Aller plus loin avec Python sur la TI-83 Premium CE Edition Python

unité 4 → unité 5 → unité 6

Auteur : Jean-Louis Balas

Texas Instruments

Sommaire

Unité 4 : Utiliser le module ti_plotlib

Compétence 1 : Paramétrer une représentation

Dans cette première leçon de l'unité 4, vous allez découvrir comment écrire et utiliser une instruction permettant de faire des représentations graphiques en Python. Vous apprendrez également à représenter un graphique et paramétrer l'affichage.

Objectifs :

- Découvrir le module **ti_plotlib**.
- Représenter un point, un segment.
- Paramétrer une représentation graphique.

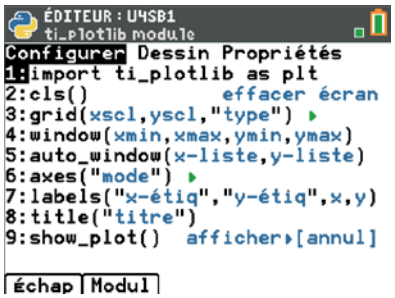
1 : La librairie ou module ti_plotlib

Pour effectuer une représentation graphique lors de l'exécution d'un script, celui-ci doit être en mesure de comprendre les instructions graphiques. Il est donc nécessaire « d'embarquer » les fonctions graphiques au sein d'une bibliothèque **ti_plotlib**.

Commencer un nouveau script en le nommant U4SB1 et inclure dans celui-ci le module ti-plotlib (touches : f(x) ◀). Choisir le menu **5 : ti_plotlib…** puis le menu **1 : import ti_plotlib as plt**.

Pour cette première partie, vous allez écrire un script permettant d'afficher un point dont les coordonnées sont connues. Ensuite, vous modifierez votre script afin de localiser votre point dans un repère et modifierez sa couleur.

Pour terminer cette première leçon, vous demanderez l'affichage du nom de chaque axe et donnerez un titre à votre graphique.

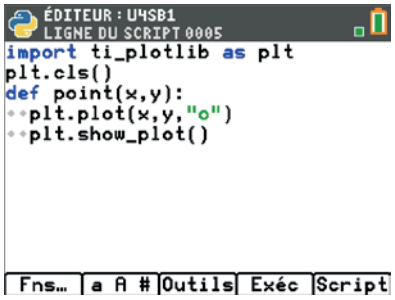


Définir une fonction ayant pour argument les coordonnées d'un point, puis demander l'affichage de ce point.

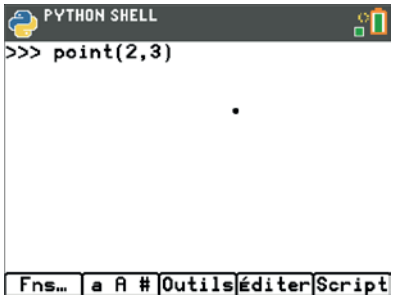
- Dans un premier temps, nettoyez votre écran en utilisant l'instruction **plt.cls()** que vous trouverez dans le module **ti_plotlib** au menu **Configurer** (**2 : cls()**).
- Pour dessiner le point, choisir l'instruction **6 : Tracé un point**, située dans le menu **Dessin** du **module ti-plotlib**.
- Choisir la marque désirée.

Conseil à l'enseignant : La représentation d'un point sous forme de pixel est à choisir dans le cas où un grand nombre de points est à représenter.

Terminer le script en demandant l'affichage de la représentation par le choix de l'instruction **plt.show_plot() (instruction n°9 du module ti-plotlib, menu configurer)**.



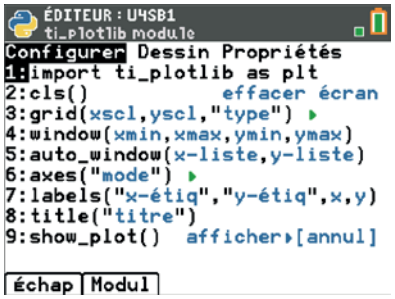
- Demander l'exécution du script (touche **F4**), puis appuyer sur la touche [var] afin de rappeler dans la console la fonction **point()**.
- Donner les coordonnées d'un point et observer votre écran.
- Appuyer sur la touche [annul] pour sortir de l'écran graphique, puis sur [var] afin de retrouver la liste des variables du script.
- Modifier les coordonnées de votre point (par exemple **point(10,10)** et constater que celui-ci n'est plus visible à l'écran.



Conseil à l'enseignant : Lors de l'écriture d'un script utilisant les fonctions graphiques, il sera nécessaire de préciser les paramètres de la fenêtre graphique et éventuellement d'afficher un repère, grille, nom des axes etc.

2 : Affiner votre représentation

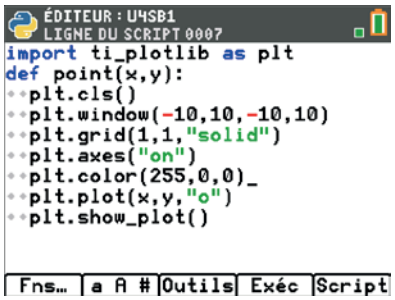
Inclure l'instruction **plt.cls()** sous la définition de la fonction, afin d'éviter d'avoir la barre de menu (**Fns** à **Script**) superposée à votre représentation graphique.



Conseil à l'enseignant : Pour couper, copier ou coller une ligne, utiliser les Outils (touche **F3**) à partir de l'éditeur de script.

A partir des différentes options du menu **Configurer** du module **ti_plotlib**, rajouter dans votre script les instructions permettant :

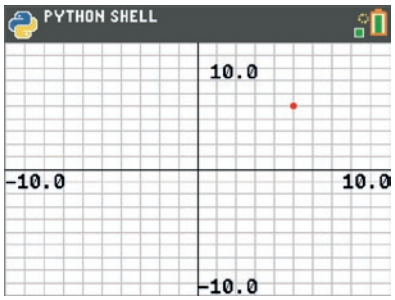
- De définir une fenêtre graphique telle que : X_{min} = -10 ; X_{max} = 10 ; Y_{min} = -10 et Y_{max} = 10 (**instruction 4 : window**).
- Afficher une grille (instruction **3 : grid**) le type de grille est laissé à votre choix.
- Afficher les axes (instruction **6 : axes**).
- Modifier la couleur du point (Menu **Dessin,** puis **1 : color(r,v,b)**).



Conseil à l'enseignant : La couleur d'un point ou d'un tracé est à préciser en code **r, v, b** (rouge, vert, bleu), chaque paramètre pouvant prendre une valeur dans l'intervalle [0 ; 255]. Les couleurs sont codées sur 8 bits, soit 2^8 = 256 possibilités, en comptant le 0 qui correspond à une absence de la composante **r** ou **v** ou **b.**

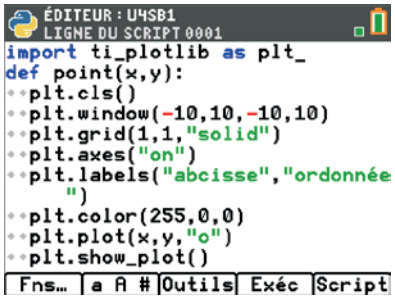
Il est également possible de tracer la grille en couleur en complétant l'instruction **grid(xscl, yscl, « style », (r,v,b)).**

Exécuter votre script et observer les changements. Vous devriez obtenir un écran identique à celui ci-contre.



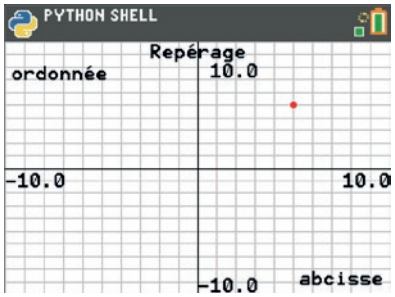
Conseil à l'enseignant : Par ailleurs lors de l'exécution d'un script, la console est réinitialisée. L'historique est accessible en utilisant les touches de direction de la calculatrice. Mais attention, cet historique est perdu lors d'une nouvelle réinitialisation.

Modifier à nouveau le script afin de donner un nom à vos axes. Par exemple (« abscisse » et « ordonnée »).
Pour cela, inclure dans votre script (peu importe l'emplacement) une ligne **plt.labels**. Celle-ci se trouve à l'emplacement **7 : labels()** du menu **Configurer** dans le module **ti_plot**.



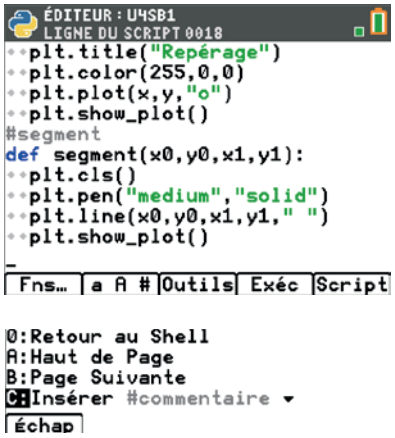
Conseil à l'enseignant : l'instruction **labels(« x-étiq », « y-étiq »,x ,y)** nommera les axes en plaçant les étiquettes aux lignes et colonnes x et y par défaut, celles-ci sont placées en ligne 12 pour x et 2 pour y, respectivement justifiées à gauche et à droite.

Si vous le souhaitez, vous pouvez également ajouter un titre à l'aide de l'instruction **plt.title(« Repérage »)**.



Rappel : Les caractères accentués sont obtenus à partir de l'éditeur de script en appuyant sur la touche F2 [a A #], puis en utilisant les touches de direction afin de choisir dans la palette le caractère souhaité.

Pour aller plus loin : Compléter votre script en écrivant une fonction vous permettant de représenter graphiquement un segment. Vous trouverez ci-contre les instructions essentielles. Vous pouvez bien entendu si vous le souhaitez modifier le script afin d'afficher les axes, une grille …etc.
L'option **plt.pen** accessible via le menu Dessin (**9 : pen(« taille », « type »)**), permet de paramétrer la largeur du trait.



Conseil à l'enseignant : A partir de l'éditeur de script, en appuyant sur les touches [2nde] [3], vous pouvez écrire un commentaire. Celui-ci est écrit en gris et précédé d'un **#**, signifiant que cette instruction ne sera pas exécutée.
Le caractère # peut également être atteint en utilisant l'option **C : Insérer du menu** Outils (touche **F3**).

Unité 4 : Utiliser le module ti_plotlib — Compétence 2 : Représenter graphiquement une fonction

Dans cette seconde leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une fonction en utilisant la librairie Python **ti_plotlib**.

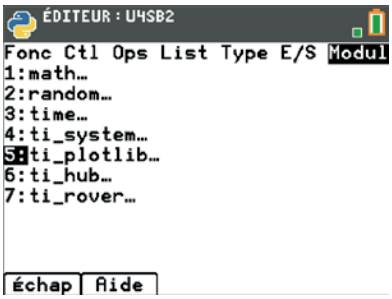
Objectifs :

- Représenter graphiquement une fonction.
- Réinvestir la notion de boucle fermée.
- Paramétrer la représentation graphique.

On se propose de réaliser un script utilisant la librairie **ti_plotlib** et permettant de tracer la représentation graphique d'une fonction pour x prenant ses valeurs dans un intervalle $[a\,;\,b]$ avec N segments.

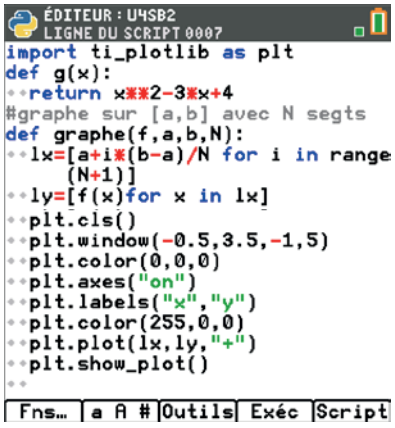
Le script que vous allez créer sera très général afin de pouvoir être réinvesti pour d'autres exemples.

Conseils à l'enseignant : Si la notion de boucle ne vous est pas familière, vous êtes invités à travailler les unités 1,2 et 3 du TI-Code Python.



- Commencer un nouveau script et le nommer U4SB2.
- Importer le module **ti_plotlib**, celui-ci s'obtient en appuyant sur la touche f(x) ◂ puis en choisissant **5 : ti_plotlib.**
- Définir la fonction $g\colon x \longmapsto x^2 - 3x + 4$.

Conseils à l'enseignant : la liste des abscisses est construite à l'aide d'une boucle fermée dont l'instruction se trouve en appuyant sur f(x) ▸, puis en choisissant dans les instructions de contrôle **4 : for i in range(taille) :**
En revanche celle des ordonnées **ly** est construite à partir de la liste **lx** des abscisses. On choisira donc l'instruction **7 : for i in liste**.



A présent vous êtes prêts pour effectuer la représentation graphique de la fonction. Insérer les instructions permettant de :

- Nettoyer l'écran : **plt.cls()**.
- Régler les paramètres de la fenêtre graphique : **plt.window(xmin, xmax, ymin, ymax)**.
- Afficher les axes du repère : **plt.axes(« on »)**.
- Afficher le nom des axes : **plt.labels(« x », « y »)**.
- Effectuer la représentation graphique **plt.plot(lx,ly, « + »)**.
- Afficher la représentation graphique : **plt.shox_plot()**.

L'ensemble de ces instructions se trouve dans le module **ti_plotlib,** les paramètres de réglages de la représentation graphique dans le menu **Configurer** et l'instruction de représentation graphique dans le menu Dessin puis **5 : Tracé connecté avec listes**.

Remarque : les instructions pour le choix de la couleur en RVB (rouge, vert, bleu) sont à coder entre 0 et 255 et à placer judicieusement, afin d'éviter par exemple d'avoir les axes en rouge.

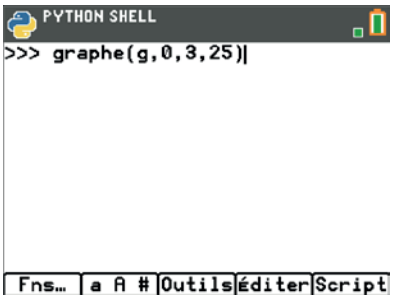
Exécuter le script puis choisir la fonction **graphe()** en appuyant sur la touche var .

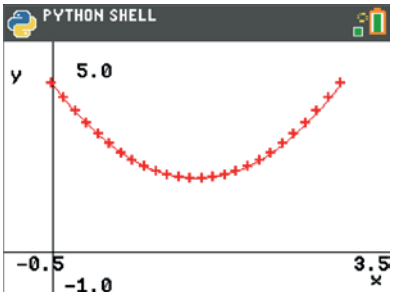
Demander la représentation graphique de la fonction contrainte sur l'intervalle [0 ; 3] avec 25 segments.

Conseils à l'enseignant : Si vous souhaitez un grand nombre de segments, préférez une représentation avec des pixels (**.** plutôt que **+**).

Prolongements possibles :

- Afficher un quadrillage.
- Changer de fonction ou en étudier plusieurs.
- Réaliser une étude mathématique (calcul différentiel), représenter par exemple la fonction avec 6 segments puis 50.





Unité 4 : Utiliser le module ti_plotlib

Compétence 3 : Nuage de points et régression

Dans cette troisième leçon de l'unité 4, vous allez découvrir comment représenter graphiquement une série de données, puis rechercher un modèle mathématique s'ajustant au mieux au nuage de points.

Objectifs :

- Réaliser la représentation d'un nuage de points.
- Rechercher et utiliser un modèle de régression.
- Utiliser les listes en Python.

Position du problème : Lors d'une inondation, afin de fournir à la population des informations pratiques, les autorités enregistrent, chaque heure à partir du début de la décrue, la hauteur d'eau maximale par rapport à un point de référence.
Les données sont fournies dans le tableau ci-dessous. On souhaite représenter le nuage de points en utilisant la librairie **ti_plotlib**, puis rechercher un modèle mathématique permettant de faire une extrapolation, afin de prévoir le temps total de la décrue.

T(h)	0	1	2	3	4	5	6	7	8	9	10	11	12
H(cm)	130	127	123	118	116	111	105	103	101	95	86	80	71

Mise en œuvre : Commencer un nouveau script et le nomme U4SB3.

1) Importer le module **ti_plotlib**.
2) Enregistrer les données dans deux listes « **lt** » et « **lh** ».
3) Préparer ensuite la représentation graphique :
   - Effacer l'écran **plt.cls()**.
   - Régler les paramètres de la fenêtre graphique : x_{min} = -2 ; x_{max} = 20 ; y_{min} = -20 ; y_{max} = 200 en écrivant **plt.window(**-2, 20, -20, 200**)**.
   - Afficher les axes **plt.axes()**.
   - Afficher la représentation graphique sous forme d'un nuage de points **plt.scatter()**.

Les instructions de paramétrage de la fenêtre graphique se trouvent dans le module **ti_plotlib** dans le menu **Configurer**. La syntaxe de la représentation du nuage de points se situe quant à elle dans le menu **Dessin.**

ÉDITEUR : U4SB3
LIGNE DU SCRIPT 0001

```
import ti_plotlib as plt
lt=[0,1,2,3,4,5,6,7,8,9,10,11,12
    ]
lh=[130,127,123,118,116,111,105,
    103,101,95,86,80,71]
#représentation graphique
```

Fns… | a A # | Outils | Exéc | Script

ÉDITEUR : U4SB3
LIGNE DU SCRIPT 0001

```
import ti_plotlib as plt
lt=[0,1,2,3,4,5,6,7,8,9,10,11,12
    ]
lh=[130,127,123,118,116,111,105,
    103,101,95,86,80,71]
#représentation graphique
plt.cls()
plt.window(-2,20,-20,200)
plt.axes("on")
plt.scatter(lt,lh,"x")
plt.show_plot()
```

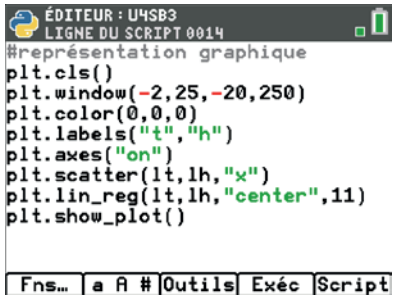
Fns… | a A # | Outils | Exéc | Script

Conseil à l'enseignant : la liste des données du temps, en heures, peut être complétée à l'aide d'une boucle fermée.

PYTHON SHELL

```
>>> lt=[i for i in range(13)]
>>> lt
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1
0, 11, 12]
>>>
```

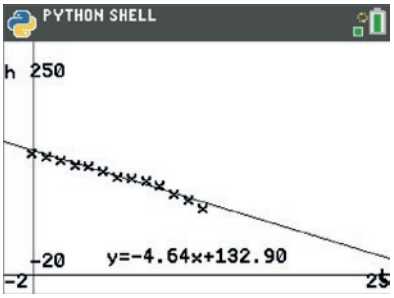
Exécuter le script et observer la représentation graphique. Si tout se passe correctement, vous devriez obtenir l'écran ci-contre.



L'ensemble des points semble-t-il aligné ?
Vous allez maintenant rechercher le modèle mathématique (affine) passant au mieux par l'ensemble des points du nuage.

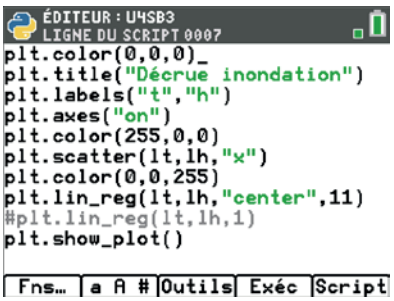
Pour cela, vous devez rajouter la ligne **plt.lin_reg(xliste,yliste, « center »,11)** dans votre script afin que la droite de régression soit calculée à partir des listes de données **lt** et **lh**, puis affichée et centré sur la ligne 11 de l'écran.

Afin de connaitre l'heure de la décrue totale, il vous reste à résoudre l'équation du premier degré $-4.64x + 132.90 = 0$

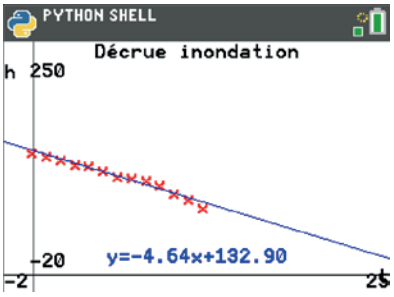


Améliorons notre représentation graphique :
Inclure dans votre script des instructions permettant de conserver les axes en noir **plt.color(0,0,0),** puis une représentation graphique du nuage de point en rouge **plt.color(255,0,0)** et enfin une représentation en bleu de la droite de régression **plt.color(0,0,255)**

Il est aussi intéressant de rajouter un titre et des étiquettes aux axes



Conseils à l'enseignant : Afin d'éviter des problèmes de superposition pour l'affichage, choisir des noms courts pour les étiquettes des axes.
L'étiquette des axes « t » et « h » peut être réglée selon vos souhaits en utilisant l'instruction **labels(« x-étiq », « y-étiq » »,x, y)**, **x** et **y** représentant la ligne où sont écrits ces labels. **Par défaut ces lignes sont respectivement 12 et 2 pour x et y et repspectivement justifiées à gauche et à droite.**
De même, l'équation de régression peut être affichée à l'emplacement souhaité, en utilisant la commande **lin_reg(xlist, ylist, « disp », row)**. Par défaut, cette équation est affichée à la ligne 11.



Unité 4 : Utiliser le module ti_plotlib

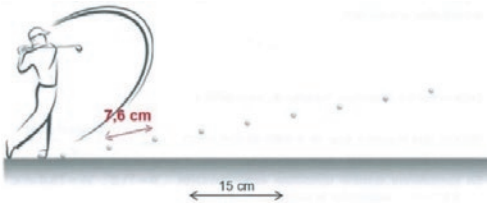
Application : Positions successive d'un mouvement

Dans l'application de l'unité 4, vous allez découvrir comment étudier un mouvement à partir des mesures effectuées sur une chronophotographie et réinvestir lesconnaissances acquises lors de l'utilisation du module **ti_plotlib**.

Objectifs :

- Représenter un nuage de points.
- Effectuer le calcul puis la représentation de vecteurs d'un système modélisé par un point.

Le problème : On étudie le mouvement d'une balle de golf à partir d'une chronophotographie (enregistrement d'un mouvement par prise de photographies selon un intervalle de temps fixé). On souhaite représenter l'évolution du vecteur vitesse au cours du temps.
Les positions sont relevées toutes les 0,066 s et sont rassemblées dans un tableau.
Remarque : dans l'unité 5, vous apprendrez à importer les mesures depuis les listes de la calculatrice en utilisant le module ti_system.



t(s)	0	0.066	0.132	0.198	0.264	0.33	0.396	0.462	0.528	0.594	0.66
x(m)	0.01	0.25	0.57	0.91	1.22	1.54	1.87	2.16	2.49	2.81	3.15
y(m)	0.015	0.34	0.681	1.01	1.297	1.559	1.768	1.95	2.08	2.158	2.193

Mise en œuvre :

1 : Entrée des mesures et création des valeurs du temps.

- Commencer un nouveau script et le nommer U4APPS.
- Importer la librairie **ti_plotlib** de représentation graphique.
- Entrer les mesures correspondant aux coordonnées d'un point repéré lors de l'analyse de la chronophotographie.

2 : Calcul des vecteurs vitesses.

Le vecteur vitesse à un instant t est donné par la relation : $\vec{V_i} = \dfrac{\overrightarrow{M_i M_{i+1}}}{t_{i+1} - t_i}$

On ne peut donc pas dessiner le vecteur vitesse du dernier point de la liste, il faut en tenir compte dans le code.

Astuce : Afin de rendre la représentation graphique lisible, on utilisera un facteur d'échelle de 2.

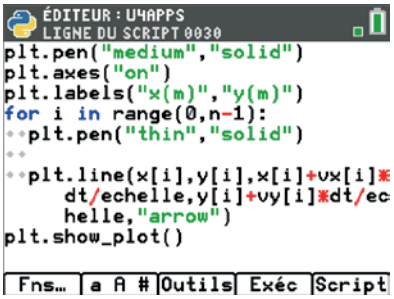
Conseil à l'enseignant : On rappelle que l'instruction **len(x)** permet d'obtenir le nombre d'éléments de la liste **x**. Celle-ci se trouve dans le menu relatif aux listes.

```
ÉDITEUR : U4APPS
LIGNE DU SCRIPT 0001
import ti_plotlib as plt_
#données
dt=0.066
x=[0.01,0.25,0.57,0.91,1.22,1.54
     ,1.87,2.16,2.49,2.81,3.15]
y=[0.01549,0.3404,0.6812,1.010,1
     .297,1.559,1.768,1.95,2.08,
     2.158,2.193]
#vecteurs vitesses
vx=[]
vy=[]
Fns… | a A # | Outils | Exéc | Script
```

```
ÉDITEUR : U4APPS
LIGNE DU SCRIPT 0005
     .297,1.559,1.768,1.95,2.08,
     2.158,2.193]
#vecteurs vitesses
vx=[]
vy=[]
n=len(x)
for i in range(0,n-1):
  vx.append((x[i+1]-x[i])/dt)
  vy.append((y[i+1]-y[i])/dt)
echelle=2
#représentation graphique
Fns… | a A # | Outils | Exéc | Script
```

3 : Régler les paramètres de la représentation graphique :

- **plt.cls()** pour effacer l'écran.
- **plt.title(« titre »)** pour donner un titre à la représentation graphique.
- **plt.window(x_{min}, x_{max}, y_{min}, y_{max})** pour régler la fenêtre graphique.
- **plt.grid(xscl, ysc, « type »)** pour afficher une grille de graduation 0.5.
- **plt.color(255,0,255)** pour un affichage en couleur magenta.
- **plt.scatter(xlist, ylist, « type »)** pour un affichage sous forme d'un nuage de points.
- **plt.color(0,0,0)** pour un affichage en couleur noire pour les axes.
- **plt.pen(« medium », « solid »)** pour un affichage des axes en épaisseur moyenne.
- **plt.labels(« x(m) », « y(m) »)** pour afficher les étiquettes aux lignes 12 et 2 par défaut.

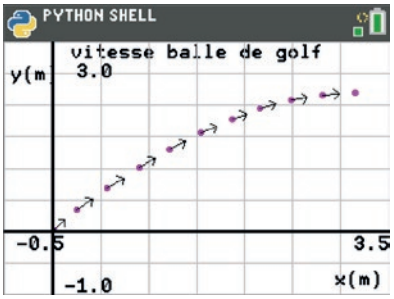


Le tracé des vecteurs est effectué au sein d'une boucle fermée de *n*-1 valeurs.

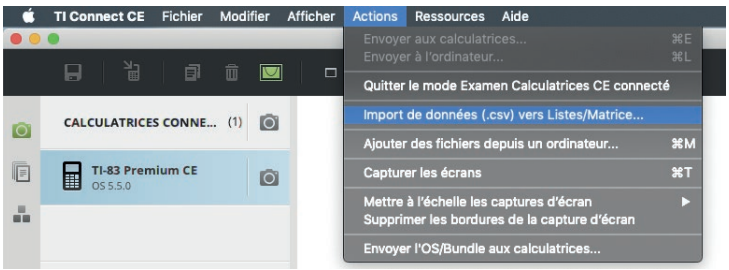
- **plt.line(x0,y0,x1,y1)** pour le tracé d'un vecteur vitesse.
- **plt.show()** pour afficher la représentation graphique.

Rappel : La position de la balle entre les instants t_i et t_{i+1} peut-être repérée dans un repère cartésien par $x_i + vx_i \times dt$ pour l'abscisse et $y_i + vy_i \times dt$ pour l'ordonnée.

Exécuter votre script ; vous devriez obtenir une représentation graphique analogue à celle de l'écran ci-contre.



Conseil à l'enseignant : En appuyant sur la touche **F3**, vous disposez d'une boîte à outils contenant des instructions de copier-coller afin de faciliter l'édition de votre script.
Pour des scripts complexes, vous disposez également de l'opportunité de dupliquer un script. Pour cela, afficher la liste des scripts, placer le curseur devant le nom du script à dupliquer, puis appuyer sur **F5** (**Gérer**) et choisir le menu **1 : Dupliquer le script**. Un nouveau nom vous sera demandé.
Si le nombre de données est important ou provient d'une expérience réalisée avec une console d'acquisition, il est possible d'utiliser le logiciel TI Connect CE afin d'importer celles-ci au format **.csv**



Unité 5 : Utiliser le module ti_system

Compétence 1 : Travailler sur des données

Dans cette première leçon de l'unité 2, vous allez découvrir comment Utiliser le module ti_**system** pour importer ou exporter des listes dans un script Python.

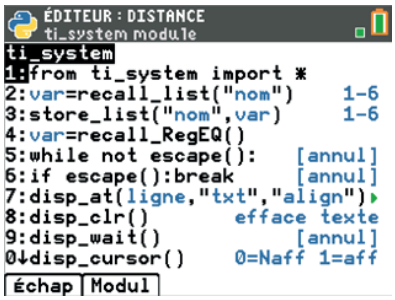
Objectifs :

- Importer-exporter des listes.
- Réinvestir les notions de l'unité 4 sur les représentations graphiques.

La librairie ou module **ti_system** utilisée seule ou en complément des autres, permet de communiquer (dans les deux sens) avec la calculatrice graphique.

2 : var=recall_list(« nom ») et **3 : store_list(« nom »,var)**

Les autres options de cette librairie seront abordées dans les autres leçons de cette unité.

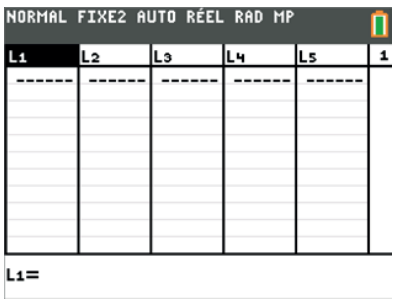


1 : Importer des données depuis la calculatrice.

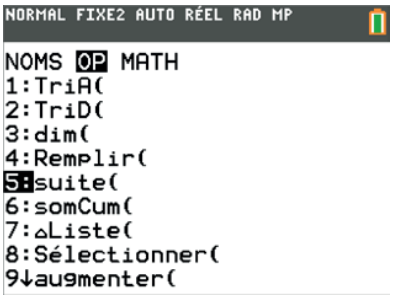
a) Création de deux listes

Dans un premier temps nous allons simplement entrer des données dans des listes (L_1 et L_2).

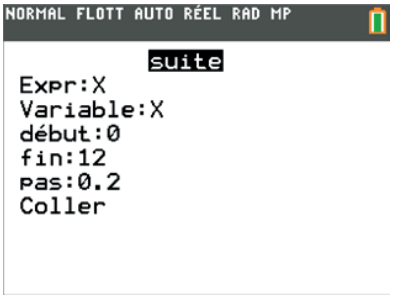
- La calculatrice étant en fonctionnement habituel (hors programmation), effacer le contenu des listes en appuyant sur 2nde + puis 4 : EffTtesListes.
- Entrer dans l'éditeur de liste et mettre le curseur sur L_1 comme sur l'écran ci-contre.



Vous allez créer dans la liste L_1 une suite de nombre entre 0 et 12 afin de pouvoir représenter graphiquement les points calculés, dont on a calculé les coordonnées à l'aide d'une fonction définie par : $x \longmapsto -x^2/2 + 3x + 1$

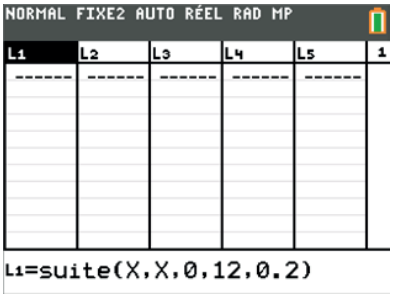


Appuyer sur [2nde] [stats] [▶] [▶] puis choisir l'instruction **5 : suite** et compléter les paramètres afin de construire une suite de nombres dans l'intervalle $[0\,;\,12]$ par pas de 0.2.

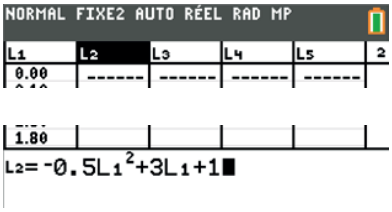


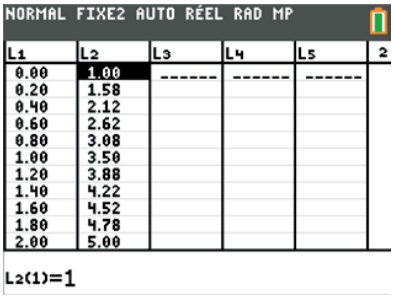
Une fois le curseur sur Coller, appuyer sur la touche [entrer] afin de valider la création de la suite.

Les valeurs calculées sont recopiées dans la liste L_1.



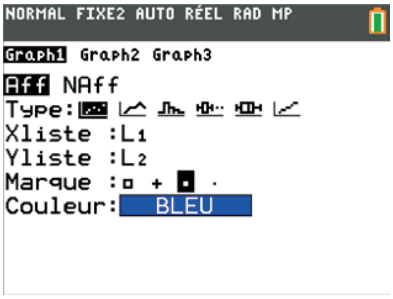
Placer le curseur sur L_2 et demander le calcul des images par la fonction : $x \longmapsto -x^2/2 + 3x + 1$ des valeurs de la liste L_1 à l'aide de l'instruction L_2=-0.5L_1^2+3L_1+1, puis valider le calcul [entrer].

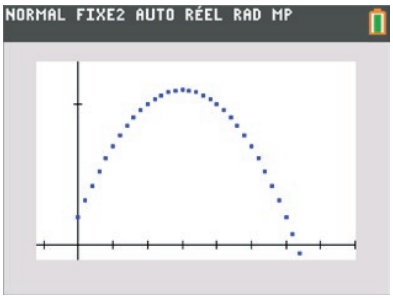




Paramétrer votre représentation graphique sous forme d'un nuage de points. Afficher la représentation graphique [zoom] [9] en vérifiant par ailleurs qu'aucune fonction n'est activée (touche [f(x)]).

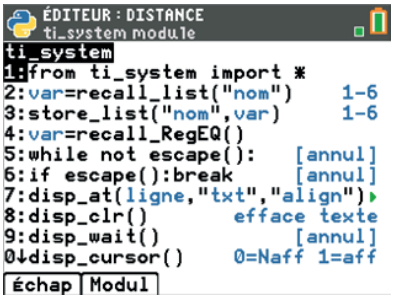
Régler les paramètres de la fenêtre graphique tels que : X_{min} = -1.2 ; X_{max} = 8 ; Y_{min}= -0.5 et enfin Y_{max}= 6.5.





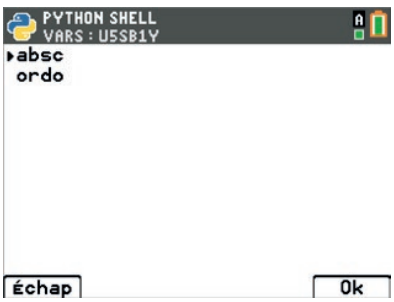
b) Import des données dans un script Python

- Commencer un nouveau script et le nommer U5SB1.
- Créer deux variables listes (vides) **absc** et **ordo**.
- Importer les librairies **ti_system** et **ti_plotlib** (il n'y a pas d'ordre)
- Créer une variable **absc** puis, à partir des options de la librairie **ti_system**, choisir l'option **2 : var=recall_list(« nom »)**. Comme les abscisses sont dans la liste L_1, le champ « nom » est complété par le nombre 1
- Créer une autre variable **ordo** et procéder de la même manière avec la liste L_2.

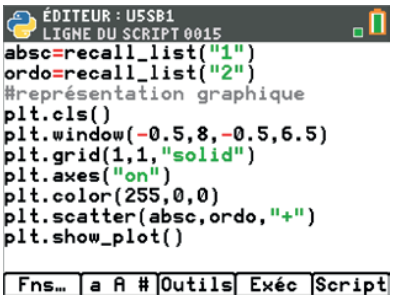


Conseil à l'enseignant : La création de listes vides absc=[ ] et ordo=[ ] n'est pas indispensable, car elles seront créés lors du rappel des listes L_1 et L_2. Cependant il est bien de conserver les bonnes habitudes apprises lorsque nous ne faisons pas appel au module ti_system qui nécessitent cette création préalable.

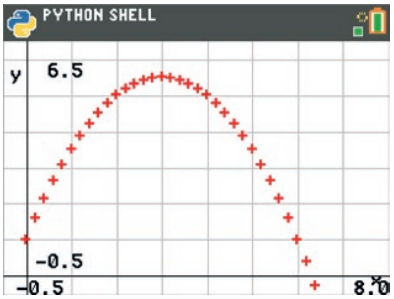
- Exécuter le script puis vérifier le contenu de vos variables **absc** et **ordo** en appuyant sur la touche [var].
- Rappeler ensuite le nom de la « liste » **absc** puis valider [entrer].
- Procéder de la même façon avec la liste des ordonnées (**ordo**)



c) Représentation graphique
Paramétrer votre représentation graphique comme proposé sur l'écran ci- contre.



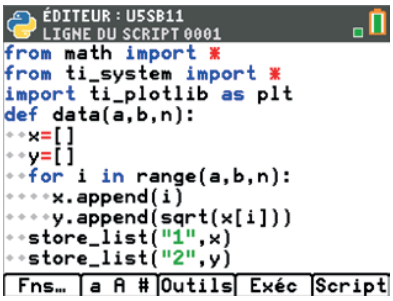
Exécuter votre script (touche **F4**).



2 : Exporter des données
Créer un nouveau script et le nommer U5SB11

Conseil à l'enseignant : Placer le curseur à la fin d'une ligne et valider. L'ordre d'écriture de l'importation des modules est sans importance.

- Créer une fonction nommée **data(a,b,n)**.
- Vous allez créer deux listes de données à représenter sous forme d'un nuage de points comportant des valeurs dans un intervalle **[a ; b]**, calculées avec un pas **n**.
- Dans la liste **y**, on calcule la racine carrée des valeurs de la liste **x**.
- Pour créer ces listes de données, nous allons construire une boucle fermée après avoir bien entendu crée deux listes vides.



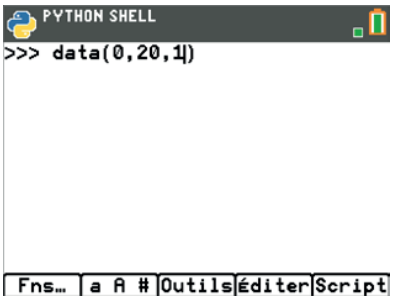
Conseil à l'enseignant : La création de deux listes vides évite le renvoi d'un message d'erreur lors de l'exécution du script.

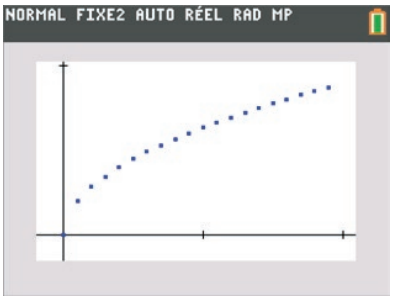
- Pour l'export de listes vers celles de la calculatrice, la syntaxe est la suivante : **store_list(« n° de liste »,nom)**. Le numéro de liste dans la calculatrice correspondant aux listes L_1 à L_6

Remarque : Attention à l'indentation, les instructions **store_list** n'ont pas à être dans la boucle. Utiliser le menu Outils (F3), puis l'option **2 : Indent <-** afin de supprimer un niveau de décalage, (Il est aussi possible d'utiliser la touche suppr.

- Exécuter votre script. On prend ici 20 valeurs de 0 à 20 par pas de 1
- Sortir de l'environnement Python et afficher la représentation graphique de vos listes L_1 et L_2.

Conseil à l'enseignant : l'export vers les listes de la calculatrice sera particulièrement intéressant pour représenter des données acquises par l'intermédiaire de capteurs avec le microcontrôleur **TI – Innovator & TI-Rover**.





Unité 5 : Utiliser le module ti_system

Compétence 2 : Modélisation

Dans cette seconde leçon de l'unité 5, vous allez découvrir comment importer les résultats d'une modélisation en utilisant la librairie **ti_system**.

Objectifs :

- Effectuer une modélisation linéaire.
- Importer les résultats de cette modélisation dans un script Python.

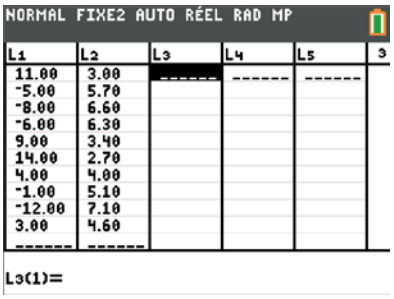
Vous allez dans cette leçon effectuer une modélisation linéaire à partir de données préalablement inscrites dans les listes de la calculatrice. Ensuite, vous écrirez un script afin d'importer les résultats de cette modélisation afin de les utiliser pour une représentation graphique, une interpolation ou extrapolation…

Le problème : Dans une journée, le pic de consommation d'électricité est atteint vers 19 h. Au niveau national, on a enregistré un pic de consommation de 96 350 mégawatts le mercredi 15 décembre 2019 à 19h02. Vous désirez prévoir les pics de consommation du prochain week-end pour la zone directement raccordée à la centrale.

Pour établir cette prévision, vous disposez de dix relevés de consommations réalisés à 19h en fonction de la températureet présentés dans le tableau ci-dessous.

T°C	11	-5	-8	-6	9	14	4	-1	-12	3
MW	3	5.7	6.6	6.3	3.4	2.7	4	5.1	7.1	4.6

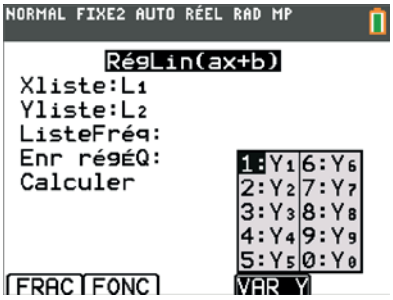
Les données sont entrées dans les listes de la calculatrice, la température dans L_1 et la consommation en MW dans L_2.



Appuyer sur [stats] [▶] [▼] [▼] [▼] pour effectuer une modélisation linéaire sous la forme $y = ax + b$.

Suivre les indications proposées et sauvegarder le résultat de la régression dans l'éditeur de fonction en Y_1.

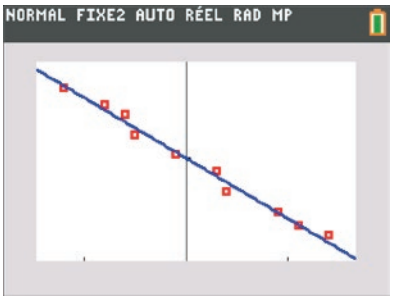
Conseil à l'enseignant : appuyer sur les touches [alpha] [trace] afin de choisir facilement Y_1.



La fonction permettant de prévoir la consommation en fonction de la température est donc : $C = -0.18 \times t + 5.01$.

Utilisation des résultats de la modélisation dans un script Python.

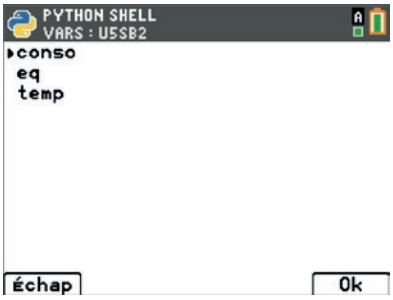
- Commencer un nouveau script et le nommer U5SB2.
- Incorporer le menu **ti_system** et **ti_plotlib**.
- Créer deux listes **temp** et **conso** vides (temp = [ ] et conso = [ ] .



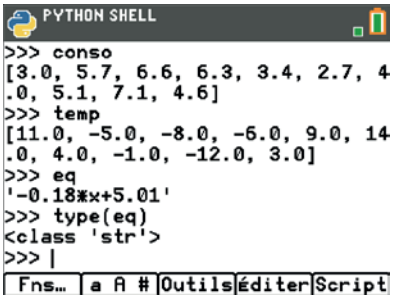
Rappeler le contenu des listes L_1 et L_2 dans leurs noms respectifs.
L'instruction **var=recall_list(« nom »)** est accessible dans le menu du module **ti_system** (voir Unité 5 Compétence 1).

Rappeler également dans une variable **eq**, l'expression du modèle linéaire calculé.

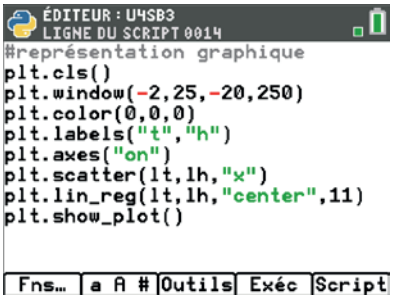
Tester votre script et demander l'affichage des différentes variables ainsi créées en appuyant sur la touche [var] puis en choisissant vos variables.



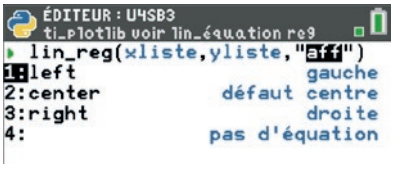
Remarque : l'équation de régression est importée en tant que chaîne de caractères.



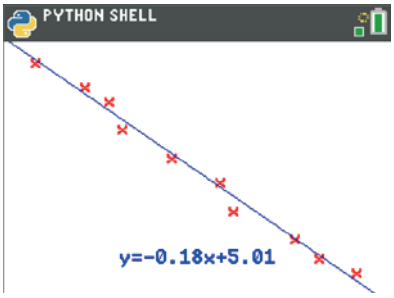
Pour compléter cette leçon et réinvestir les compétences acquises lors de l'unité 4, vous pouvez effectuer la représentation graphique de vos mesures ainsi que du modèle de régression calculé.



L'instruction **plt.lin_reg ( )** correspond à l'option **8 : plt.lin_reg(x-liste, y-liste, aff)** du menu **Dessin** de la librairie **ti_plotlib.**
Remarque : l'instruction **plt.auto_window(x_list, y_list)** permet d'ajuster automatiquement les paramètres de la fenêtre graphique. C'est en quelque sorte l'équivalent du ZOOM 9 (ZOOM Statistiques) de la calculatrice TI-83 Premium CE.



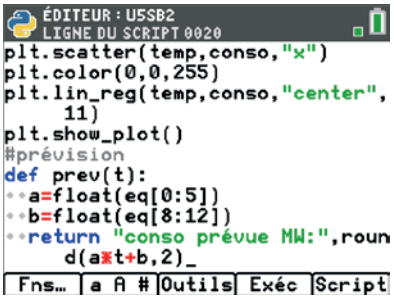
Conseil à l'enseignant : l'instruction **lin_reg(xliste, yliste, « aff », row)** comporte une instruction supplémentaire row donnant la possibilité de placer sur une autre ligne l'équation de régression. Par défaut, celle-ci est placée à la ligne 11.



Prolongement : prévoir la consommation pour une température donnée.
Vous allez récupérer dans deux variables **a** et **b** les coefficients de l'équation afin de les utiliser dans une fonction vous permettant de réaliser ainsi une extrapolation ou une interpolation de la consommation électrique, lorsque la température est connue.

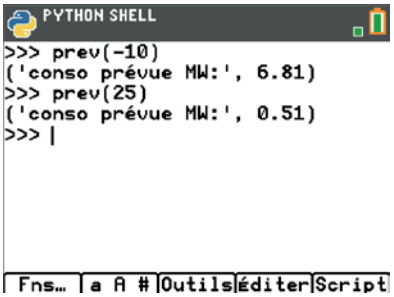
Conseil à l'enseignant : l'instruction **v = float(eq[0 : i+1])** permet de récupérer dans la variable **v** les **i** premiers éléments de la chaîne de caractère **eq**.

Définir une fonction **prev(t)** qui retournera la consommation prévue pour une température **t**.



Exécuter votre script et effectuer quelques tests pour différentes températures. On rappelle que le modèle représente une prévision de consommation électrique à 19h02 en fonction de la température.

Vous pouvez ainsi déterminer les limites de validité de votre modèle.



Unité 5 : Utiliser le module ti_system

Compétence 3 : Affichage et temporisation

Dans cette troisième leçon de l'unité 5, vous allez découvrir comment utiliser les options d'affichage et de « temporisation » de la librairie Python **ti_system**.

Objectifs :

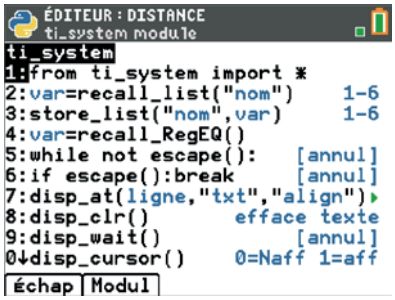
- Comprendre le fonctionnement des instructions **disp…**
- Utiliser ces instructions dans un script en complément de celles des autres librairies.

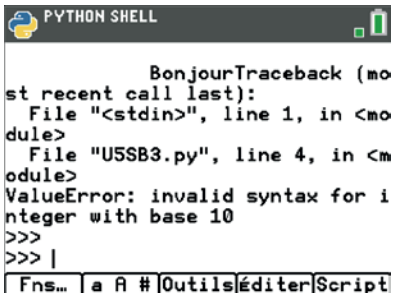
1 : Les instructions disp

Lors des leçons 1 et 2 de cette unité, vous avez appris à importer exporter des listes de données puis à travailler sur l'équation de régression.

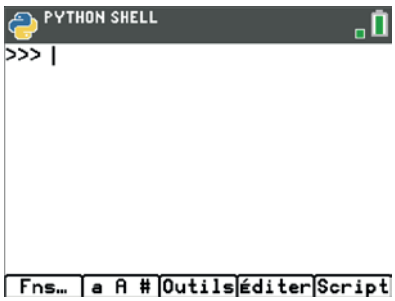
Conseil à l'enseignant : l'instruction **4 : var=recall_RegEQ()** permet de rappeler l'équation de régression calculée à partir de deux listes. Cette équation ne correspond pas nécessairement à un modèle linéaire.

- Commencer un nouveau script et le nommer U5SB3.
- Importer le module **ti_system**.
- Incorporer la ligne **disp_clr()**.
- Appuyer sur la touche **F5** afin d'afficher la liste des scripts et passer en mode console (shell, touche **F4**).
- Observer dans votre console la présence de lignes de texte ou messages d'information.

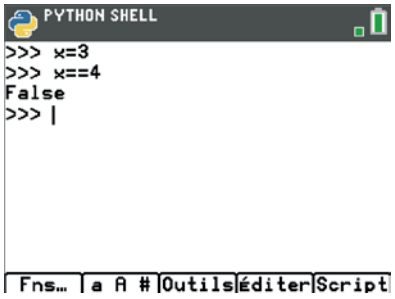




- Rappeler votre script U5SB3 et demander son exécution. Vous devriez obtenir l'écran ci-contre.

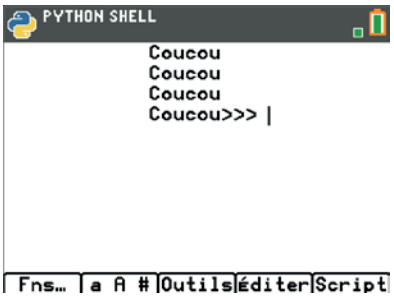


L'instruction **disp_clr()** nettoie l'écran et place le prompt de la console en haut de l'écran. Vous obtenez ainsi au sein d'un programme un écran débarrassé de ses précédents affichages.

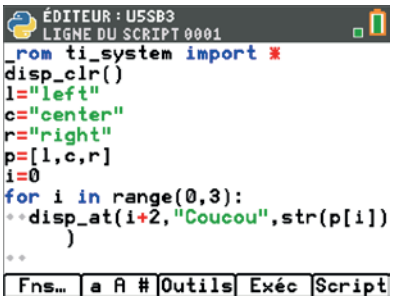


- Continuer l'édition du script en insérant dans une boucle fermée, l'instruction **7 : disp_at(ligne, « txt », « align »)** que vous trouverez dans le module **ti_system**. Cette instruction affiche à la ligne **i** de l'écran, le texte en l'alignant à gauche, droite ou centre de la ligne.

- Exécuter à nouveau votre script et observer ce que vous obtenez.

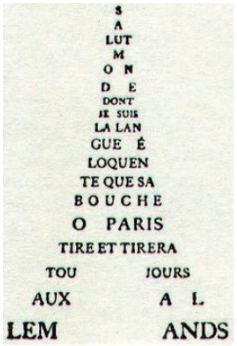


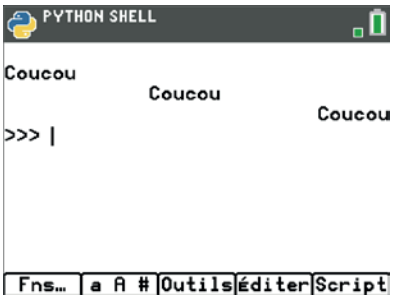
Reprendre le script et le modifier comme sur l'écran ci-contre en créant trois variables chaînes de caractères **l, c, r** que l'on inclue ensuite dans une liste **p** (position).



Par la suite, lors de l'exécution de la boucle fermée, le mot devrait être décalé. L'indice d'écriture du mot est décalé, puisqu'il est commun avec celui de la liste **p.** On ne peut pas écrire sur une ligne d'indice 0, cela causera un message d'erreur lors du fonctionnement du script.

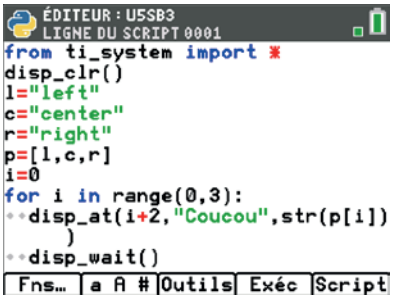
Une suggestion pour prolonger cette instruction : réaliser des poèmes graphiques.



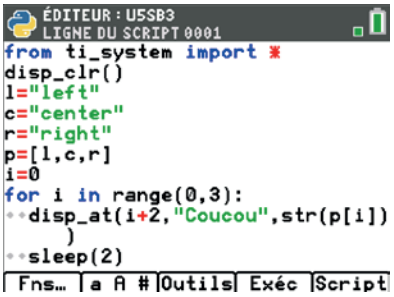


L'instruction **disp_wait()** met l'exécution du script en pause. Appuyer sur la touche [entrer] afin de poursuivre son exécution.

Conseil à l'enseignant : Cette instruction est intéressante afin d'observer le fonctionnement pas à pas d'un script incluant une boucle. Le mot proposé ici peut également être inclus dans une liste afin de constituer une phrase. Par exemple, « sujet », « verbe », « complément ».

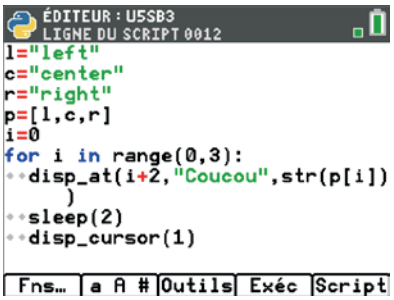


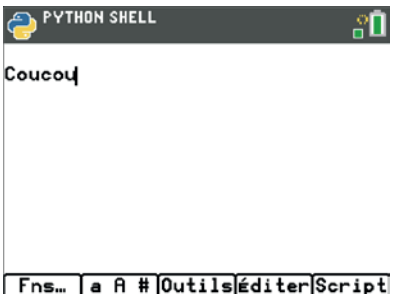
Supprimer l'instruction **disp_wait()** et lui substituer **sleep(secondes).** Cette dernière instruction introduit une temporisation de la durée indiquée.
Ainsi, le mot proposé s'affichera dans sa position correspondant à la valeur de la liste p[i] toutes les 2 secondes.



L'instruction **disp_cursor(1)** affichera le curseur sous la forme d'un trait vertical, à la fin du mot à afficher.

disp_cursor(0) ne l'affichera pas.





2 : les instructions escape()

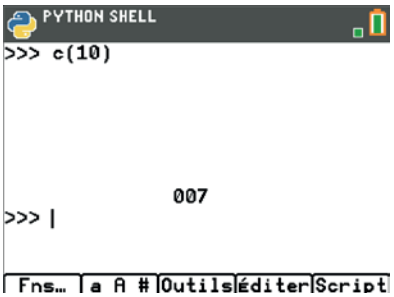
Créer un nouveau script et le nommer USB531.

- Mettre l'instruction **disp_clr()** afin d'effacer l'écran.
- Définir une fonction ayant pour argument un entier.

Dans une boucle fermée, l'instruction **if escape() : break** arrêtera l'exécution du script si vous appuyez sur la touche annul.

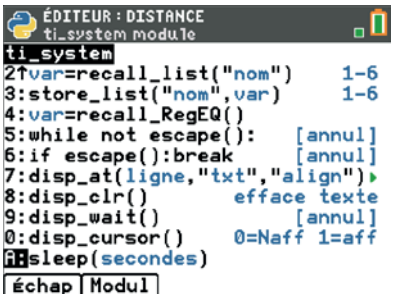
- Sinon la valeur **i** de l'incrément de la boucle sera stockée dans une variable ***x*** .
- Une pause de 1 seconde permet de voir affiché à l'écran le texte « 00 ».

Lors de l'exécution du script, vous verrez votre écran scintiller très légèrement. L'objectif de ce script est d'afficher lors de l'appel de la fonction c(n) de provoquer l'affichage ci-contre.



Le fonctionnement de l'instruction **while not escape()** est identique quant à la touche à presser, mais son action est semblable à la précédente puisqu'ici, le script poursuivra son exécution tant que l'on n'appuie pas sur la touche annul.

Conseil à l'enseignant : L'utilisation de ces instructions seront très utiles lors de la mise en œuvre de script utilisant le **TI-Innovator** et le **TI-Rover**.



Unité 5 : Utiliser le module ti_system & ti_plotlib

Application : Etude de la chute libre

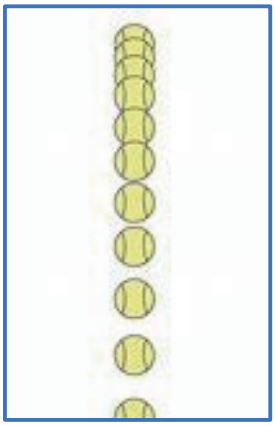
Dans cette application de l'unité 5, vous allez réinvestir les notions vues lors dans l'unité 4 et 5 afin de créer un simulateur permettant de décrire un mouvement.

Objectifs :

- Effectuer une simulation d'une chronophotographie.
- Exporter les résultats dans les listes de la calculatrice.

On propose dans cette application de l'unité 5, d'utiliser le langage Python afin de créer un simulateur d'un phénomène physique. Nous retenons l'étude de la chute libre :

- Le script devra permettre la description d'un mouvement ;
- La représentation graphique des données sous forme d'une chronophotographie ;
- L'export des données vers les listes de la calculatrice.



Le problème : Une balle est lâchée sans vitesse initiale d'une hauteur h. Les photographies sont effectuées au cours du temps toutes les 60 ms.

Vous allez écrire un script permettant de calculer, au cours du temps, la position de la balle, puis de la représenter graphiquement.

a) Calcul des positions successives

Lors de la chute libre, à partir d'une origine des temps et des espaces choisis (y>0), la position de la balle peut être calculée en utilisant la relation $y = -\frac{g}{2} \times t^2 + h_0$.

Rappelons que, $g = 9{,}81$ m.s^{-2}.

Créer un nouveau script et le nommer U5Apps.

- Importer les modules **ti_system** et **ti_plotlib.**
- Nettoyer l'écran.
- Créer une fonction **chrono(h)** prenant comme paramètre la hauteur initiale à laquelle est lâchée la balle et qui affiche la position de la balle en fonction du temps.
- Créer trois listes vides, abscisse **x**, ordonnée **y** et temps **te.**
- Les ordonnées sont calculées toutes les 60 ms.
- Les valeurs sont sauvegardées dans les listes si y >0 (la balle ne s'enfonce pas dans le sol)

```
ÉDITEUR : USAPPS
LIGNE DU SCRIPT 0001
from ti_system import *
import ti_plotlib as plt
disp_clr()
def chrono(h):
    g=9.81
    x=[]
    y=[]
    te=[]
    dt=0.06
    for i in range(0,50,1):
        t=i*dt
        e=-g/2*t**2+h
        if e>0:
            te.append(t)
            x.append(0*i)
            y.append(round(e,2))
            store_list("1",te)
            store_list("2",y)
#repré_entation graphique
Fns… | a A # | Outils | Exéc | Script
```

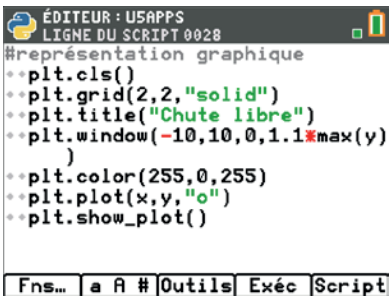
Créer une boucle fermée permettant de calculer l'altitude de la balle en fonction du temps. Les résultats sont exprimés à 10^{-2} près et stockés dans leurs listes respectives. Comme on souhaite avoir une représentation graphique de la chronophotographie correspondant à un cas réel, la liste des abscisses est complétée avec des 0.

Enfin, les listes **te** et **y** sont respectivement exportées dans les listes L_1 et L_2 de la calculatrice.

b) Représentation graphique

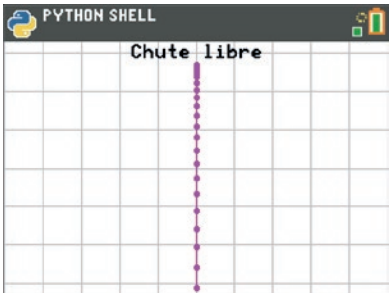
Paramétrer une représentation graphique :

- Nettoyer l'écran **plt.cls()**.
- Afficher une grille d'unité 2 **plt.grid(xscl, yscl, type,(r,v,b))**.
- Régler la fenêtre graphique **plt.window(x_{min}, x_{max}, y_{min}, y_{max})**.
- Fixer la couleur du point au magenta **plt.color(255,0,255)**.
- Représenter le nuage de point **plt.plot(x-list, y-list, marque)**.
- Afficher le graphe **plt.show_plot()**.



Exécuter votre script et appeler la fonction chrono [var] puis lui fournir comme argument la hauteur initiale du lâché (8m par exemple).

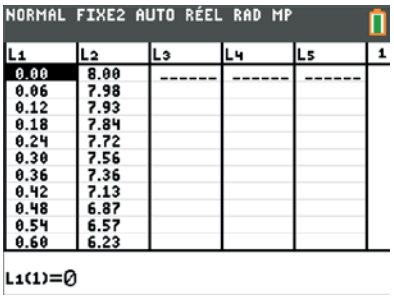
La chronophotographie est représentée.

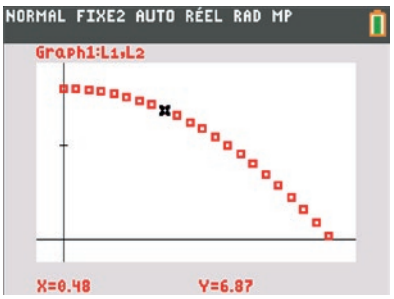


Conseil à l'enseignant : A partir de la liste des positions de la balle, on pourra éventuellement calculer la vitesse de la balle, puis afficher les vecteurs vitesses…

c) Visualisation des données exportées

Quitter l'éditeur Python et afficher les listes.
Demander la représentation graphique de ces listes.
Utiliser l'outil [trace] pour explorer la représentation graphique (durée totale de la chute…).





Unité 6 : Utiliser le module ti_hub & ti_rover

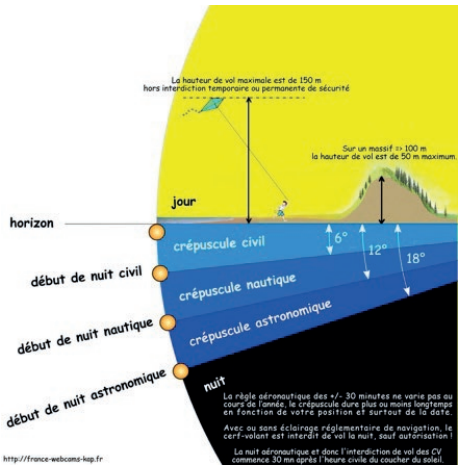
Compétence 1 : Les capteurs intégrés au hub

Dans cette première leçon de l'unité 6, vous allez découvrir comment utiliser la librairie ti_hub afin de commander les dispositifs intégrés au hub TI-Innovator™.

Objectifs :

- Découvrir le module ti-hub.
- Écrire un script intégrant la librairie **ti-hub** pour les dispositifs intégrés.

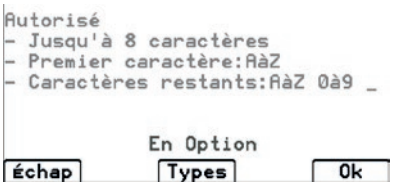
Vous allez, dans cette leçon, utiliser la librairie **ti_Innovator** afin de noter visuellement un changement de luminosité pour, par la suite, simuler un interrupteur crépusculaire, ou bien enregistrer une série de mesures lors du lever du soleil ou du crépuscule.
Vous commencerez par ailleurs à envisager comment associer cette librairie à celle que vous connaissez déjà (**ti_plotlib & ti_system**) afin de créer un projet scientifique complet.
Le script que vous allez écrire correspond à l'algorithme simple suivant :

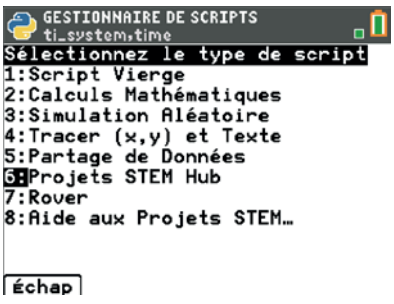



```
Mesurer l'intensité lumineuse ambiante :
Lum0 ← mesure ± (tolérance ?)
Modifier l'intensité lumineuse (lampe ; cache devant le capteur)
Lum1 ← mesure
  Si Lum1 > Lum 0 :
  |  Alors allumer la DEL RVB en rouge (2s)
  Sinon Si Lum0 < Lum1 :
  |  Alors allumer la DEL RVB en vert (2s)
  Sinon : Ne rien faire
```

Commencer un nouveau script et le nommer U6SB1
Ce script doit intégrer la librairie **ti_hub**. Pour cela, vous avez plusieurs possibilités.

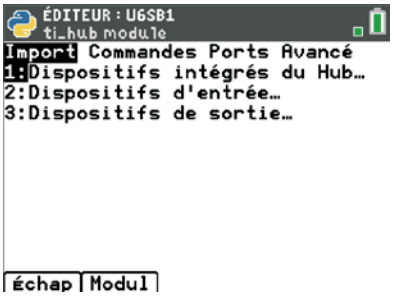
a) Appuyer sur la touche **F3** [Types] puis choisir le menu 6 : Hub Projet. Valider ensuite en appuyant sur [entrer] deux fois ou une seule fois puis OK.
b) Vous pouvez également partir d'un script vierge, puis incorporer manuellement les librairies dont vous avez besoin.
c) Enfin, vous pouvez aussi utiliser les librairies et variables qui ont été écrites lors des projets TI STEM. Celles-ci seront immédiatement recopiées comme « Objet du projet ». Cette dernière option n'est intéressante que pour créer un script analogue à ceux ayant été définis lors du projet STEM. https://education.ti.com/en/resources/stem





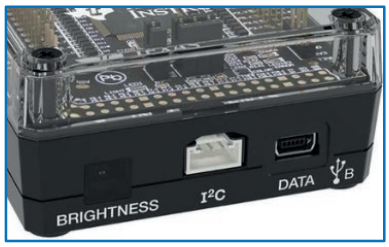
En choisissant le type **6 : Hub Projet**, Vous devriez obtenir l'écran ci-contre.

Nous allons utiliser le capteur de luminosité intégré au **TI-Innovator**, ainsi que la diode RVB. Afin que le script soit en mesure de les gérer, intégrons les librairies correspondantes. Pour cela choisir dans le menu **Modul** puis **6 : ti_hub…** et enfin dans le sous-menu **1 : Dispositifs intégrés du Hub**.

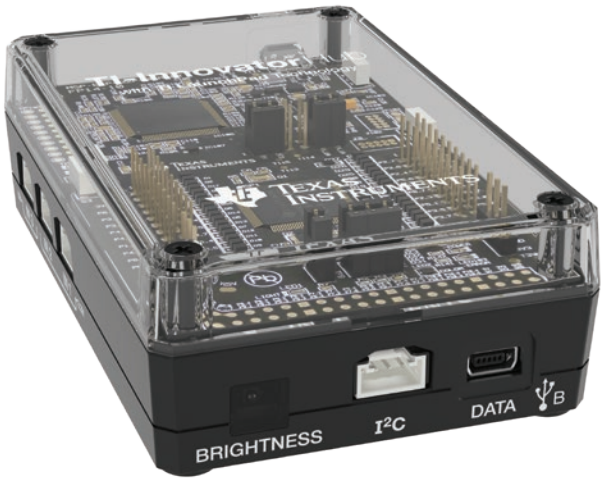


Conseil à l'enseignant : Les deux autres possibilités concerneront les capteurs et actionneurs que l'on connectera directement sur les ports d'entrée sortie IN… et OUT… du Hub ou éventuellement sur les ports BBx.

- Afin que le programme affiche des informations sur un écran « propre », nettoyer celui-ci, à l'aide l'instruction **disp_clr()** se trouvant dans la librairie **ti_system.**
- Créer une variable l**um0**, à laquelle on affecte une mesure de la luminosité. L'instruction **brightns.measurement()** se trouve dans le module **9 : Brightness**, chargé dès lors que le module correspondant est importé.
- Afficher un message invitant l'utilisateur à modifier l'intensité lumineuse dans le voisinage du capteur intégré **disp_at()**.



- Mettre le programme en attente **disp_wait()**, le temps d'effectuer cette modification.
- Créer une variable **lum1** à laquelle, on affecte la nouvelle mesure.
- Les mesures sont ensuite comparées. Selon le résultat de l'instruction conditionnelle, la DEL RVB s'allumera en vert ou rouge pendant un délai de 2 secondes.




```
ÉDITEUR : U6SB1
LIGNE DU SCRIPT 0002
# Hub Projet
from ti_system import *
from time import *
import color
import brightns
#mesure
disp_clr()
lum0=brightns.measurement()
disp_at(5,"modifier la luminosit
    é","left")
disp_wait()
lum1=brightns.measurement()
#comparaison
if lum0<lum1:
  color.rgb(255,0,0)
  sleep(2)
  color.off()
elif lum0>lum1:
  color.rgb(0,255,0)
  sleep(2)
  color.off()
if lum0<lum1:
  color.rgb(255,0,0)
  sleep(2)
  color.off()
elif lum0>lum1:
  color.rgb(0,255,0)
  sleep(2)
  color.off()
else:
  color.off()
Fns… | a A # | Outils | Exéc | Script
```

Prolongement de l'activité :
Modifier le script précédent ou en créer un autre U6SB11
Ce nouveau script doit sur 40 minutes enregistrer les mesures de la luminosité.

Conseil à l'enseignant : Attention le capteur brightness du TI-Innovator n'est pas étalonné en Lux, mais cela n'a pas d'importance dans la mesure où l'on ne s'intéresse qu'aux variations de la luminosité et non à leur mesure en Lux.

Les mesures sont sauvegardées dans une liste **r[ ]**.
Celles correspondant à la valeur du temps dans une liste **t[ ]**.

Le script proposé ci-dessous propose également la représentation graphique des mesures afin de le comparer à la représentation ci-contre.
Il est également proposé un export vers les listes de la calculatrice,

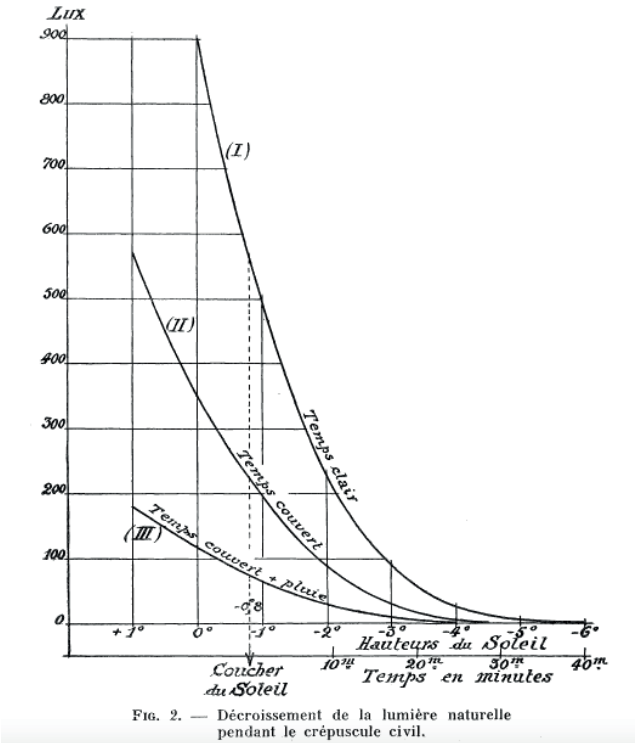


Fig. 2. — Décroissement de la lumière naturelle pendant le crépuscule civil.

a) Acquisition des données

```
ÉDITEUR : U6SB11
LIGNE DU SCRIPT 0001
# Hub Projet_
from ti_system import *
from time import *
import ti_plotlib as plt
import brightns
disp_clr()
def bri(n):
  r=[]
  t=[]
  for i in range(n):
    r.append(brightns.measuremen
      t())
    t.append(i)
    sleep(60)
  return t,r
#représentation graphique
Fns… | a A # | Outils | Exéc | Script
```

b) Représentation graphique

```
ÉDITEUR : U6SB11
LIGNE DU SCRIPT 0025
#représentation graphique
def graphe(t,r):
  plt.cls()
  plt.auto_window(t,r)
  plt.labels("t(min)","r")
  plt.title("Crépuscule")
  plt.color(255,0,255)
  plt.scatter(t,r,"+")
  store_list("1",t)
  store_list("2",r)
  plt.show_plot()
Fns… | a A # | Outils | Exéc | Script
```

La fonction **bri(n)** réalise l'acquisition de données toutes les minutes pendant **n** minutes et renvoie les liste **t** et **r**
La fonction **graphe(t,r)** réalise la représentation graphique des données **t[ ]** et **r[ ],** puis les exporte vers les listes L_1 et L_2 de la calculatrice.

Unité 6 : Utiliser le module ti_hub & ti_rover

Compétence 2 : Les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter et utiliser un dispositif d'entrée-sortie du TI-Innovator™ à l'aide de la librairie **ti_hub**.

Objectifs :

- Découvrir le module **ti_hub**.
- Écrire et utiliser un script permettant d'utiliser un composant d'entrée sortie « grove ».

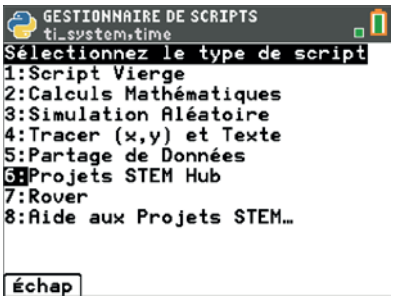
Vous allez dans cette leçon, utiliser un composant essentiel dans toute chaîne de mesure utilisant des capteurs, il s'agit d'un potentiomètre.

Un **potentiomètre** est un type de résistance variable à trois bornes, dont une est reliée à un curseur se déplaçant sur une piste résistante terminée par les deux autres à laquelle est soumise la résistance.
Les potentiomètres sont couramment employés dans les circuits électroniques. Ils servent par exemple à contrôler le volume d'une radio. Les potentiomètres peuvent aussi être utilisés comme des transducteurs, puisqu'ils convertissent une position en une tension. Ce type de dispositif peut être rencontré dans des joysticks.

Vous allez écrire un script permettant de mesurer la tension électrique entre deux bornes du potentiomètre, puis de l'afficher à l'écran.

Remarque : L'esprit de cette leçon ne porte pas sur l'étude du composant en lui-même, mais sur son intégration au sein d'un script Python afin d'obtenir les informations que celui-ci doit fournir. Ainsi le script que vous allez réaliser sera aisément transposable à tout autre type de transducteur.

Mise en œuvre :
Commencer un nouveau script et le nommer U6SB2. Choisir en appuyant sur la touche **F3**, lors de l'écriture du nom du script.

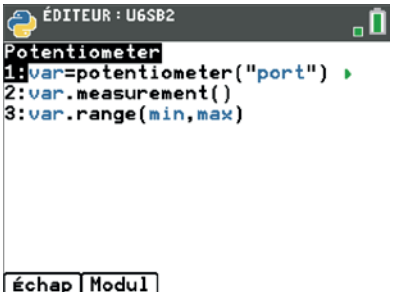


Les librairies **ti_system** et **time** sont importées, vous allez à présent importer la librairie du Hub correspondant au potentiomètre et éventuellement la librairie **ti_plot**.

Appuyer sur la touche F1 (Fns) [f(x)] [◄] puis afin d'accéder au menu **modul**, puis choisir **6 : ti_hub** puis **2 : Dispositifs d'entrée…**et enfin **0 : Potentiometer**.

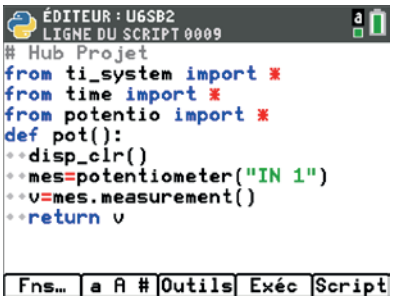
Toutes les commandes relatives au potentiomètre seront maintenant accessibles dans le menu **modul**, en complément de ceux déjà présent.

- Commencer par nettoyer l'écran afin d'obtenir une console vide de toute information. Pour cela, inclure l'instruction **dispclr()** disponible dans le module **ti_system**.
- Créer une fonction **pot( )** ne comportant pas d'argument pour le moment.
- Créer une variable **mes** et laisser le curseur clignoter derrière le nom de cette variable. Puis rechercher dans le menu **modul** puis **8 : Potentiometer…** la première instruction **var=potentiometer(« port »)**
- Valider en appuyant sur la touche [entrer], l'affectation de l'instruction permettant de connecter le potentiomètre au port IN 1est achevée.

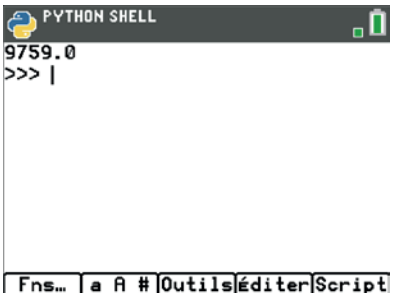


Conseil à l'enseignant : tous les capteurs (**dispositifs d'entrée**) comportent un minimum de deux variables **1 : var=capteur(« port »)** et **2 : var.measurement()**

- Créer une variable **v** permettant de collecter la mesure du capteur connecté (variable **mes**).



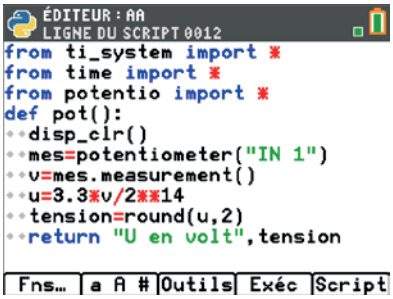
- Vérifier le fonctionnement de votre capteur après avoir préalablement placé le potentiomètre sur une position centrale.
- Connecter le TI-Innovator à la calculatrice puis le potentiomètre au port IN1
- Appeler la fonction **pot( )**.
- Vous devriez obtenir une information du même ordre de grandeur que celle de l'écran ci-contre, mais constater qu'il ne s'agit pas d'une tension puisque votre potentiomètre est alimenté par une tension de 3.3V. La valeur à trouver appartiendra àl'intervalle [0 ; 3.3].



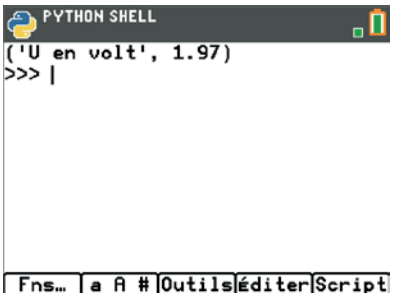
- Modifier votre script afin de prendre en compte la résolution du convertisseur analogique numérique (14 bits). Ainsi la mesure de la tension sera :

$$u = U_{alim} \times \frac{v}{2^{14}}$$

- La tension sera arrondie à 10^{-2} près.



- La tension précédente est d'environ 1.97 V.



Quelques idées pour prolonger la leçon :

- Utiliser un potentiomètre pour réaliser un capteur angulaire (une mesure de tension correspond à la valeur d'un angle lue sur un rapporteur), puis réaliser la représentation graphique de la fonction modélisée $\alpha = f(u)$.
- Entre 0 et 3.3v, associer une plage de tension à une couleur en utilisant la DEL RVB du TI-Innovator™.
- Associer la mesure de la tension aux coordonnées d'un point repéré (principe d'un joystick)…etc.

Unité 6 : Utiliser le module ti_hub & ti_rover

Compétence 3 : Les dispositifs d'entrée-sortie

Dans cette seconde leçon de l'unité 6, vous allez découvrir comment connecter le ti-rover à l'aide de la librairie **ti_rover**.

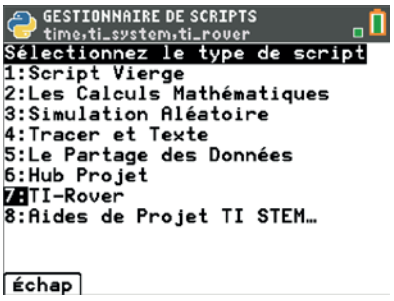
Objectifs :

- Découvrir le module **ti_rover**.
- Écrire et utiliser un script permettant d'utiliser le ti-rover et ses actionneurs associés.
- Utiliser une boucle ouverte et une instruction conditionnelle.

Vous allez, dans cette leçon, réaliser un script donnant au ROVER la possibilité d'effectuer un parcours marqué par l'illumination de la diode RVB, tant que la distance (mesurée par le capteur RANGER) respecte une limite inscrite dans une instruction conditionnelle.

```
Avancer sur une distance de 2m
    Tant que le mouvement n'est pas stoppé par l'utilisateur
    | a ⟵ distance à un objet mesurée
    |       si a < 0.2
    |       |   alors afficher une couleur rouge et s'arrêter
    |       |   sinon afficher une couleur verte et continuer
    S'arrêter, afficher une couleur bleue
    Attendre 1s
    Eteindre la diode
Allumer la diode en bleu pour marquer la fin
```

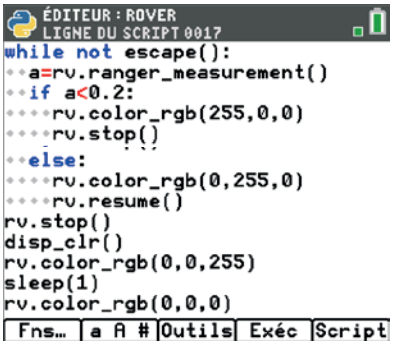
- Commencer un nouveau script et le nommer U6SB3.
- A partir du menu **modul**, importer la librairie **ti-rover**.
- Valider en appuyant sur la touche [entrer].



- Effacer votre écran à l'aide de l'instruction **disp_clr()** située dans le menu **ti_system**.
- Toujours dans le menu **ti_system**, choisir l'instruction **disp_cursor( )** affectée de la valeur 0 afin de ne pas afficher le curseur.
- Demander au ROVER de se déplacer en avant. L'unité de mesure de la distance est laissée à votre choix sachant que par défaut, celle-ci est fixée à 0.1 m. Ainsi **rv.forward(20)** assignera au robot un déplacement en avant sur une distance de 2 m. L'instruction rv.forward( ) est située dans le menu **modul**, puis **7 : ti_rover** et enfin **2 : forward(distance)** dans le menu **Conduire.**
- Inscrire ensuite, le début d'une boucle ouverte que l'on trouve dans le menu **modul** puis dans celui de la librairie **ti_system**.

Conseil à l'enseignant : Un grand nombre d'instructions disponibles dans le menu de la librairie **ti_system**, le sont également dans celui de la librairie **ti_rover** sous le menu Commandes.

- Créer une variable **a** à laquelle est affectée la distance mesurée par le RANGER. Pour cela, commencer à écrire la lettre **a**, puis laisser le curseur à la fin de cette lettre. Inscrire ensuite l'instruction **rv.ranger_measurement( )** située dans le menu **modul** puis **7 : ti_rover** puis **E/S** (entrées sorties) ; **1 : Entrées** et enfin **1 : rv.ranger_measurement( ).** L'unité de mesure est le mètre.
- Créer à présent l'instruction conditionnelle. Si la distance mesurée est inférieure à 20 cm, le robot s'arrête et la diode RVB s'allume en rouge. L'instruction **rv.color( )** est disponible dans la librairie **ti_rover** au menu **E/S** puis **2 : sortie**. **rv.stop( )** est une instruction de conduite et donc placée sous le menu correspondant. Sinon la diode RVB est de couleur verte, et le robot poursuit son parcours jusqu'à attendre la distance fixée. L'instruction **rv.resume()** termine le traitement des actions en court dans la file d'attente.
- A la fin de la boucle :
  - Le robot s'arrête. **rv.stop()**.
  - L'écran est effacé.
  - La diode affiche une couleur bleue.
  - Un délai d'attente de 1s précède l'extinction de la diode.



- Effacer votre écran à l'aide de l'instruction **disp_clr()** située dans le menu **ti_system**.
- Toujours dans le menu **ti_system**, choisir l'instruction **disp_cursor( )** affectée de la valeur 0 afin de ne pas afficher le curseur.
- Demander au ROVER de se déplacer en avant. L'unité de mesure de la distance est laissée à votre choix sachant que par défaut, celle-ci est fixée à 0.1 m. Ainsi **rv.forward(20)** assignera au robot un déplacement en avant sur une distance de 2 m. L'instruction rv.forward( ) est située dans le menu **modul**, puis **7 : ti_rover** et enfin **2 : forward(distance)** dans le menu **Conduire.**
- Inscrire ensuite, le début d'une boucle ouverte que l'on trouve dans le menu **modul** puis dans celui de la librairie **ti_system**.

Conseil à l'enseignant : Un grand nombre d'instructions disponibles dans le menu de la librairie **ti_system**, le sont également dans celui de la librairie **ti_rover** sous le menu Commandes.

ÉDITEUR : U6SB3
LIGNE DU SCRIPT 0006

```
# TI-Rover
from time import *
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)_
```

Fns… | a A # | Outils | Exéc | Script

ÉDITEUR : U6SB3
LIGNE DU SCRIPT 0010

```
# TI-Rover
from time import *
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"[annul] pour arrêter"
    ,"center")
rv.forward(20)
while not escape():
  _
```

Fns… | a A # | Outils | Exéc | Script

- Créer une variable **a** à laquelle est affectée la distance mesurée par le RANGER. Pour cela, commencer à écrire la lettre **a**, puis laisser le curseur à la fin de cette lettre. Inscrire ensuite l'instruction **rv.ranger_measurement( )** située dans le menu **modul** puis **7 : ti_rover** puis **E/S** (entrées sorties) ; **1 : Entrées** et enfin **1 : rv.ranger_measurement( ).** L'unité de mesure est le mètre.
- Créer à présent l'instruction conditionnelle. Si la distance mesurée est inférieure à 20 cm, le robot s'arrête et la diode RVB s'allume en rouge. L'instruction **rv.color( )** est disponible dans la librairie **ti_rover** au menu **E/S** puis **2 : sortie**. **rv.stop( )** est une instruction de conduite et donc placée sous le menu correspondant. Sinon la diode RVB est de couleur verte, et le robot poursuit son parcours jusqu'à attendre la distance fixée. L'instruction **rv.resume()** termine le traitement des actions en court dans la file d'attente.
- A la fin de la boucle :
  - Le robot s'arrête. **rv.stop()**.
  - L'écran est effacé.
  - La diode affiche une couleur bleue.
  - Un délai d'attente de 1s précède l'extinction de la diode.

ÉDITEUR : ROVER
LIGNE DU SCRIPT 0017

```
while not escape():
  a=rv.ranger_measurement()
  if a<0.2:
    rv.color_rgb(255,0,0)
    rv.stop()
  else:
    rv.color_rgb(0,255,0)
    rv.resume()
rv.stop()
disp_clr()
rv.color_rgb(0,0,255)
sleep(1)
rv.color_rgb(0,0,0)
```

Fns… | a A # | Outils | Exéc | Script

Unité 6 : Utiliser le module ti_hub & ti_rover

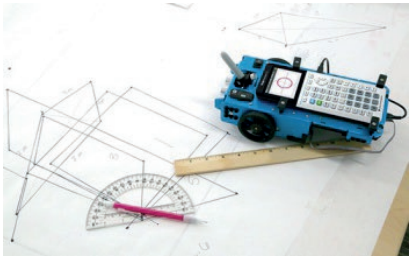
Application : Représenter un parcours

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 :

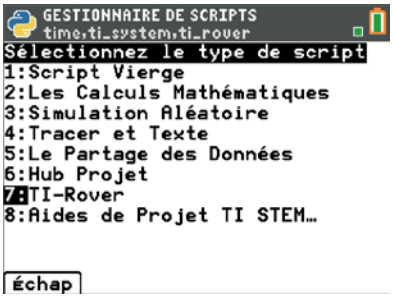
- Découvrir le module **ti_rover**.
- Écrire et utiliser un script permettant d'utiliser le ti-rover et ses actionneurs associés.
- Utiliser une boucle fermée.
- Représenter graphiquement des données.

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**.

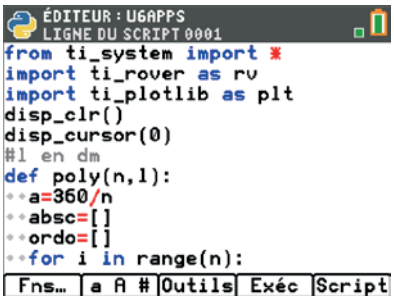


Mise en œuvre :

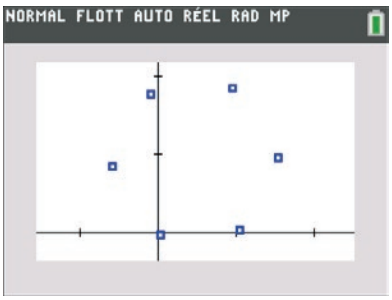
- Commencer un nouveau script et le nommer U6APPS
- A partir du menu **modul**, importer la librairie ti-rover
- Valider en appuyant sur la touche [entrer].
- Importer également la librairie **ti_plotlib**



- 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 **abs**c et **ordo** destinées à recevoir les coordonnées de chacun de ces sommets.



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.

education.ti.france

@TIEducationFR

TIedtechFR

Contactez notre service client TI-Cares
education.ti.com/fr/csc
01 41 04 60 40

education.ti.com/fr