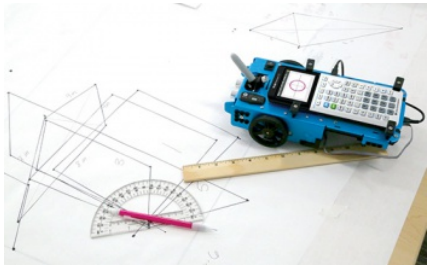
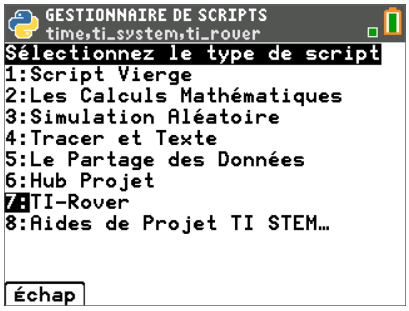
Unité 6 : utiliser la librairie ti_hub & ti_rover	Application : Représenter un parcours
Dans cette application de l'unité 6, vous allez connecter le ti-rover à l'aide de la librairie **ti_hub** et construire un script permettant d'enregistrer des coordonnées de points lors d'un parcours, puis de les représenter graphiquement.	**Objectifs :** • Découvrir le module **ti_rover.** • Écrire et utiliser un script permettant d'utiliser le ti-rover et ses actionneurs associés. • Utiliser une boucle fermée. • Représenter graphiquement des données.

Vous allez, dans cette leçon, réaliser un script donnant au ti-rover la possibilité d'effectuer un parcours correspondant au dessin d'un polygone.
Les coordonnées des sommets seront sauvegardées dans des listes puis représentées graphiquement à l'aide des instructions de la librairie **ti_plot**.

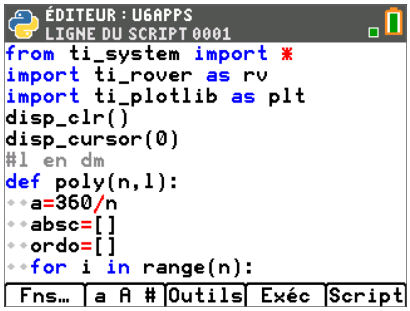


Mise en œuvre :

- Commencer un nouveau script et le nommer U6APPS
- A partir du menu **modul**, importer la librairie ti-rover
- Valider en appuyant sur la touche entrer.
- Importer également la librairie **ti_plotlib**



- Bien que cela ne soit pas impérativement nécessaire, les instructions permettant d'effacer l'écran et de ne pas afficher le curseur sont toujours d'un rendu plus agréable. Insérer **disp_clr()** et **disp_cursor(0)** se trouvant dans le menu **modul** puis **ti_system**.
- Créer une fonction **poly(),** prenant en argument n le nombre de côté du polygone et l la longueur en unité par défaut pour le TI-Rover, c'est-à-dire le dm.
- L'angle au sommet de chaque polygone sera donc égal $a = \frac{360}{n}$.
- Créer deux listes vides **absc** et **ordo** destinées à recevoir les coordonnées de chacun de ces sommets.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

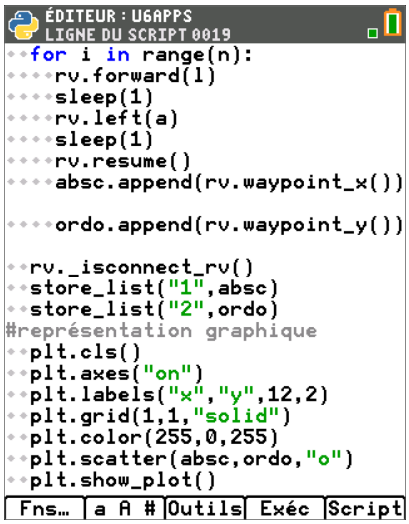
- Créer une boucle ouverte de taille n.

Les instructions suivantes se trouvent dans la librairie **ti_rover**.

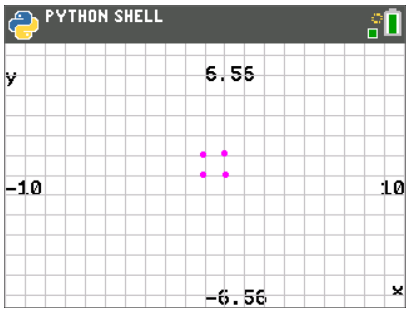
- Faire avancer le rover de $\boldsymbol{l}$ décimètre pour $\boldsymbol{n}$ fois.
- Tourner sur la gauche d'un angle $\boldsymbol{a}$.
- Mettre entre chaque étape un délai de 1s.
- Stocker dans les listes x et y les coordonnées des points.
- Déconnecter le rover à la fin du parcours.
- Exporter les coordonnées **absc** et **ordo** respectivement dans les listes L_1 et L_2.

Conseil à l'enseignant : les instructions **rv.waypoint_x()** et **rv.waypoint_y()** se trouvent dans le menu **E/S** puis **3 : le chemin** de la librairie **TI_Rover**.

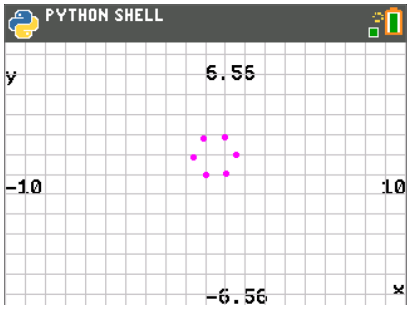
- Passer ensuite à la représentation graphique, celle-ci est ici incluse dans la fonction, mais il est toujours possible de créer une fonction graphe séparément comme nous avons pu le faire dans d'autres leçons (Unité 5).



Fonctionnement du script :
Exécuter votre script **F4** et appeler la fonction **poly( )** en lui fournissant par exemple les arguments poly(4, 1) afin de dessiner un carré de 1 dm de côté.

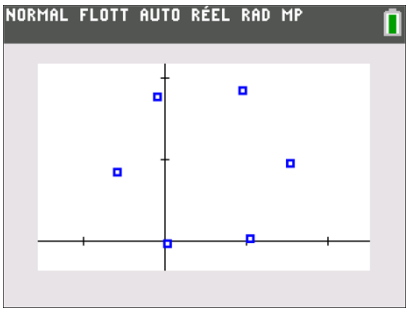


Un essai avec un hexagone de 1 dm fournit la représentation suivante.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/

L'export des listes vers la calculatrice donne la représentation graphique ci-contre. L'utilisation de la touche trace permettra de retrouver les coordonnées de chaque point.



Conseil à l'enseignant : Pour ce type d'exercice, éviter d'utiliser les instructions **rv.pathlist_x()** et **rv.pathlist_y()**.
En fait lors du tracé d'un segment, la calculatrice enregistre les coordonnées des points d'un segment du polygone, puis réenregistre les coordonnées du dernier point comme premier point du segment suivant. D'autant plus que nous avons placé, entre chaque tracé, une temporisation de 1s.
Ainsi les instruction rv.path… sont dans notre cas inappropriées.
En utilisant les instructions **rv.pathlist_x()** et **rv.pathlist_y(),** on obtiendrai deux fois les coordonnées des extrémités.

Remarque : ajuster le format de votre grille, en fonction des polygones que vous souhaitez tracer. Celle-ci a volontairement été réglée ici aux paramètres par défaut, afin d'observer la précision du tracé du ROVER.

Prendre garde également à la nature de la surface sur laquelle se déplace le robot. Celle-ci ne doit pas opposer trop de résistance au déplacement ou au contraire, favoriser les glissements.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/