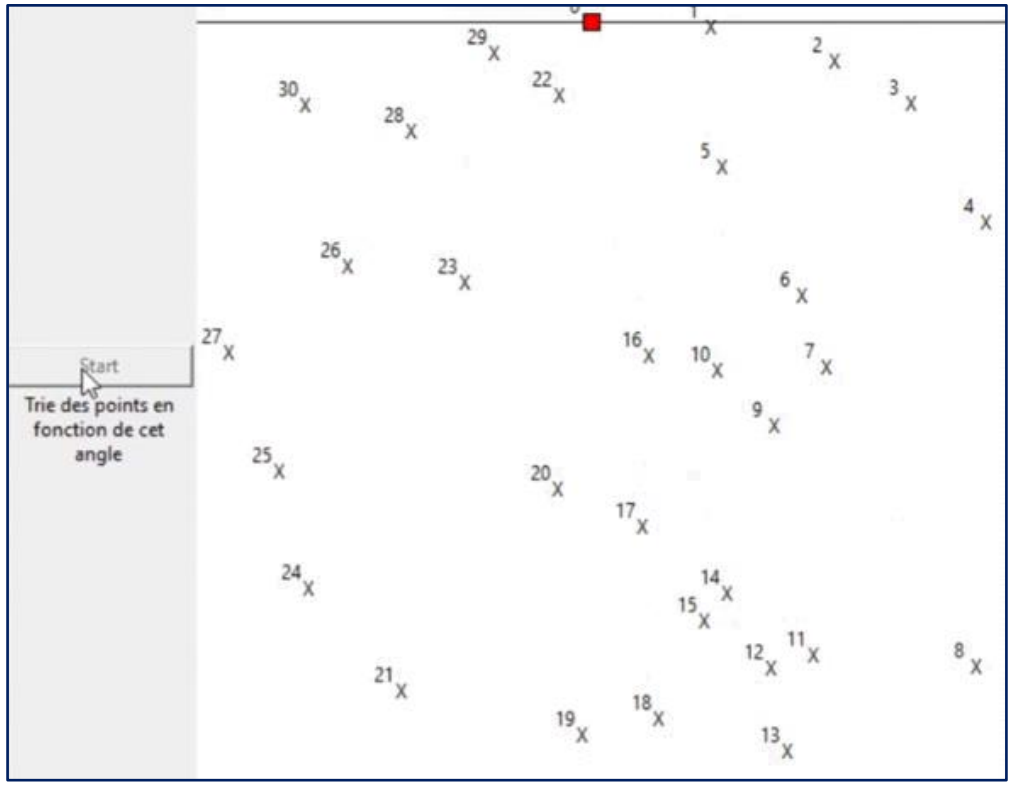
RECHERCHE DE L'ENVELOPPE CONVEXE

On trace un segment entre les points

On recherche le point le plus à droite du segment

```
def find_hull(self,
list_point: list,
point_a: list,
point_b: list)
```

On divise le nuage en deux



LIMITES DU PROGRAMME

CONCLUSION