# TP2 – Algorithmes Gloutons

## Exercice 1 : Voyageur glouton

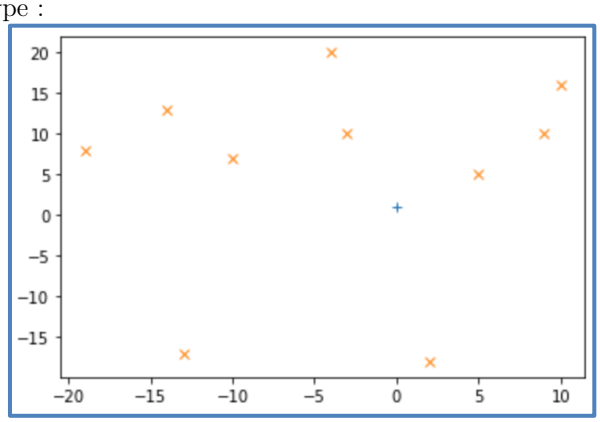
Dans cet exercice, on va mettre en place un algorithme glouton sur le voyageur de commerce. Pour cela on va créer un nombre de points aléatoires dans un carré. On définira un point de départ, et à chaque étape du chemin on utilisera une optimisation locale c'est-à-dire on ira au voisin le plus proche. Afin de vérifier notre algorithme on utilisera une représentation graphique pour visualiser tout cela.

1. Dans un premier temps on va créer une fonction « points(n,c) » qui va générer une liste de points de coordonnées (x,y). En entrée de la fonction on aura le nombre de points désirés, et la dimension du carré choisi.

```
from random import randint
nbpoints=10 #Nombre de points choisis
dim=20 #Dimension du carré

def points(n,c):
  liste=[]
  while len(liste)<n:
    x=randint(-c,c)
    y=randint(-c,c)
    if [x, y] not in liste:
      
  return liste
```

Compléter la ligne manquante du programme et rajouter des commentaires sur les différentes lignes du programme.

2. On va vérifier graphiquement que notre fonction points a bien généré le nombre de points voulus. Pour cela écrire sous python le script qui permettra de représenter les différents points.
On attend un résultat de ce type :




3. Afin de tester le corps principal du programme on va avoir besoin de quelques fonctions non inclues :

a) Créer une fonction « distance(p1,p2) » qui retourne la distance entre deux points p1 et p2.

b) Créer une fonction « distances(pts,dep) » qui retourne dans un tableau la distance du point i par rapport à tous les autres points. On s'aidera de la fonction distance créée précédemment. Compléter le script suivant et rajouter des lignes de commentaires.

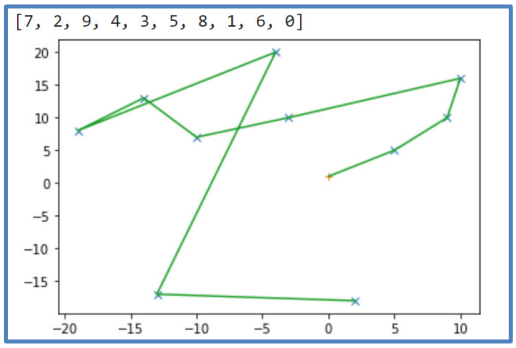
```
def distances(pts,dep):
  n=len(pts)
  tab=[(n+1)*[0] for i in range(n+1)]
  for i in range(n):
    for j in range(i):




  return tab
```

4. On va maintenant s'intéresser à la stratégie gloutonne du voyageur. Pour cela on va créer une fonction indice puis une fonction glouton.
   a) La fonction indice est donnée. Elle cherche le point le plus proche du point i dans une liste puis retourne la valeur de l'indice du point le plus proche jusqu'à la fin des n points demandés.

```
#On crée une fonction indice qui applique la stratégie gloutonne en allant au
#point le plus proche du précédent.
def indice(position, dist, dispo):
  n=len(dist)-1
  mini=2*sqrt(2)*dim #distance maximale entre 2 points dans le carré de -dim à dim
  for i in range(n): #On va tester tous les points parmi les restants
    if dispo[i]: #Si true, c'est ok, si false c'est déjà pris
      d=dist[position][i]
      if d<mini:
        mini=d
        ind=i
  return ind
```

   b) Il va falloir utiliser cette fonction indice pour avoir le chemin glouton. A l'aide de la fonction indice, créer une fonction « glouton(dist) » qui retourne la liste de points de la stratégie gloutonne.

5. Pour finir le programme on va réaliser deux affichages
   a) On afficher la liste des indices à l'aide de la fonction glouton.
   b) On trace le chemin glouton sur le graphique et on vérifiera que l'algorithme glouton généré effectue bien la démarche souhaitée.




## Exercice 2 : Rendu de monnaie

La fonction monnaie du cours utilise une liste pour représenter les pièces rendues dans le problème du rendu de monnaie. Ecrire une fonction qui utilise un dictionnaire dont les clés sont les valeurs des pièces rendues et les valeurs associées aux clés sont le nombre de pièces correspondantes. Tester la fonction pour rendre 19 euros avec le système (1, 2, 5, 10, 20, 50, 100, 200).

## Exercice 3 : Activités scolaires

Supposons avoir une liste d'activités, chacune associée à un créneau horaire défini par une heure de début et une heure de fin. Deux activités sont compatibles si leurs créneaux horaires ne se recouvrent pas. On souhaite sélectionner un nombre maximal d'activités toutes compatibles entre elles.

1. On se donne des activités sur les créneaux suivants : 8h-13h, 12h-17h, 9h-11h, 14h-16h, 11h-12h. Combien de ses activités peuvent-elles être conciliées sur une journée ?
2. On propose une stratégie gloutonne pour sélectionner des activités en commençant par le début de la journée : choisir l'activité dont l'heure de fin arrive le plus tôt (parmi les activités dont l'heure de début est bien postérieure aux créneaux des activités déjà choisies). Appliquer cette stratégie à la situation précédente.
3. On suppose avoir n activités numérotées de 0 à (n-1) et deux tableaux début et fin de taille n tels que debut[i] et fin[i] contiennent respectivement l'heure de début et l'heure de fin de l'activité numéro i.
   a. Ecrire une fonction prochaine(debut, fin, h) qui sélectionne parmi les activités dont l'heure de début n'est pas antérieure à h, une s'arrêtant plus tôt. On demandera à la fonction de renvoyer None s'il n'y a aucun créneau disponible.
   b. En déduire une fonction selection(debut,fin) qui, en supposant que toutes les heures sont positives, sélectionne autant d'activités que possible en suivant la stratégie gloutonne. On demandera à la fonction d'afficher les numéros des activités sélectionnées.