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

Version 5.4.0

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.

Sommaire

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 v5.4.0 est disponible pour la TI-83 Premium CE *Édition Python* Les informations incluses dans ce guide électronique s'appliquent à la calculatrice TI-83 Premium CE *Édition Python* mise à jour avec le bundle CE v5.4.0.

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 v5.4.0 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*.

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:.

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 Adapter

Utilisez les touches de raccourci affichées à l'écran pour naviguer entre les différents espaces de travail de l'application Python Adapter. 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.

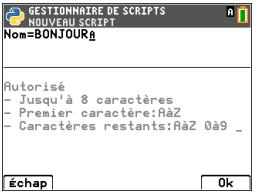
Remarque : Les écrans réels peuvent présenter de légères différences par rapport aux images fournies.

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

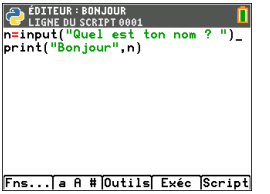
- L'exemple de script utilisé est BONJOUR. Saisissez le nom du script, puis appuyez sur [graphe] ([Ok]).
- Notez que le curseur est en verrouillage ALPHA. Saisissez toujours un nom de script conforme aux règles affichées à l'écran.



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 BONJOUR.

- 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.

Configuration d'une session Python avec vos scripts

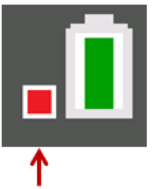
Lorsque vous exécutez l'application Python, la connexion CE établie avec l'expérience TI-Python lance la synchronisation pour la session Python en cours. Votre liste de scripts, présents dans la mémoire RAM, s'affiche lors de la synchronisation avec l'expérience Python.

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.

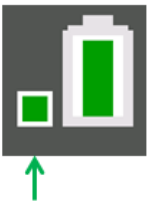
Vous observerez peut-être une synchronisation complète de vos programmes avec l'expérience TI-83 Premium CE *Édition Python* lorsque vous mettrez à jour votre version à partir du site education.ti.com/83ceupdate.

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

TI Connect™ CE v5.4.0 permet d'effectuer des captures de n'importe quel écran de l'application Python.

Espaces de travail Python

L'application Python Adapter comprend trois espaces de travail pour développer votre programmation 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.
[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. • Dupliquer, supprimer ou renommer un script sélectionné. • Afficher l'écran À propos. • Quitter l'application. Vous pouvez également utiliser 2nde [quitter].

Éditeur Python

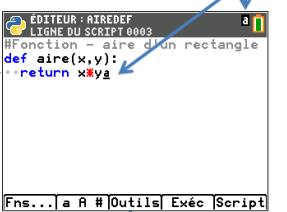
L'Éditeur Python s'affiche à partir d'un script sélectionné dans le Gestionnaire de scripts ou à partir du Shell. L'Éditeur affiche en couleur les mots-clés, les opérateurs, les commentaires, les chaînes et les retraits. Le collage rapide de fonctions et mots-clés Python courants est disponible, de même 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.

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.

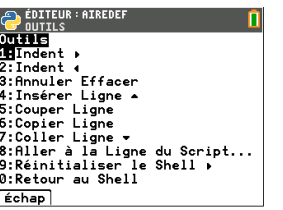
Emplacement du curseur sur la ligne de script.

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 math et random. **Remarque :** 2nde [catalog] est également

<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>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>
<tr><td>9: Réinitialiser le Shell</td><td>Affiche la console Shell réinitialisée.</td></tr>
</table></td></tr>
</table>

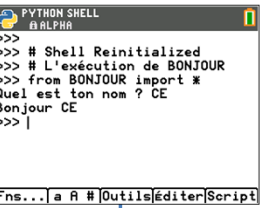
Menus et touches de raccourci de l'Éditeur Python		
Menus	**Touche d'accès**	**Description**
		0: Retour au Shell — Affiche le Shell dans son état actuel.
[Exéc]	trace	Sélectionnez [Exéc] pour exécuter votre script.
[Script]	graphe	Sélectionnez [Script] pour afficher le Gestionnaire de scripts.

La console Python (Shell)

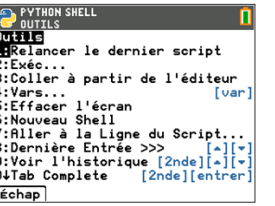
La console Python (Shell) vous permet d'interagir avec l'interpréteur Python ou d'exécuter des scripts Python. Le collage rapide de fonctions et mots-clés Python courants est disponible, aussi bien par la saisie directe au clavier que par l'entrée de caractères [a A #]. L'invite du Shell peut vous servir 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.

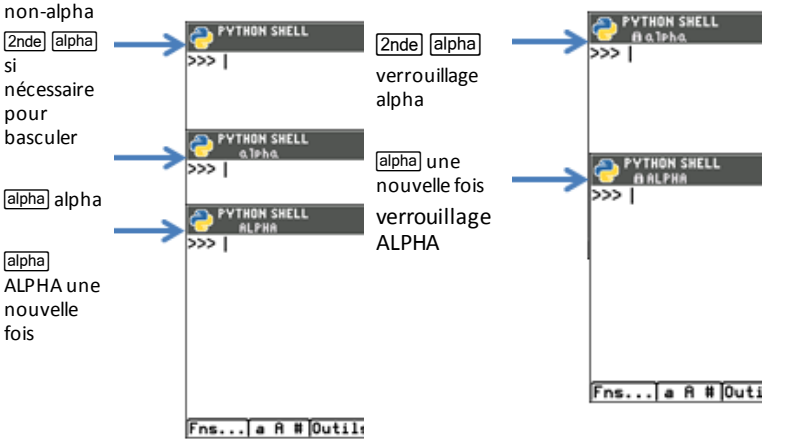
Le Shell est réinitialisé lors de l'exécution d'un nouveau script.



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



États du curseur Shell



Menus et touches de raccourci du Shell 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 math et random. **Remarque :** 2nde [catalog] est également pratique pour le collage rapide.
[a A #]	fenêtre	Sélectionnez [a A #] afin d'accéder à une palette de caractères servant de méthode alternative pour saisir de nombreux caractères.
[Outils]	zoom	Sélectionnez [Outils] pour afficher les éléments de menu suivants. 1: Relancer le dernier script — Relance le dernier script exécuté dans le Shell. 2: Exéc… — Affiche la liste des scripts Python qu'il est possible d'exécuter dans le Shell. 3: Coller à partir de l'éditeur — Colle la dernière ligne de script copiée à partir de l'Éditeur dans l'invite du Shell.

Menus et touches de raccourci du Shell Python			
Menus	**Touche d'accès**	**Description**	
		4: Vars...	Affiche les variables du dernier script exécuté. N'affiche pas les variables définies dans un script importé.
		5: Effacer l'écran	Efface l'écran du Shell. Ne réinitialise pas le Shell.
		6: Nouveau Shell	Réinitialise le Shell.
		7: Aller à la Ligne du Script...	Affiche l'Éditeur à partir du Shell en plaçant le curseur sur la ligne de script spécifiée.
		8: Dernière Entrée >>> 2nde ▲ ▼	Affiche jusqu'aux 8 dernières entrées à l'invite de la console au cours d'une session Shell.
		9: Voir l'historique 2nde ▲ 2nde ▼	Permet de faire défiler l'écran du Shell pour afficher les 60 dernières lignes générées dans la console au cours d'une session Shell.
		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 via la commande 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.	

Menus et touches de raccourci du Shell Python		
Menus	**Touche d'accès**	**Description**
[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 programme en cours d'exécution.

Entrées – Clavier, catalogue, jeu de caractères et menus

Conseils de saisie rapide

- Clavier
- Catalogue
- Jeu de caractères [a A #]
- Menus [Fns…]

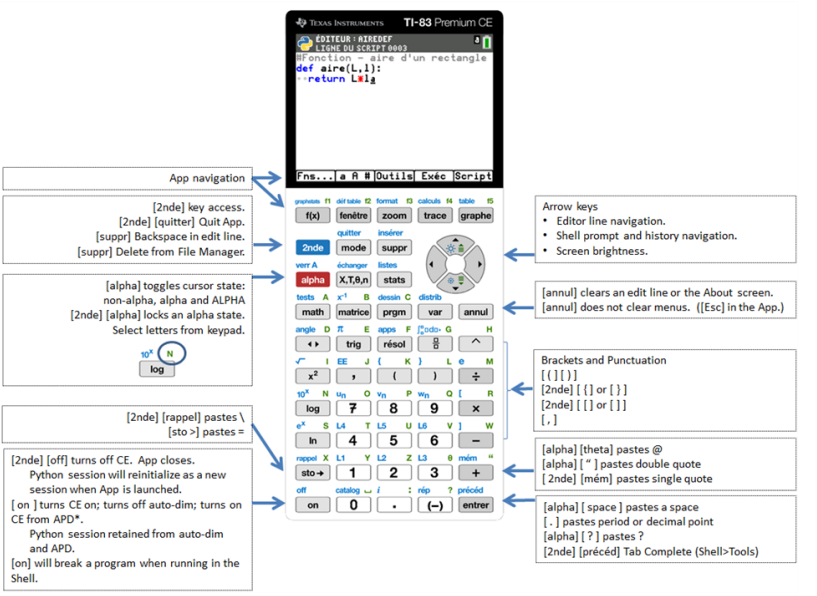
Utilisation du clavier, du catalogue, du jeu de caractères [a A #] et des menus Fns…

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.

Clavier

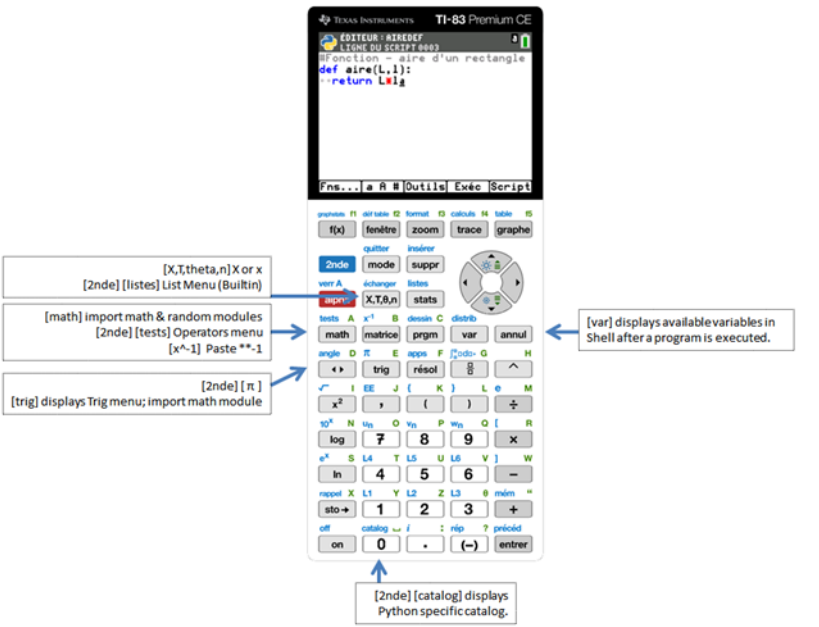
Lorsque l'application Python est en cours d'exécution, le clavier est prévu pour coller les opérations Python appropriées ou pour ouvrir des menus destinés à faciliter la saisie des fonctions, mots-clés, méthodes, opérateurs, etc. Les touches 2nde et alpha vous permettent d'accéder aux deuxième et troisième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

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



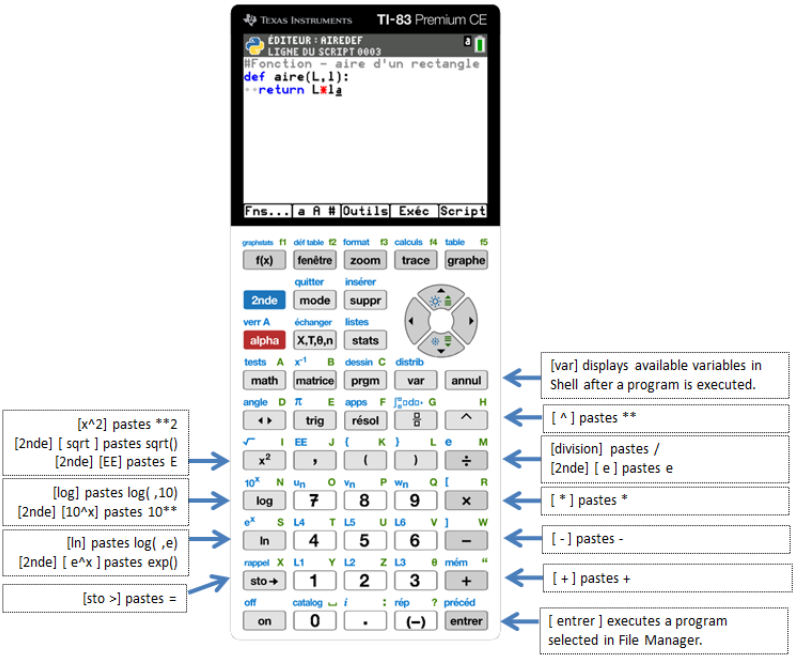
Avis de non-responsabilité : Les images de la calculatrice CE figurant dans ce document ont été prises à partir de TI-SmartView CE pour la famille TI-83 v5.4.0. Même si ce modèle n'affiche pas la mention « Édition Python » sur l'image de l'émulateur, il assure réellement l'émulation de la TI-83 Premium CE *Édition Python* qui exécute l'application Python.

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



Avis de non-responsabilité : Les images de la calculatrice CE figurant dans ce document ont été prises à partir de TI-SmartView CE pour la famille TI-83 v5.4.0. Même si ce modèle n'affiche pas la mention « Édition Python » sur l'image de l'émulateur, il assure réellement l'émulation de la TI-83 Premium CE *Édition Python* qui exécute l'application Python.

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



Avis de non-responsabilité : Les images de la calculatrice CE figurant dans ce document ont été prises à partir de TI-SmartView CE pour la famille TI-83 v5.4.0. Même si ce modèle n'affiche pas la mention « Édition Python » sur l'image de l'émulateur, il assure réellement l'émulation de la TI-83 Premium CE *Édition Python* qui exécute l'application Python.

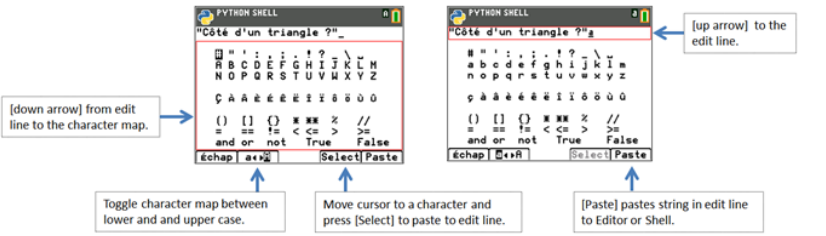
Catalogue

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.

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.



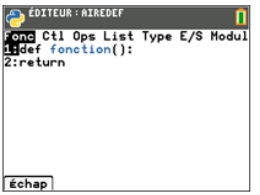
Remarque : Lorsque le curseur se trouve dans la ligne d'édition [a A #], certaines touches du clavier ne sont pas disponibles. Lorsque le curseur se trouve dans le jeu de caractères, les fonctions du clavier sont limitées.

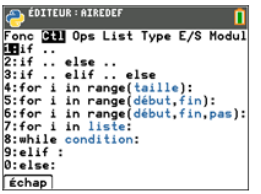
Menus [Fns…]

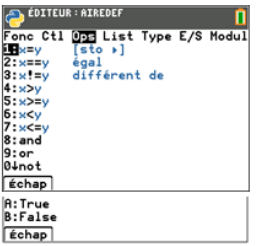
L'onglet de raccourci [Fns…] affiche les menus contenant les fonctions, mots-clés et opérateurs Python fréquemment utilisés. Les menus permettent également d'accéder aux fonctions et constantes sélectionnées dans les modules math et random. Même si vous pouvez saisir du code caractère par caractère à 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 Catalogue et Clavier pour d'autres méthodes de saisie.

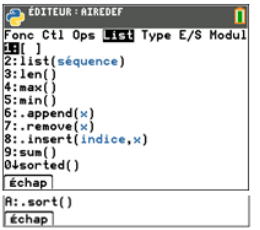
Sous-menus des fonctions et modules

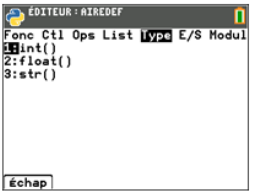
Éléments intégrés (Built-ins), opérateurs et mots-clés

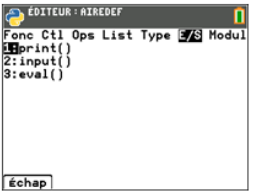












Sous-menus des modules

Lorsque vous utilisez une fonction ou une constante Python à partir d'un module, utilisez toujours une instruction d'importation pour indiquer l'emplacement du module de la fonction ou de la constante. Reportez-vous à la section Python pour la TI-83 Premium CE

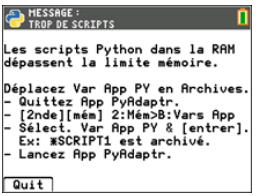
Messages de l'application Python

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

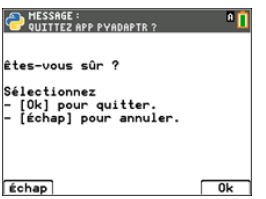
Les fichiers Python sont synchronisés avec l'adaptateur. Si la mémoire de l'adaptateur* n'est pas suffisante pour gérer le nombre d'AppVars Python stockées dans la mémoire RAM de votre CE, lorsque l'application Python Adapter se synchronise avec l'adaptateur TI-Python, vous êtes invité à déplacer certains scripts de la mémoire RAM vers la mémoire d'archive.

*La mémoire de l'adaptateur TI-Python peut héberger jusqu'à 40 K ou 80 scripts Python, selon la première de ces deux éventualités.



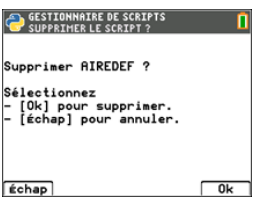
Utilisez [2nde] [Quitter] pour quitter l'application

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 Adapter, vos scripts AppVar Python sont synchronisés avec l'adaptateur. Le Shell est réinitialisé.



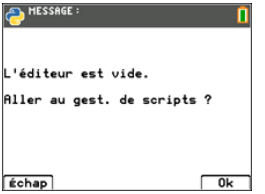
Dans le Gestionnaire de scripts, appuyez sur la touche [suppr] dans un script Python sélectionné ou choisissez Gestionnaire de scripts > Gérer, puis 2:Supprimer le script...

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

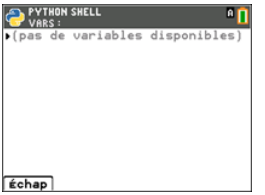


Vous tentez de créer un nouveau script ou de dupliquer un script Python existant déjà sur votre CE, soit dans la mémoire d'archive, soit désactivé pour le mode Examen. Saisissez un autre nom.

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.



Erreurs

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.

Consultez la section Sélection de contenus de module dans les Built-ins pour obtenir la liste des erreurs (exceptions) Python prises en charge dans cette version.

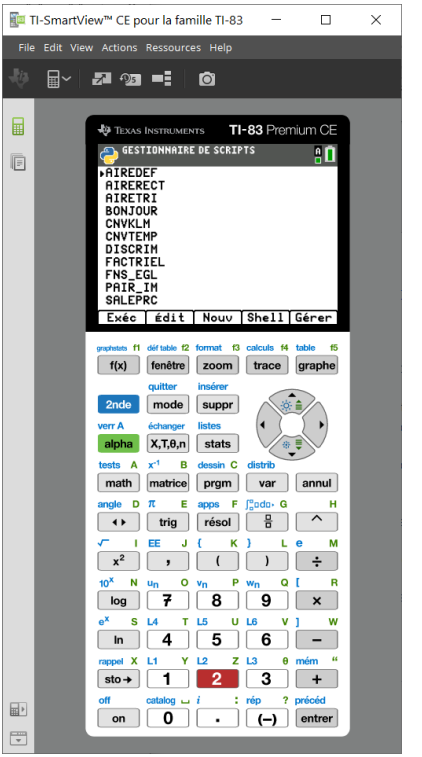
Utilisation de TI-SmartView™ CE pour les démonstrations d'expérience Python

Nouveautés pour TI-SmartView™ CE v5.4.0

L'application Python v5.4.0 sera chargée dans l'émulateur 83 CE, qui inclut le système d'exploitation 83 CE OS v5.4.0 dans cette version.

Mettez à jour votre TI-SmartView™ CE vers la version v5.4.0 pour l'expérience Python pour enseignants.

- Mettez à jour TI-SmartView™ CE pour la famille TI-83 v5.4.0 à partir du site *education.ti.com/83ceupdate*.
- Lancez TI-SmartView CE v5.4.0.
- Exécutez l'application Python sur l'émulateur TI-83 Premium CE.*
- L'application Python propose
  - Gestionnaire de scripts
  - Éditeur
  - Exécution de votre script Python dans le Shell
- L'application SmartPad CE active le clavier à distance lorsque l'application Python est en cours d'exécution.
- Vous pouvez envoyer des fichiers *.py stockés sur votre ordinateur à l'espace de travail de l'Explorateur de l'émulateur afin de convertir vos scripts en AppVars PY.



Remarque : Quittez l'application Python avant de basculer dans l'Explorateur de l'émulateur pour envoyer/recevoir des AppVars PY. Si une application est laissée en cours d'exécution, les modifications d'un script apportées dans l'éditeur Python ne sont pas immédiatement prises en compte dans l'espace de travail de l'Explorateur de l'émulateur.

***Remarque :** L'émulateur s'appelle toujours TI-83 Premium CE sans la mention *Édition Python*, mais il prend en charge l'exécution de l'application Python. Aucun adaptateur n'est nécessaire.

Rappel : Pour tout ordinateur/toute expérience TI-Python : Une fois que vous avez créé un script Python sur l'ordinateur, validez son exécution sur la calculatrice dans l'expérience TI-Python. Modifiez le script si nécessaire.

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

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 plus de détails, consultez le Guide électronique de la 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 CircuitPython 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
- math*
- random*

***Remarque :** Si vous possédez des scripts Python créés dans un autre environnement de développement Python, modifiez-les de manière à n'accéder qu'aux éléments disponibles dans la solution TI-Python Adapter.

Comme dans n'importe quelle version de Python, vous devrez inclure « from math import * » et/ou « from random import * » pour utiliser les fonctions, les méthodes ou les constantes présentes dans le module math ou le module random. 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 :

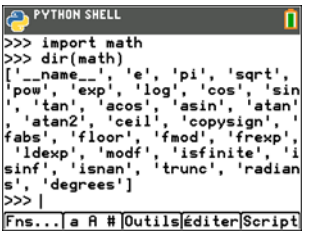
```
>>> help("modules")
__main__ sys gc
random time array
```

```
math builtins collections
```

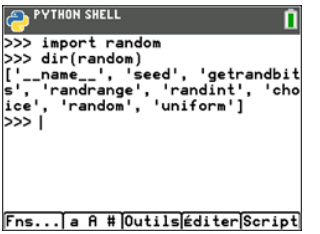
Vous pouvez afficher le contenu des modules dans le Shell comme illustré en utilisant « import module » et « dir(module) ».

Ces écrans affichent le contenu des modules math et random.

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



module math



module random

Contenu d'une sélection de modules et mots-clés

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

Contenu du module sélectionné pour l'application 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.

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
- I
- L
- M
- N
- O
- P
- R
- S
- T
- U
- W
- Y

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

abs()

Module : Built-in

[2nde] [catalog]

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.

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

Exemple :

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

acos()

Module : math

[trig] 7:acos()

Syntaxe : acos(x)

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]

B

break

Mot-clé

[2nde] [catalog]

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

.

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]

class

Mot-clé 2nde [catalog]

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

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
```

D

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

trig Trig
2:degrees()

Syntaxe : degrees(x)

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

Exemple :

2nde
[catalog]

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

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

del

Mot-clé

2nde [catalog]

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

E

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:

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

2nde [catalog]

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]
```

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]

frexp()

Module : math

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.

Exemple :

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

math Modul 1:math A:frexp()

2nde [catalog]

[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]

G

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.

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:

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

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.

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 ».

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

.insert(indice,x)

Module : Built-in

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

Syntaxe : listname.insert(indice,x)

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

2nde [catalog]

Exemple :

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

[Fns…] > List
8:.insert
(indice,x)

int()

Module : Built-in

2nde [catalog]

Syntaxe : int(x)

Description : Retourne un objet integer x.

[Fns…] > Type 1:int()

Exemple :

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

is

Mot-clé

2nde [catalog]

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

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]

Exemple :

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

[Fns…] > List 3:len()

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]

M

math.fonction

Module : math

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
```

2nde [catalog]

max()

Module : Built-in

Syntaxe : max(séquence)

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

Exemple :

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

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

2nde [catalog]

[Fns…] > List 4:max()

min()

Module : Built-in

Syntaxe : min(séquence)

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

Exemple :

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

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

2nde [catalog]

[Fns…] > List 5:min()

N

None

Mot-clé

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

Exemple :

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

2nde [catalog]

[a A #]

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 #]

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}
```

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.

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-dessous.

Exemple :

[Fns…] > Modul 1:math… > Const 2:pi

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

Autre exemple :

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

pow(x,y)

Module : math

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.

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

Exemple :

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

Exemple avec : Built-in:

[Outils] > 6:Nouveau Shell

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

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

2nde [catalog]

[Fns…] > Modul 1:math 5:pow(x,y)

les commandes import sont disponibles via 2nde [catalog]

print()

Module : Built-in

Syntaxe : print(argument)

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

Exemple :

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

2nde [catalog]

[Fns…] > E/S 1:print()

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

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.

[2nde] [catalog]

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
```

.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]

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]

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
```

S

seed()

Module : random

math Modul
2:random…
Random
7:seed()

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

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

2nde [catalog]

les commandes import sont disponibles via 2nde [catalog]

Les résultats varient avec une sortie aléatoire.

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]

.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()

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 import sont disponibles via 2nde [catalog].

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.

Exemple :

2nde [catalog]

```
>>>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]

True

Mot-clé

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

Exemple :

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

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

[2nde] [catalog]

[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.

U

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]

W

while condition:

Mot-clé

Syntaxe : while condition:

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

[Fns…] Ctl
8:while
condition:

[2nde] [catalog]

Exemple :

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

@

Opérateur

alpha [θ] (au-dessus de 3)

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

2nde [catalog]

<<

Opérateur

2nde [catalog]

Syntaxe : x<<n

Description : Décalage vers la gauche bit à bit de n bits.

>>

Opérateur

2nde [catalog]

Syntaxe : x>>n

Description : Décalage vers la droite bit à bit de n bits.

|

Opérateur

2nde [catalog]

Syntaxe : x|y

Description : Opérateur or (ou) bit à bit.

&

Opérateur

2nde [catalog]

Syntaxe : x&y

Description : Opérateur and (et) bit à bit.

^

Opérateur

2nde [catalog]

Syntaxe : x^y

Description : Opérateur exclusive or (ou exclusif) bit à bit.

~

Opérateur

2nde [catalog]

Syntaxe : ~x

Description : Opérateur not bit à bit ; les bits de x sont inversés.

x<=y

Opérateur

math
1:math > Ops
7:x<=y

Syntaxe : x<=y

Description : Comparaison ; x inférieur ou égal à y.

Exemple :

2nde [catalog]

```
>>>2<=5
True
>>>3<=0
False
```

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Opérateur

Syntaxe : x<y

Description : Comparaison; x strictement inférieur à y.

Exemple :

```
>>>6<10
True
>>>12<-15
False
```

math
1:math > Ops
6:x<y

2nde [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Opérateur

Syntaxe : x>=y

Description : Comparaison ; x supérieur ou égal à y.

Exemple :

```
>>>35>=25
True
>>>14>=65
False
```

math
1:math > Ops
5:x>=y

2nde [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Opérateur

Syntaxe : x>y

Description : Comparaison; x strictement supérieur à y.

Exemple :

```
>>>35>25
True
>>>14>65
False
```

math
1:math > Ops
4:x>y

2nde [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Opérateur

Syntaxe : x!=y

Description : Comparaison ; x différent de y.

Exemple :

```
>>>35!=25
True
>>>14!=10+4
False
```

[math]
1:math > Ops
3:x!=y

[2nde] [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Opérateur

Syntaxe : x==y

Description : Comparaison ; x égal à y.

Exemple :

```
>>>75==25+50
True
>>>1/3==0.333333
False
>>>1/3==0.3333333 #égal à une valeur Python enregistrée
True
```

[math]
1:math > Ops
2:x==y

[2nde] [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Opérateur

Syntaxe : x=y

Description : y est enregistré dans la variable x

Exemple :

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3 #Utilisez [ ^ ] sur le clavier pour **
>>>print(B)
8
```

sto→

math
1:math > Ops
1:x=y

2nde [catalog]

[Fns…] > Ops
1:x=y

[a A #]

\

Séparateur

Description : Barre oblique inverse.

2nde [catalog]

[a A #]

\t

Séparateur

Description : Espace de tabulation entre des chaînes ou des caractères.

2nde [catalog]

\n

Séparateur

Description : Retour à la ligne permettant d'afficher la chaîne de caractères de manière claire à l'écran.

2nde [catalog]

' '

Séparateur

Description : Deux guillemets simples sont ajoutés.

Exemple :

```
>>>eval('a+10')
17
```

2nde [mém] (au-dessus de +)

2nde [catalog]

[a A #]

" "

Séparateur

Description : Deux guillemets doubles sont ajoutés.

Exemple :

```
>>>print("Ok")
```

alpha ["] (au-dessus de +)

2nde [catalog]

[a A #]

with

Mot-clé

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

2nde [catalog]

Y

yield

Mot-clé

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

2nde [catalog]

Annexe

Selected TI-Python Module Content

Built-ins	math	random	keywords
__name__	__name__	__name__	False
__build_class__ -- <function>	e -- 2.71828	seed -- <function>	None
__import__ -- <function>	pi -- 3.14159	getrandbits -- <function>	True
__repl_print__ -- <function>	sqrt -- <function>	randrange -- <function>	and
bool -- <class 'bool'>	pow -- <function>	randint -- <function>	as
bytes -- <class 'bytes'>	exp -- <function>	choice -- <function>	assert
bytearray -- <class 'bytearray'>	log -- <function>	random -- <function>	break
dict -- <class 'dict'>	cos -- <function>	uniform -- <function>	class
enumerate -- <class 'enumerate'>	sin -- <function>		continue
filter -- <class 'filter'>	tan -- <function>		def
float -- <class 'float'>	acos -- <function>		del
int -- <class 'int'>	asin -- <function>		elif
list -- <class 'list'>	atan -- <function>		else
map -- <class 'map'>	atan2 -- <function>		except
memoryview -- <class 'memoryview'>	ceil -- <function>		finally
object -- <class 'object'>	copysign -- <function>		for
property -- <class 'property'>	fabs -- <function>		from
range -- <class 'range'>	floor -- <function>		global
set -- <class 'set'>	fmod -- <function>		if
slice -- <class 'slice'>	frexp -- <function>		import

Built-ins	math	random	keywords
str -- <class 'str'>	ldexp -- <function>		in
super -- <class 'super'>	modf -- <function>		is
tuple -- <class 'tuple'>	isfinite -- <function>		lambda
type -- <class 'type'>	isinf -- <function>		nonlocal
zip -- <class 'zip'>	isnan -- <function>		not
classmethod -- <class 'classmethod'>	trunc -- <function>		or
staticmethod -- <class 'staticmethod'>	radians -- <function>		pass
Ellipsis -- Ellipsis	degrees -- <function>		raise
abs -- <function>			return
all -- <function>			try
any -- <function>			while
bin -- <function>			with
callable -- <function>			yield
chr -- <function>			
dir -- <function>			
divmod -- <function>			
eval -- <function>			
exec -- <function>			
getattr -- <function>			
setattr -- <function>			
globals -- <function>			

Built-ins	math	random	keywords
hasattr -- <function>			
hash -- <function>			
help -- <function>			
hex -- <function>			
id -- <function>			
input -- <function>			
isinstance -- <function>			
issubclass -- <function>			
iter -- <function>			
len -- <function>			
locals -- <function>			
max -- <function>			
min -- <function>			
next -- <function>			
oct -- <function>			
ord -- <function>			
pow -- <function>			
print -- <function>			
repr -- <function>			
round -- <function>			
sorted -- <function>			

Built-ins	math	random	keywords
sum -- <function>			
BaseException -- <class 'BaseException'>			
ArithmeticError -- <class 'ArithmeticError'>			
AssertionError -- <class 'AssertionError'>			
AttributeError -- <class 'AttributeError'>			
EOFError -- <class 'EOFError'>			
Exception -- <class 'Exception'>			
GeneratorExit -- <class 'GeneratorExit'>			
ImportError -- <class 'ImportError'>			
IndentationError -- <class 'IndentationError'>			
IndexError -- <class 'IndexError'>			
KeyboardInterrupt -- <class 'KeyboardInterrupt'>			
ReloadException -- <class 'ReloadException'>			
KeyError -- <class 'KeyError'>			
LookupError -- <class 'LookupError'>			
MemoryError -- <class 'MemoryError'>			
NameError -- <class 'NameError'>			
NotImplementedError -- <class 'NotImplementedError'>			
OSError -- <class 'OSError'>			
OverflowError -- <class 'OverflowError'>			

Built-ins	math	random	keywords
RuntimeError -- <class 'RuntimeError'>			
StopIteration -- <class 'StopIteration'>			
SyntaxError -- <class 'SyntaxError'>			
SystemExit -- <class 'SystemExit'>			
TypeError -- <class 'TypeError'>			
UnicodeError -- <class 'UnicodeError'>			
ValueError -- <class 'ValueError'>			
ZeroDivisionError -- <class 'ZeroDivisionError'>			
help -- <function>			
input -- <function>			
open -- <function>			

.

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.