Programmation en Python pour la calculatrice graphique TI-83 Premium CE *Édition Python*

Version 5.7.0 Bundle 83CE

Pour en savoir plus sur les technologies TI, consultez l'aide en ligne disponible à l'adresse education.ti.com/eguide.

Informations importantes

Sauf disposition contraire stipulée dans la licence qui accompagne un programme, Texas Instruments n'émet aucune garantie expresse ou implicite, y compris sans s'y limiter, toute garantie implicite de valeur marchande et d'adéquation à un usage particulier, concernant les programmes ou la documentation, ceux-ci étant fournis "tels quels" sans autre recours. En aucun cas, Texas Instruments ne peut être tenue responsable vis à vis de quiconque pour quelque dommage de nature spéciale, collatérale, fortuite ou indirecte occasionné à un tiers, en rapport avec ou découlant de l'achat ou de l'utilisation desdits matériels, la seule et exclusive responsabilité de Texas Instruments, pour quelque forme d'action que ce soit, ne pouvant excéder le montant indiqué dans la licence du programme. Par ailleurs, la responsabilité de Texas Instruments ne saurait être engagée pour quelque réclamation que ce soit en rapport avec l'utilisation desdits matériels par toute autre tierce partie.

« Python » et les logos Python sont des marques commerciales ou des marques déposées de Python Software Foundation, utilisées par Texas Instruments Incorporated avec l'autorisation de la Foundation.

Remarque : Les écrans réels peuvent présenter de légères différences par rapport aux images fournies.

Sommaire

Nouveautés

Nouveautés de l'application Python

TI-83 Premium CE *Édition Python*

Programmation en Python

- Lorsque l'application Python est chargée, vous y accédez via [prgm]. L'application Python figure également dans [2nd] [apps].
  - Tenez-vous informé des dernières nouveautés en consultant le site education.ti.com/83ceupdate.
  - Pour en savoir plus sur l'application Python, consultez le guide de programmation en Python, disponible sur le site education.ti.com/eguide.
- Collage rapide d'instructions d'importation pour les modules complémentaires. Les modules complémentaires sont disponibles dans les activités Python publiées sur le site education.ti.com.
- Les nouveaux modules complémentaires ti_draw et ti_image sont chargés à l'aide du bundle CE.
  - Dessinez et utilisez des images dans vos scripts Python.
- Le menu du module ti_system comprend désormais la méthode wait_key() pour simplifier l'utilisation.
- Les modules ti_hub et ti_rover prennent en charge la dernière version du logiciel TI-Innovator™ Hub Sketch v1.5.
  - Collecte de données : collectez plusieurs échantillons de données à l'aide d'une seule commande
  - Instructions composées permettant de synchroniser plusieurs résultats
  - TI-RGB Array : permet de commander des diodes DEL
  - Son : lecture de bips répétés à l'aide d'une seule commande
  - Capteur Ranger : renvoie « le temps de vol »

Transfert de scripts Python

Lors du transfert de scripts Python d'une plateforme non-TI vers une plateforme TI OU d'un produit TI vers une solution tierce :

- Les scripts Python qui utilisent des fonctions de base du langage et des bibliothèques standard (math, random etc.) peuvent être migrés sans modifications.
  Remarque : La longueur des listes est limitée à 100 éléments.
- Les scripts qui utilisent des bibliothèques propres à une plateforme – matplotlib (pour ordinateur),
  ti_plotlib, ti_system, ti_hub, etc. pour les plateformes TI – devront être modifiés avant de pouvoir être exécutés sur une plateforme différente.

Cela peut même s'appliquer à des scripts devant être transférés entre plateformes TI.

Pour de plus amples informations sur les nouveautés et les fonctionnalités mises à jour, rendez-vous sur le site education.ti.com/83ceupdate.

Application Python

Les sections suivantes décrivent l'utilisation, la navigation et l'exécution de l'application Python.

Utilisation de l'application Python

L'application Python est disponible pour la TI-83 Premium CE *Édition Python* Les informations incluses dans ce guide électronique s'appliquent à la TI-83 Premium CE *Édition Python* mise à jour avec le dernier bundle CE.

Lorsque vous exécutez pour la première fois l'application Python sur votre TI-83 Premium CE *Édition Python*, vous serez peut-être invité à mettre à jour votre version vers le bundle CE disponible pour la dernière version de l'application Python. Consultez le site education.ti.com/83ceupdate pour mettre à jour votre TI-83 Premium CE *Édition Python*.

Gestion de la mémoire des scripts Python (AppVar PY)

L'application Python propose un Gestionnaire de scripts, un Éditeur pour créer des scripts et une console (Shell) pour exécuter les scripts et interagir avec l'interpréteur Python. Les scripts Python enregistrés ou créés en tant que variables Python (AppVars) sont exécutés à partir de la mémoire RAM. Vous pouvez stocker les scripts Python AppVars dans la mémoire archive à des fins de gestion de la mémoire [2nde] [mém] 2:. Si l'écran Python App File Manager (Gestionnaire de fichiers de l'application Python) n'affiche pas l'un de vos scripts **AppVar PY**, vous pouvez déplacer un script Python **AppVar PY** de la calculatrice entre la RAM et la mémoire Archive comme indiqué. Le * indique un fichier dans la mémoire Archive. Appuyez sur [enter] (83CE FR [entrer]) pour déplacer le fichier entre les mémoires RAM et Archive.

[2nde][mém] 2 > B:Var App... [entrer] RAM <> *ARC

Remarque : Si vous possédez une calculatrice TI-83 Premium CE, consultez le site education.ti.com/83ceupdate pour prendre connaissance des dernières informations sur votre CE, notamment votre expérience Python spécifique.

Navigation dans l'application Python

Utilisez les touches de raccourci affichées à l'écran pour naviguer entre les différents espaces de travail de l'application Python. Dans l'image, les onglets de raccourci indiquent :

* Accès au Gestionnaire de scripts [Script]

** Accès à l'Éditeur : [Édit] ou [Éditer]

*** Accès à la console Shell [Shell]

Accédez aux onglets de raccourci de l'écran en utilisant la ligne de touches graphiques située immédiatement en dessous de l'écran. Reportez-vous également à la section Clavier. Le menu Éditeur > Outils et le menu Shell > Outils comportent également des options de navigation.

Exemple d'activité

L'exemple d'activité présenté ici a pour objectif de vous familiariser avec les espaces de travail disponibles dans l'application Python.

- Créez un nouveau script à partir du Gestionnaire de scripts.
- Écrivez le script dans l'Éditeur.
- Exécutez le script dans le Shell de l'application Python.

Pour en savoir plus sur la programmation en Python sur votre calculatrice CE, consultez les ressources relatives à la TI-83 Premium CE *Édition Python*.

Pour commencer :

- Exécutez l'application Python.

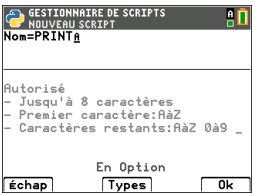
Remarque : Les écrans réels peuvent présenter de légères différences par rapport aux images fournies.

Saisissez le nom du nouveau script à partir du Gestionnaire de scripts.

- Appuyez sur [zoom] ([Nouv]) pour créer un nouveau script.

Saisie du nom du nouveau script

- L'exemple de script utilisé est PRINT. Saisissez le nom du script, puis appuyez sur [graphe] ([Ok]).
- Notez que le curseur est en verrouillage ALPHA. Saisissez toujours un nom de script conforme aux règles affichées à l'écran.



Astuce : Si le curseur n'est pas en verrouillage ALPHA, appuyez sur [2nde] [alpha] [alpha] pour activer les lettres majuscules.

Saisissez le nom du script comme indiqué.

Astuce : L'application offre la saisie rapide. Vérifiez toujours l'état du curseur au début d'un script !

Caractères alphabétiques du clavier 	[alpha] affiche en alternance le curseur d'insertion dans l'Éditeur et dans le Shell. _ non-alpha a alpha en minuscules

	A ALPHA en majuscules
Où se trouve le signe égal ?	Appuyez sur sto→ lorsque le curseur correspond à _. 
Où se trouvent ces fonctions ? input() print()	[Fns...] E/S 1:print() 2:input()
Où se trouve le guillemet double ?	alpha ["] 
Où se trouvent (et) ?	Utilisez le clavier lorsque le curseur correspond à _. 

Essayez ! [a A #] et [2nde] [catalog] sont également des aides facilitant la saisie rapide si nécessaire.

Exécutez le script PRINT.

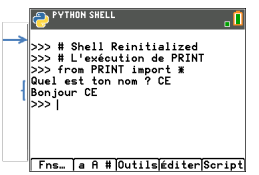
- Dans l'Éditeur, appuyez sur [trace] ([Exéc]) pour exécuter votre script dans la console Shell.
- Saisissez votre nom en réponse à l'invite « Quel est ton nom ? ».
- Le résultat affiche « Bonjour » suivi de votre nom.

Remarque : À l'invite du Shell >>>, vous pouvez exécuter une commande telle que 2+3. Si vous utilisez des fonctions provenant des modules math ou random, pensez à toujours exécuter au préalable une instruction import, comme dans n'importe quel environnement de codage en Python.

Indicateur d'état du curseur Shell.

Saisissez votre nom.

Le résultat du script BONJOUR s'affiche.



Configuration d'une session Python avec vos scripts

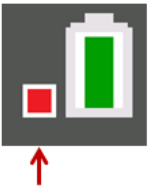
Lorsque vous exécutez l'application Python, la connexion CE établie avec l'expérience TI-Python lance la synchronisation pour la session Python en cours. Votre liste de scripts, présents dans la mémoire RAM, s'affiche lors de la synchronisation avec l'expérience Python.

Lorsque la session Python est établie, la barre d'état contient un indicateur carré vert près de l'icône de la batterie signalant que la session Python est prête à être utilisée. Si l'indicateur est rouge, patientez jusqu'à ce qu'il redevienne vert, lorsque l'expérience Python est à nouveau disponible.

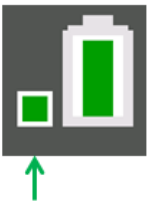
Vous observerez peut-être une synchronisation complète de vos programmes avec l'expérience TI-83 Premium CE *Édition Python* lorsque vous mettrez à jour votre version à partir du site education.ti.com/83ceupdate.

Déconnexion et reconnexion de l'application Python

Lorsque l'application Python est exécutée, la barre d'état affiche un indicateur signalant si l'adaptateur est prêt à fonctionner. Tant que la connexion n'est pas établie, le clavier CE ne répond pas forcément. Au cours d'une session Python, il est recommandé de consulter l'indicateur de connexion de la barre d'état.



Python non prêt



Python prêt

Captures d'écran

Avec TI Connect™ ce sur education.ti.com/83ceupdate, les captures d'écran de n'importe quel écran d'application Python sont autorisées.

Espaces de travail Python

L'application Python comprend trois espaces de travail pour développer votre programmation en Python.

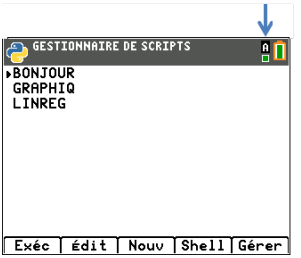
- Gestionnaire de scripts
- Éditeur
- Console (Shell)

Gestionnaire de scripts Python

Le Gestionnaire de scripts dresse la liste des scripts Python AppVars disponibles dans la mémoire RAM de votre calculatrice. Il vous permet de créer, de modifier et d'exécuter des scripts, de même que d'accéder au Shell.

En mode alpha, il vous suffit d'appuyer sur une lettre du clavier pour accéder directement aux scripts dont le nom commence par cette lettre.

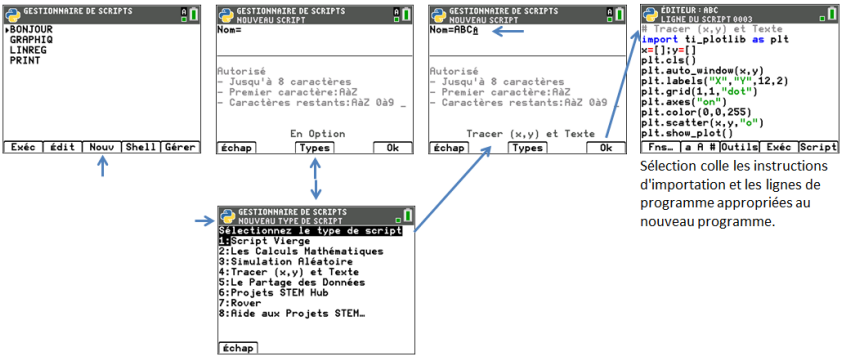
Appuyez au besoin sur la touche [alpha] lorsque l'indicateur **A** n'est pas visible sur la barre d'état.



Menus et touches de raccourci du Gestionnaire de scripts		
Menus	**Touche d'accès**	**Description**
[Exéc]	[f(x)]	Sélectionnez un script à l'aide des touches ▲ ou ▼. Sélectionnez ensuite [Exéc] pour exécuter votre script.
[Édit]	[fenêtre]	Sélectionnez un script à l'aide des touches ▲ ou ▼. Sélectionnez ensuite [Édit] pour afficher le script dans l'Éditeur afin de le modifier.
[Nouv]	[zoom]	Sélectionnez [Nouv] pour saisir le nom d'un nouveau script et accéder à l'Éditeur afin d'écrire ce nouveau script. Dans l'écran [Nouveau script], sélectionnez [Types] (appuyez sur [zoom]) pour sélectionner un type de script. Suite à cette sélection, un modèle d'instructions d'importation et de fonctions et méthodes fréquemment utilisées seront collés dans votre nouveau script pour cette activité.
[Shell]	[trace]	Sélectionnez [Shell] pour afficher l'invite de la console Shell (l'interpréteur Python). Le Shell s'affiche dans l'état actif.
[Gérer]	[graphe]	Sélectionnez [Gérer] pour : • Afficher le numéro de version.

Menus et touches de raccourci du Gestionnaire de scripts		
Menus	**Touche d'accès**	**Description**
		• Dupliquer, supprimer ou renommer un script sélectionné. • Afficher l'écran À propos. • Quitter l'application. Vous pouvez également utiliser 2nde [quitter].

Création d'un nouveau script à l'aide de modèles de type de script

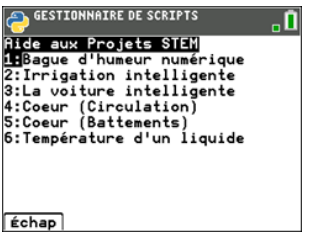


Sélection colle les instructions d'importation et les lignes de programme appropriées au nouveau programme.

- Sélectionnez [types] pour afficher le menu Sélectionner le type de programme.
- Importation(s) s'affiche(s) dans la barre d'état.

Création d'un nouveau script d'activité STEM à l'aide de modèles

Lorsque l'AppVar TISTEMFR est chargée dans la mémoire d'archive, l'élément « Aide aux Projects STEM… » s'affiche dans le menu Sélectionnez le type de script. Sélectionnez le modèle d'activité STEM approprié afin de commencer un nouveau script STEM.



Éditeur Python

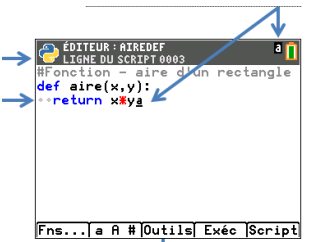
L'Éditeur Python s'affiche à partir d'un script sélectionné dans le Gestionnaire de scripts ou à partir du Shell. L'Éditeur affiche en couleur les mots-clés, les opérateurs, les commentaires, les chaînes et les retraits. Le collage rapide de fonctions et mots-clés Python courants est disponible, de même que la saisie directe au clavier et l'entrée des caractères [a A #]. Lorsque vous collez un bloc de code tel que if.. elif.. else, l'Éditeur vous propose le retrait automatique, que vous pouvez modifier au besoin à mesure que vous écrivez votre script.

Emplacement du curseur sur la ligne de script.

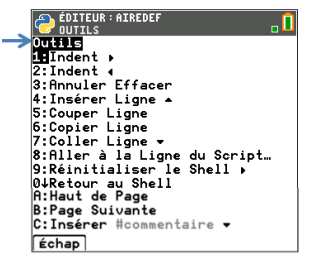
Le curseur est toujours en mode d'insertion. Les touches 2nde et alpha permettent d'alterner entre les états du curseur : numérique, a et A. La touche [suppr] se comporte comme le retour arrière et supprime un caractère.

Blocs de code avec retrait automatique.

La mise en retrait des lignes est indiquée visuellement par des points gris.



Outils pratiques pour éditer et travailler dans le Shell. Une description complète est fournie ci-dessous.



Menus et touches de raccourci de l'Éditeur Python		
Menus	**Touche d'accès**	**Description**
[Fns…]	f(x)	Sélectionnez [Fns…] pour accéder aux menus des fonctions, mots-clés et opérations courantes. Il vous permet également d'accéder à une sélection de contenus dans les modules

<table>
<tr><th colspan="3">Menus et touches de raccourci de l'Éditeur Python</th></tr>
<tr><th>Menus</th><th>Touche d'accès</th><th>Description</th></tr>
<tr><td></td><td></td><td>math et random.
Remarque : 2nde [catalog] est également pratique pour le collage rapide.</td></tr>
<tr><td>[a A #]</td><td>fenêtre</td><td>Sélectionnez [a A #] afin d'accéder à une palette de caractères servant de méthode alternative pour saisir de nombreux caractères.</td></tr>
<tr><td>[Outils]</td><td>zoom</td><td>Sélectionnez [Outils] pour accéder à des fonctions d'aide à l'édition ou aux interactions avec le Shell.
<table>
<tr><td>1 : Indent ▶</td><td>Met en retrait la ligne de script vers la droite et positionne le curseur sur le premier caractère de la ligne.</td></tr>
<tr><td>2 : Indent ◀</td><td>Réduit la mise en retrait de la ligne de script vers la gauche. Le curseur se positionne sur le premier caractère de la ligne.</td></tr>
<tr><td>3 : Annuler Effacer</td><td>Colle la dernière ligne effacée sur une nouvelle ligne placée sous la ligne de script sur laquelle se trouve le curseur. Le curseur s'affiche à la fin de la ligne collée.</td></tr>
<tr><td>4 : Insérer Ligne (flèche vers le haut)</td><td>Insère une ligne au-dessus de la ligne de script sur laquelle se trouve le curseur. La ligne est mise en retrait et affiche au besoin des points de mise en retrait.</td></tr>
<tr><td>5 : Couper Ligne</td><td>La ligne de script active sur laquelle se trouve le curseur est coupée. Le curseur s'affiche sur la ligne de script située en dessous de la ligne coupée.</td></tr>
<tr><td>6 : Copier Ligne</td><td>Copie la ligne de script active sur laquelle se trouve le curseur. Il est possible de coller une ligne de script copiée sur l'invite du Shell. Voir la section Shell ci-dessous.</td></tr>
<tr><td>7 : Coller Ligne (flèche vers le bas)</td><td>Colle la dernière ligne de script conservée sur la ligne située en dessous de la position du curseur.</td></tr>
<tr><td>8 : Aller à la Ligne du Script...</td><td>Affiche le curseur au début de la ligne de script spécifiée.</td></tr>
</table></td></tr>
</table>

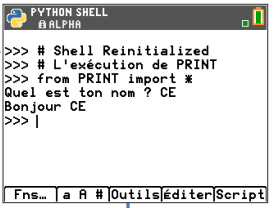
<table>
<tr><th colspan="3">Menus et touches de raccourci de l'Éditeur Python</th></tr>
<tr><th>Menus</th><th>Touche d'accès</th><th>Description</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>9 : Réinitialiser le Shell</td><td>Affiche la console Shell réinitialisée.</td></tr>
<tr><td>0 : Retour au Shell</td><td>Affiche le Shell dans son état actuel.</td></tr>
<tr><td>A : Page Précédente</td><td>Affiche 11 lignes de script disponibles au-dessus de la position actuelle du curseur.</td></tr>
<tr><td>B : Page Suivante</td><td>Affiche 11 lignes de script disponibles sous la position actuelle du curseur.</td></tr>
<tr><td>C : Insérer #comment en dessous</td><td>Insère # sur une nouvelle ligne située en dessous de la position du curseur.</td></tr>
</table></td></tr>
<tr><td>[Exéc]</td><td>trace</td><td>Sélectionnez [Exéc] pour exécuter votre script.</td></tr>
<tr><td>[Script]</td><td>graphe</td><td>Sélectionnez [Script] pour afficher le Gestionnaire de scripts.</td></tr>
</table>

La console Python (Shell)

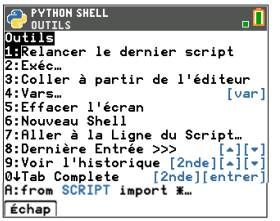
La console Python (Shell) vous permet d'interagir avec l'interpréteur Python ou d'exécuter des scripts Python. Le collage rapide de fonctions et mots-clés Python courants est disponible, aussi bien par la saisie directe au clavier que par l'entrée de caractères [a A #]. L'invite du Shell peut vous servir à tester une ligne de code collée à partie de l'Éditeur. Il est également possible de saisir plusieurs lignes de code et de les exécuter depuis l'invite du Shell >>>.

Indicateur d'état du curseur Shell.

Le Shell est réinitialisé lors de l'exécution d'un nouveau script.



Outils pratiques pour travailler dans le Shell. Voir les détails ci-dessous.



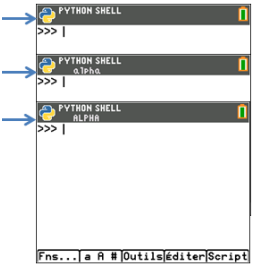
États du curseur Shell

non-alpha

2nde alpha si nécessaire pour basculer

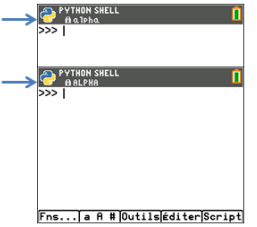
alpha alpha

alpha ALPHA une nouvelle fois



2nde alpha verrouillage alpha

alpha une nouvelle fois verrouillage ALPHA



<table>
<tr><th colspan="3">Menus et touches de raccourci du Shell Python</th></tr>
<tr><th>Menus</th><th>Touche d'accès</th><th>Description</th></tr>
<tr><td>[Fns…]</td><td>f(x)</td><td>Sélectionnez [Fns…] pour accéder aux menus des fonctions, mots-clés et opérations courantes. Il vous permet également d'accéder à une sélection de contenus dans les modules math et random.
Remarque : 2nde [catalog] est également pratique pour le collage rapide.</td></tr>
<tr><td>[a A #]</td><td>fenêtre</td><td>Sélectionnez [a A #] afin d'accéder à une palette de caractères servant de méthode alternative pour saisir de nombreux caractères.</td></tr>
<tr><td>[Outils]</td><td>zoom</td><td>Sélectionnez [Outils] pour afficher les éléments de menu suivants.
<table>
<tr><td>1 : Relancer le dernier script</td><td>Relance le dernier script exécuté dans le Shell.</td></tr>
<tr><td>2 : Exéc…</td><td>Affiche la liste des scripts Python qu'il est possible d'exécuter dans le Shell.</td></tr>
<tr><td>3 : Coller à partir de l'éditeur</td><td>Colle la dernière ligne de script copiée à partir de l'Éditeur dans l'invite du Shell.</td></tr>
<tr><td>4 : Vars…</td><td>Affiche les variables du dernier script exécuté. N'affiche pas les variables définies dans un script importé.</td></tr>
<tr><td>5 : Effacer l'écran</td><td>Efface l'écran du Shell. Ne réinitialise pas le Shell.</td></tr>
<tr><td>6 : Nouveau Shell</td><td>Réinitialise le Shell.</td></tr>
<tr><td>7 : Aller à la Ligne du Script…</td><td>Affiche l'Éditeur à partir du Shell en plaçant le curseur sur la ligne de script spécifiée.</td></tr>
<tr><td>8 : Dernière Entrée >>> ▲ ▼</td><td>Affiche jusqu'aux 8 dernières entrées à l'invite de la console au cours d'une session Shell.</td></tr>
<tr><td>9 : Voir l'historique
2nde ▲
2nde ▼</td><td>Permet de faire défiler l'écran du Shell pour afficher les 60 dernières lignes générées dans la console au cours d'une session Shell. Après avoir utilisé les fonctions de dessin dans le Shell à l'aide du module ti_plotlib, ti_draw ou ti_image,</td></tr>
</table></td></tr>
</table>

Menus et touches de raccourci du Shell Python			
Menus	**Touche d'accès**	**Description**	
			appuyez sur [annul] pour effacer le dessin et revenir à l'invite du Shell. L'historique ne s'affichera pas. Le cas échéant, appuyez sur les touches [2nde] ▲ et [2nde] ▼ pour afficher l'historique.
		0 : Tab Complete [2nde] [entrer]	Affiche les noms des variables et des fonctions accessibles pendant la session Shell en cours. Lorsque vous entrez la première lettre d'une variable ou d'une fonction disponible, appuyez sur [2nde] [entrer] pour compléter automatiquement le nom si une correspondance est disponible dans la session Shell en cours.
		A: from SCRIPT import *...	Lors de sa première exécution dans une session Shell, le SCRIPT est exécuté et les variables sont uniquement visibles en utilisant Tab Complete. Lorsque vous relancez le script au cours de la même session Shell, l'exécution apparaît comme non effectuée. Cette commande peut également être collée à partir de [2nde] [catalog].
[Éditer]	[trace]	Sélectionnez [Éditer] pour afficher l'Éditeur avec le dernier script édité. Si la fenêtre de l'Éditeur est vide, vous pouvez afficher le Gestionnaire de scripts.	
[Script]	[graphe]	Sélectionnez [Script] pour afficher le Gestionnaire de scripts.	

Remarque :

- Pour interrompre un script Python en cours d'exécution, par exemple lorsqu'un script se trouve dans une boucle infinie, appuyez sur [on]. Appuyez sur **[Outils]** ([zoom]) > **6:Nouveau Shell** comme méthode alternative pour arrêter un script en cours d'exécution.
- Lorsque vous utilisez les modules ti_plotlib, ti_draw ou ti_image pour accéder aux tracés dans le Shell, appuyez sur la touche [annul] pour effacer le tracé et revenir à l'invite du Shell située en haut de l'écran. Pour afficher l'historique du Shell, appuyez sur les touches [2nde] ▲ et [2nde] ▼ selon vos besoins.

Erreur D'Exécution : Accédez À La Ligne De Programme À L'Aide De Shell > Outils

Lors de l'exécution du code, l'adaptateur TI-Python affiche les messages d'erreur Python dans le Shell. Si un message d'erreur s'affiche lorsqu'un script est en cours d'exécution, un numéro de ligne de script est indiqué. Choisissez Shell > Outils 7:Aller à la Ligne du Script… Entrez le numéro de ligne, puis appuyez sur **[OK]**. Le curseur s'affiche au niveau du premier caractère de la ligne de script appropriée dans l'Éditeur. Le numéro de la ligne de script s'affiche sur la deuxième ligne de la barre d'état dans l'Éditeur.

Support d'édition rapide

Pour saisir du code dans l'Éditeur ou dans le Shell, utilisez les méthodes suivantes afin de coller rapidement une entrée dans la ligne d'édition.

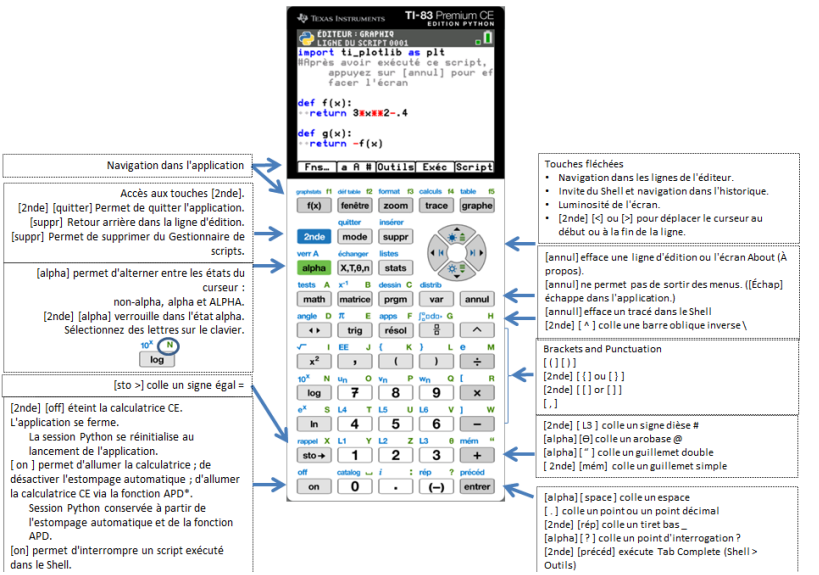
Conseils de saisie rapide

- Utilisation du clavier Python
- Utilisation du catalogue Python
- Utilisation du jeu de caractères [a A #]

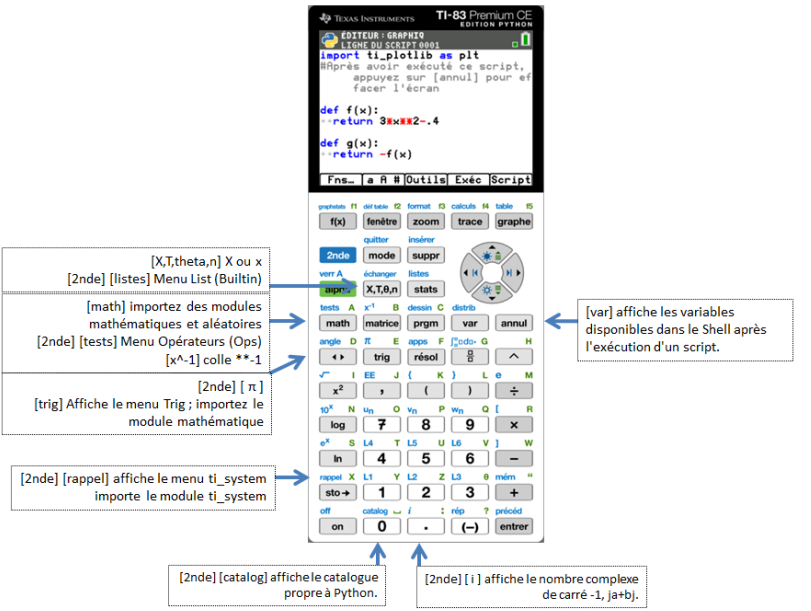
Utilisation du clavier Python

Lorsque l'application Python est en cours d'exécution, le clavier est prévu pour coller les opérations Python appropriées ou pour ouvrir des menus destinés à faciliter la saisie des fonctions, mots-clés, méthodes, opérateurs, etc. Les touches [2nde] et [alpha] vous permettent d'accéder aux deuxième et troisième fonctions d'une touche comme dans le système d'exploitation.

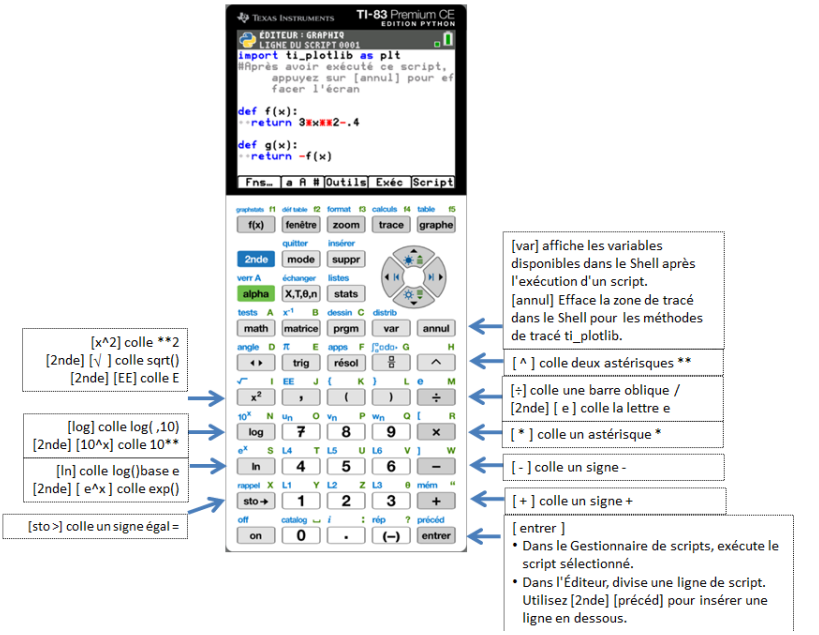
Navigation, édition et caractères spéciaux par rangées de touches dans l'application Python



Touches spécifiques dans l'application Python pour accéder aux menus et fonctions par rangées de touches



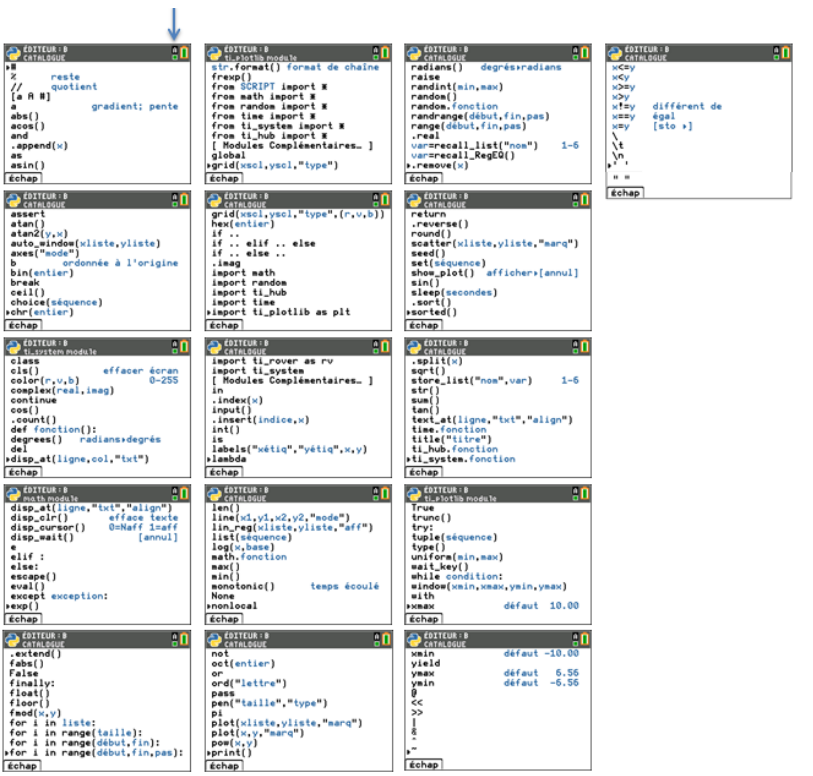
Touches spécifiques dans l'application Python pour accéder aux menus et fonctions par rangées de touches (suite)



Utilisation du catalogue Python

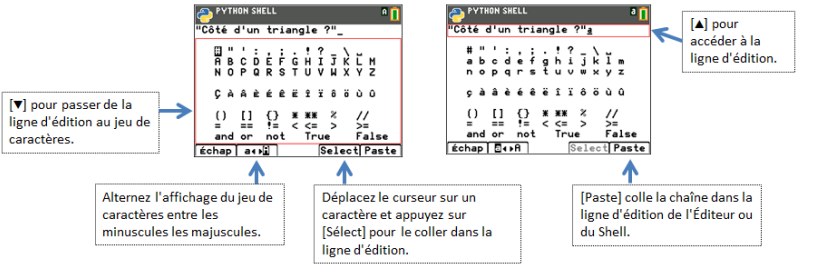
Lorsque l'application Python est en cours d'exécution, 2nde [catalog] affiche une liste de séparateurs, mots-clés, fonctions et opérateurs fréquemment utilisés pour que vous puissiez facilement les coller dans une ligne d'édition. 2nde [catalog] est uniquement disponible dans l'Éditeur et le Shell. Pour une description détaillée de chaque élément du catalogue, consultez le Guide de référence. En haut du menu Catalogue, appuyez sur ▲ pour parcourir le catalogue d'un bout à l'autre.

Dans l'écran du catalogue, sélectionnez alpha et une touche représentant une lettre pour afficher la liste à partir de cette lettre.



Utilisation du jeu de caractères [a A #]

L'onglet de raccourci [a A #], qui permet d'accéder à une palette de caractères, est une fonction pratique pour saisir des chaînes de caractères dans l'Éditeur ou dans le Shell.



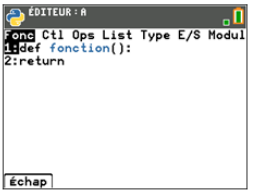
Remarque : Lorsque le curseur se trouve dans la ligne d'édition [a A #], certaines touches du clavier ne sont pas disponibles. Lorsque le curseur se trouve dans le jeu de caractères, les fonctions du clavier sont limitées.

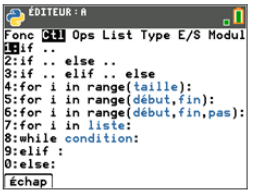
Menus [Fns], modules et modules complémentaires

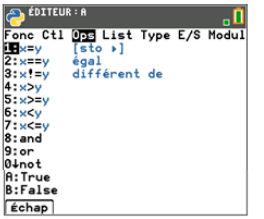
Menus [Fns…]

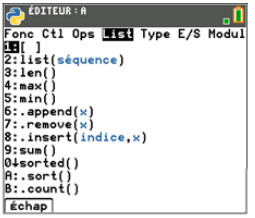
L'onglet de raccourci [Fns…] affiche les menus contenant les fonctions, mots-clés et opérateurs Python fréquemment utilisés. Les menus permettent également d'accéder aux fonctions et constantes sélectionnées dans les modules et modules complémentaires. Même si vous pouvez saisir du code caractère par caractère à partir du clavier, ces menus vous offrent un moyen rapide de coller des données dans l'Éditeur ou le Shell. Appuyez sur [Fns…] dans l'Éditeur ou le Shell. Reportez-vous également aux sections Utilisation du catalogue Python et Utilisation du clavier Python pour d'autres méthodes de saisie.

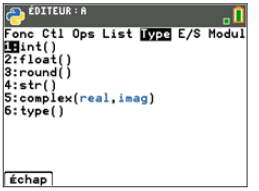
[Fns...] Éléments intégrés (Built-ins), opérateurs et mots-clés

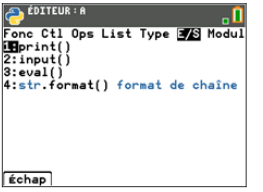










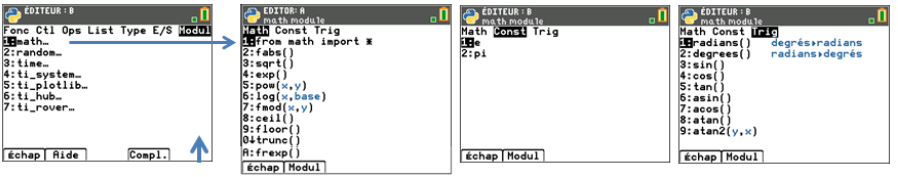


[Fns...] Modules

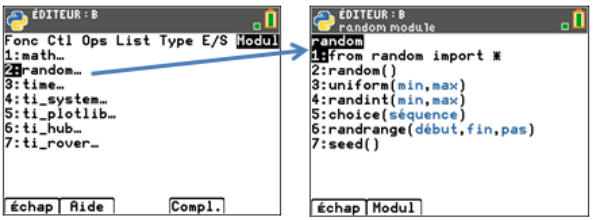
Lorsque vous utilisez une fonction ou une constante Python à partir d'un module, utilisez toujours une instruction d'importation pour indiquer dans quel module se trouve la fonction, la méthode ou la constante.

Voir Présentation de l'expérience de programmation Python

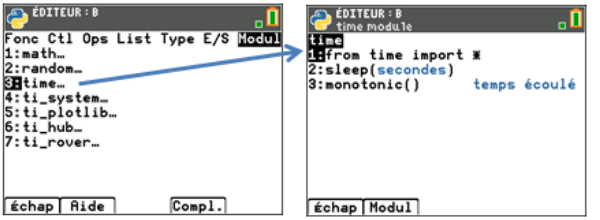
[Fns…]>Modul : module math



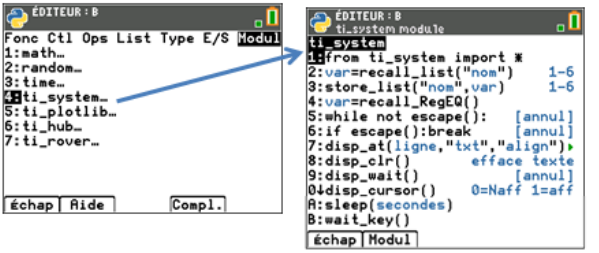
[Fns…]>Modul : module random



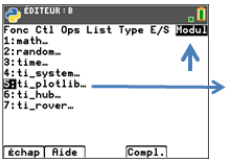
[Fns…]>Modul : module time

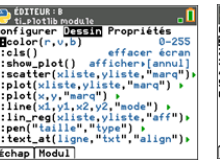


[Fns…]>Modul : module ti_system



[Fns…]>Modul : ti_plotlib





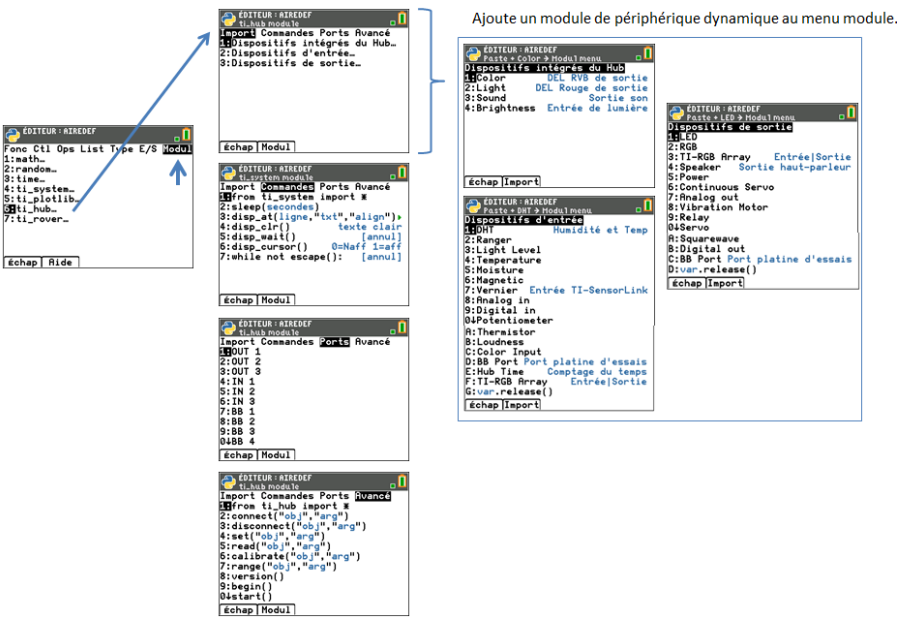
Remarque importante concernant les tracés :

- Afin de vous assurer d'obtenir les résultats attendus, vérifiez que l'ordre des lignes du script à utiliser pour le tracé suit celui indiqué dans le menu Configurer.
- Le tracé s'affiche lorsque plt.show_plot() est exécutée dans un script, après les objets du tracé. Pour effacer la zone de tracé dans le Shell, appuyez sur annul. Lorsque le Shell s'affiche, pour afficher l'historique du Shell, appuyez sur [2nde] ▲ et [2nde] ▼ ([Tools (Outils)] 9 :View History (Afficher l'historique)).
- L'exécution d'un deuxième script qui présuppose que les valeurs par défaut sont définies au sein du même environnement Shell aboutit généralement à un comportement inattendu au niveau de la couleur ou d'autres paramètres d'argument par défaut. Modifiez les scripts en utilisant des valeurs d'argument attendues ou réinitialisez le Shell avant d'exécuter un autre script de tracé.

[Fns…]>Modul : module ti_hub

Comme les méthodes ti_hub ne sont pas répertoriées dans le catalogue, elles ne figurent pas non plus dans le Guide de référence. Référez-vous aux informations affichées dans les écrans des menus concernant les arguments et les valeurs par défaut ou les valeurs admises correspondantes. Des informations complémentaires sur la programmation en Python pour TI-Innovator™ Hub et TI-Innovator™ Rover sont disponibles sur le site education.ti.com.

Remarque : Assurez-vous que le TI-Innovator™ Hub est connecté lorsque vous exécutez des scripts en Python.



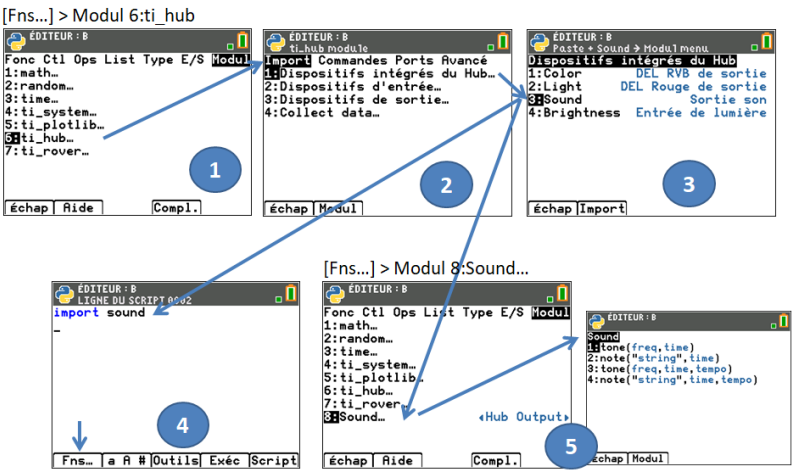
Module ti_hub – Ajout d'import à l'Éditeur et ajout du module de capteur ti_hub au menu Modul

Exemple d'écran : Importation d'un son

Pour importer des méthodes de capteur TI-Innovator™ dans votre script en Python, à partir de l'Éditeur, procédez comme suit :

1. Sélectionnez [Fns…] > Modul 6:ti_hub.
2. Sélectionnez le menu ti_hub Import. Sélectionnez un type de capteur dans Dispositifs intégrés du Hub, Dispositifs d'entrée et Dispositifs de sortie.
3. Sélectionnez un capteur.
4. Une instruction d'importation est collée dans l'Éditeur et le module du capteur devient disponible sous [Fns…] > Modul lorsque vous revenez à ce menu à partir de votre script.
5. Sélectionnez [Fns…] > Modul 8:Sound… pour coller des méthodes adaptées à ce capteur.

[Fns…]>Modul 6:ti_hub



Remarque : Brightns est un objet "intégré" (Built-in) dans TI-Innovator Hub.

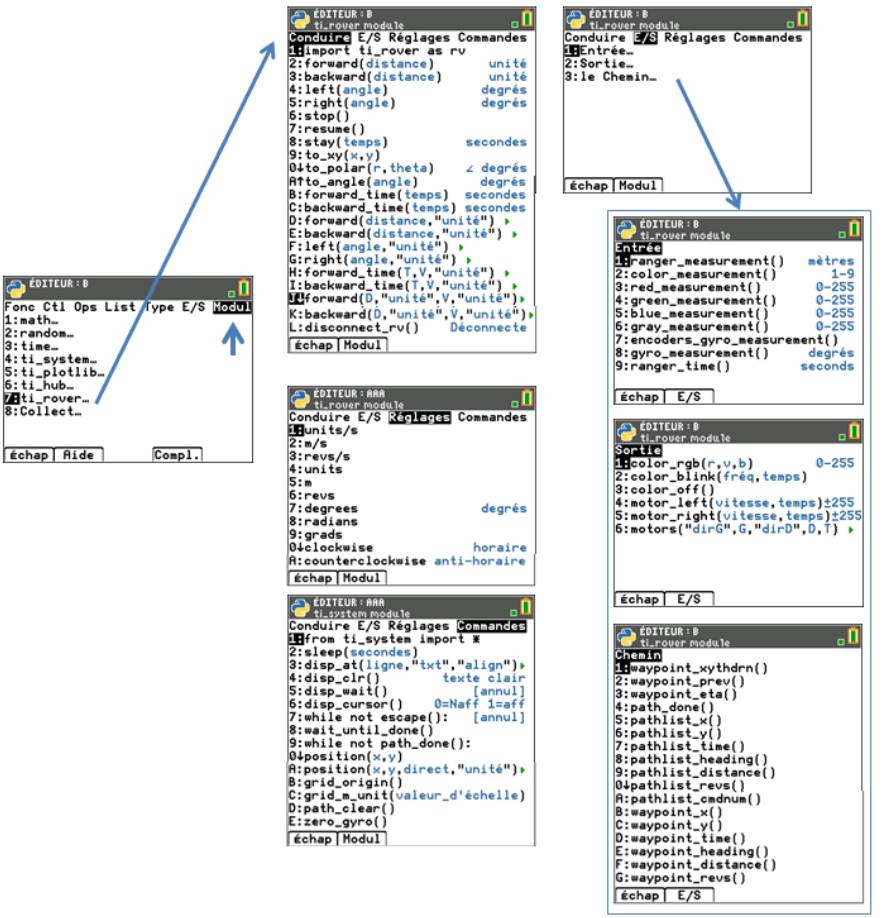
Lorsque vous utilisez l'instruction « import brightns », saisissez « brightns.range (0,100) » pour garantir l'exactitude de la plage par défaut au début de l'exécution du script.

Exemple :

```
import brightns
brightns.range(0,100)
b=brightns.measurement()
print(b)
```

[Fns…]>Modul module ti_rover

Comme les méthodes ti_rover ne sont pas répertoriées dans le catalogue, elles ne figurent pas non plus dans le Guide de référence. Référez-vous aux informations affichées dans les écrans des menus concernant les arguments et les valeurs par défaut ou les valeurs admises correspondantes. Des informations complémentaires sur la programmation en Python pour TI-Innovator™ Hub et TI-Innovator™ Rover sont disponibles sur le site education.ti.com.



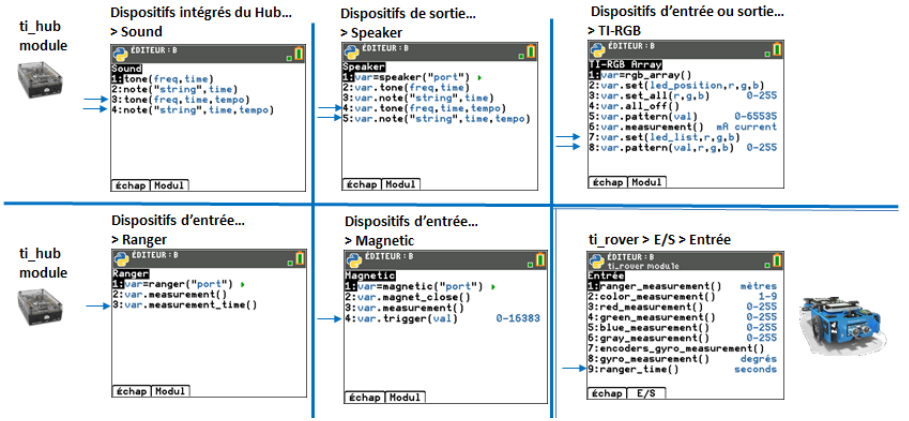
Remarques :

- En programmation TI-Python, il est inutile d'inclure des méthodes permettant de connecter et de déconnecter TI-Innovator™ Rover. Les méthodes Python pour TI-Innovator™ Rover gèrent parfaitement les connexions et les déconnexions sans

nécessiter de méthodes additionnelles. Ce langage est légèrement différent de la programmation de TI-Innovator™ Rover en TI-Basic.

- rv.stop() s'exécute en tant que pause, puis la commande de reprise « resume » continue avec les mouvements Rover placés dans la file d'attente. Si une autre commande de mouvement est exécutée après rv.stop(), alors la file d'attente des mouvements est effacée. Comme indiqué précédemment, ce langage est légèrement différent de la programmation de TI-Innovator™ Rover en TI-Basic.

Support Python de la version du logiciel TI-Innovator Sketch v1.5



[Fns...] Modules complémentaires

Les modules complémentaires améliorent l'expérience des modules de l'application Python avec des fonctionnalités supplémentaires et un accès facile aux méthodes Python supplémentaires à partir des menus de l'application Python.

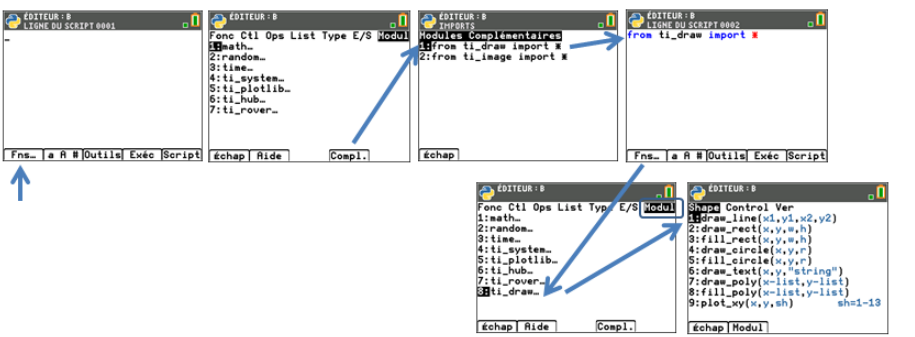
Vous remarquerez peut-être un module complémentaire à charger, à l'aide de TI Connect™ CE, dans le cadre d'une activité Python publiée sur education.ti.com comme ce_turtl, ce_chart, ce_box, ce_quivr et microbit selon la région. Vous aurez besoin de la dernière version des modules complémentaires actuellement publiés. Certains modules complémentaires seront chargés dans votre calculatrice, tels que ti_draw et ti_image, lorsque vous effectuez la mise à jour avec le dernier bundle CE.

L'application Python affiche les menus du module complémentaire dans le menu [Fns…]> Modul uniquement si votre script dans l'Éditeur démarre avec une instruction d'importation appropriée.

Collage d'une instruction d'importation de module complémentaire dans l'Éditeur

Étapes :

1. Création d'un nouveau script.
2. Dans l'Éditeur, sélectionnez [Fns…]> Modul.
3. Sélectionnez [Compl.] et lorsqu'un module complémentaire est chargé sur la calculatrice, un menu d'instruction d'importation pour les modules chargés s'affiche.
4. Sélectionnez l'instruction d'importation à coller dans l'Éditeur.
5. Sélectionnez [Fns…]> Modul pour localiser les menus du module complémentaire importé.



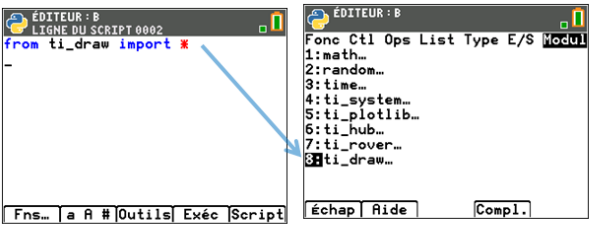
Les faits :

- [Add-On Modules Imports (Importation des modules complémentaires)…] figure également dans 2nde [catalog].

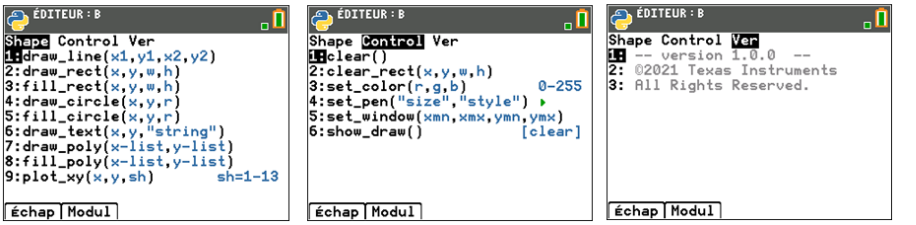
- Les modules complémentaires sont les fichiers “AppVar” de la calculatrice qui sont mémorisés dans la mémoire Archive et apparaissent dans [mém] comme une AppVar. Il est recommandé de conserver ces fichiers dans la mémoire Archive pour une expérience améliorée du module de l’application Python.
- Un script Python s'exécute dans l'application Python à partir du Gestionnaire de fichiers ou de l’Éditeur lorsque le script "AppVar PY" est dans la RAM. Si un script Python AppVar PY est placé dans la mémoire Archive, il ne sera pas disponible pour l'exécution ou l'édition dans l'application Python.

[Fns...] Module complémentaire ti_draw

Le module ti_draw est inclus dans le dernier bundle CE. Utilisez [Fns…]> Modul [Compl.] pour coller l'instruction d’importation dans votre script. Le menu ti_draw s'affiche alors dans le menu [Fns…] > Modul comme indiqué ici.



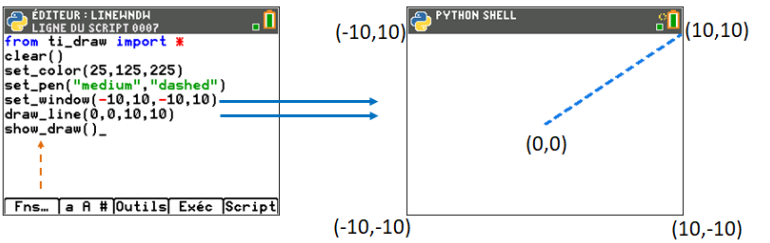
ti_draw menus



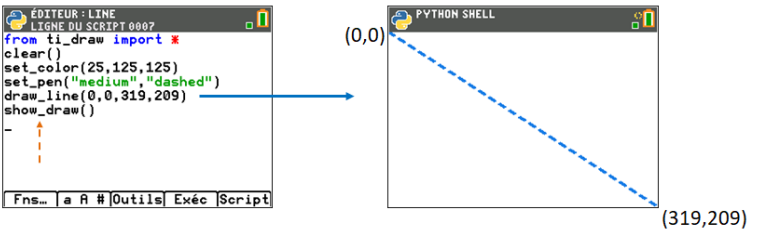
Informations sur le script lors de l'utilisation de ti_draw :

- Après l'instruction d’importation, utilisez la méthode clear() pour effacer la zone de dessin du Shell si nécessaire.
- Les scripts doivent contenir la commande show_draw() pour afficher le dessin lors de l'exécution du script.
- À l’aide des méthodes draw_rect(), draw_circle() ou draw_poly() dessinez la bordure de la construction tandis qu’avec les méthodes fill_rect(), fill_circle() et fill_poly(), vous remplissez l'intérieur de la figure spécifiée (selon la taille du crayon).
- Appuyez sur [annul] pour effacer le dessin et revenir à l’invite du Shell. L'historique du Shell peut être affiché à l'aide de [2nde] ▲ et [2nde] ▼.

- Veuillez lire les informations des menus Shape et Control dans le tableau ci-dessous. Vos dessins créés avec les méthodes du menu Shape dépendent des méthodes du menu Control telles que set_color() et set_ Pen().
- **Les arguments de coordonnées** sont les coordonnées de l'écran en pixels ou définies par la méthode set_window() .
  - méthodes ti_draw utilisant les coordonnées set_window()



  - méthodes ti_draw utilisant les coordonnées de l'écran en pixels



	Menu Shape	Description
1 :	draw_line(x1,y1,x2,y2)	Dessine un segment entre les points spécifiés (x1,y1) et (x2,y2).
2 :	draw_rect(x,y,w,h)	Dessine un rectangle ayant le coin supérieur gauche à (x,y) avec une largeur de w pixels et une hauteur de h.
3 :	fill_rect(x,y,w,h)	Remplit l'intérieur d'un rectangle ayant le coin supérieur gauche à (x,y) d'une largeur de w pixels et d'une hauteur de h pixels.
4 :	draw_circle(x,y,r)	Dessine un cercle dont le centre est situé à (x,y) et un rayon de r pixels.
5 :	fill_circle(x,y,r)	Dessine un cercle dont le centre est situé à (x,y) et un rayon de r pixels et rempli avec la couleur spécifiée (en utilisant set_ color ou le noir en l'absence de définition).
6 :	draw_text(x,y,"chaîne")	Dessine la chaîne de caractères sous

	Menu Shape	Description
		forme de texte à l'écran avec le coin supérieur gauche du texte commençant à (x,y).
7 :	draw_poly(x-list,y-list)	Dessine un ensemble de droites pouvant représenter un polygone. Les droites sont tracées en utilisant la taille et la couleur actuelles du crayon.
8 :	fill_poly(x-list,y-list)	Les arguments x-list et y-list doivent avoir la même longueur que les arguments de la liste des sommets (x,y). Le polygone est dessiné en connectant chaque paire de sommets et en remplissant la région avec la couleur actuelle du crayon.
9 :	poly_xy(x,y,sh) sh=1-13	En utilisant les arguments x et y comme emplacement du centre, la valeur (sh) de figure demandée ci-dessous sera dessinée. Les figures sont dessinées en utilisant la couleur actuelle du crayon. **Figure** **Description** 1 Cercle plein de rayon 2 2 Cercle vide de rayon 2 3 Carré plein 3x3 4 Carré vide 3 x 3 5 Le signe x est dessiné 6 Le signe + est dessiné 7 Pixel unique 8 Cercle plein de rayon 4 en pixels 9 Cercle vide de rayon 4 en pixels 10 Cercle plein de rayon 6 en pixels 11 Cercle vide de rayon 6 en pixels 12 Cercle plein de rayon 8 en pixels 13 Cercle vide de rayon 8 en pixels

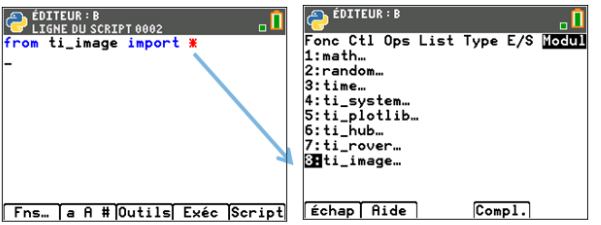
	Menu Control	Description
1 :	clear()	Efface la zone de dessin dans le Shell. Cette méthode doit être exécutée avant de dessiner pour s'assurer que la zone de dessin dans le Shell est effacée pour

	Menu Control	Description
		afficher les résultats attendus.
2 :	clear_rect(x,y,w,h)	Remplit l'intérieur d'un rectangle ayant le coin supérieur gauche à (x,y) et une largeur de w une hauteur de h. Le blanc est la couleur de remplissage par défaut. Après avoir collé la méthode dans l'Éditeur, la méthode peut accepter un cinquième argument facultatif pour spécifier une couleur différente via l'utilisation d'un tuple spécifiant une valeur (r,g,b). Un tuple valide (r,g,b) contient des valeurs entières comprises entre 0 et 255.
3 :	set_color(r,g,b) 0-255	Définit la couleur du crayon de dessin à l'aide du tuple (r,g,b).
4 :	set_pen("taille","style")	Règle la « taille » et le « style » du crayon de dessin pour tous les dessins suivants jusqu'à ce qu'une modification soit spécifiée. Lors de l'importation de ti_draw, la taille est "mince", "moyenne" ou "épaisse" et le style est "continu", "pointillés" ou "tirets". S'ils ne sont pas spécifiés, les arguments par défaut sont "mince" et "continu". L'assistant d'argument > permet de remplir correctement les chaînes d'arguments. **Remarque :** Lors de l'importation du module ti_plotlib, l'argument correspondant au style de la méthode pen() est "continu", "pointillés" ou "tirets".
5 :	set_window (xmn,xmx,ymn,ymx)	Définit la zone de dessin avec des plages de coordonnées [xmin,xmax] et [ymin,ymax] avec (0,0) au milieu des plages. Notez les points suivants : Si les valeurs d'argument sont en dehors de la zone de dessin spécifiée, aucune erreur n'est signalée. Si set_window(xmin,xmax,ymin,ymax) n'est pas exécuté dans un script, la taille de la fenêtre en pixels est la taille par

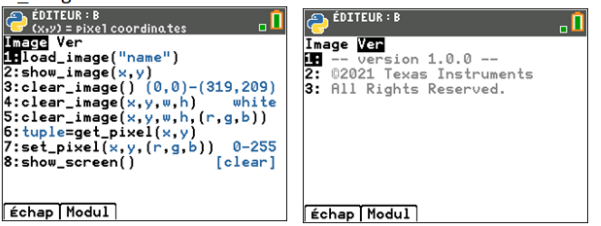
	Menu Control	Description
		défaut avec (xmin,xmax,ymin,ymax) = (0,319,0,209) avec (0,0) pour les coordonnées en pixels de la zone dans le coin supérieur gauche.
6 :	show_draw() [clear]	Doit être inclus pour afficher le dessin. Appuyez sur [annul] pour effacer le dessin et revenir à l'invite du Shell. Pour afficher l'historique du Shell, appuyez sur [2nde] ▲ et [2nde] ▼.

[Fns...] Module complémentaire ti_image

Le module ti_image est inclus dans le dernier bundle CE. Utilisez [Fns…]> Modul [Compl.] pour coller l'instruction d'importation dans votre script. Le menu ti_image s'affiche alors dans le menu [Fns…] > Modul comme indiqué ici.



ti_image menus



- Le module TI_image peut être utilisé pour
  - afficher votre image Python nommée AppVar mémorisée dans la mémoire Archive de votre CE.
  - afficher les rectangles remplis avec la couleur spécifiée dans un emplacement de coordonnées en pixels.
  - définir ou obtenir une couleur de pixel.
  - effacer l'intérieur d'une zone d'écran rectangulaire.
  - effacer la zone de dessin plein écran dans le Shell à partir des coordonnées de pixels allant de (0,0) à (319,209).
- Après l'instruction d'importation, utilisez la méthode clear_image() pour effacer la zone de dessin Shell si nécessaire.
- L'image Python AppVar est un fichier d'image Python spécial (*.8xv).
  - Actuellement, les images des programmes d'études de la bibliothèque sont publiées à l'adresse https://resources.t3europe.eu/t3europe-home?country=15&q=images&cHash=d50a2b65ab1b875dfa3ec11bca12154c
  - Lors de l'utilisation d'une image Python AppVar, il est recommandé de
    - mémoriser l'image Python AppVar dans la mémoire Archive. [2nde] [mém]
    - connaître les dimensions en pixels de votre image à utiliser lors du codage.

- connaître le nom exact de votre image Python AppVar. Vous devez saisir le nom en respectant l'orthographe et les majuscules et minuscules. Aucune erreur ne sera signalée pour les noms des images Python AppVar mal tapés.

- Veillez à toujours utiliser la dernière mise à jour de TI Connect™ CE et TI-SmartView™ CE à partir du site education.ti.com/83ceupdate

- Les arguments de coordonnées (X,y) sont des coordonnées de pixels UNIQUEMENT dans les méthodes ti_image et vont de (0,0) à (319,209). Veuillez lire plus d'informations sur chaque méthode dans le tableau ci-dessous. Certaines méthodes sont proposées pour coller des entrées dans l'éditeur dans différents formats lorsque des arguments facultatifs sont proposés.
- Appuyez sur [annul] pour effacer le dessin et revenir à l'invite du Shell. L'historique du Shell peut être affiché à l'aide de [2nde] ▲ et [2nde] ▼.

	Menu Control	Description
1 :	load_image("nom")	Charge une image Python AppVar "nom" à utiliser dans le script. L'image Python "nom" doit respecter la casse et l'orthographe de l'image Python AppVar. Notez les points suivants : Aucun message d'erreur n'est généré si le nom AppVar n'est PAS spécifié exactement comme indiqué. L'image Python "nom" sera l'image utilisée pour l'affichage dans show_image (x,y). **Meilleures pratiques :** • Connaître les dimensions en pixels de votre image Python. • Conseil mémoire : Les images Python AppVars doivent être mémorisées dans la mémoire Archive.
2 :	show_image(x,y)	Affiche l'image spécifiée dans load_image ("name"). Affiche l'image avec le coin pixel supérieur gauche (x,y) de la zone de dessin dans le Shell. Les coordonnées (x,y) de l'écran en pixels vont de (0,0) dans le coin supérieur gauche à (319,209) dans le coin inférieur droit. Si aucun nom d'image n'a été spécifié à

	Menu Control	Description
		l'aide de load_image(), une erreur est signalée lors de l'exécution du script. Si le “nom” est entré de manière incorrecte, aucune erreur ne s'affiche. Utilisez la méthode show_screen() pour conserver l'image affichée jusqu'à ce que [annul] revienne au shell. Pour afficher l'historique du Shell, appuyez sur [2nde] ▲ et [2nde] ▼.
3 :	clear_image() (0,0)-(319,209)	La méthode clear_image() sans argument est utilisée pour effacer la zone de dessin du Shell. La zone de dessin s'affiche sous forme d'écran blanc. Les coordonnées en pixels vont de (0,0) dans le coin supérieur gauche à (319,209) dans le coin inférieur droit. Une fois que toute la zone de dessin est "effacée" avec cette méthode, utilisez la méthode load_image("nom") et show_image(x,y) pour afficher l'image "nom" selon les besoins. Lorsque vous utilisez également les méthodes du module ti_draw, notez que la couleur set_pen() passera à noir lors de l’exécution de la méthode ti_image, clear_image().
4 :	clear_image(x,y,w,h) blanc	À partir des coordonnées de pixel (x,y) données pour le coin supérieur gauche d'un rectangle de largeur w pixels et de hauteur h pixels, cette méthode « effacera » la zone intérieure du rectangle pour la mettre en blanc.
5 :	clear_image(x,y,w,h,(r,g,b))	À partir des coordonnées de pixel (x,y) données pour le coin supérieur gauche d'un rectangle de w pixels de large et de h pixels de haut, cette méthode « effacera » la zone intérieure du rectangle pour la mettre à la couleur RGB spécifiée dans le tuple (r,g,b).
6 :	tuple=get_pixel(x,y)	Renvoie les valeurs RGB du pixel de coordonnées (x,y) sous forme de tuple (r,g,b).
7 :	set_pixel(x,y,(r,g,b))	Définit la couleur du pixel de

	Menu Control	Description
		coordonnées (x,y) à la couleur RGB spécifiée par (r,g,b).

Messages de l'application Python

Différents messages sont susceptibles de s'afficher au cours d'une session Python. Le tableau suivant présente une sélection de ces messages. Suivez les instructions affichées à l'écran et naviguez dans l'application à l'aide des commandes [quitter], [Échap] ou [Ok], selon les besoins.

Gestion de la mémoire

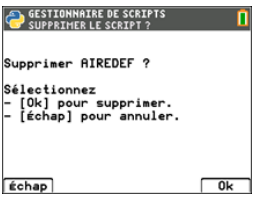
La mémoire disponible pour l'expérience Python correspond à un maximum de 100 scripts Python (AppVars PY) ou 50 K de mémoire. Les modules livrés avec l'application dans cette version de Python utiliseront l'espace commun à tous les fichiers.

Utilisez [2nde] [Quitter] pour quitter l'application

Un message vous invite à confirmer la fermeture de l'application. Si vous quittez l'application, votre session Python est interrompue. Lorsque vous rouvrez l'application Python, vos modules et scripts AppVar Python se synchronisent. Le Shell est réinitialisé.

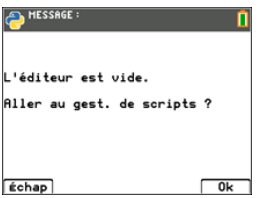
Dans le Gestionnaire de scripts, appuyez sur la touche [suppr] dans un script Python sélectionné ou choisissez Gestionnaire de scripts > Gérer, puis 2:Supprimer le script...

Une boîte de dialogue vous invite alors à confirmer la suppression ou à annuler et à revenir au Gestionnaire de scripts.

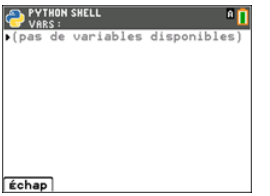


Vous tentez de créer un nouveau script ou de dupliquer un script Python existant déjà sur votre CE soit dans la RAM soit dans la mémoire Archive, ou désactivé pour le mode Examen. Saisissez un autre nom.

Vous tentez de passer du Shell à l'Éditeur, mais ce dernier est vide. Sélectionnez une option appropriée à votre tâche.



Lorsque vous exécutez un script Python, les variables définies à partir du dernier script exécuté sont répertoriées dans le menu Shell > Outils > 4:Vars… afin que vous puissiez les réutiliser dans le Shell. Si aucune variable ne s'affiche, vous devrez peut-être réexécuter le script.



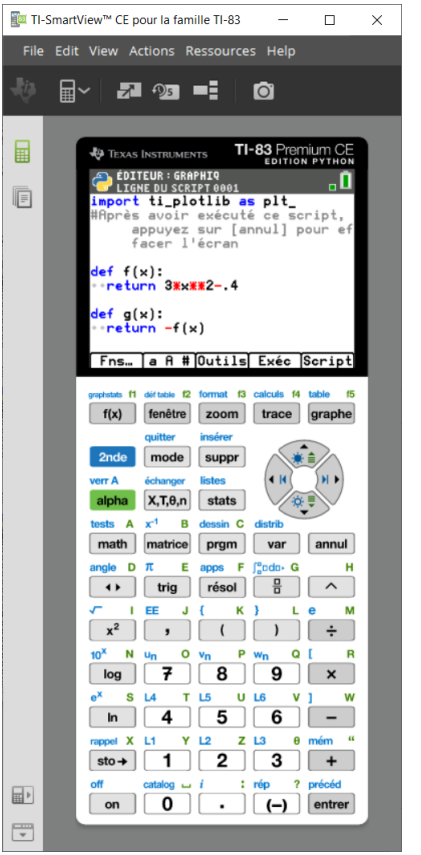
Utilisation de TI-SmartView™ CE et de l'expérience Python

Ce guide d'utilisation suppose que la dernière mise à jour de TI-SmartView™ CE pour la famille TI-83 est à disposition. Installez la dernière mise à jour de TI-SmartView™ CE pour la famille TI-83 à partir du site *education.ti.com/83ceupdate*.

Cette mise à jour comprend la dernière version de l'OS de l'émulateur TI-83 Premium CE Édition Python qui exécute la version la plus récente de l'application Python. Les modules time, ti_system, ti_plotlib, ti_rover* et ti_hub* mis à jour sont inclus.

Exécutez l'application Python sur l'émulateur TI-83 Premium CE *Édition Python*.

- L'application Python propose :
  - Gestionnaire de scripts
  - Éditeur
  - Exécution de votre script Python dans le Shell*



Hub/Rover Programs

- Créez des scripts ti_hub/ti_rover en Python dans l'émulateur CE qui exécute l'application Python.

  *** Remarque :** Aucune connexion ne peut être établie entre TI-SmartView™ CE et TI-Innovator™ Hub ou TI-Innovator™ Rover. Vous pouvez créer des scripts, puis les exécuter sur la calculatrice CE.

- Quittez l'application Python pour vous préparer à transférer les AppVars Python à partir de l'émulateur. L'émulateur ne doit pas être « occupé » à exécuter une application ou un script lors de la prochaine étape.
- Basculez dans l'espace de travail Emulator Explorer (Explorateur de l'émulateur) et envoyez le(s) script(s) à l'ordinateur.

– Utilisez TI Connect™ CE pour envoyer les AppVars Python de l'ordinateur à la calculatrice CE afin de les exploiter avec TI-Innovator™ Hub/TI-Innovator™ Rover.

Remarque : Pour interrompre un script Python en cours d'exécution dans le Shell, par exemple lorsqu'un script se trouve dans une boucle infinie, appuyez sur [on]. Appuyez sur [Outils] [zoom] > 6:Nouveau Shell comme méthode alternative pour arrêter un script en cours d'exécution.

Rappel : Pour tout ordinateur/toute expérience TI-Python : Après avoir créé un programme Python dans un environnement de développement Python sur l'ordinateur, veuillez valider votre programme s'exécute sur la calculatrice/l'émulateur dans l'expérience TI-Python. Modifiez le programme si nécessaire.

Clavier à distance via l'application SmartPad CE

– Lorsque vous utilisez l'application SmartPad CE sur votre calculatrice CE connectée, elle se comporte comme un clavier à distance, notamment pour le mappage spécial du *clavier* lorsque l'application Python est en cours d'exécution.

Espace de travail Emulator Explorer (Explorateur de l'émulateur)

– Quittez l'application Python pour éviter que l'émulateur ne soit occupé lorsque vous accédez à toutes les fonctions de l'espace de travail Emulator Explorer (Explorateur de l'émulateur).

– Les conversions script.py < > AppVar PY sont autorisées. Ceci est similaire au comportement de TI Connect™ CE lors de l'envoi de scripts à la calculatrice CE connectée.

– Lorsque vous envoyez un script.py créé dans un autre environnement Python, vous devez modifier votre AppVar PY pour qu'elle s'exécute comme prévu en langage TI-Python. Utilisez l'Éditeur d'application Python pour modifier le script selon les besoins des modules propres, tels que ti_plotlib, ti_system, ti_hub et ti_rover.

Assistant d'importation de données

– Les fichiers de données *.csv, formatés comme indiqué dans la boîte de dialogue de l'assistant, seront convertis en variables de liste CE. Les méthodes incluses dans le module ti_system peuvent ensuite être utilisées pour partager les listes entre l'OS CE de l'émulateur et l'application Python. Cette fonction est similaire à l'assistant d'importation de données disponible dans la TI Connect™ CE.

  Remarque : si des nombres décimaux sont représentés par des virgules dans le fichier *.csv, le fichier ne sera pas converti à l'aide de l'assistant d'importation de données. Vérifiez le formatage du numéro du système d'exploitation de votre ordinateur et convertissez le fichier *.csv pour utiliser la représentation décimale. La liste des calculateurs ce et l'éditeur de matrice utilisent le format numérique comme, par exemple, 12.34 et non 12.34.

Conversion de scripts Python à l'aide de TI Connect™ CE

Mettez à jour vers TI Connect™ CE pour bénéficier des dernières fonctionnalités disponibles, telles que la conversion de scripts *.py en AppVar PY comme format de fichier de calculatrice CE.

Pour de plus amples informations sur la calculatrice CE, TI-SmartView CE et TI Connect CE, consultez le guide électronique (eGuide) TI-83 Premium CE.

Présentation de l'expérience de programmation Python

TI-Python est basé sur CircuitPython, une variante de Python conçue pour les petits microcontrôleurs. L'implémentation CircuitPython d'origine a été spécialement adaptée par TI.

Le stockage interne des nombres pour les calculs à effectuer dans cette variante du langage Circuit Python est réalisé en virgule flottante d'une précision limitée et ne peut donc pas représenter avec exactitude toutes les valeurs décimales possibles. Les différences par rapport aux représentations décimales réelles qui surviennent lors de l'enregistrement de ces valeurs peut produire des résultats inattendus dans les calculs ultérieurs.

- **Pour les nombres à virgule flottante** : affiche jusqu'à 16 chiffres significatifs de précision. En interne, les valeurs sont enregistrées à l'aide de 53 bits de précision, ce qui équivaut approximativement à 15-16 décimales.
- **Pour les nombres entiers** : la taille des nombres entiers est uniquement limitée par la mémoire disponible au moment de l'exécution des calculs.

Modules inclus dans la TI-83 Premium CE Édition Python

- Built-ins
- module math
- module random
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Remarque : Si vous possédez des scripts Python créés dans d'autres environnements de développement Python, modifiez-les pour la solution TI-Python. Les modules peuvent employer des méthodes, des arguments et un ordre des méthodes dans un script qui sont différents de ceux utilisés dans les modules ti_system, ti_plotlib, ti_hub et ti_rover. De manière générale, soyez toujours attentif aux questions de compatibilité lorsque vous utilisez une quelconque version du langage et des modules Python.

Lors du transfert de scripts Python d'une plateforme non-TI vers une plateforme TI OU d'un produit TI vers une solution tierce :

- Les scripts Python qui utilisent des fonctions de base du langage et des bibliothèques standard (math, random etc.) peuvent migrer sans modifications.

  Remarque : La longueur des listes est limitée à 100 éléments.

- Les scripts qui utilisent des bibliothèques propres à une plateforme – matplotlib (pour ordinateur), ti_plotlib, ti_system, ti_hub, etc. pour les plateformes TI – devront être modifiés avant de pouvoir être exécutés sur une plateforme différente.

- Cela peut même s'appliquer à des scripts devant être transférés entre plateformes TI.

Comme dans n'importe quelle version de Python, vous devrez inclure des commandes d'importation telles que « from math import * » pour utiliser les fonctions, les méthodes ou les constantes présentes dans le module math. À titre d'exemple, pour exécuter la fonction cos(), spécifiez import afin d'importer le module math pour l'utiliser.

Voir Liste du CATALOGUE.

Exemple :

```
>>>from math import *
>>>cos(0)
1.0
```

Autre exemple :

```
>>>import math
>>>math.cos(0)
1.0
```

Pour afficher dans le Shell les modules disponibles, utilisez la commande suivante :

```
>>> help("modules")
__main__ sys gc
random time array
math builtins collections
```

Vous pouvez afficher le contenu des modules dans le Shell comme illustré en utilisant « import module » et « dir(module) ».

Le contenu complet du module n'apparaît pas dans les menus de collage rapide tels que [Fns…] ou 2nde [catalog].

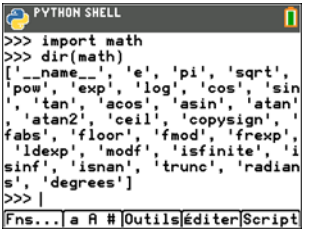
Contenu d'une sélection de modules et mots-clés

Pour obtenir la liste des modules inclus dans cette version, consultez la section :

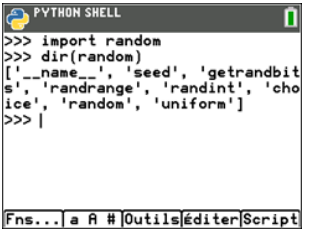
Annexe : Sélection de dispositifs intégrés, de mots-clés et de contenus de modules TI-Python

Rappel : pour n'importe quel ordinateur/TI-Python expérience : après la création d'un programme Python sur l'ordinateur, veuillez valider votre programme s'exécute sur la calculatrice dans le TI-Python expérience. Modifier le programme au besoin.

Ces écrans affichent le contenu des modules math et random.

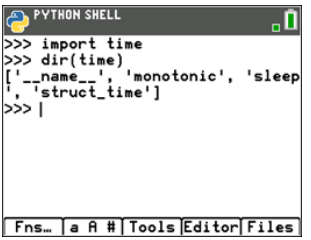


module math

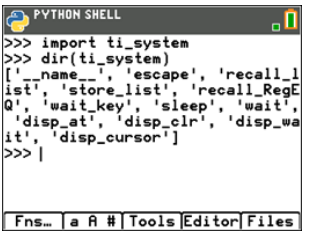


module random

Ces écrans affichent le contenu des modules time and ti_system.



time

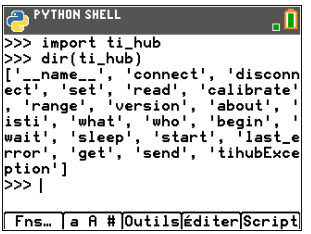


ti_system

Ces écrans affichent le contenu du module ti_plotlib.

ti_plotlib

Ces écrans affichent le contenu du module ti_hub.



ti_hub

Ces écrans affichent le contenu du module ti_rover.

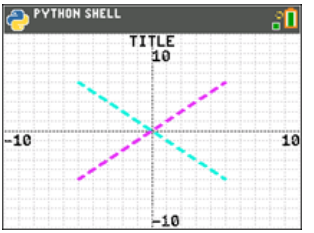
ti_rover

Exemples de scripts

Utilisez les exemples de scripts suivants pour vous familiariser avec les méthodes décrites à la section Référence. Par ailleurs, ces exemples comprennent plusieurs scripts TI-Innovator™ Hub et TI-Innovator Rover™ qui faciliteront votre prise en main du langage TI-Python.

COLORLIN

```
import ti_plotlib as plt
plt.cls()
plt.window(-10,10,-10,10)
plt.axes("on")
plt.grid(1,1,"dot")
plt.title("TITLE")
plt.pen("medium","solid")
plt.color(28,242,221)
plt.pen("medium","dash")
plt.line(-5,5,5,-5,"")
plt.color(224,54,243)
plt.line(-5,-5,5,5,"")
plt.show_plot()
```



Appuyez sur [annul] pour afficher l'invite du Shell.

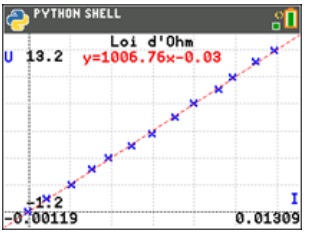
REGEQ1

Tout d'abord, saisissez deux listes dans le système d'exploitation (OS) CE. Puis, par exemple, calculez [stat] CALC 4:LinReg(ax+b) pour vos listes. Cela permet de stocker l'équation de régression dans RegEQ dans l'OS. Voici un script destiné à rappeler RegEQ dans l'expérience Python.

```
# Exemple d'utilisation de recall_RegEQ()
from ti_system import *

reg=recall_RegEQ()
print(reg)
x=float(input("Input x = "))
print("RegEQ(x) = ",eval(reg))
```

LINREG (inclus dans le bundle CE)

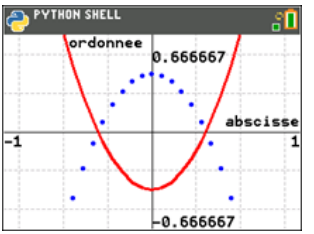
```
import ti_plotlib as plt

# intensité du courant
I = [0.0, 0.9, 2.1, 3.1, 3.9, 5.0, 6.0, 7.1, 8.0, 9.2, 9.9, 11.0,11.9]

# Convertir des milliampères en ampères
for n in range (len(I)):
I[n] /= 1000

# tension
U = [0, 1, 2, 3.2, 4, 4.9, 5.8, 7, 8.1, 9.1, 10, 11.2, 12]

plt.cls()
plt.auto_window(I,U)
plt.pen("thin","solid")
plt.axes("on")
plt.grid(.002,2,"dot")
plt.title("Loi d'Ohm")
plt.color (0,0,255)
plt.labels("I","U",11,2)
plt.scatter(I,U,"x")
plt.color (255,0,0)
plt.pen("thin","dash")
plt.lin_reg(I,U,"center",2)
plt.show_plot()
plt.cls()
a=plt.a
b=plt.b
print ("a =",round(plt.a,2))
print ("b =",round(plt.b,2))
```



Appuyez sur annul pour afficher l'invite du Shell.

GRAPHIQ (inclus dans le bundle CE)

```
import ti_plotlib as plt
#Après avoir exécuté ce script, appuyez sur [annul] pour effacer
l'écran

def f(x):
  return 3*x**2-.4

def g(x):
  return -f(x)

def plot(res,xmin,xmax):
  #configurer l'écran graphique
  plt.window(xmin,xmax,xmin/1.5,xmax/1.5)
  plt.cls()
  gscale=5
  plt.grid((plt.xmax-plt.xmin)/gscale*(3/4),(plt.ymax-
plt.ymin)/gscale,"dash")
  plt.pen("thin","solid")
  plt.color(0,0,0)
  plt.axes("on")
  plt.labels("abscisse"," ordonnee",6,1)
  plt.pen("medium","solid")

# représenter les fonctions f et g
dX=(plt.xmax -plt.xmin)/res
x=plt.xmin
x0=x
  for i in range(res):
    plt.color(255,0,0)
    plt.line(x0,f(x0),x,f(x),"")
    plt.color(0,0,255)
    plt.plot(x,g(x),"o")
    x0=x
    x+=dX
  plt.show_plot()

#plot(résolution,xmin,xmax)
plot(30,-1,1)
# Créer un graphique avec les paramètres (résolution,xmin,xmax)

# Après avoir effacé le premier graphique, appuyez sur la touche [var].
La fonction plot() permet de modifier les paramètres d'affichage
(résolution,xmin,xmax).
```

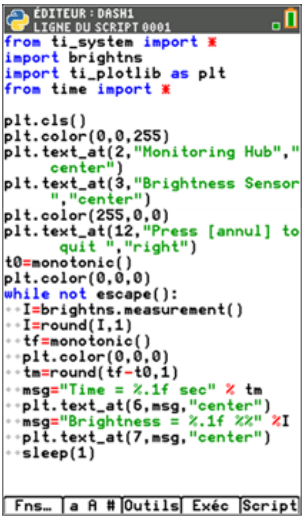


Appuyez sur [annul] pour afficher l'invite du Shell.

DASH1 – Exemple de script TI-Innovator™ Hub

Voir : [Fns…]>Modul: module ti_hub

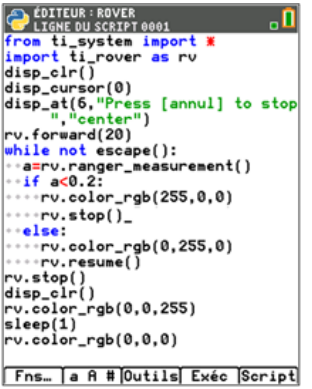
```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *

plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [annul] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
while not escape():
  I=brightns.measurement()
  I=round(I,1)
  tf=monotonic()
  plt.color(0,0,0)
  tm=round(tf-t0,1)
  msg="Time = %.1f sec" % tm
  plt.text_at(6,msg,"center")
  msg="Brightness = %.1f %%" %I
  plt.text_at(7,msg,"center")
  sleep(1)
```



ROVER – Exemple de script TI-Innovator™ Rover

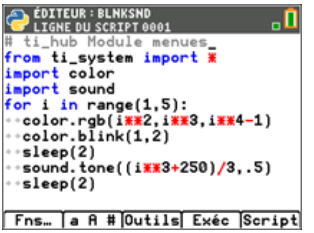
Voir : [Fns…]>Modul module ti_rover

```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [annul] to stop","center")
rv.forward(20)
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)
```



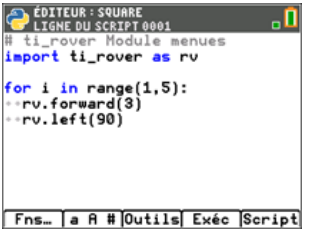
BLNKSND – Exemple de script TI-Innovator™ Hub

Voir : [Fns…]>Modul: module ti_hub



CARRÉ – Exemple de script TI-Innovator™ Rover

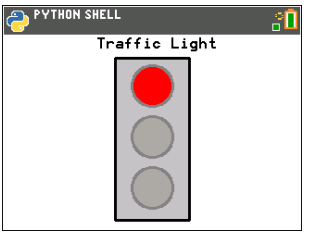
Voir : [Fns…]>Modul module ti_rover



STOP_GO - Sample ti_draw, ti_image, time Program

See:[Fns…]>Modul [Add-On]

```
from ti_draw import *
from ti_image import *
from time import *
clear()
# Pixel screen upper left (0,0) to (319,209)
draw_text(100,20,"Traffic Light")
set_pen("medium","solid")

draw_rect(120,25,80,175)
set_color(192,192,192)
fill_rect(120,25,80,175)
set_color(128,128,128)
draw_circle(160,55,22)
draw_circle(160,110,22)
draw_circle(160,165,22)

def off(x,y):
  set_color(169,169,169)
  fill_circle(x,y,22)
  set_color(128,128,128)
  draw_circle(x,y,22)

for i in (1,20,1):
# Green
  set_color(51,165,50)
  fill_circle(160,165,22)
  sleep(3)
  off(160,165)
# Yellow
  set_color(247,239,10)
  fill_circle(160,110,22)
  sleep(2)
  off(160,110)
# Red
  set_color(255,0,0)
  fill_circle(160,55,22)
  sleep(3)
  off(160,55)
  show_draw()
```



Guide de référence pour l'expérience TI-Python

L'application Python Adapter contient des menus de fonctions, de classes, de commandes, d'opérateurs et de mots-clés destinés à faciliter le collage d'entrées dans l'Éditeur ou le Shell. Le tableau de référence suivant contient la liste des fonctionnalités accessibles via 2nde [catalog] lorsque l'application est en cours d'exécution. Pour obtenir la liste complète des fonctions, classes, opérateurs et mots-clés Python disponibles dans cette version, consultez la section « Contenu d'une sélection de modules et mots-clés ».

Ce tableau n'est pas destiné à fournir une liste exhaustive des fonctions Python disponibles dans cette offre. D'autres fonctions prises en charge dans cette offre Python sont accessibles à partir des touches alphabétiques du clavier.

La plupart des exemples présentés dans ce tableau s'exécutent sur l'invite du Shell (>>>).

Liste du CATALOGUE

Liste alphabétique

- A
- B
- C
- D
- E
- F
- G
- H
- I
- L
- M
- N
- O
- P
- R
- S
- T
- U
- W
- X
- Y

- Symbols

A

#

Séparateur 2nde [catalog]

Syntaxe : #Votre commentaire concernant le script.

Description : En langage Python, un commentaire débute par le caractère hashtag (#) et s'étend jusqu'à la fin de la ligne. [a A #]

Exemple :

```
#Une courte explication du code.
```

%

Opérateur 2nde [catalog]

Syntaxe : x%y ou x % y

Description : Renvoie le reste de la division euclidienne de x par y. (modulo) Utilisation conseillée lorsque x et y sont des nombres entiers. [a A #]

Exemple :

```
>>>57%2
1
```

Voir aussi fmod(x,y).

//

Opérateur 2nde [catalog]

Syntaxe : x//y ou x // y

Description : Renvoie le quotient de la division euclidienne de x par y. [a A #]

Exemple :

```
>>>26//7
3
>>>65.4//3
21.0
```

[a A #]

Description : Lancez le jeu de caractères [a A #].

Comprend ç à â è é ê ë î ï ô ö ù û

[a A #]
le raccourci apparaît à l'écran via [fenétre] dans l'Éditeur ou dans le Shell

a pente

Module : ti_plotlib

Syntaxe : plt.a pente

Description : Après l'exécution de la commande plt.linreg() qui intervient en dernier dans un script, les valeurs calculées pour la pente, a, et l'ordonnée à l'origine, b, sont stockées dans plt.a et dans plt.b.

Valeurs par défaut : = 0.0

Exemple :

Voir l'exemple de script : LINREG.

[2nde] [catalog]

[Fns...]>Modul ou [math]
5:ti_plotlib...>
Propriétés
5:a

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu Configurer de ti_plotlib.

abs()

Module : Built-in

Syntaxe : abs(x)

Description : Renvoie la valeur absolue d'un nombre. Dans cette version, l'argument peut être un nombre entier ou un nombre à virgule flottante.

Exemple :

```
>>>abs(-35.4)
35.4
```

[2nde] [catalog]

Remarque : fabs() est une fonction du module math.

acos()

Module : math

Syntaxe : acos(x)

[trig] 7:acos()

acos()

Description : Renvoie l'arc cosinus de x en radians. 2nde [catalog]

Exemple :

```
>>>from math import *
>>>acos(1)
0.0
```

[Fns…] Modul 1:math… > Trig 7:acos()

Autre exemple : [Outils] > 6:Nouveau Shell

```
>>>import math
>>>math.acos(1)
0.0
```

les commandes import sont disponibles via 2nde [catalog]

and

Mot-clé

[?] [tests]
Ops 8:and

Syntaxe : x and y

Description : Peut retourner Vrai ou faux. Renvoie « x » si « x » est égal à False et « y » dans le cas contraire. Un espace est collé avant et après and. Modifiez selon vos besoins.

[Fns…] > Ops 8:and

Exemple :

2nde [catalog]

```
>>>2<5 and 5<10
True
>>>2<5 and 15<10
False
>>>{1} and 3
3
>>>0 and 5 < 10
0
```

[a A #]

.append(x)

Module : Built-in

2nde [listes]
List
6: .append(x)

Syntaxe : listname.append(item)

Description : La méthode append() ajoute un élément à la liste.

2nde [catalog]

Exemple :

```
>>>listA = [2,4,6,8]
>>>listA.append(10)
>>>print(listA)
[2,4,6,8,10]
```

[Fns…] > List
6:.append(x)

as

Mot-clé

2nde [catalog]

Description : Utilisez as pour créer un alias lorsque vous importez un module. Pour plus de détails, consultez la documentation de Python.

asin()

Module : math

trig 6:asin()

Syntaxe : asin()

Description : Renvoie l'arc sinus de x en radians.

2nde [catalog]

Exemple :

```
>>>from math import *
>>>asin(1)
1.570796326794897
```

[Fns…] > Modul 1:math... > Trig 6:asin()

Autre exemple :

```
>>>import math
>>>math.asin(1)
1.570796326794897
```

les commandes import sont disponibles via 2nde [catalog]

assert

Mot-clé

2nde [catalog]

Description : Utilisez assert pour tester une condition dans votre code. Renvoie None (Aucun), sinon, l'exécution du script génère une erreur « AssertionError ».

atan()

Module : math

trig 8:atan()

Syntaxe : atan(x)

Description : Renvoie l'arc tangente de x en radians.

[Fns…] > Modul 1:math... > Trig 8 :atan()

Exemple :

```
>>>from math import *
>>>atan(1)*4
3.141592653589793
```

2nde [catalog]

Autre exemple :

```
>>>import math
>>>math.atan(1)*4
3.141592653589793
```

les commandes import sont disponibles via 2nde [catalog]

atan2(y,x)

Module : math

Syntaxe : atan2(y,x)

Description : Renvoie l'arc tangente de y/x en radians. Le résultat est dans [-pi, pi].

Exemple :

```
>>>from math import *
>>>atan2(pi,2)
1.003884821853887
```

Autre exemple :

```
>>>import math
>>>math.atan2(math.pi,2)
1.003884821853887
```

[trig] 9:atan2 ()

[Fns…] > Modul 1:math... > Trig 9:atan2()

[2nde] [catalog]

les commandes import sont disponibles via [2nde] [catalog]

auto_window(xliste,yliste)

Module : ti_plotlib

Syntaxe : plt.auto_window(xliste,yliste)

Description : Met automatiquement à l'échelle la fenêtre de tracé pour faire tenir les plages de données spécifiées dans le script par les listes xliste et yliste avant l'utilisation de auto_window().

Remarque : max(list) - min(list) > 0.00001

Exemple :

Voir l'exemple de script : LINREG.

[2nde] [catalog]

[Fns...]>Modul ou [math] 5:ti_plotlib...> Configurer 5:auto_window()

Les commandes d'importation sont disponibles via [2nde][catalog] ou dans le menu Configurer de ti_ plotlib.

axes("mode")

Module : ti_plotlib

Syntaxe : plt.axes("mode")

Description : Affiche les axes sur la fenêtre spécifiée dans la zone de tracé.

Argument :

Options de l'argument "mode" :

"off"	pas d'axes
"on"	axes+étiquettes
"axes"	axes seuls
"window"	étiquettes de fenêtre uniquement

plt.axes() utilise le paramètre de couleur de stylo actif. Pour garantir le traçage correct des axes plt.axes(), utilisez plt.color() AVANT plt.axes() afin de vous assurer que les couleurs s'affichent comme prévu.

Exemple :

Voir l'exemple de script LINREG.

2nde [catalog]

[Fns...]>Modul ou
math
5:ti_plotlib...>
Configurer
6:axes()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

b ordonnée à l'origine

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.b ordonnée à l'origine

Description : Après l'exécution de plt.linreg() dans un script, les valeurs calculées pour la pente, a, et l'ordonnée à l'origine, b, sont stockées dans plt.a et dans plt.b.

[Fns...]>Modul ou math 5:ti_plotlib...> Propriétés 6:b

Valeurs par défaut : = 0.0

Exemple :

Voir l'exemple de script LINREG.

Les commandes d'importation sont disponibles via 2nde[catalog] ou dans le menu Configurer de ti_plotlib.

bin(entier)

Module : Built-in

2nde [catalog]

Syntaxe : bin(entier)

Description : Affiche l'argument entier au format binaire.

Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> bin(2)
'0b10'
>>> bin(4)
'0b100'
```

break

Mot-clé

2nde [catalog]

Description : Utilisez break pour sortir d'une boucle for ou while.

.

C

ceil()

Module : math

Syntaxe : ceil(x)

Description : Renvoie le plus petit entier supérieur ou égal à x.

Exemple :

```
>>>from math import *
>>>ceil(34.46)
35
>>>ceil(678)
678
```

[math] Modul
1:math... Math
8:ceil()

[2nde] [catalog]

[Fns...] Modul
1:math...Math
8:ceil()

les commandes import sont disponibles via [2nde] [catalog]

choice(séquence)

Module : random

Syntaxe : choice(séquence)

Description : Renvoie un élément aléatoire provenant d'une liste non vide.

Exemple :

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA) #Votre résultat peut être différent.
4
```

[math] Modul
2:random...
Random
5:choice(séquence)

[2nde] [catalog]

[Fns...] Modul
2:random...
Random
5:choice(séquence)

les commandes import sont disponibles via [2nde] [catalog]

chr(entier)

Module : Built-in

2nde [catalog]

Syntaxe : chr(entier)

Description : Renvoie une chaîne de caractères à partir d'un nombre entier représentant un caractère unicode.

Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> char(40)
'('
>>> char(35)
'#'
```

class

Mot-clé

2nde [catalog]

Description : Utilisez class pour créer une classe. Pour plus de détails, consultez la documentation de Python.

cls() effacer écran

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.cls() effacer écran

[Fns...]>Modul ou math
5:ti_plotlib...>
Configurer
2:cls()

Description : Efface l'écran du Shell pour le tracé. Les touches de raccourci ne sont pas affichées lors du tracé.

Remarque :plt.cls() se comporte différemment de la commande disp_clr() du module ti_system.

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
2:cls()

Exemple :

Voir l'exemple de script : GRAPHIQ.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

color(r,v,b) 0-255

Module : ti_plotlib

Syntaxe : plt.color(r,v,b) 0-255

Description : Définit la couleur de tous les graphiques/tracés qui suivent. Les valeurs (r,v,b) doivent être spécifiées dans la plage 0-255. La couleur spécifiée est utilisée dans l'affichage du tracé jusqu'à ce que la commande color() soit à nouveau exécutée en précisant une couleur différente.

La couleur par défaut est le noir lors de l'importation du module ti_plotlib.

Exemple :

Voir l'exemple de script : COLORLIN.

2nde [catalog]

[Fns...]>Modul ou
math
5:ti_plotlib...>
Dessin
1:color()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

complex(real,imag)

Module : Built-in

Syntaxe : complex(real, imag)

Description : Type nombre complexe.

Exemple :

```
>>>z = complex(2, -3)
>>>print(z)
(2-3j)
>>>z = complex(1)
>>>print(z)
(1+0j)
>>>z = complex()
>>>print(z)
0j
>>>z = complex("5-9j")
>>>print(z)
(5-9j)
```

2nde [catalog]

[Fns...]>Type>
5:complex()

Remarque :"1+2j" est la syntaxe correcte. Les espaces tels que "1 + 2j" génèrent une exception.

continue

Mot-clé

2nde [catalog]

Description : Utilisez continue dans une boucle for ou while pour mettre fin à l'itération actuelle. Pour plus de détails, consultez la documentation de Python.

cos()

Module : math

trig Trig
4: cos()

Syntaxe : cos(x)

Description : Renvoie le cosinus de x. L'argument Angle est exprimé en radians.

2nde [catalog]

Exemple :

```
>>>from math import *
>>>cos(0)
1.0
>>>cos(pi/2)
6.123233995736767e-17
```

[Fns...] Modul
1:math... > Trig
4:cos()

Autre exemple :

```
>>>import math
>>>math.cos(0)
1.0
```

Remarque : Python affiche en notation scientifique à l'aide de e ou E. Certains résultats du module math en langage Python seront différents de ceux du système d'exploitation CE.

.count()

Module : Built-in

2nde [catalog]

Syntaxe : listname.count(item)

Description : count()est une méthode qui renvoie le nombre d'occurrences d'un élément dans un objet list, tuple, bytes, str, bytearray ou array.array.

Exemple :

```
>>>listA = [2,4,2,6,2,8,2,10]
>>>listA.count(2)
4
```

def fonction ():

Mot-clé

2nde [catalog]

Syntaxe : def fonction(var, var,...)

Description : Définit une fonction dépendant de variables spécifiées. Elle est généralement utilisée avec le mot-clé return.

[Fns…] > Fonc 1:def fonction ():

Exemple :

[Fns…] > Fonc 2:return

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

degrees()

Module : math

Syntaxe : degrees(x)

Description : Convertit l'angle x défini en radians en degrés.

Exemple :

```
>>>from math import *
>>>degrees(pi)
180.0
>>>degrees(pi/2)
90.0
```

[trig] Trig
2:degrees()

[2nde] [catalog]

[Fns…] > Modul
1:math… > Trig
2:degrees()

del

Mot-clé

Description : Utilisez del pour supprimer des objets tels que des variables, listes, etc.
Pour plus de détails, consultez la documentation de Python.

[2nde] [catalog]

disp_at(ligne,col,"txt")

Module : ti_system

Syntaxe : disp_at(ligne,col,"txt")

Description : Affiche un texte débutant à la position définie par une ligne et une colonne sur la zone de tracé.

REPL avec le curseur >>>| s'affiche après le texte si la fin du script a été atteinte. Utilisez disp_cursor() pour contrôler l'affichage du curseur.

Argument :

ligne	entier, 1 - 11

[2nde] [catalog]

[2nde] [rappel]
ti_system
7:disp_at()

[Fns…]>Modul ou [math]
4:ti_system
7:disp_at()

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu

disp_at(ligne,col,"txt")

colonne	1 - 32, entier
"txt"	est une chaîne de caractères qui s'affiche à l'écran avec retour à la ligne automatique

Modul de ti_system.

Arguments facultatifs pour définir la couleur et l'arrière-plan illustrés ici : disp_at (ligne,col,"txt","align",color 0-15, background color 0-5)

Exemple :

Exemple de script :

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,6,"hello")
disp_cursor(0)
disp_wait()
```

disp_at(ligne,"txt","align")

Module : ti_system

2nde [catalog]

Syntaxe : disp_at(ligne,"txt","align")

2nde [rappel] ti_system 7:disp_at()

Description : Affiche le texte aligné comme spécifié sur l'écran du tracé sur une ligne comprise dans 1-11. Les données de la ligne sont effacées avant l'affichage du texte. En cas d'utilisation dans une boucle, le contenu est actualisé à chaque nouvel affichage de données.

[Fns...]>Modul ou math 4:ti_system 7:disp_at()

REPL avec le curseur >>>| s'affiche après le texte si la fin du script a été atteinte. Utilisez disp_cursor() pour contrôler l'affichage du curseur avant l'utilisation de disp_at() dans votre script.

Argument :

ligne	entier, 1 - 11
"txt"	est une chaîne de caractères qui s'affiche à l'écran avec retour à la

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

disp_at(ligne,"txt","align")

	ligne automatique
"align"	"gauche" (par défaut)
	"centre"
	"droite"

Argument facultatif illustré ici : disp_at (ligne,col,"txt","align",color 0-15, background color 0-15)

Exemple :

Exemple de script :

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

disp_clr() **effacer texte**

Module : ti_system

Syntaxe : disp_clr() effacer texte

Description : Efface l'écran dans l'environnement Shell. Ligne 0-11, un entier peut être utilisé comme argument facultatif pour effacer une ligne d'affichage de l'environnement Shell.

Exemple :

Exemple de script :

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

2nde [catalog]

2nde [rappel]
ti_system
8:disp_clr()

[Fns…]>Modul ou
math
4:ti_system
8:disp_clr()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

disp_cursor() **0=off 1=on**

Module : ti_system

Syntaxe : disp_cursor() 0=off 1=on

2nde [catalog]

2nde [rappel]

disp_cursor() **0=off 1=on**

Description : Contrôle l'affichage du curseur dans le Shell lorsqu'un script est en cours d'exécution.

Argument :

0 = off

différent de 0 = on

Exemple :

Exemple de script :

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

ti_system
0:disp_cursor()

[Fns...]>Modul ou
[math]
4:ti_system
0:disp_cursor()

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu Modul de ti_system.

disp_wait() **[annul]**

Module : ti_system

Syntaxe : disp_wait() [annul]

Description : Arrête l'exécution du script à ce niveau et affiche le contenu de l'écran jusqu'à ce que l'utilisateur appuie sur la touche [annul], les données de l'écran sont alors effacées.

Exemple :

Exemple de script :

```
from ti_system import *
disp_clr() #clears Shell screen
disp_at(5,"hello","left")
disp_cursor(0)
disp_wait()
```

[2nde] [catalog]

[2nde] [rappel]
ti_system
9:disp_wait()

[Fns...]>Modul ou
[math]
4:ti_system
9:disp_wait()

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu Modul de ti_system.

e

Module : math

Syntaxe : math.e ou e si le module math a été importé

Description : La constante e s'affiche comme illustré ci-dessous.

Exemple :

```
>>>from math import *
>>>e
2.718281828459045
```

Autre exemple :

```
>>>import math
>>>math.e
2.718281828459045
```

2nde [e] (au-dessus de ÷)

[Fns…] > Modul 1:math… > Const 1:e

elif :

Mot-clé

Voir if..elif..else.. pour plus de détails.

2nde [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Mot-clé

2nde [catalog]

Voir if..elif..else.. pour plus de détails.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Module : ti_system

Syntaxe : escape()

Description : escape() renvoie True (Vrai) ou False (Faux).

La valeur initiale est False (Faux).

Lorsque vous appuyez sur la touche [annul] de la calculatrice CE, la valeur est définie sur True (Vrai).

Lorsque la fonction est exécutée, la valeur est réinitialisée sur False (Faux).

Exemple d'utilisation :

while not escape():

Dans une boucle while exécutée dans un script qui propose de terminer la boucle en laissant l'exécution du script se poursuivre.

if escape():break

Peut s'utiliser dans un script de débogage pour inspecter les variables en utilisant Shell [var] après avoir exécuté le script et utilisé ce « break ».

2nde [catalog]

En tant que ligne de script :

2nde [rappel]
ti_system
5:while not escape():
6:if escape():break

[Fns…]>Modul ou math
4:ti-system
5:while not escape():
6:if escape():break

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

eval()

Module : Built-in

2nde [catalog]

Syntaxe : eval(x)

Description : Renvoie l'évaluation de l'expression x.

[Fns…] E/S 3:eval()

Exemple :

```
>>>a=7
>>>eval("a+9")
16
>>>eval('a+10')
17
```

except exception:

Mot-clé

2nde [catalog]

Description : Utilisez except dans un bloc de code try..except. Pour plus de détails, consultez la documentation de Python.

exp()

Module : math

Syntaxe : exp(x)

Description : Renvoie e**x.

Exemple :

```
>>>from math import *
>>>exp(1)
2.718281828459046
```

Autre exemple : [Outils] > 6:Nouveau Shell

```
>>>import math
 >>>math.exp(1)
2.718281828459046
```

2nde [e^x] (au-dessus de ln)

2nde [catalog]

[Fns…] > Modul
1:math…
4:exp()

les commandes import sont disponibles via 2nde [catalog].

.extend()

Module : Built-in

Syntaxe : listname.extend(newlist)

Description : La méthode extend() permet d'ajouter newlist à la fin de la liste.

Exemple :

```
>>>listA = [2,4,6,8]
>>>listA.extend([10,12])
>>>print(listA)
[2,4,6,8,10,12]
```

2nde [catalog]

fabs()

Module : math

2nde [catalog]

Syntaxe : fabs(x)

Description : Renvoie la valeur absolue de x.

[Fns…] > Modul
1:math...
2:fabs()

Exemple :

```
>>>from math import *
>>>fabs(35-65.8)
30.8
```

les commandes import sont disponibles via 2nde [catalog].

Voir aussi la fonction Built-in abs().

False

Mot-clé

[?] [tests] (au-dessus de math)

Description : Renvoie False lorsque l'instruction exécutée est Fausse. « False » représente la valeur fausse d'objets de type booléen.

Exemple :

2nde [catalog]

```
>>>64<=32
False
```

[Fns…] > Ops
B:False

[a A #]

finally

Mot-clé 2nde [catalog]

Description : Utilisez finally dans un bloc de code try..except..finally. Pour plus de détails, consultez la documentation de Python.

float()

Module : Built-in 2nde [catalog]

Syntaxe : float(x)

Description : Renvoie x sous forme de nombre flottant. [Fns…] > Type 2:float()

Exemple :

```
>>>float(35)
35.0
>>>float("1234")
1234.0
```

floor()

Module : math math Modul 1:math 9:floor()

Syntaxe : floor(x)

Description : Renvoie le plus grand entier inférieur ou égal à x (partie entière de x).

2nde [catalog]

Exemple :

```
>>>from math import *
>>>floor(36.87)
36
>>>floor(-36.87)
-37
>>>floor(254)
254
```

[Fns…] > Modul 1:math 9:floor()

les commandes import sont disponibles via 2nde [catalog]

fmod(x,y)

Module : math

Syntaxe : fmod(x,y)

Description : Peut retourner Vrai ou faux. Utilisation conseillée lorsque x et y sont des nombres flottants.

Peut ne pas renvoyer le même résultat que x%y.

Exemple :

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0 #validation à partir de la description
-2.0
```

Voir aussi : x%y.

[math] Modul
1:math
7:fmod()

[2nde] [catalog]

[Fns…] >
Modul
1:math…
7:fmod()

les commandes import sont disponibles via [2nde] [catalog]

for i in liste:

Mot-clé

Syntaxe : for i in liste:

Description : Permet d'itérer sur les éléments d'une liste.

Exemple :

```
>>> for i in [2,4,6]:
… print(i)
…
…
…
2
4
6
```

[Fns…] Ctl
7:for i in liste:

[2nde] [catalog]

for i in range(taille):

Mot-clé

[Fns…] Ctl 4:for i in range (taille):

Syntaxe : for i in range(taille)

Description : Permet d'itérer sur une plage.

Exemple :

2nde [catalog]

```
>>> for i in range(3):
… print(i)
…
…
…
0
1
2
```

for i in range(début,fin):

Mot-clé

[Fns…] Ctl 5:for i in range (début,fin):

Syntaxe : for i in range(début,fin)

Description : Permet d'itérer sur une plage.

Exemple :

2nde [catalog]

```
>>> for i in range(1,4):
… print(i)
…
…
…
1
2
3
```

for i in range(début,fin,pas):

Mot-clé

Syntaxe : for i in range(début,fin,pas)

Description : Permet d'itérer sur une plage.

Exemple :

```
>>> for i in range(1,8,2):
… print(i)
…
…
…
1
3
5
7
```

[Fns…] Ctl 6:for i in range (début,fin,pas):

2nde [catalog]

str.format() format de chaîne

Module : Built-in

2nde [catalog]

Syntaxe :str.format()

Description : Formate la chaîne de caractères spécifiée. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> print("{+f}".format(12.34))
+12.340000
```

frexp()

Module : math

math Modul 1:math A:frexp()

Syntaxe : frexp(x)

Description : Renvoie une paire (y,n) telle que x == y * 2**n où y est un nombre flottant, avec 0.5<abs(y)<1 et n un entier.

2nde [catalog]

Exemple :

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11 #valide la description
2000.0
```

[Fns…] > Modul 1:math A:frexp()

les commandes import sont disponibles via 2nde [catalog]

from SCRIPT import *

Mot-clé

Syntaxe : from SCRIPT import *

Description : Permet d'importer un script. Importe les attributs publics d'un module Python dans l'espace de nom actuel.

Shell [Outils]
A:from SCRIPT import *

2nde [catalog]

from math import *

Mot-clé

Syntaxe : from math import *

Description : Permet d'importer toutes les fonctions et constantes à partir du module math.

math Modul
1:math...
1:from math import *

[Fns..] > Modul
1:math...
1:from math import *

2nde [catalog]

from random import *

Mot-clé

Syntaxe : from random import *

Description : Permet d'importer toutes les fonctions à partir du module random.

math Modul
2:random…
1:from random import *

[Fns..] > Modul
2:random…
1:from random import *

2nde [catalog]

from time import *

Mot-clé

Syntaxe : from time import *

Description : Permet d'importer toutes les méthodes à partir du module time.

Exemple :

Voir l'exemple de script : DASH1.

2nde [catalog]

math Modul
3:time…
1:from time import *

[Fns…]>Modul
3:time…
1:from time import *

from ti_system import *

Mot-clé

Syntaxe : from ti_system import *

Description : Permet d'importer toutes les méthodes à partir du module ti_system.

Exemple :

Voir l'exemple de script : REGEQ1.

2nde[catalog]

math Modul
4:ti_system…
1:from system import *

[Fns…]>Modul
4:ti_system…
1:from system import *

from ti_hub import *

Mot-clé 2nde [catalog]

Syntaxe : from ti_hub import *

Description : Permet d'importer toutes les méthodes à partir du module ti_hub. Pour des dispositifs d'entrée et de sortie spécifiques, utilisez la fonctionnalité de module dynamique en sélectionnant le dispositif via [Fns…] > Modul > ti_hub > menu Import dans l'Éditeur.

Voir : Module ti_hub – Ajout d'import à l'Éditeur et ajout du module de capteur ti_hub au menu Modul.

Exemple :

Voir l'exemple de script : DASH1.

global

Mot-clé

2nde [catalog]

Description : Utilisez global pour créer des variables globales au sein d'une fonction.

Pour plus de détails, consultez la documentation de CircuitPython.

grid(xscl,yscl,"style")

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.grid(xscl,yscl,"style")

[Fns...]>Modul ou math
5:ti_plotlib...>
Configurer
3:grid()

Description : Affiche une grille qui utilise l'échelle spécifiée pour les axes x et y. Remarque : L'ensemble du tracé est réalisé lorsque plt.show_plot() est exécutée.

La couleur de la grille se définit à l'aide de l'argument optionnel (r,v,b) utilisant des valeurs comprises dans la plage 0-255, le gris (192,192,192) étant la valeur par défaut.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

La valeur par défaut de xscl ou yscl = 1.0.

"style" = "dot" (pointillés – par défaut), "dash" (tirets), "solid" (trait continu) ou "point" (pixel)

Exemple :

Voir les exemples de scripts : COLORLIN ou GRAPHIQ.

grid(xscl,yscl,"style",(r,v,b))

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.grid(xscl,yscl,"style",(r,v,b))

[Fns...]>Modul ou math
5:ti_plotlib...>
Configurer
3:grid()

Description : Affiche une grille qui utilise l'échelle spécifiée pour les axes x et y. Remarque : L'ensemble du tracé est réalisé lorsque plt.show_plot() est exécutée.

grid(xscl,yscl,"style",(r,v,b))

La couleur de la grille se définit à l'aide de l'argument optionnel (r,g,b) utilisant des valeurs comprises dans la plage 0-255, le gris (192,192,192) étant la valeur par défaut.

La valeur par défaut de xscl ou yscl = 1.0.

« style" = "dot" (pointillés – par défaut), "dash" (tirets), "solid" (trait continu) ou "point" (pixel).

Si les valeurs xscl ou yscl sont inférieures à 1/50e de la différence xmax-xmin ou ymax-ymin, alors une exception « Invalid grid scale value » (Valeur d'échelle de grille incorrecte) est générée.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Exemple :

Voir l'exemple de script : GRAPHIQ.

hex(entier)

Module : Built-in

2nde [catalog]

Syntaxe : hex(entier)

Description : Affiche l'argument entier au format hexadécimal. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> hex(16)
'0x10'
>>> hex(16**2)
'0x100'
```

"if ":

Voir if..elif..else.. pour plus de détails.

2nde [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

if..elif..else..

Mot-clé

2nde [catalog]

Syntaxe : Identifiants de mise en retrait gris •• générés automatiquement dans l'application Python pour simplifier l'utilisation.

[Fns…] > Ctl

if :

1:if..

••

2:if..else..

elif :

3:if..elif..else

••

9:elif :

else:

0:else:

Description : if..elif..else est une instruction conditionnelle. L'Éditeur offre la mise en retrait automatique sous forme de points gris pour vous aider à utiliser la mise en retrait de programmation appropriée.

Exemple : Créez et exécutez ce script, que nous appellerons S01, à partir de l'Éditeur :

```
def f(a):
••if a>0:
••••print(a)
••elif a==0:
••••print("zéro")
••else:
••••a=-a
••••print(a)
```

Interactions avec le Shell

```
>>> # Shell Reinitialized
>>> # Exécution de S01
>>>from S01 import * #colle automatiquement
>>>f(5)
5
>>>f(0)
zéro
>>>f(-5)
5
```

if..else..

Mot-clé

2nde [catalog]

Voir if..elif..else.. pour plus de détails.

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Module : Built-in

2nde [catalog]

Syntaxe :var.imag

Description : Renvoie la partie imaginaire d'une variable donnée de type nombre complexe.

Exemple :

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

import math

Mot-clé

Syntaxe : import math

2nde [catalog]

Description : Le module math est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module « math » dans son propre espace nom.

import random

Mot-clé

import random

Syntaxe : import random

2nde [catalog]

Description : Le module random est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module « random » dans son propre espace nom.

import ti_hub

Mot-clé

2nde [catalog]

Syntaxe : import ti_hub

Description : Le module ti_hub est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module ti_hub dans son propre espace nom.

Pour des dispositifs d'entrée et de sortie spécifiques, utilisez la fonctionnalité de module dynamique en sélectionnant le dispositif via [Fns…] > Modul > ti_hub > menu Import dans l'Éditeur.

Voir :[Fns…]>Modul : module ti_hub.

import time

Mot-clé

2nde [catalog]

Syntaxe : import time

Description : Le module time est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module time dans son propre espace nom.

Voir :[Fns…]>Modul : modules time et ti_system.

import ti_plotlib as plt

Mot-clé

2nde [catalog]

Syntaxe : import ti_plotlib as plt

math Modul 5:ti_plotlib... 1:import ti_plotlib as plt

Description : Le module ti_plotlib est accessible à l'aide de cette commande. Cette instruction importe

import ti_plotlib as plt

les attributs publics du module ti_plotlib dans son propre espace nom. Les attributs du module ti_plotlib doivent être saisis sous la forme plt.attribute.

[Fns…]>Modul
5:ti_plotlib...
1:import ti_plotlib
as plt

Exemple :

Voir l'exemple de script : COLORLIN.

import ti_rover as rv

Mot-clé

2nde [catalog]

Syntaxe : import ti_rover as rv

math Modul
7:ti_rover...
1:import ti_rover
as rv

Description : Le module ti_rover est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module ti_rover dans son propre espace nom. Les attributs du module ti_rover doivent être saisis sous la forme rv.attribute.

[Fns…]>Modul
7:ti_rover...
1:import ti_rover
as rv

Exemple :

Voir l'exemple de script : ROVER.

import ti_system

Mot-clé

2nde [catalog]

Syntaxe : import ti_system

Description : Le module ti_system est accessible à l'aide de cette commande. Cette instruction importe les attributs publics du module ti_system dans son propre espace nom.

Exemple :

Voir l'exemple de script : REGEQ1.

in

Mot-clé

2nde [catalog]

Description : Utilisez « in » pour vérifier si une valeur se trouve dans une séquence ou pour itérer une séquence dans une boucle « for ».

in

.index(x)

Module : Built-in 2nde [catalog]

Syntaxe :var.index(x)

Description : Renvoie l'indice ou la position d'un élément d'une liste. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> a=[12,35,45]
>>> print(a.index(12))
0
>>> print(a.index(35))
1
>>> print(a.index(45))
2
```

input()

Module : Built-in 2nde [catalog]

Syntaxe : input()

Description : Invite à saisir des données [Fns…] E/S 2:input()

Exemple :

```
>>>input("Name? ")
Name? Moi
'Moi'
```

Autre exemple :

```
CréezScript A
len=float(input("len: "))
print(len)
```

```
ExécutezScript A
>>> # Shell Reinitialized
```

input()

```
>>> # Exécution de A
>>>from A import *
len: 15 (saisissez15)
15.0 (sortiefloat 15.0)
```

.insert(indice,x)

Module : Built-in

Syntaxe : listname.insert(indice,x)

Description : La méthode insert() insère un élément x après indice dans une séquence.

Exemple :

```
>>>listA = [2,4,6,8]
>>>listA.insert(3,15)
>>>print(listA)
[2,4,6,15,8]
```

2nde [listes] List 8:.insert (indice,x)

2nde [catalog]

[Fns…] > List 8:.insert (indice,x)

int()

Module : Built-in

Syntaxe : int(x)

Description : Retourne un objet integer x.

Exemple :

```
>>>int(34.67)
34
>>>int(1234.56)
1234
```

2nde [catalog]

[Fns…] > Type 1:int()

is

Mot-clé

Description : Utilisez « is » pour vérifier si deux objets sont identiques.

2nde [catalog]

labels("x-étiq","y-étiq",x,y)

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.labels("x-étiq","y-étiq",x,y)

[Fns...]>Modul ou math
5:ti_plotlib...>
Configurer
7:labels()

Description : Affiche les étiquettes "x-étiq" et "y-étiq" sur les axes du tracé aux positions de lignes x et y. Ajustez selon le besoin pour l'affichage du tracé.

"x-étiq" est positionnée sur la ligne x spécifiée (ligne 12 par défaut) et est justifiée à droite.

"y-étiq" est positionnée sur la ligne y spécifiée (ligne 2 par défaut) et est justifiée à gauche.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Remarque : plt.labels("|","",12,2) sera collé avec les valeurs par défaut des lignes x et y (12,2), que vous pouvez ensuite modifier en fonction de votre script.

Exemple :

Voir l'exemple de script : GRAPHIQ.

lambda

Mot-clé

2nde [catalog]

Syntaxe : arguments lambda : expression

Description : Utilisez lambda pour définir une fonction anonyme. Pour plus de détails, consultez la documentation de Python.

len()

Module : Built-in

2nde [listes] (au-dessus de stats)
List
3:len()

Syntaxe : len(séquence)

Description : Renvoie le nombre d'éléments présents dans l'argument. L'argument peut correspondre à une séquence ou à une collection. Pour plus de détails, consultez la documentation de Python.

2nde [catalog]

len()

Exemple :

```
>>>mylist=[2,4,6,8,10]
>>>len(mylist)
5
```

[Fns…] > List
3:len()

line(x1,y1,x2,y2,"mode")

Module : ti_plotlib

Syntaxe : plt.line(x1,y1,x2,y2,"mode")

Description : Affiche un segment de droite allant de (x1,y1) à (x2,y2).

La taille et le style sont définis à l'aide de pen() et de color() avant line().

Arguments :

x1,y1, x2,y2 sont des nombres réels flottants.

"mode": Avec la valeur par défaut "", aucune pointe de flèche n'est dessinée.
Avec la valeur "arrow", une pointe de flèche de vecteur est dessinée à la position (x2,y2).

Exemple :

Voir l'exemple de script : COLORLIN.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
7:line ou vector

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

lin_reg(xliste,yliste,"disp",ligne)

Module : ti_plotlib

Syntaxe : plt.lin_reg(xliste,yliste,"disp",ligne)

Description : Calcule et dessine le modèle de régression linéaire, ax+b, de xliste,yliste. Cette méthode doit suivre la méthode du diagramme de dispersion. L'affichage par défaut de l'équation est "center" à la ligne 11.

Argument :

"disp"	"left"
	"center"
	"right"
ligne	1 - 12

Les commandes plt.a (pente) et plt.b (ordonnée à l'origine) sont stockées lors de l'exécution de lin_reg.

Exemple :

Voir l'exemple de script : LINREG.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
8:lin_reg()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

list(séquence)

Module : Built-in

2nde [listes] (au-dessus de stats) List 2:list(séquence)

Syntaxe : list(séquence)

Description : Séquence (mutable) d'éléments du type de sauvegarde.

list()" convertit son argument en type « list ». À l'instar de nombreuses autres séquences, les éléments d'une liste ne doivent pas nécessairement être du même type.

2nde [catalog]

Exemple :

[Fns…] > List 2:list(séquence)

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
```

Exemple :

```
>>>mylist=[2,4,6,8]
>>>print(mylist)
[2,4,6,8]
>>> list({1,2,"c", 7})
[7, 1, 2, 'c']
>>> list("foobar")
['f', 'o', 'o', 'b', 'a', 'r']
```

log(x,base)

Module : math

Syntaxe : log(x,base)

Description : log(x) sans base renvoie le logarithme népérien x.

Exemple :

```
>>>from math import *
>>>log(e)
1.0
>>>log(100,10)
2.0
>>>log(32,2)
5.0
```

2nde log for log(x,10)

2nde ln for log(x) (logarithme népérien)

math Modul 1:math… 6:log(x,base)

2nde [catalog]

[Fns…] > Modul 1:math… 6:log(x,base)

les commandes import sont disponibles via 2nde [catalog]

math.fonction

Module : math

2nde [catalog]

Syntaxe : math.fonction

Description : Utilisez après la commande « import math » pour insérer une fonction dans le module math.

Exemple :

```
>>>import math
>>>math.cos(0)
1.0
```

max()

Module : Built-in

2nde [listes] (au-dessus de stats) List 4:max()

Syntaxe : max(séquence)

Description : Renvoie la valeur maximale dans la séquence. Pour plus d'informations sur max(), consultez la documentation de Python.

2nde [catalog]

Exemple :

```
>>>listA=[15,2,30,12,8]
>>>max(listA)
30
```

[Fns…] > List 4:max()

min()

Module : Built-in

2nde [listes] (au-dessus de stats) List 5:min()

Syntaxe : min(séquence)

Description : Renvoie la valeur minimale dans la séquence. Pour plus d'informations sur min(), consultez la documentation de Python.

2nde [catalog]

Exemple :

```
>>>listA=[15,2,30,12,8]
>>>min(listA)
2
```

[Fns…] > List 5:min()

monotonic() **temps écoulé**

Module : time

Syntaxe : monotonic() temps écoulé

Description : Renvoie la valeur du temps écoulé à partir du point d'exécution. Vous pouvez comparer la valeur renvoyée à d'autres valeurs en provenance de monotonic().

Exemple :

Exemple de script :

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

```
Exécutez le script EXEMPLE jusqu'à l'arrêt
de l'exécution.
>>>15.0
```

2nde [catalog]

[Fns...]>Modul ou math
3:time
3:momotonic()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de time.

None

Mot-clé 2nde [catalog]

Description : None représente l'absence d'une valeur.

Exemple : [a A #]

```
>>> def f(x):
…x
…
…
…
>>> print(f(2))
None
```

nonlocal

Mot-clé 2nde [catalog]

Syntaxe : nonlocal

Description : Utilisez nonlocal pour déclarer une variable qui n'est pas locale. Pour plus de détails, consultez la documentation de Python.

not

Mot-clé [?] [tests] Ops 0:not

Syntaxe : not x

Description : Donne True si x estFaux et False dans le cas contraire. Un espace est collé avant et après le mot-clé not. Éditez selon les besoins. [Fns…] > Ops 0:not

Exemple :

```
>>> not 2<5 #supprimez l'espace avant not
False
>>>3<8 and not 2<5
False
```

2nde [catalog]

[a A #]

oct(entier)

Module : Built-in

2nde [catalog]

Syntaxe : oct(entier)

Description : Renvoie la représentation de l'entier en base 8. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> oct(8)
'0o10'
>>> oct(64)
'0o100'
```

or

Mot-clé

[?] [tests] Ops 9:or

Syntaxe : x or y

[Fns…] > Ops 9:or

Description : Peut retourner Vrai ou faux. Renvoie x si x s'évalue à True et y dans le cas contraire. Un espace est collé avant et après or. Éditez selon les besoins.

2nde [catalog]

Exemple :

[a A #]

```
>>>2<5 or 5<10
True
>>>2<5 or 15<10
True
>>>12<5 or 15<10
False
>>> 3 or {}
3
>>> [] or {2}
{2}
```

ord("caractère")

Module : Built-in

2nde [catalog]

Syntaxe : ord("caractère")

Description : Renvoie la valeur unicode du caractère. Pour plus de détails, consultez la documentation de Python.

ord("caractère")

Exemple :

```
>>> ord("#")
35
>>> ord("/")
47
```

pass

Mot-clé

2nde [catalog]

Description : Utilisez pass dans une fonction ou une définition de classe vide comme une zone réservée dans laquelle vous ajouterez du code par la suite, à mesure que vous développerez votre script. Les définitions vides ne génèrent pas d'erreur lors de l'exécution du script.

pen("taille","style")

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.pen("taille","style")

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
9:pen()

Description : Définit l'apparence de toutes les lignes suivantes tracées jusqu'à la prochaine exécution de la commande pen().

Argument :

Les valeurs par défaut de pen() sont "thin" et "solid".

"taille"	"thin"
	"medium"
	"thick"
"style"	"solid"
	"dot"
	"dash"

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Exemple :

Voir les exemples de scripts : COLORLIN ou GRAPHIQ.

pi

Module : math

2nde [π] (au-dessus de trig)

Syntaxe : math.pi ou pi si le module math a été importé.

Description : La constante pi s'affiche comme illustré ci-

pi

dessous.

Exemple :

[Fns…] > Modul 1:math… > Const 2:pi

```
>>>from math import *
>>>pi
3.141592653589793
```

Autre exemple :

```
>>>import math
>>>math.pi
3.141592653589793
```

plot(xliste,yliste,"marq")

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.plot(xliste,yliste,"marq")

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
5:Connected Plot with Lists

Description : Une ligne polygonale est tracée en utilisant les paires ordonnées à partir des listes xliste et yliste spécifiées. Le style et la taille de la ligne sont définis à l'aide de la commande plt.pen().

xliste et yliste doivent correspondre à des nombres réels flottants et les listes doivent avoir la même dimension.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Argument :

"marq" désigne le symbole utilisé de la façon suivante :

o	point rempli (par défaut)
+	croix
x	x
.	pixel

Exemple :

Voir l'exemple de script : LINREG.

plot(x,y,"marq")

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.plot(x,y,"marq")

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
6:plot a Point

Description : Un tracé de points (x,y) s'affiche à l'aide des valeurs x et y spécifiées.

xliste et yliste doivent correspondre à des nombres réels flottants et les listes doivent avoir la même dimension.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Argument :

"marq" désigne le symbole utilisé de la façon suivante :

o	point rempli (par défaut)

plot(x,y,"marq")

+	croix
x	x
.	pixel

Exemple :

Voir l'exemple de script : LINREG.

pow(x,y)

Module : math

math Modul 1:math 5:pow(x,y)

Syntaxe : pow(x,y)

Description : Renvoie x élevé à la puissance y. Convertit x et y en nombres flottants. Pour plus d'informations, consultez la documentation de Python.

2nde [catalog]

Utilisez la fonction built-in pow(x,y) ou ** pour calculer des puissances entières exactes.

[Fns…] > Modul 1:math 5:pow(x,y)

Exemple :

```
>>>from math import *
>>>pow(2,3)
>>>8.0
```

les commandes import sont disponibles via 2nde [catalog]

Exemple avec : Built-in:

[Outils] > 6:Nouveau Shell

```
>>>pow(2,3)
8
>>>2**3
8
```

print()

Module : Built-in

2nde [catalog]

Syntaxe : print(argument)

Description : Affiche l'argument sous forme de chaîne de caractères.

[Fns…] > E/S 1:print()

Exemple :

```
>>>x=57.4
>>>print("mon nombre est =", x)
Mon nombre est = 57.4
```

radians()

Module : math

trig Trig
1:radians()

Syntaxe : radians(x)

Description : Convertit l'angle x exprimé en degrés en radians.

2nde [catalog]

Exemple :

```
>>>from math import *
>>>radians(180.0)
3.141592653589793
>>>radians(90.0)
1.570796326794897
```

[Fns…] >
Modul
1:math… >
Trig
1:radians()

raise

Mot-clé

2nde [catalog]

Syntaxe : raise exception

Description : Utilisez raise pour lever une exception spécifique et arrêter le script.

randint(min,max)

Module : random

Syntaxe : randint(min,max)

Description : Renvoie un entier aléatoire compris entre des valeurs min et max.

Exemple :

```
>>>from random import *
>>>randint(10,20)
>>>15
```

Autre exemple :

```
>>>import random
>>>random.randint(200,450)
306
```

Les résultats varient avec une sortie aléatoire.

[math] Modul
2:random
4:randint
(min,max)

[Fns…] >
Modul
2:random…
4:randint
(min,max)

[2nde] [catalog]

les
commandes
import sont
disponibles
via
[2nde] [catalog]

random()

Module : random

Syntaxe : random()

Description : Renvoie un nombre à virgule flottante compris entre 0 et 1.0. Cette fonction n'accepte aucun argument.

Exemple :

```
>>>from random import *
>>>random()
0.5381466990230621
```

Autre exemple :

```
>>>import random
>>>random.random()
0.2695098437037318
```

Les résultats varient avec une sortie aléatoire.

[math] Modul
2:random…
Random
2:random()

[Fns…] >
Modul
2:random…
Random
2:random()

[2nde] [catalog]

les commandes import sont disponibles via [2nde] [catalog]

random.fonction

Module : random

[2nde] [catalog]

Syntaxe : random.fonction

Description : Utilisez après la commande « import random » pour accéder à une fonction du module random.

Exemple :

```
>>>import random
>>>random.randint(1,15)
2
```

Les résultats varient avec une sortie aléatoire.

randrange(début,fin,pas)

Module : random

Syntaxe : randrange(début,fin,pas)

Description : Renvoie un nombre aléatoire entre début et fin selon le pas.

Exemple :

```
>>>from random import *
>>>randrange(10,50,2)
12
```

Autre exemple :

```
>>>import random
>>>random.randrange(10,50,2)
48
```

Les résultats varient avec une sortie aléatoire.

[math] Modul
2:random…
Random
6:randrange
(début,fin,pas)

[math] Modul
2:random…
Random
6:randrange
(début,fin,pas)

[2nde] [catalog]

les commandes import sont disponibles via [2nde][catalog]

range(début,fin,pas)

Module : Built-in

[2nde] [catalog]

Syntaxe : range(début,fin,pas)

Description : Utilisez la fonction range pour renvoyer une séquence de nombres. Tous les arguments sont facultatifs. La valeur de début par défaut est 0, le pas par défaut est égal à 1 et la séquence se termine à la valeur de fin.

Exemple :

```
>>> x = range(2,10,3)
>>> for i in x
… print(i)
…
…
2
5
8
```

.real

Module : Built-in

2nde [catalog]

Syntaxe :var.real

Description : Renvoie la partie réelle d'une variable donnée de type nombre complexe.

Exemple :

```
>>>a=complex(4,5)
>>>a.real
4
>>>a.imag
5
```

var=recall_list("nom") 1-6

Module : ti_system

2nde [catalog]

Syntaxe :var=recall_list("nom") 1-6

2nde [rappel]
ti_system
4:var=recall_list()

Description : Rappelle une liste prédéfinie de l'OS. La longueur de la liste doit être inférieure ou égale à 100.

Argument : "**nom**"

Pour OS L1-L6

1 - 6
"1" - "6"
'1' - '6'

[Fns…]>Modul ou math
4:ti_system
4:var=recall_list()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

Pour la liste personnalisée "nom" de l'OS

----- Nombre maximal de 5 caractères, chiffres ou lettres, commençant par des lettres, lesquelles doivent être en majuscules.

Exemples :

"ABCDE"

"R12"

La ligne "L1" est personnalisée en L1 et pas OS L1

Rappel : Le langage Python est à double précision. Python prend en charge plus de chiffres que l'OS.

Exemple :

var=recall_list("nom") 1-6

Exemple de script :

Créez une liste dans l'OS.
LIST={1,2,3}

Exécutez l'application Python.
Créez un nouveau script AA.

```
import ti_system as *
xlist=recall_list("LIST")
print xlist
```

Exécutez le script AA.
Le Shell affiche le résultat obtenu.

```
[1.0, 2.0, 3.0]
```

var=recall_RegEQ()

Module : ti_system

2nde [catalog]

Syntaxe :var=recall_RegEQ()

2nde [rappel]
ti_system
4:var=recall_
REGEQ()

Description : Rappelle la variable RegEQ à partir de l'OS CE. L'équation de régression doit être calculée dans l'OS avant le rappel de RegEQ dans l'application Python.

[Fns…]>Modul ou
math
4:ti_system
4:var=recall_
REGEQ()

Exemple :

Voir l'exemple de script : REGEQ1.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

.remove(x)

Module : Built-in

2nde [listes]
List
7:.remove(x)

Syntaxe : listname.remove(élément)

Description : La méthode remove() supprime la première instance d'un élément dans une séquence.

2nde [catalog]

.remove(x)

Exemple :

```
>>>listA = [2,4,6,8,6]
>>>listA.remove(6)
>>>print(listA)
[2,4,8,6]
```

[Fns…] > List
7:.remove(x)

return

Module : Built-in

2nde [catalog]

Syntaxe : return expression

Description : Une instruction « return » définit la valeur générée par une fonction. Par défaut, les fonctions Python renvoient None. Voir aussi : def fonction ():

[Fns…] > Fonc
1:def fonction():

Exemple :

[Fns…] > Fonc
2:return

```
>>> def f(a,b):
…return a*b
…
…
…
>>> f(2,3)
6
```

.reverse()

Module : Built-in

2nde [catalog]

Syntaxe : listname.reverse()

Description : Inverse l'ordre des éléments dans une séquence.

Exemple :

```
>>>list1=[15,-32,4]
>>>list1.reverse()
>>>print(list1)
[4,-32,15]
```

round()

Module : Built-in

2nde [catalog]

round()

Syntaxe : round(nombre, chiffres)

Description : Utilisez la fonction « round » pour renvoyer un nombre à virgule flottante arrondi aux chiffres spécifiés. Le chiffre par défaut est 0 ; la fonction renvoie l'entier le plus proche.

Exemple :

```
>>>round(23.12456)
23
>>>round(23.12456,3)
23.125
```

scatter(xliste,yliste,"marq")

Module : ti_plotlib

Syntaxe : plt.scatter(xliste,yliste,"marq")

Description : Une séquence de points de coordonnées provenant de (xliste,yliste) sera tracée avec le style de marque spécifié. Le style et la taille de la ligne sont définis à l'aide de la commande plt.pen().

xliste et yliste doivent correspondre à des nombres réels flottants et les listes doivent avoir la même dimension.

Argument :

"marq" désigne le symbole utilisé de la façon suivante :

o	point rempli (par défaut)
+	croix
x	x
.	pixel

Exemple :

Voir l'exemple de script : LINREG.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Dessin
4:scatter()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

seed()

Module : random

Syntaxe : seed() ou seed(x) où x est un entier

Description : Initialise un générateur de nombres aléatoires.

Exemple :

```
>>>from random import *
>>>seed(12)
>>>random()
0.9079708720366826
>>>seed(10)
>>>random()
0.9063990882481896
```

math Modul
2:random...
Random
7:seed()

[Fns...] > Modul
2:random...
Random
7:seed()

2nde [catalog]

seed()

```
>>>seed(12)
>>>random()
0.9079708720366826
```

Les résultats varient avec une sortie aléatoire.

les commandes import sont disponibles via 2nde [catalog]

set(séquence)

Module : Built-in

2nde [catalog]

Syntaxe : set(séquence)

Description : Renvoie une séquence sous forme d'ensemble. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> print(set("83CE")
{'E', '8', '3', 'C'}
```

show_plot() afficher > [annul]

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.show_plot() afficher > [annul]

[Fns...]>Modul ou math
5:ti_plotlib...>
Configurer
9:show_plot

Description : Exécute l'affichage du tracé tel que configuré dans le script.

show_plot() doit être placé après la configuration de tous les objets à tracer. L'ordre des objets à tracer dans le script est donné par l'ordre défini dans le menu Configurer.

[Fns...]>Modul ou [math]
5:ti_plotlib... >
Dessin
9:show_plot()

Pour obtenir de l'aide sur le tracé d'un modèle, dans le Gestionnaire de scripts, choisissez [New] ([zoom]), puis [Types] ([zoom]) afin de sélectionner le type de script "Plotting (x,y) & Text".

Une fois le script exécuté, vous pouvez effacer le tracé affiché en appuyant sur [annul] afin de revenir à l'invite du Shell.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

Exemple :

Voir les exemples de scripts : COLORLIN ou GRAPHIQ.

sin()

Module : math

Syntaxe : sin()

Description : Renvoie le sinus de x. L'angle passé en argument est exprimé en radians.

Exemple :

```
>>>from math import *
>>>sin(pi/2)
1.0
```

[trig] 3:sin()

[2nde] [catalog]

[Fns…] > Modul 1:math… > Trig 3:sin()

les commandes import sont disponibles via [2nde] [catalog]

sleep(secondes)

Module : ti_system; time

Syntaxe : sleep(secondes)

Description : Se met en veille pendant un nombre donné de secondes. L'argument en secondes est un nombre flottant.

Exemple :

Exemple de script :

```
from time import *
a=monotonic()
sleep(15)
b=monotonic()
print(b-a)
```

Exécutez le script TIME.

```
>>>15.0
```

[2nde] [catalog]

[2nde] [rappel] ti_system A:sleep()

[Fns…]>Modul ou [math] 4:ti_system A:sleep()

[Fns…]>Modul ou [math] 3:time 2:sleep()

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu Modul de ti_system.

.sort()

Module : Built-in

Syntaxe : listname.sort()

Description : La méthode trie une liste en place. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA) #listA mise à jour en liste triée
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nde [listes]
(au-dessus de
stats
List A:.sort()

2nde [catalog]

[Fns…] >
List
A:sort()

sorted()

Module : Built-in

Syntaxe : sorted(séquence)

Description : Renvoie une liste triée à partir de la séquence.

Exemple :

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>sorted(listA)
[2,3,3,4,4,4,5,6,6,7,8,9]
>>>print(listA) #listA n'a pas été modifiée
[4,3,6,2,7,4,8,9,3,5,4,6]
```

2nde [listes] (au-dessus de stats) List 0:sorted()

2nde [catalog]

[Fns…] > List 0:sorted()

.split(x)

Module : Built-in

Syntaxe :var.split(x)

Description : La méthode renvoie une liste définie par le séparateur spécifié. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>> a="rouge,bleu,vert"
>>> a.split(",")
['rouge', 'bleu', 'vert']
```

2nde [catalog]

sqrt()

Module : math

Syntaxe : sqrt(x)

Description : Renvoie la racine carrée de x.

Exemple :

```
>>>from math import *
>>>sqrt(25)
5.0
```

math Modul 1:math 3:sqrt()

2nde [catalog]

[Fns…] > Modul 1:math 3:sqrt()

les commandes

sqrt()

import sont disponibles via 2nde [catalog].

store_list("nom",var) 1-6

Module : ti_system

Syntaxe : store_list("nom",var) 1-6

Description : L'exécution du script Python stocke la liste Python var dans une variable "nom" de type liste de l'OS. La longueur de la liste doit être inférieure ou égale à 100.

Argument : "**nom**"

Pour OS L1-L6

1 - 6
"1" - "6"
'1' - '6'

Pour la liste OS personnalisée "nom"

----- Nombre maximal de 5 caractères, chiffres ou lettres, commençant par des lettres, lesquelles doivent être en majuscules.

Exemples :

"ABCDE"

"R12"

La ligne "L1" est personnalisée en L1 et pas OS L1

Rappel : Le langage Python est à double précision. Il prend donc en charge plus de chiffres que l'OS.

Exemple :

```
>>>a=[1,2,3]
>>>store_list("1",a)
>>>
```

Quittez l'application Python, puis appuyez sur 2nde [L1] (au-dessus de [1]) et entrer dans l'écran de calcul pour afficher la liste [L1] sous la forme {1 2 3}.

2nde [catalog]

2nde [rappel]
ti_system
3:var=store_list()

[Fns...]>Modul ou math
4:ti_system
3:var=store_list()

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Modul de ti_system.

str()

Module : Built-in

2nde [catalog]

Syntaxe : str(argument)

Description : Convertit l'argument en une chaîne de caractères.

[Fns…] > Type 3 :str()

Exemple :

```
>>>x=2+3
>>>str(x)
'5'
```

sum()

Module : Built-in

2nde [listes] (au-dessus de stats) List 9:sum()

Syntaxe : sum(séquence)

Description : Renvoie la somme des éléments inclus dans une séquence.

2nde [catalog]

Exemple :

```
>>>listA=[2,4,6,8,10]
>>>sum(listA)
30
```

[Fns…] > List 9:sum()

tan()

Module : math

Syntaxe : tan(x)

Description : Renvoie la tangente de x. L'argument Angle est exprimé en radians.

Exemple :

```
>>>from math import *
>>>tan(pi/4)
1.0
```

[trig] 5:tan()

[Fns…] > Modul 1:math… > Trig 5:tan()

[2nde] [catalog]

les commandes import sont disponibles via [2nde] [catalog]

text_at(ligne,"txt","align")

Module : ti_plotlib

Syntaxe : plt.text_at(ligne,"txt","align")

Description : Affiche le texte dans la zone de tracé selon l'alignement spécifié.

ligne	entier compris entre 1 et 12
"txt"	chaîne trop longue tronquée
"align"	"left" (par défaut) "center" "right"
optional	"1" efface la ligne avant le texte (par défaut)

[2nde] [catalog]

[Fns…]>Modul ou [math] 5:ti_plotlib…> Dessin 0:text_at()

Les commandes d'importation sont disponibles via [2nde] [catalog] ou dans le menu Configurer de ti_plotlib.

text_at(ligne,"txt","align")

"0" ligne ne s'efface pas

Exemple :

Voir l'exemple de script : DASH1.

time.function

Module : Built-in

2nde [catalog]

Syntaxe : time.function

Description : Utilisez après la commande d'importation de time pour accéder à une fonction du module time.

Exemple :

Voir :[Fns...]>Modul : modules time et ti_system.

title("titre")

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.title("titre")

[Fns...]>Modul ou math 5:ti_plotlib...> Configurer 8:title()

Description : Le "titre" s'affiche centré sur la première ligne de la fenêtre. Le "titre" est tronqué s'il est trop long.

Exemple :

Voir l'exemple de script : COLORLIN.

Les commandes d'importation sont disponibles via 2nde[catalog] ou dans le menu Configurer de ti_plotlib.

ti_hub.function

Module : ti_hub

2nde [catalog]

Syntaxe : ti_hub.function

ti_hub.function

Description : Utilisez après la commande d'importation de ti_hub pour accéder à une fonction du module ti_hub.

Exemple :

Voir :[Fns…]>Modul : module ti_hub.

ti_system.function

Module : ti_system 2nde [catalog]

Syntaxe : ti_system.function

Description : Utilisez après la commande d'importation de ti_system pour accéder à une fonction du module ti_system.

Exemple :

```
>>> # Shell Reinitialized
>>>import ti_system
>>>ti_system.disp_at(6,8,"texte")

texte>>>|

# s'affiche à la ligne 6, colonne 8 avec
l'invite du Shell comme indiqué.
```

True

Mot-clé

[?] [tests] (au-dessus de [math])

Description : Renvoie True lorsque l'instruction exécutée est Vraie. « True » représente la valeur vraie pour les objets de type booléen.

Exemple :

[2nde] [catalog]

```
>>>64>=32
True
```

[Fns…] > Ops
A:True

[a A #]

trunc()

Module : math

Syntaxe : trunc(x)

Description : Renvoie la valeur réelle x tronquée sous forme d'un entier.

Exemple :

```
>>>from math import *
>>>trunc(435.867)
435
```

[math] Modul
1:math...
0:trunc()

[2nde] [catalog]

[Fns...] >
Modul
1:math...
0:trunc()

les commandes import sont disponibles via [2nde] [catalog]

try:

Mot-clé

[2nde] [catalog]

Description : Utilisez le bloc de code « try » pour vérifier l'absence d'erreurs dans un bloc de code. Il s'utilise également avec « except » et « finally ». Pour plus de détails, consultez la documentation de Python.

tuple(séquence)

Module : Built-in

[2nde] [catalog]

Syntaxe : tuple(séquence)

Description : Convertit une séquence en tuple. Pour plus de détails, consultez la documentation de Python.

Exemple :

```
>>>a=[10,20,30]
>>>tuple(a)
(10,20,30)
```

type()

Module : Built-in

2nde [catalog]

Syntaxe : type(objet)

[Fns…]>Type>6:type ()

Description : Renvoie le type de l'objet.

Exemple :

```
>>>a=1,25
>>>print(type(a))
<class 'float'>
>>>b=100
>>>print(type(b))
<class 'int'>
>>>c=10+2j
>>>print(type(c))
<class 'complex'>
```

uniform(min,max)

Module : random

Syntaxe : uniform(min,max)

Description : Renvoie un nombre aléatoire x (flottant) tel que min <= x <= max.

Exemple :

```
>>>from random import *
>>>uniform(0,1)
0.476118
>>>uniform(10,20)
16.2787
```

Les résultats varient avec une sortie aléatoire.

[math] Modul
2:random…
Random
3:uniform
(min,max)

[2nde] [catalog]

[Fns…] > Modul
2:random…
Random
3:uniform
(min,max)

les commandes import sont disponibles via [2nde] [catalog]

wait_key()

Module : ti_system

2nde [catalog]

Syntaxe : wait_key()

Description : Renvoie une combinaison de codes de touche représentant la touche enfoncée associée à 2nde et/ou alpha. La méthode attend qu'une touche soit enfoncée avant de revenir au script.

Exemple :

Voir : [Fns…]>Modul : modules time et ti_system.

Voir : Mappage du clavier pour wait_key().

while condition:

Mot-clé

[Fns…] Ctl 8:while condition:

Syntaxe : while condition:

Description : Exécute les instructions figurant dans le bloc de code suivant jusqu'à ce que la « condition » soit égale à False.

2nde [catalog]

Exemple :

```
>>> x=5
>>> while x<8:
… x=x+1
… print(x)
…
…
6
7
8
```

window(xmin,xmax,ymin,ymax)

Module : ti_plotlib

2nde [catalog]

Syntaxe : plt.window(xmin,xmax,ymin,ymax)

[Fns...]>Modul ou math 5:ti_plotlib...> Configurer 4:window()

Description : Définit la fenêtre du tracé en faisant correspondre l'intervalle horizontal (xmin, xmax) et l'intervalle vertical (ymin, ymax) spécifiés à la zone de tracé allouée (pixels).

window(xmin,xmax,ymin,ymax)

Cette méthode doit être exécutée avant toutes les commandes du module ti_plotlib.

Les variables de Propriétés, xmin, xmax, ymin et ymax du module ti_plotlib seront mises à jour d'après les valeurs des arguments. Les valeurs par défaut sont (-10, 10, -6.56, 6.56).

Exemple :

Voir l'exemple de script : GRAPHIQ.

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

with

Mot-clé

2nde [catalog]

Description : Pour plus de détails, consultez la documentation de Python.

xmax défaut10.00

Module : ti_plotlib

Syntaxe : plt.xmax défaut10.00

Description : Variable spécifiée pour les arguments de window, définie en tant que plt.xmax.

Valeurs par défaut :

xmin	défaut -10.00
xmax	défaut 10.00
ymin	défaut -6.56
ymax	défaut 6.56

Exemple :

Voir l'exemple de script : GRAPHIQ.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Propriétés
2:xmax

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

xmin défaut-10.00

Module : ti_plotlib

Syntaxe : plt.xmin défaut-10.00

Description : Variable spécifiée pour les arguments de window, définie en tant que plt.xmin.

Valeurs par défaut :

xmin	défaut -10.00
xmax	défaut 10.00
ymin	défaut -6.56
ymax	défaut 6.56

Exemple :

Voir l'exemple de script : GRAPHIQ.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Propriétés
1:ymax

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

yield

Mot-clé

2nde [catalog]

Description : Utilisez yield pour mettre fin à une fonction. Renvoie un générateur. Pour plus de détails, consultez la documentation de Python.

ymax défaut6.56

Module : ti_plotlib

Syntaxe : plt.ymax défaut6.56

Description : Variable spécifiée pour les arguments de window, définie en tant que plt.ymax.

Valeurs par défaut :

xmin	défaut -10.00
xmax	défaut 10.00
ymin	défaut -6.56
ymax	défaut 6.56

Exemple :

Voir l'exemple de script : GRAPHIQ.

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Propriétés
4:ymax

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_plotlib.

ymin défaut-6.56

Module : ti_plotlib

Syntaxe : plt.ymin défaut-6.56

Description : Variable spécifiée pour les arguments de window, définie en tant que plt.ymin.

Valeurs par défaut :

xmin	défaut -10.00
xmax	défaut 10.00
ymin	défaut -6.56
ymax	défaut 6.56

2nde [catalog]

[Fns...]>Modul ou math
5:ti_plotlib...>
Propriétés
3:ymin

Les commandes d'importation sont disponibles via 2nde [catalog] ou dans le menu Configurer de ti_

ymin défaut-6.56

Exemple : plotlib.

Voir l'exemple de script : GRAPHIQ.

Annexe

Contenu Du Module Ti-Python Sélectionné

Built-ins

Built-ins	Built-ins	Built-ins
__name__	abs -- <function>	BaseException -- <class 'BaseException'>
__build_class__ -- <function>	all -- <function>	ArithmeticError -- <class 'ArithmeticError'>
__import__ -- <function>	any -- <function>	AssertionError -- <class 'AssertionError'>
__repl_print__ -- <function>	bin -- <function>	AttributeError -- <class 'AttributeError'>
bool -- <class 'bool'>	callable -- <function>	EOFError -- <class 'EOFError'>
bytes -- <class 'bytes'>	chr -- <function>	Exception -- <class 'Exception'>
bytearray -- <class 'bytearray'>	dir -- <function>	GeneratorExit -- <class 'GeneratorExit'>
dict -- <class 'dict'>	divmod -- <function>	ImportError -- <class 'ImportError'>
enumerate -- <class 'enumerate'>	eval -- <function>	IndentationError -- <class 'IndentationError'>
filter -- <class 'filter'>	exec -- <function>	IndexError -- <class 'IndexError'>
float -- <class 'float'>	getattr -- <function>	KeyboardInterrupt -- <class 'KeyboardInterrupt'>
int -- <class 'int'>	setattr -- <function>	ReloadException -- <class

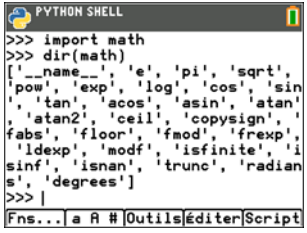
Built-ins	Built-ins	Built-ins
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <class 'MemoryError'>
object -- <class 'object'>	help -- <function>	NameError -- <class 'NameError'>
property -- <class 'property'>	hex -- <function>	NotImplementedError -- <class 'NotImplementedError'>
range -- <class 'range'>	id -- <function>	OSError -- <class 'OSError'>
set -- <class 'set'>	input -- <function>	OverflowError -- <class 'OverflowError'>
slice -- <class 'slice'>	isinstance -- <function>	RuntimeError -- <class 'RuntimeError'>
str -- <class 'str'>	issubclass -- <function>	StopIteration -- <class 'StopIteration'>
super -- <class 'super'>	iter -- <function>	SyntaxError -- <class 'SyntaxError'>
tuple -- <class 'tuple'>	len -- <function>	SystemExit -- <class 'SystemExit'>
type -- <class 'type'>	locals -- <function>	TypeError -- <class 'TypeError'>
zip -- <class 'zip'>	max -- <function>	UnicodeError -- <class 'UnicodeError'>
classmethod -- <class 'classmethod'>	min -- <function>	ValueError -- <class 'ValueError'>
staticmethod -- <class 'staticmethod'>	next -- <function>	ZeroDivisionError -- <class 'ZeroDivisionError'>

Built-ins	Built-ins	Built-ins
Ellipsis -- Ellipsis	oct -- <function>	
	ord -- <function>	
	pow -- <function>	
	print -- <function>	
	repr -- <function>	
	round -- <function>	
	sorted -- <function>	
	sum -- <function>	

Mots-clés

mots-clés	mots-clés	mots-clés
False	elif	lambda
None	else	nonlocal
True	except	not
and	finally	or
as	for	pass
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield

math

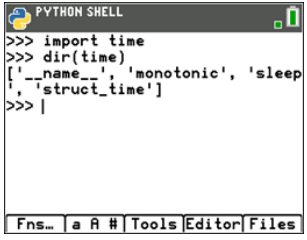


math	math	math
__name__	acos -- <function>	frexp -- <function>
e -- 2.71828	asin -- <function>	ldexp -- <function>
pi -- 3.14159	atan -- <function>	modf -- <function>
sqrt -- <function>	atan2 -- <function>	isfinite -- <function>
pow -- <function>	ceil -- <function>	isinf -- <function>
exp -- <function>	copysign -- <function>	isnan -- <function>
log -- <function>	fabs -- <function>	trunc -- <function>
cos -- <function>	floor -- <function>	radians -- <function>
sin -- <function>	fmod -- <function>	degrees -- <function>
tan -- <function>		

random

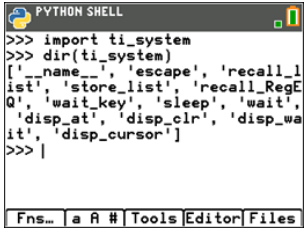
random	random	random
__name__	randint -- <function>	
seed -- <function>	choice -- <function>	
getrandbits -- <function>	random -- <function>	
randrange -- <function>	uniform -- <function>	

time



time	time	time
__name__		
monotonic		
sleep		
struc_time		

ti_system



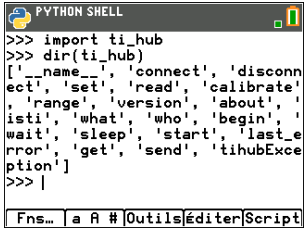
ti_system	ti_system	ti_system
__name__	recall_RegEQ	disp_at
escape	wait_key	disp_clr
recall_list	sleep	disp_wait
store_list	wait	disp_cursor

ti_plotlib

ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color

ti_plotlib	ti_plotlib	ti_plotlib
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle
sys	ymin	copysign
xmin	tiplotlibException	gr
ymax	lables	xmax
yscl	cls	sleep
_xy	sqrt	auto_window
_rdelta	xscl	
_ydelta	axes	
scatter		

ti_hub



ti_hub	ti_hub	ti_hub
__name__	version	last_error
connect	begin	sleep
disconnect	start	tihubException
set	about	wait
read	isti	get
calibrate	what	send
range	who	

ti_rover

ti_rover	ti_rover	ti_rover
__name__	color_blink	_rv
motor_right	motor_left	stay

ti_rover	ti_rover	ti_rover
to_angle	waypoint_heading	waypoint_xythdrn
to_xy	_motor	ranger_measurement
red_measurment	gyro_measutrment	left
rvmovement	wait_until_done	pathlist_cmdnum
gray_measurment	encoders_gyro_measurement	waypoint_y
_excpt	pathlist_distance	waypoint-x
ti_hub	position	pathlist_y
waypoint_prev	blue_measurement	pathlist_x
pathlist_time	forward	right
waypoint_revs	waypoint_distance	color_rgb
to_polar	grid_origin	pathlist-revs
waypoint_eta	resume	color_measurement
color_off	path_done	tiroverException
grid_m_unit	disconnect_rv	forward_time
path_clear	backward_time	pathlist_heading
green_measurement	zero-gyro	
waypoint_time	_rv_connected	

ti_rover	ti_rover	ti_rover
motors	stop	
backward		

.

Mappage du clavier pour wait_key()

graphstats f1 f(x)	73 48 73	déf table f2 fenêtre	72 75 72	format f3 zoom	46 87 46	calculs f4 trace	90 59 90	table f5 graphe	68 74 68
2nde	---- ---- ----	quitter mode	69 64 69	insérer suppr	10 11 10	◀| ◀	2 14 2	|▶ ▶	1 15 1
verr A alpha	---- ---- ----	échanger X,T,θ,n	180 65 180	listes stats	49 58 49	☀ ≡ ▲	3 ---- 3	☼ ≡ ▼	4 ---- 4
tests A math	50 51 154	x^{-1} B matrice	55 182 155	dessin C prgm	45 47 156	distrib var	53 56 53	 annul	9 9 9
angle D ◀ ▶	64018 57 157	π E trig	64017 181 158	apps F résol	64020 44 159	$\int_a^b$ G ▯	64458 64016 160	H ^	132 --- 161
√‾ I x^2	189 190 162	EE J ,	139 152 163	{ K (	133 236 164	} L)	134 237 165	e M ÷	131 239 166
10^x N log	193 194 167	u_n O 7	149 249 168	v_n P 8	150 250 169	w_n Q 9	151 251 170	[R ×	130 135 171
e^x S ln	191 192 172	L4 T 4	146 246 173	L5 U 5	147 247 174	L6 V 6	148 248 175	] W −	129 136 176
rappel X sto→	138 12 177	L1 Y 1	143 243 178	L2 Z 2	144 244 179	L3 θ 3	145 245 204	mém " +	128 54 203
off on	break off break	catalog _ 0	142 62 153	i : .	141 238 198	rép ? (–)	140 197 202	précéd entrer	5 13 5

Informations générales

Aide en ligne

education.ti.com/eguide

Sélectionnez votre pays pour obtenir d'autres informations relatives aux produits.

Contacter l'assistance technique TI

education.ti.com/ti-cares

Sélectionnez votre pays pour obtenir une assistance technique ou d'autres types de support.

Informations sur le service et la garantie

education.ti.com/warranty

Sélectionnez votre pays pour obtenir des informations sur la durée et les conditions de la garantie ou sur le service après-vente.

Garantie limitée. Cette garantie n'affecte pas vos droits statutaires.