Programação Python para a calculadora gráfica TI-84 Plus CE-T *Python Edition*

Versão 84CE Bundle 5.6.0.

Saiba mais sobre a Tecnologia TI através da ajuda online em education.ti.com/eguide.

Informações importantes

Exceto se expressamente indicado na Licença fornecida juntamente com um programa, a Texas Instruments renuncia a todas as garantias, quer sejam explícitas ou implícitas, incluindo mas não se limitando às garantias implícitas de comercialização e adequabilidade a um fim específico, relativas a qualquer programa ou livro e disponibiliza apenas estes materiais numa base “como está” A TI não se responsabiliza por qualquer dano indireto, especial ou acidental, relacionado ou decorrente da utilização destes materiais, e a única e exclusiva responsabilidade da Texas Instruments, independentemente da forma de ação, não excederá o montante definido na licença do programa. Além disso, a Texas Instruments não se responsabiliza por qualquer reclamação relacionada com a utilização destes materiais por terceiros.

"Python" e os logótipos Python são marcas comerciais ou marcas comerciais registadas da Python Software Foundation, utilizadas pela Texas Instruments Incorporated com permissão da Foundation.

Índice

O que há de novo?

O que há de novo na aplicação de programação Python v5.5.0

TI-84 Plus CE-T *Python Edition*

Programação Python

TI-84 Plus CE-T *Python Edition*

- Apoia a programação Python utilizando a aplicação Python a partir da versão 5.6.0 do 84CE. Atualize para a mais recente em education.ti.com/84cetupdate.
- Aceda à aplicação Python a partir de [2nd] [apps] ou [prgm] quando a aplicação Python estiver carregada.

Nota: Qual é a dua experiência na calculadora CE para TI-Python?

- TI-84 Plus CE-T *Python Edition* com 84CE Bundle v5.6.0 ou superior

---

Transferir programas Python

Ao transferir programas Python de uma plataforma não TI para uma plataforma TI OU de um produto TI para outro:

- Os programas Python que utilizam funcionalidades da linguagem central e as bibliotecas padrão (math, random, etc.) podem ser portados sem alterações. **Nota:** Cada lista tem no máximo 100 elementos.
- Os programas que utilizam bibliotecas específicas da plataforma matplotlib (for PC), ti_plotlib, ti_system/ti_hub/etc. para plataformas TI, necessitam de ser editados antes de serem executados numa plataforma diferente.

Isto pode aplicar-se mesmo entre plataformas TI.

Funções e módulos TI-Python novos

- Suporta tipo de números complexos como a+bj.
  - Consulte o menu [Fns…] Types a partir do Editor ou do Shell (Interpretador).
- módulo time
- Módulos TI
  - ti_system
    - Recupere lista de SO e equação de regressão de SO num programa Python. Crie listas num programa Python e armazene em variáveis de lista de SO. Cada lista tem no máximo 100 elementos.
  - ti_plotlib
    - Execute programas Python para fazer gráficos estatísticos e funcionais.
  - ti_hub
    - Crie programas TI-Innovator™ Hub Python.
  - ti_rover
    - Controle o TI-Innovator™ Rover utilizando programação Python.

Crie "New" "Types" de programas com modelos

Quando o seu programa requer instruções de importação necessárias para módulos, utilize o separador Types quando criar um Novo programa. As linhas essenciais do programa serão pré-coladas ao seu novo programa no Editor. Isto é especialmente útil para as atividades STEM! O modelo do método de desenho suporta a primeira experiência com a escrita de um programa utilizando ti_plotlib.

Auxiliares de argumento e sugestões do ecrã de menu

Uma ajuda de argumentos peremitir-lhe-á selecionar o argumento correto de um menu quando os métodos contêm argumentos em cadeia. Sem escrever! Sem necessidade de procurar a cadeia correta!

As sugestões do ecrã de menu são fornecidas com intervalos de argumentos, predefinições ou sugestões de teclas.

Atualizações do teclado da aplicação Python

math continua a exibir todos os módulos disponíveis.

2nd [i] (acima de [.]) exibe j imaginário para o número complexo Python a+bj.

Consulte também: Teclado

Informação do software

TI Connect™ CE

Suporte de conectividade e conversão *.py <> PY AppVar para a TI-84 Plus CE-T *Python Edition*.

TI-SmartView™ CE-T

O emulador TI-84 Plus CE-T *Python Edition* suporta Python App v5.5.0

Os programas exemplo HELLO, GRAPH e LINREGR são carregados na instalação e reposição.

O assistente de importação de dados converte ficheiros *csv devidamente formatados em listas de calculadora para o emulador CE. Esta funcionalidade é útil quando se utiliza o módulo ti_system e dados externos para a programação Python.

- Se os números decimais forem representados com o uso de uma vírgula no ficheiro *.csv, o ficheiro não será convertido utilizando o assistente de importação de dados. Verifique a formatação dos números do sistema operativo do seu computador e converta o *.csv para utilizar a representação de pontos decimais. A edição de listas e matrizes na calculadoras CE utiliza o formato numérico como, por exemplo, 12.34 e não 12,34.

Nota: Para executar os programas TI-Innovator™ Hub ou TI-Innovator™ Rover, envie os programas para a calculadora utilizando o TI Connect™ CE. Saia da aplicação Python antes da transferência do explorador do emulador para o computador e depois para a calculadora.
Os programas TI-Innovator™ Hub e TI-Innovator™ Rover não são executados a partir do
TI-SmartView™ CE-T.

Para mais informações sobre a funcionalidade nova e atualizada, consulte education.ti.com/84cetupdate.

Aplicação Python

Para utilizar, navegar e executar a aplicação Python, veja o seguinte.

- Utilizar a aplicação Python
- Navegação na aplicação Python
- Atividade de exemplo

Utilizar a aplicação Python

A aplicação Python está disponível para a TI-84 Plus CE-T *Python Edition*. A informação neste eGuia é para a utilização com a TI-84 Plus CE-T *Python Edition* atualizada com o CE Bundle mais recente.

Quando executar a aplicação Python pela primeira vez na sua TI-84 Plus CE-T *Python Edition*, a aplicação pode direcioná-lo para a atualização do CE Bundle mais recente para a aplicação Python.
Consulte education.ti.com/84cetupdate para atualizar a sua TI-84 Plus CE-T *Python Edition*.

A aplicação Python oferece um gestor de ficheiros, um editor para criar programas e um Shell (Interpretador) para executar programas e interagir com o interpretador Python. Os programas Python guardados ou criados como Python AppVars serão executados a partir da RAM. Arquive Python AppVars através do ecrã de gestão de memória do SO para auxiliar na gestão de armazenamento de ficheiros Python.

Nota: Se a sua calculadora for a TI-84 Plus CE-T, consulte education.ti.com/84cetupdate para encontrar a informação mais recente para o seu CE.

Navegação na aplicação Python

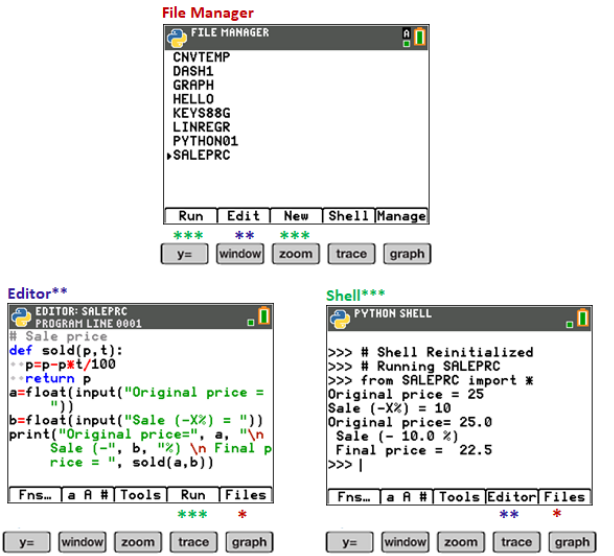
Utilize as teclas de atalho no ecrã da aplicação para navegar entre as áreas de trabalho na aplicação Python. Na imagem, as etiquetas do separador de atalho indicam:

* Navegação para o File Manager [Files]

** Navegação para o Editor [Edit] ou [Editor]

*** Navegação para o Shell (Interpretador) [Shell]

Aceda aos separadores de atalho no ecrã utilizando a linha de teclas gráficas imediatamente abaixo do ecrã. Consulte também Teclado. O menu Editor>Tools menu e o menu Shell>Tools também contêm ações de navegação.



Atividade de exemplo

Utilize o exemplo de atividade fornecido como uma experiência para se familiarizar com as áreas de trabalho na aplicação Python.

- Crie um novo programa a partir do Gestor de ficheiros
- Escreva o programa no Editor
- Execute o programa no Shell (Interpretador) na aplicação Python.

Para mais informações sobre programação Python na sua CE, consulte os recursos para a
TI-84 Plus CE-T *Python Edition*.

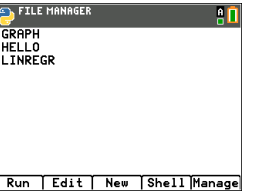
Como começar:

- Execute a aplicação Python.

Nota: Os ecrãs reais podem variar ligeiramente das imagens fornecidas.

Introduza o nome do novo programa a partir do Gestor de ficheiros.

- Prima [zoom] ([New]) para criar um novo programa.



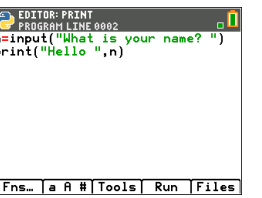
Introdução do nome do novo ficheiro

- O programa exemplo terá o nome de "PRINT". Introduza o nome do programa e prima [graph] ([Ok]).
- Note que o cursor está em ALPHA lock. Introduza sempre o nome do programa seguindo o requisitos dados no ecrã.

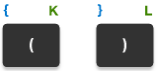
Sugestão: Se o cursor não estiver em ALPHA lock, prima [2nd] [alpha] [alpha] para letras maiúsculas.

Introduza o programa conforme indicado.

Sugestão: A aplicação permite introdução rápida! Observe sempre o estado do cursor quando introduz o programa!



Carateres do alfabeto no teclado  	[alpha] alterna o estado do cursor de introdução no Editor e no Shell (Interpretador). _ não alfabético a alfabeto em

	minúsculas A alfabeto em maiúsculas
Onde está o sinal de igual?	Prima [sto→] quando o cursor for _. 
Onde é que estes estão? input() print()	[Fns…] I/O 1:print() 2:input()
Onde estão as aspas duplas?	[alpha] ["] 
Onde estão (e)?	Utilize o teclado quando o cursor for _. 

Experimente! [a A #] e [2nd] [catalog] são também auxiliares para introdução rápida quando necessário!

Executar o programa PRINT

- No Editor, prima [trace] ([Run]) para executar o seu programa no Shell (Interpretador).
- Introduza o seu nome no prompt "What is your name?".
- É exibido "HELLO" com o seu nome.

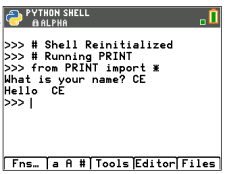
Nota: No prompt do Shell (Interpretador) >>>, pode executar um comando, como 2+3. Se utilizar qualquer método a partir de módulos math, random ou outros disponíveis, execute primeiro uma instrução do módulo import, como em qualquer ambiente de codificação Python.

Cursor do Shell (Interpretador)

indicador de estado.

Introduza o seu nome.

É exibido o resultado de PRINT.



Configurar uma sessão Python com os seus programas

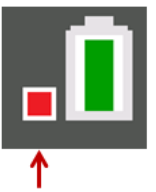
Quando a aplicação Python for lançada, a conexão CE com a experiência TI-Python será sincronizada para a sua sessão Python atual. Irá ver a sua lista de programas em RAM e módulos dinâmicos, à medida que estes se sincronizam com a experiência Python.

Quando a sessão Python for estabelecida, a barra de estado contém um indicador quadrado verde perto do ícone da bateria que sinaliza que a sessão Python está pronta para ser utilizada. Caso o indicador esteja vermelho, aguarde que o indicador mude novamente para verde quando a experiência Python estiver novamente disponível.

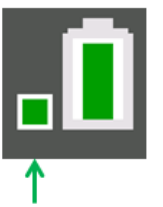
Pode ver uma atualização da distribuição Python ao lançar a aplicação Python, juntamente com a sincronização do programa após a última atualização para a sua TI-84 Plus CE-T *Python Edition* a partir de education.ti.com/84cetupdate.

Desconectar e voltar a conectar a aplicação Python

Quando a aplicação Python estiver a ser executada, a barra de estado contém um indicador que sinaliza se a Python está pronta a ser utilizada. Até a conexão ser estabelecida, o teclado CE não responde. A melhor prática é estar atento ao indicador de conexão da barra de estado durante a sua sessão Python.



Python não pronta



Python pronta

Capturas de ecrãs

Utilizando o TI Connect™ CE em education.ti.com/84cetupdate, é permitido fazer capturas de ecrãs de qualquer ecrã da aplicação Python.

Áreas de trabalho Python

A aplicação Python contém três áreas de trabalho para o desenvolvimento da sua programação Python.

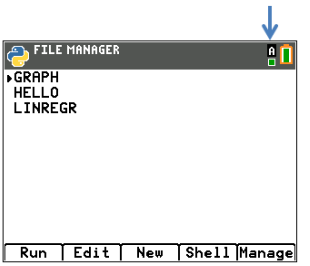
- File Manager
- Editor
- Shell (Interpretador)

Python File Manager

O File Manager lista as Python AppVars disponíveis na RAM da calculadora. Pode criar, editar e executar programas, assim como navegar até ao Shell (Interpretador).

No estado alfabético, prima qualquer letra no teclado para saltar para programas que comecem com essa letra.

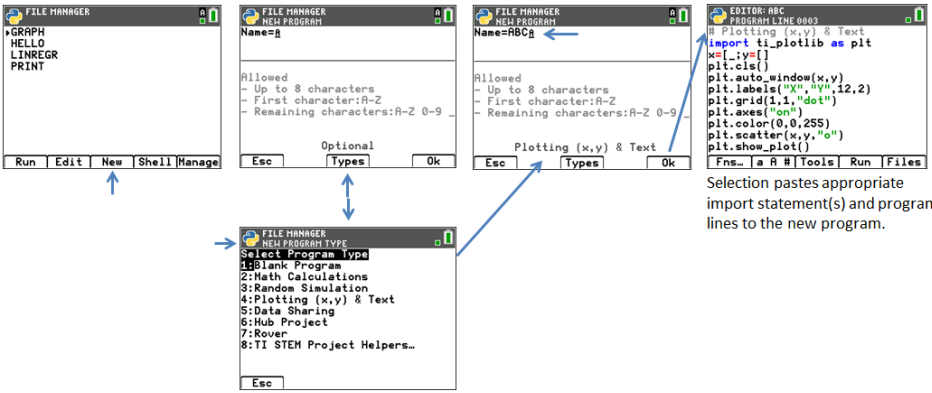
Prima alpha, se necessário, quando o indicador **A** não estiver na barra de estado.



Teclas de atalho e menus do File Manager		
Menus	**Tecla premida**	**Descrição**
[Run]	y=	Selecione um programa utilizando ▲ ou ▼. A seguir, selecione [Run] para executar o seu programa.
[Edit]	window	Selecione um programa utilizando ▲ ou ▼. A seguir, selecione [Edit] para exibir o programa no Editor para editar o programa.
[New]	zoom	Selecione [New] para introduzir um novo nome de programa e continue para o Editor para introduzir o seu novo programa. No ecrã [New], selecione [Types] (prima [zoom]) para selecionar um Tipo de programa. Ao selecionar um tipo de programa, um modelo de instruções de importação e funções e métodos frequentemente utilizados serão colados ao seu novo programa para essa atividade.
[Shell]	trace	Selecione [Shell] para exibir o prompt do Shell (Interpretador Python). O Shell (Interpretador) irá estar no estado atual.
[Manage]	graph	Selecione [Manage] para: • Ver o número de versão. • Replicar, apagar ou mudar nome de um

Teclas de atalho e menus do File Manager		
Menus	**Tecla premida**	**Descrição**
		programa selecionado. • Ver o ecrã About. • Sair da aplicação. Utilize também [2nd] [quit]

Criar um novo programa utilizando modelos de tipo de programa

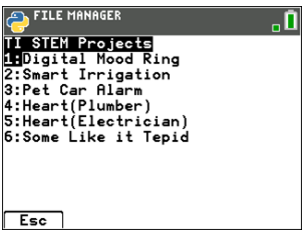


Selection pastes appropriate import statement(s) and program lines to the new program.

- Select [Types] to display Select Program Type menu.
- Import(s) displays on status bar.

Criar um novo programa de atividade STEM utilizando modelos

Quando a TISTEMEN AppVar está carregada para o arquivo, o item menu "TI STEM Project Helpers..." é exibido no menu Select Program Type. Selecione o modelo de atividade STEM, se necessário, para ajudar a iniciar um novo programa STEM.



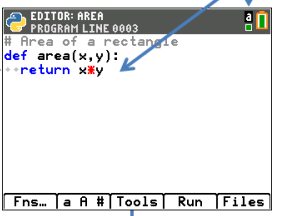
Python Editor

O Python Editor é exibido a partir de um programa selecionado no File Manager ou a partir do Shell (Interpretador). O Editor exibe palavras-chave, operadores, comentários, cadeias e indentações em cor. Estão disponíveis uma colagem rápida de palavras-chave e funções Python comuns, bem como uma introdução direta pelo teclado e uma introdução de carateres [a A #]. Ao colar um bloco de código, como if.. elif.. else, o Editor oferece indentação automática que pode ser alterada, conforme necessário, à medida que vai escrevendo o programa.

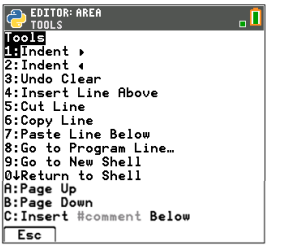
O cursor está sempre em modo de introdução. Utilize 2nd e alpha para alternar o cursor. Os estados do cursor são numérico, a, e A. [del] comporta-se como retrocesso e apaga um caráter.

Localização do cursor na linha do programa.

Indentação automática de blocos de código. Pontos cinzentos como indicador visual de linhas com indentação.



Ferramentas úteis para editar e trabalhar no Shell (Interpretador). Descrição completa em baixo.



Teclas de atalho e menus do Python Editor		
Menus	**Tecla premida**	**Descrição**
[Fns…]	y=	Selecione [Fns…] para aceder os menus das funções mais usadas, palavras-chave e operadores. Também acede a conteúdos selecionados nos módulos math e random. **Nota:** 2nd [catalog] também é útil para colagem rápida.

<table>
<tr><th colspan="4">Teclas de atalho e menus do Python Editor</th></tr>
<tr><th>Menus</th><th>Tecla premida</th><th colspan="2">Descrição</th></tr>
<tr><td>[a A #]</td><td>window</td><td colspan="2">Selecione [a A #] para aceder a uma palete de carateres como forma alternativa de introduzir muitos carateres.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td colspan="2">Selecione [Tools] para aceder a funcionalidades para o ajudar da edição ou na sua interação com o Shell (Interpretador).</td></tr>
<tr><td></td><td></td><td>1: Indentação ▶</td><td>Indenta a linha do programa para a direita, o cursor desloca-se para o primeiro caráter da linha.</td></tr>
<tr><td></td><td></td><td>2: Indentação ◀</td><td>Reduz a indentação da linha do programa para a esquerda. O cursor desloca-se para o primeiro caráter da linha.</td></tr>
<tr><td></td><td></td><td>3: Anular limpar</td><td>Cola a última linha limpa para uma linha nova por baixo da linha de programa que contém o cursor. O cursor apresenta-se no fim da linha colada.</td></tr>
<tr><td></td><td></td><td>4: Inserir linha acima</td><td>Introduz uma linha por cima da linha de programa com o cursor. A linha irá indentar e exibir pontos de indentação quando apropriado.</td></tr>
<tr><td></td><td></td><td>5: Cortar linha</td><td>A linha atual do programa com o cursor está cortada. O cursor é exibido na linha de programa abaixo da linha de corte.</td></tr>
<tr><td></td><td></td><td>6: Copiar linha</td><td>Copia a linha do programa atual com o cursor. Uma linha de programa copiada pode ser colada no prompt do Shell (Interpretador). Ver Shell (Interpretador) em baixo.</td></tr>
<tr><td></td><td></td><td>7: Colar linha em baixo</td><td>Cola a última linha de programa guardada na linha por baixo da posição do cursor.</td></tr>
<tr><td></td><td></td><td>8: Go to Program Line...</td><td>Exibe o cursor no início da linha de programa especificada.</td></tr>
<tr><td></td><td></td><td>9: Ir para Novo Shell (Interpretador)</td><td>Exibe o Shell (Interpretador) reinicializado.</td></tr>
<tr><td></td><td></td><td>0: Regressar a</td><td>Exibe o Shell (Interpretador) no</td></tr>
</table>

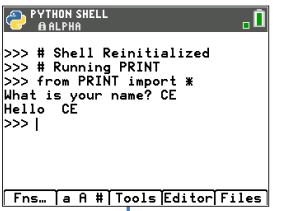
<table>
<tr><th colspan="3">Teclas de atalho e menus do Python Editor</th></tr>
<tr><th>Menus</th><th>Tecla premida</th><th>Descrição</th></tr>
<tr><td></td><td></td><td><table>
<tr><td>Shell (Interpretador)</td><td>estado atual.</td></tr>
<tr><td>A: Subir uma página</td><td>Exibe 11 linhas de programa acima da posição atual do cursor, conforme disponível.</td></tr>
<tr><td>B: Descer uma página</td><td>Exibe 11 linhas de programa abaixo da posição atual do cursor, conforme disponível.</td></tr>
<tr><td>C: Inserir #comment abaixo</td><td>Insere # numa linha nova por baixo da posição do cursor.</td></tr>
</table></td></tr>
<tr><td>[Run]</td><td>trace</td><td>Selecione [Run] para executar o seu programa.</td></tr>
<tr><td>[Files]</td><td>graph</td><td>Selecione [Files] para exibir o File Manager.</td></tr>
</table>

Shell (Interpretador) Python

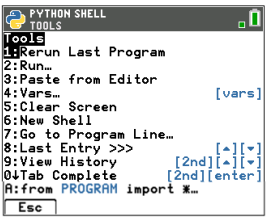
O Shell (Interpretador) Python é a consola onde pode interagir com o interpretador Python ou executar os seus programas Python. Estão disponíveis uma colagem rápida de palavras-chave e funções Python comuns, bem como uma introdução direta pelo teclado e uma introdução de carateres [a A #]. O prompt do Shell (Interpretador) pode ser utilizado para testar uma linha de código colada a partir do Editor. Podem também ser introduzidas e executadas várias linhas de código num prompt do Shell (Interpretador) >>>.

Indicador de estado do cursor do Shell (Interpretador).

O Shell (Interpretador) reinicia quando um novo programa é executado.



Ferramentas úteis para trabalhar no Shell (Interpretador). Consulte os detalhes abaixo.



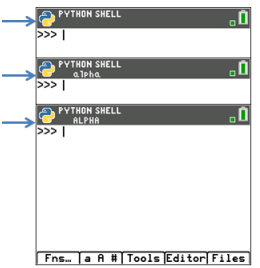
Estados do cursor do Shell (Interpretador).

não alfabético

2nd alpha se necessário para alternar

alpha alfa

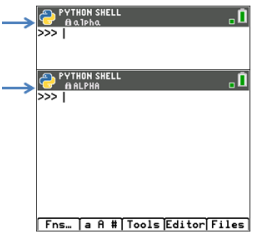
alpha novamente ALFA



2nd alpha

alpha bloqueado

alpha novamente ALPHA bloqueado



<table>
<tr><th colspan="3">Teclas de atalho e menus do Shell (Interpretador) Python</th></tr>
<tr><th>Menus</th><th>Tecla premida</th><th>Descrição</th></tr>
<tr><td>[Fns…]</td><td>y=</td><td>Selecione [Fns…] para aceder os menus das funções mais usadas, palavras-chave e operadores. Também acede a conteúdos selecionados nos módulos math e random.
Nota: 2nd [catalog] também é útil para colagem rápida.</td></tr>
<tr><td>[a A #]</td><td>window</td><td>Selecione [a A #] para aceder a uma palete de carateres como forma alternativa de introduzir muitos carateres.</td></tr>
<tr><td>[Tools]</td><td>zoom</td><td>Selecione [Tools] para exibir os seguintes itens de menu.
<table>
<tr><td>1: Rerun last program</td><td>Executa novamente o último programa que foi executado no Shell (Interpretador).</td></tr>
<tr><td>2: Run…</td><td>Exibe uma lista dos programas Python disponíveis para execução no Shell (Interpretador).</td></tr>
<tr><td>3: Paste from Editor</td><td>Cola a última linha de programa copiada do Editor para o prompt do Shell (Interpretador).</td></tr>
<tr><td>4: Vars…</td><td>Mostra os vars do último programa que foi executado. Não exibe vars definidas de programa de um programa importado.</td></tr>
<tr><td>5: Clear Screen</td><td>Apaga o ecrã do Shell (Interpretador). Não reinicia um novo Shell (Interpretador).</td></tr>
<tr><td>6: New Shell</td><td>Reinicia um novo Shell (Interpretador).</td></tr>
<tr><td>7: Go to Program Line…</td><td>Exibe o Editor a partir do Shell (Interpretador) com o cursor sobre a linha de programa especificada.</td></tr>
<tr><td>8: Last Entry>>> ▲▼</td><td>Exibe até as últimas 8 entradas no prompt do Shell (Interpretador) durante uma sessão do Shell (Interpretador).</td></tr>
<tr><td>9: View History 2nd ▲ 2nd ▼</td><td>Percorra o ecrã Shell para ver até às últimas 60 linhas de saída no Shell (Interpretador) durante uma sessão do Shell (Interpretador).</td></tr>
<tr><td>0: Tab</td><td>Exibe os nomes das variáveis e</td></tr>
</table></td></tr>
</table>

Teclas de atalho e menus do Shell (Interpretador) Python			
Menus	**Tecla premida**	**Descrição**	
		Complete [2nd] [enter]	funções disponíveis para acesso na atual sessão do Shell (Interpretador). Quando for introduzida uma letra de uma variável ou função disponível, prima [2nd] [enter] para completar automaticamente o nome se estiver disponível uma correspondência na sessão atual do Shell (Interpretador).
		A: from PROGRAM import *…	Quando executado pela primeira vez numa sessão do Shell (Interpretador), o PROGRAMA será executado e as vars só poderão ser visualizadas utilizando o separador Complete. Quando executado novamente na mesma sessão do Shell (Interpretador), a execução aparece como sem execução. Este comando também pode ser colado de [2nd] [catalog].
[Editor]	[trace]	Selecione [Editor] para exibir o Editor com os últimos programas no Editor. Se o Editor estiver vazio, pode exibir o Gestor de ficheiros.	
[Files]	[graph]	Selecione [Files] para exibir o Gestor de ficheiros.	

Nota:

- Para interromper um programa Python em execução, como se um programa estivesse em loop contínuo, prima [on]. Prima **[Tools]** ([zoom]) > **6:New Shell** como método alternativo para parar um programa em execução.
- Quando utilizar o módulo ti_plotlib para traçar a área do gráfico no Shell (Interpretador), prima [clear] para limpar o gráfico e voltar ao prompt do Shell (Interpretador).

Erro de execução: Vá para a linha de programa utilizando Shell >Tools

A experiência TI-Python irá exibir mensagens de erro Python no Shell (Interpretador) quando o código é executado. Se for exibido um erro quando um programa é executado, será exibido um número de linha de programa. Utilize **Shell>Tools 7:Go to Program Line…** Introduza os números da linha e prima **[OK]**. O cursor irá aparecer no primeiro caráter da linha de programa apropriada no Editor. O número da linha do programa é exibido na segunda linha da barra de estado no Editor.

Introduções - teclado, catálogo, mapa de carateres e menus

Sugestões para introduções rápidas

- Teclado
- Catálogo
- Mapa de carateres [a A #]
- [Fns…] Menus

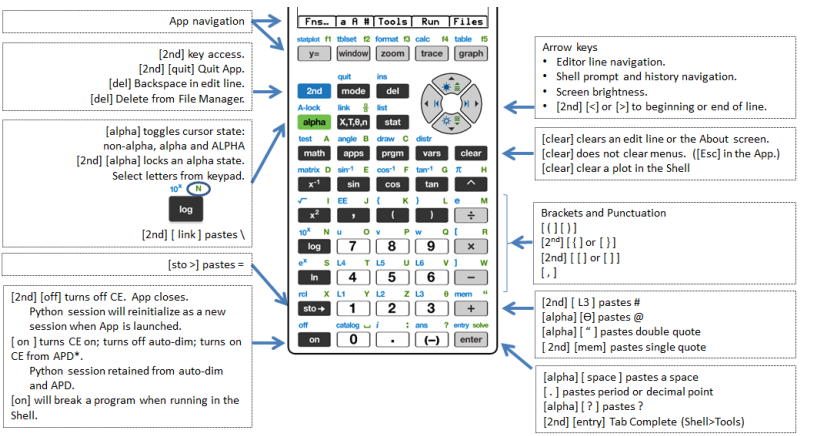
Utilizar os menus Keypad, Catalog, [a A #] e Fns…

Ao introduzir o código no Editor ou no Shell (Interpretador), utilize os seguintes métodos de introdução para colar rapidamente na linha de edição.

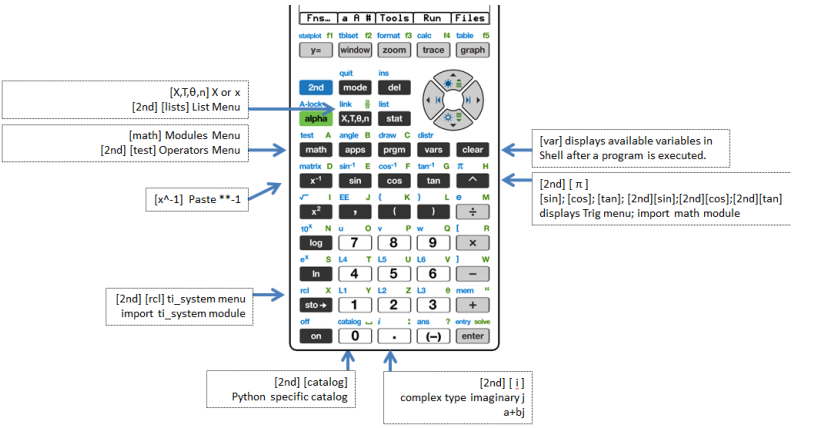
Teclado

Quando a aplicação Python está a ser executada, o teclado foi concebido para colar as operações Python apropriadas ou abrir menus concebidos para facilitar a introdução de funções, palavras-chave, métodos, operadores, etc. Ao premir 2nd e alpha irá aceder à segunda e terceira função numa tecla como no sistema operativo.

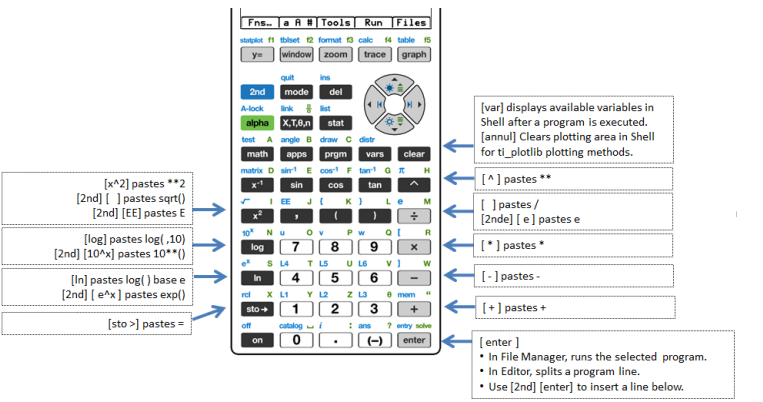
Navegação na aplicação Python, editar e carateres especiais por linha de teclado



Teclas específicas da aplicação Python para menus e funções por linha de teclado



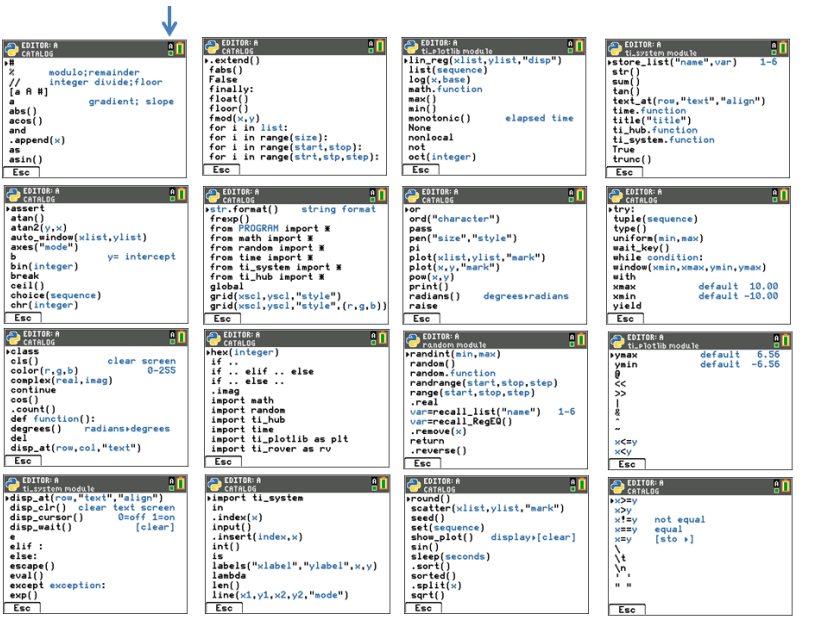
Teclas específicas da aplicação Python para menus e funções por linha de teclado (Continuação)



Catálogo

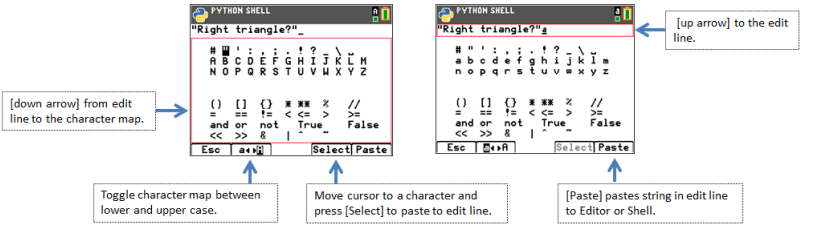
Quando a aplicação Python está a ser executada, 2nd [catalog] exibe uma lista dos delimitadores, palavras-chave, funções e operadores mais frequentes para colar rapidamente para uma linha de edição.
2nd [catalog] está disponível apenas no Editor e no Shell (Interpretador). Para uma descrição mais detalhada de cada item de catálogo, consulte o Guia de referência. A partir do topo do menu do catálogo, utilize ▲ para deslocação circular do catálogo.

Quando estiver no catálogo, selecione alpha e uma tecla de letra para exibir a listagem a iniciar nessa letra.



Mapa de carateres [a A #]

O separador de atalho [a A #] para uma palete de caracteres é uma funcionalidade prática para introduzir cadeias no Editor ou Shell (Interpretador).



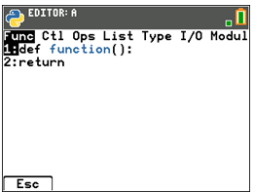
Nota: Quando o foco do cursor estiver na linha de edição [a A #], as teclas do teclado selecionado não estão disponíveis. Quando o foco estiver no mapa de carateres, o teclado está limitado.

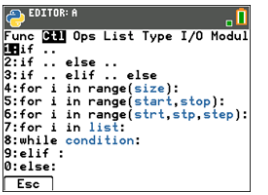
[Fns…] Menus

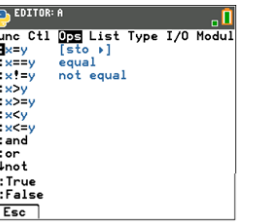
O separador de atalho [Fns…] exibe menus contendo funções, palavras-chave e operadores Python frequentemente utilizados. Os menus também dão acesso às funções e constantes selecionadas a partir dos módulos math e random. Apesar de poder introduzir caráter a caráter a partir do teclado, estes menus proporcionam uma forma rápida de colar no Editor ou Shell (Interpretador). Prima [Fns…] quando estiver no Editor ou no Shell (Interpretador). Ver também Catálogo e Teclado para métodos de introdução alternativos.

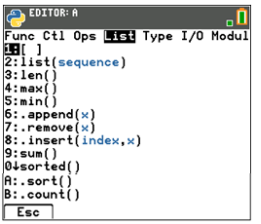
Funções e submenus dos módulos

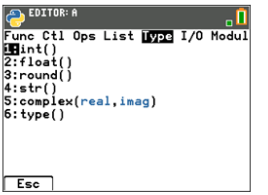
Incorporado, operadores e teclados

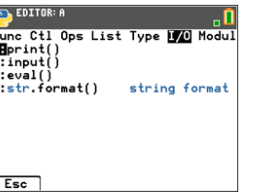












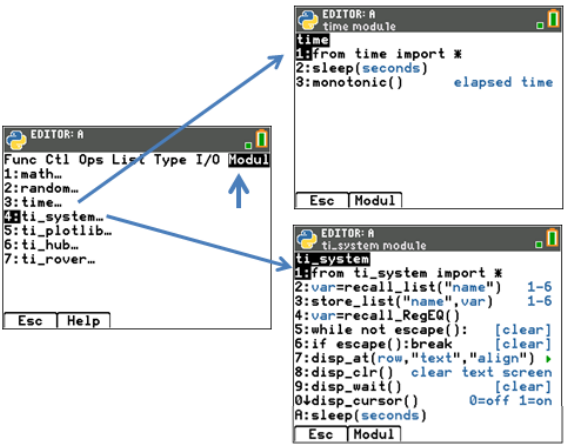
Submenus dos módulos

Ao utilizar uma função ou constante Python a partir de um módulo, utilize sempre uma instrução de importação para indicar a localização do módulo da função, método ou constante.

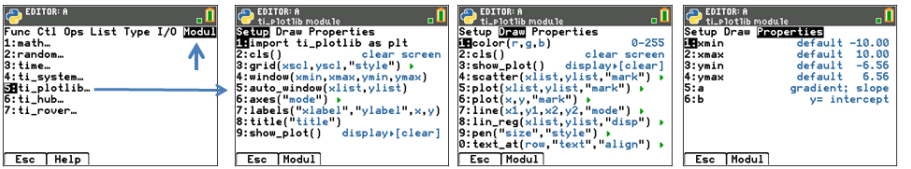
Ver Qual é a experiência de programação Python?

[Fns…]>Modul: módulos math e random

[Fns…]>Modul: módulos time e ti_system



[Fns…]>Modul: ti_plotlib



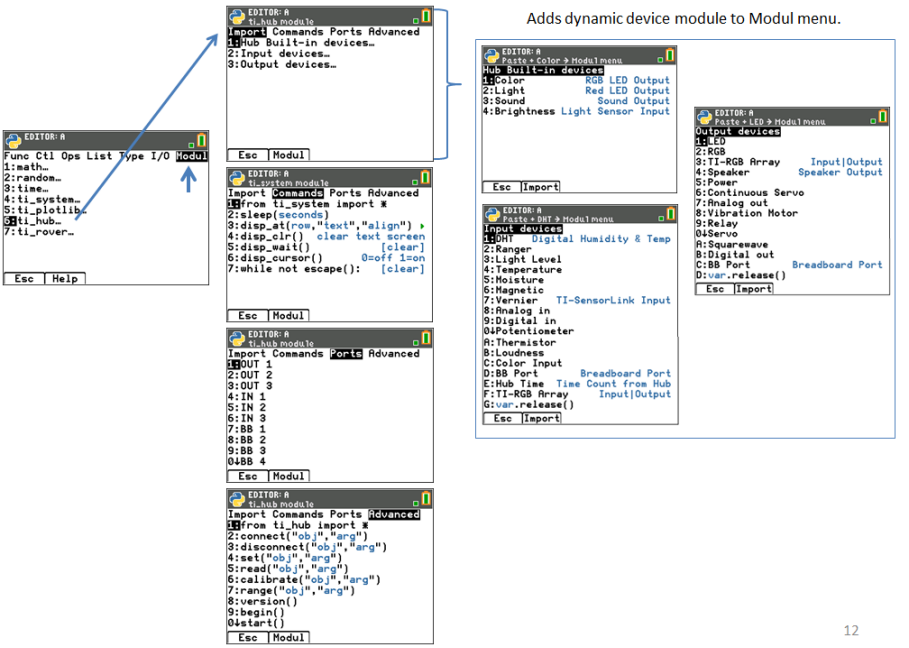
Nota importante para a representação gráfica:

- A ordem das linhas de programa para a representação gráfica deve seguir a ordem como no menu Setup para garantir os resultados esperados.
- O gráfico é exibido quando plt.show_plot() é executado no fim dos objetos de desenho num programa. Para limpar a área de representação gráfica no Shell (Interpretador), prima [clear].
- A execução de um segundo programa que assume que os valores predefinidos são definidos dentro do mesmo ambiente Shell (Interpretador), geralmente resulta em comportamentos inesperados, tais como cor ou outras configurações de argumento predefinidas. Edite programas com valores de argumento esperados ou reinicie o Shell (Interpretador) antes de executar outro programa de representação gráfica.

[Fns…]>Modul: módulo ti_hub

Os métodos ti_hub não estão listados no catálogo e, por isso, não estão listados no guia de referência. Por favor, utilize a informação do ecrã nos menus para detalhes sobre argumentos e predefinições dos argumentos ou valores permitidos. Mais informações sobre programação Python para TI-Innovator™ Hub e TI-Innovator™ Rover estará disponível em education.ti.com.

Nota: O TI-Innovator™ Hub deve estar conectado quando executar os seus programas Python.



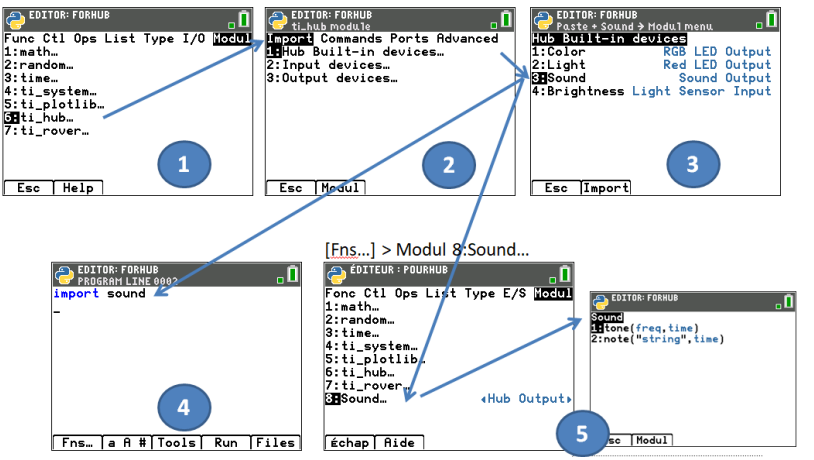
Módulo ti_hub – Adicionar importação ao Editor e adicionar o módulo ti_hub sensor ao menu Modul.

Exemplo do ecrã: Importar som

Para importar métodos de sensor do TI-Innovator™ para o seu programa Python, a partir do Editor,

1. Selecione **[Fns…] > Modul 6:ti_hub**
2. Selecione o menu ti_hub Import. Selecione um tipo de sensor a partir de Built-in, Input e Output.
3. Selecione um sensor.
4. Uma instrução de importação será colada ao Editor e o módulo do sensor estará disponível em **[Fns…] > Modul** quando voltar a esse menu a partir do seu programa.
5. Selecione **[Fns…] > Modul 8:Sound…** para colar os métodos adequados para este sensor«.

[Fns…]>Modul 6:ti_hub



Nota: Brightns é um objeto "built-in" no TI-Innovator Hub.

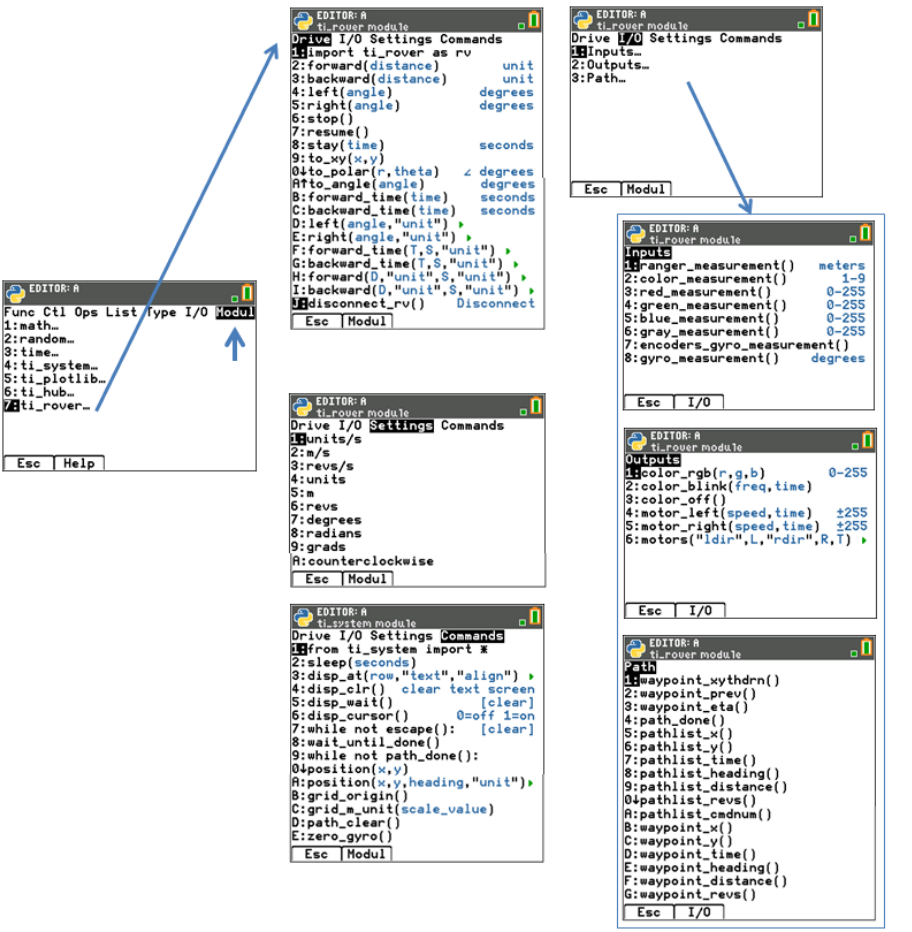
Quando utilizar a instrução 'import brightns', introduza 'brightns.range(0,100)' para garantir o intervalo predefinido correto no início da execução do programa.

Exemplo:

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

[Fns…]>Modul módulo ti_rover

Os métodos ti_rover não estão listados no catálogo e, por isso, não estão listados no guia de referência. Por favor, utilize a informação do ecrã nos menus para detalhes sobre argumentos e predefinições dos argumentos ou valores permitidos. Mais informações sobre programação Python para TI-Innovator™ Hub e TI-Innovator™ Rover estará disponível em education.ti.com.



Notas:

- Na programação TI-Python não necessita de incluir métodos para conectar e desconectar o TI-Innovator™ Rover. Os métodos TI-Innovator™ Rover Python tratam da conexão e desconexão sem métodos adicionais. Isto é um pouco diferente de programar TI-Innovator™ Rover em TI-Basic.

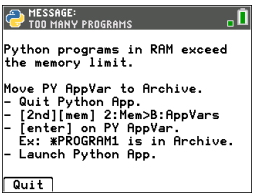
- rv.stop() é executado como uma pausa e depois resume continua com os movimentos Rover em fila. Se for executado outro comando de movimento depois de
  rv.stop(), a fila de movimentos é limpa. Novamente, isto é um pouco diferente de programar TI-Innovator™ Rover em TI-Basic.

Mensagens da aplicação Python

Há várias mensagens que podem ser exibidas enquanto estiver numa sessão Python. Algumas mensagens selecionadas são apresentadas na tabela. Siga as instruções no ecrã e navegue usando [Quit], [Esc] ou [Ok], conforme necessário.

Gestão da memória

A memória disponível para a experiência Python será no máximo de 100 programas Python (PY AppVars) ou 50K de memória. Os módulos que estão incorporados na aplicação nesta versão Python irão partilhar o mesmo espaço com todos os ficheiros.



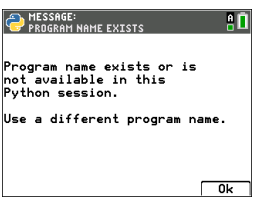
Utilizar [2nd] [quit] para sair da aplicação

Irá ser questionado para se certificar de que pretende sair da aplicação. Sair da aplicação irá parar a sua sessão Python. Quando executar novamente a aplicação Python, os seus programas e módulos Python AppVar irão sincronizar. O Shell (Interpretador) irá reiniciar.

No Gestor de ficheiros, prima del num programa Python selecionado ou selecione a partir de **File Manager>Manage 2:Delete Program...**.

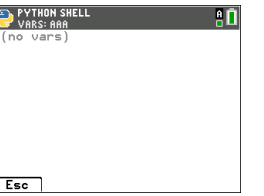
Verá uma caixa de diálogo para apagar ou voltar para o Gestor de ficheiros.

Tentou criar um novo programa Python ou duplicar um programa que já exista na sua CE, seja em RAM ou Arquivo ou desativado para modo de exame. Introduza um nome diferente.



Tentou navegar do Shell (Interpretador) para o Editor, mas o Editor está vazio. Selecione uma opção adequada para o seu trabalho.

Quando executa um programa Python, as variáveis definidas do último programa executado estão listadas no menu **Shell>Tools> 4:Vars…** para utilizar e estão disponíveis para utilização no Shell (Interpretador). Se não forem exibidas variáveis, poderá ter de executar novamente o seu programa.



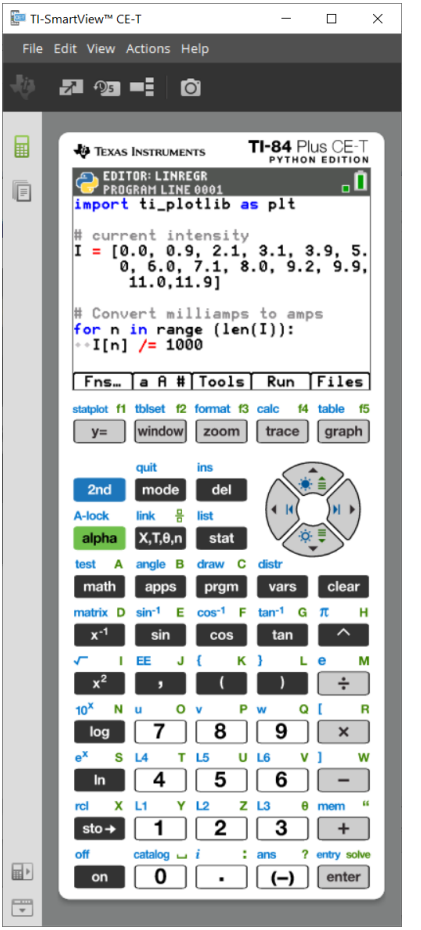
Utilizar o TI-SmartView™ CE-T e a experiência Python

Este guia assume a última atualização do TI-SmartView™ CE-T. Atualize para o TI-SmartView™ CE-T mais recente em education.ti.com/84cetupdate.

A atualização inclui o SO emulador TI-84 Plus CE-T *Python Edition* mais recente a executar a aplicação Python mais recente. Os módulos atualizados de time, ti_system, ti_plotlib, ti_rover* e ti_hub* estão incluídos.

Execute a aplicação Python no emulador TI-84 Plus CE-T *Python Edition*.

- A aplicação Python App oferece
  - Gestor de ficheiros
  - Editor
  - Execução do seu programa Python no Shell (Interpretador)*



Programas Hub/Rover

- Crie programas ti_hub/ti_rover Python no emulador CE que esteja a executar a aplicação Python.

  *** Nota:** Não há conectividade entre o TI-SmartView™ CE e o TI-Innovator™ Hub ou o TI-Innovator™ Rover. Os programas podem ser criados e depois executados na calculadora CE.

- Saia da aplicação Python para preparar a transferência da(s) Python AppVar(s) do emulador. Para o passo seguinte, o emulador não deve "estar ocupado" a executar uma aplicação ou programa.

- Mude para a área de trabalho do explorador do emulador e envie o(s) programa(s) para o computador.
- Utilize o TI Connect™ CE para enviar Python AppVars do computador para a calculadora CE para a experiência TI-Innovator™ Hub/TI-Innovator™ Rover.

Nota: Para interromper um programa Python em execução no Shell (Interpretador), como se um programa estivesse em loop contínuo, prima **[on]**. Prima **[Tools] [zoom] > 6:New Shell** como método alternativo para parar um programa em execução.

Lembrete: Para qualquer computador/experiência TI-Python: Depois de criar um programa Python num ambiente de desenvolvimento Python no computador, confirme se o seu programa é executado na calculadora/emulador na experiência TI-Python. Altere o programa conforme necessário.

Teclado remoto da aplicação SmartPad CE

- Ao executar a aplicação SmartPad CE App na sua CE conectada, irá comportar-se como um teclado remoto incluindo o mapeamento especial de *teclado* oferecido quando a aplicação Python está a ser executada.

Área de trabalho do explorador do emulador

- Saia da aplicação Python para que o emulador não esteja ocupado quando aceder a todas as funcionalidades da área de trabalho do explorador do emulador.
- As conversões program.py < > PY AppVar são permitidas. Isto é semelhante à experiência
  TI Connect™ CE ao enviar programas para a calculadora CE conectada.
- Quando enviar um ficheiro program.py criado noutro ambiente Python, o seu PY AppVar terá de ser editado para ser executado como esperado no TI-Python. Use o editor da aplicação Python para alterar conforme necessário para os módulos únicos, tais como
  ti_plotlib, ti_system, ti_hub e ti_rover.

Assistente de importação de dados

- Os ficheiros *.csv de dados, formatados como indicado na caixa de diálogo do assistente, irão converter os dados em variáveis de listagem CE. Os métodos em ti_system podem então ser utilizados para partilhar listas entre o emulador CE OS e a aplicação Python. Esta funcionalidade é semelhante à do assistente de importação de dados no TI Connect™ CE.
- Se os números decimais forem representados com o uso de uma vírgula no ficheiro *.csv, o ficheiro não será convertido utilizando o assistente de importação de dados. Verifique a formatação dos números do sistema operativo do seu computador e converta o *.csv para utilizar a representação de pontos decimais. A edição de listas e matrizes na calculadoras CE utiliza o formato numérico como, por exemplo, 12.34 e não 12,34.

Utilizar o TI Connect™ CE para converter programas Python

Atualize para o TI Connect™ CE para as funcionalidades mais recentes, incluindo converter programas *.py para uma PY AppVar como o formado de ficheiro da calculadora CE.

Consulte o e-Guia da TI-84 Plus CE-T para mais informações sobre a calculadora CE, o TI-SmartView™ CE-T e o TI Connect CE.

Qual é a experiência de programação Python?

A TI-Python é baseada na CircuitPython, uma variante de Python concebida para ser instalada em pequenos microcontroladores. A implementação original do CircuitPython foi adaptada para ser utilizada pela TI.

O armazenamento interno de números para cálculo nesta variante da Circuit Python está em flutuações binárias de precisão limitada e, portanto, não pode representar exatamente todos os valores decimais possíveis. As diferenças em relação às representações decimais reais que surgem ao guardar estes valores podem conduzir a resultados inesperados em cálculos subsequentes.

- **Para números de ponto flutuante** - Exibe até 16 dígitos significantes de precisão. Internamente, os valores são armazenados utilizando 53 bits de precisão, o que equivale aproximadamente a 15-16 dígitos decimais.
- **Para números inteiros** - O tamanho dos números inteiros está limitado apenas pela memória disponível na altura em que os cálculos são realizados.

Módulos incluídos na TI-84 Plus CE-T Python Edition

- Planos integrados
- módulo math
- módulo random
- time
- ti_system
- ti_plotlib
- ti_hub
- ti_rover

Nota: Se tiver programas Python existentes criados noutros ambientes de desenvolvimento Python, edite o(s) seu(s) programa(s) para a solução TI-Python. Os módulos podem utilizar diferentes métodos, argumentos e ordenação de métodos num programa em comparação com o módulos ti_system, ti_plotlib, ti_hub e ti_rover. De um modo geral, tenha consciência da compatibilidade ao utilizar qualquer versão do Python e dos módulos Python.

Ao transferir programas Python de uma plataforma não TI para uma plataforma TI OU de um produto TI para outro:

- Os programas Python que utilizam funcionalidades da linguagem central e as bibliotecas padrão (math, random, etc.) podem ser portados sem alterações.

  Nota: Cada lista tem no máximo 100 elementos.

- Os programas que utilizam bibliotecas específicas da plataforma matplotlib (for PC), ti_plotlib,

ti_system, ti_hub, etc. para plataformas TI, necessitam de ser editados antes de serem executados numa plataforma diferente.

- Isto pode aplicar-se mesmo entre plataformas TI.

Como em qualquer versão do Python, terá de incluir importações como, a partir de math import *, para utilizar quaisquer funções, métodos ou constantes contidas no módulo math. Para um exemplo, para executar a função cos(), utilize importar para importar o módulo math.

Consulte Lista do CATÁLOGO.

Exemplo:

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

Exemplo alternativo:

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

Os módulos disponíveis podem ser apresentados no Shell (Interpretador) utilizando o seguinte comando

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

O conteúdo dos módulos pode ser visualizado no Shell (Interpretador) conforme exibido utilizando “import module” e “dir(module)”.

Nem todo o conteúdo dos módulos aparece nos menus de colagem rápida, como [Fns…] ou 2nd [catalog].

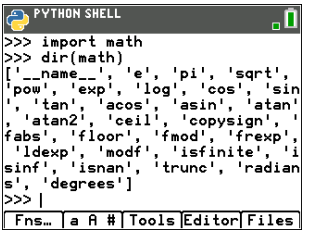
Conteúdo dos módulos e palavras-chave selecionados

Para a lista de módulos incluídos nesta versão, consulte:

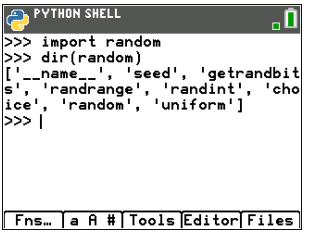
Anexo: Conteúdo do módulo, palavras-chave e planos integrados da TI-Python

Lembrete: Para qualquer computador/experiência TI-Python: Depois de criar um programa Python no computador, confirme se o seu programa é executado na calculadora na experiência TI-Python. Altere o programa conforme necessário.

Estes ecrãs apresentam o conteúdo dos módulos para math e random.

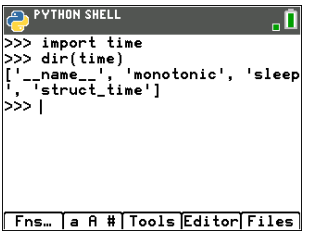


módulo math

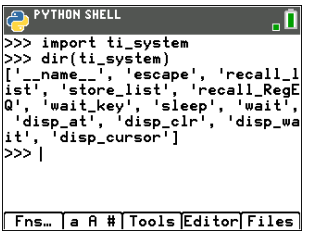


módulo random

Estes ecrãs apresentam o conteúdo dos módulos para time e ti_system.

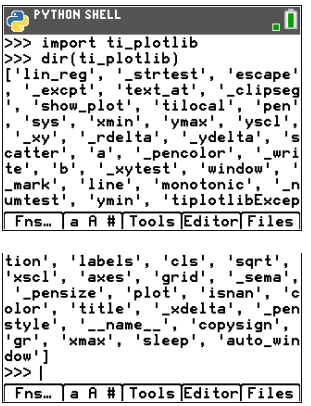


time



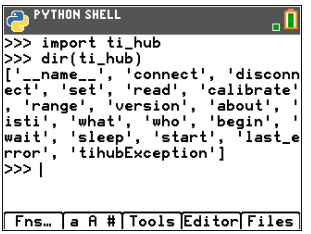
ti_system

Estes ecrãs apresentam o conteúdo do módulo para ti_plotlib.



ti_plotlib

Este ecrã apresenta o conteúdo do módulo para ti_hub.



ti_hub

Estes ecrãs apresentam o conteúdo do módulo para ti_rover.

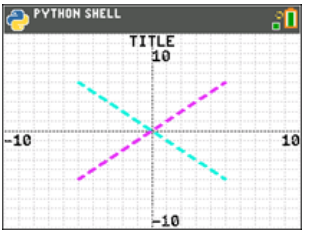
ti_rover

Programas exemplo:

Utilize os seguintes programas exemplo para se familiarizar com os métodos da secção de referência. Estes exemplos contêm também alguns programas TI-Innovator™ Hub e
TI-Innovator Rover™ para o ajudar a começar com o 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()
```



Prima clear para exibir o prompt do Shell (Interpretador).

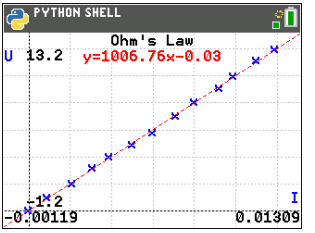
REGEQ1

Configure uma equação de regressão antes de executar o programa Python na aplicação Python. Um exemplo seria, em primeiro lugar, introduzir duas listas no SO CE. Depois, por exemplo, calcule [stat] CALC 4:LinReg(ax+b) para as suas listas. Isto guarda a equação de regressão para RegEQ no SO. Aqui está um programa para recuperar RegEQ para a experiência Python.

```
# Exemplo de recall_RegEQ()
from ti_system import *

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

LINREGR (fornecido no CE Bundle)

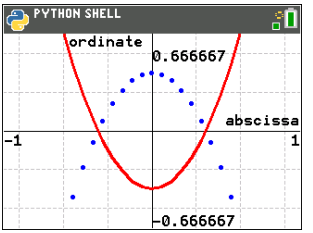
```
import ti_plotlib as plt

# current intensity
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]

# voltage
n in range (len(I)):
I[n] /= 1000

# la 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("Ohm's Law")
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))
```



Prima clear para exibir o prompt do Shell (Interpretador).

GRAPH (fornecido no CE Bundle)

```
import ti_plotlib as plt
#Depois de executar o programa, prima [clear] para limpar a
representação gráfica e regressar ao Shell (Interpretador).

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

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

def plot(res,xmin,xmax):
  #setup plotting area
  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")

# plot f(x) and g(x)
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(resolution,xmin,xmax)
plot(30,-1,1)
# Criar um gráfico com parameters(resolution,xmin,xmax)
# Depois de limpar o primeiro gráfico, prima a tecla [var]. A função
plot() permite-lhe alterar as definições de exibição
(resolution,xmin,xmax).
```

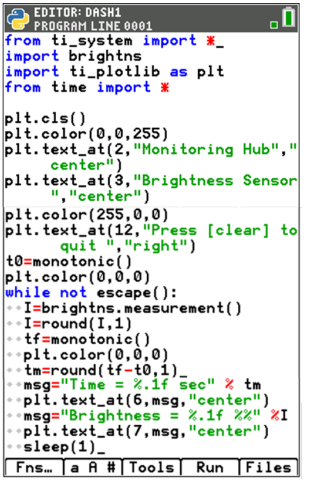


Prima clear para exibir o prompt do Shell (Interpretador).

DASH1 – Programa amostra de TI-Innovator™ Hub

Consultar: [Fns…]>Modul: módulo ti_hub

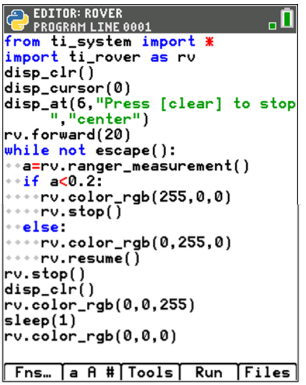
```
from ti_system import *
import brightns
import ti_plotlib as plt
from time import *
plt.cls()
plt.color(0,0,255)
plt.text_at(2,"Monitoring Hub","center")
plt.text_at(3,"Brightness Sensor","center")
plt.color(255,0,0)
plt.text_at(12,"Press [clear] to quit ","right")
t0=monotonic()
plt.color(0,0,0)
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 – Programa amostra de TI-Innovator™ Rover

Consultar: [Fns…]>Modul módulo ti_rover

```
from ti_system import *
import ti_rover as rv
disp_clr()
disp_cursor(0)
disp_at(6,"Press [clear] 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)
```

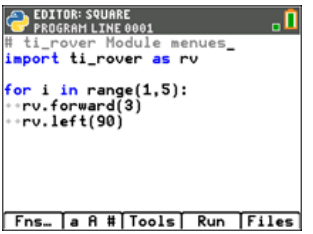


BLNKSND - Programa exemplo para o TI-Innovator™ Hub

Consultar: [Fns…]>Modul: módulo ti_hub

SQUARE - Programa exemplo para o TI-Innovator™ Rover

Consultar: [Fns…]>Modul módulo ti_rover



Guia de referência para a experiência TI-Python

A aplicação Python contém menus de funções, classes, controlos, operadores e palavras-chave para colagem rápida no Editor ou no Shell (Interpretador). A tabela de referência seguinte contém a listagem das funções em [2nd] [catalog] quando a aplicação está a ser executada. Para uma lista completa das funções, classes, operadores e palavras-chave Python disponíveis nesta versão, consulte "Conteúdo do módulo, palavras-chave e incorporado da TI-Python."

Esta tabela não pretende ser uma lista exaustiva de funções Python disponíveis na presente oferta. Outras funções suportadas nesta oferta Python podem ser introduzidas utilizando as teclas alfabéticas do teclado.

A maioria dos exemplos dados nesta tabela é executada no prompt do Shell (Interpretador) (>>>).

Lista do CATÁLOGO

Lista do alfabeto

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

#

Delimitador

2nd [catalog]

Sintaxe: #O seu comentário sobre o seu programa.

[a A #]

Descrição: No Python, um comentário inicia com o caráter de hashtag, #, e estende até ao fim da linha.

Exemplo:

```
#A short explanation of the code.
```

%

Operador

2nd [catalog]

Sintaxe: x%y ou x % y

Descrição: Devolve o resto de x/y. A utilização preferencial é quando x e y são números inteiros.

[a A #]

Exemplo:

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

Consulte também fmod(x,y).

//

Operador

2nd [catalog]

Sintaxe: x//y ou x // y

[a A #]

Descrição: Devolve a divisão do piso de x/y.

Exemplo:

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

[a A #]

Descrição: Iniciar a palete de carateres [a A #].

Inclui carateres acentuados como ç à â è é ê ë î ï ô ö ù û

O atalho [a A #] está no ecrã em window no Editor ou no Shell (Interpretador)

a gradiente; declive

Módulo: ti_plotlib

Sintaxe: plt.a gradiente; declive

Descrição: Depois de plt.linreg() ser executado pela última vez num programa, os valores calculados de declive, a, e interceção, b, são guardados em plt.a e plt.b.

Valores padrão: = 0.0

Exemplo:

Consulte o programa de amostra: LINREGR.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_ plotlib Setup.

abs()

Módulo: Built-in

Sintaxe: abs(x)

Descrição: Devolve o valor absoluto de um número. Nesta versão, o argumento pode ser um número inteiro ou de ponto flutuante.

Exemplo:

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

2nd [catalog]

Nota: fabs() é uma função do módulo math.

acos()

Módulo: math

Sintaxe: acos(x)

Descrição: Devolve o arco cosseno de x em radianos.

Exemplo:

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

Exemplo alternativo: [Tools] > 6:New Shell

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

sin 7:acos()

2nd [catalog]

[Fns…]
Modul
1:math… >
Trig
7:acos()

os comandos de importação encontram-se em 2nd [catalog]

and

Palavra-chave

Sintaxe: x and y

Descrição: Pode devolver True ou False. Devolve "x" se "x" for False e "y" se não for. Cola com espaço antes e depois de and. Editar conforme necessário.

Exemplo:

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

2nd [test]
Ops 8:and

[Fns…] > Ops
8:and

2nd [catalog]

[a A #]

.append(x)

Módulo: Built-in

Sintaxe: listname.append(item)

Descrição: O método append() acrescenta um item à lista.

Exemplo:

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

2nd [list]
List
6: .append(x)

2nd [catalog]

[Fns…] > List
6:.append(x)

as

Palavra-chave

Descrição: Utilize as para criar um pseudónimo quando importar um módulo. Para mais informações, consulte a documentação Python.

2nd [catalog]

asin()

Módulo: math

Sintaxe: asin()

Descrição: Devolve o arco seno de x em radianos.

Exemplo:

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

Exemplo alternativo:

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

sin 6:asin()

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog]

assert

Palavra-chave

2nd [catalog]

Descrição: Utilize assert para testar uma condição no seu código. Devolve None ou, caso contrário, a execução do programa exibe um AssertionError.

atan()

Módulo: math

sin 8:atan()

Sintaxe: atan(x)

Descrição: Devolve o arco tangente de x em radianos.

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

Exemplo:

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

2nd [catalog]

Exemplo alternativo:

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

os comandos de importação encontram-se em 2nd [catalog]

atan2(y,x)

Módulo: math

sin 9:atan2()

Sintaxe: atan2(y,x)

Descrição: Devolve o arco tangente de y/x em radianos. O resultado é em [-pi, pi].

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

Exemplo:

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

2nd [catalog]

Exemplo alternativo:

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

os comandos de importação

atan2(y,x)

encontram-se em 2nd [catalog]

auto_window(xlist,ylist)

Módulo: ti_plotlib

Sintaxe: plt.auto_window(xlist,ylist)

Descrição: Dimensiona automaticamente a janela de representação gráfica para se ajustar aos intervalos de dados dentro de xlist e ylist especificados no programa antes de auto_window().

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

Exemplo:

Consulte o programa de amostra: LINREGR.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

axes("mode")

Módulo: ti_plotlib

Sintaxe: plt.axes("mode")

Descrição: Exibe eixos na janela especificada na área de representação gráfica.

Argumento:

Opções de argumento "mode":

"off"	sem eixos
"on"	eixos+etiquetas
"axes"	só eixos
"window"	só etiquetas da janela

plt.axes() utiliza a definição de cor de caneta atual. Para garantir que plt.axes() são sempre desenhados como esperado, utilize plt.color() ANTES de plt.axes() para garantir que as cores são as esperadas.

Exemplo:

Consulte o programa de amostra LINREGR.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

b y= intersecção

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.b y= intersecção

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

Descrição: Depois de plt.linreg() ser executado num programa, os valores calculados de declive, a, e intersecção, b, são guardados em plt.a e plt.b.

Valores padrão: = 0.0

Exemplo:

Consulte o programa exemplo LINREGR.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

bin(integer)

Módulo: Plano integrado

2nd [catalog]

Sintaxe: bin(integer)

Descrição: Exibe o formato binário do argumento de número inteiro.

Para mais informações, consulte a documentação Python.

Exemplo:

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

break

Palavra-chave

2nd [catalog]

Descrição: Utilize break para sair de a para ou durante o ciclo.

C

ceil()

Módulo: math

Sintaxe: ceil(x)

Descrição: Devolve o menor número inteiro que é igual ou superior a x.

Exemplo:

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

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

[2nd] [catalog]

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

os comandos de importação encontram-se em [2nd] [catalog]

choice(sequence)

Módulo: random

Sintaxe: choice(sequence)

Descrição: Devolve um elemento aleatório de uma sequência não vazia.

Exemplo:

```
>>>from random import *
>>>listA=[2,4,6,8]
>>>choice(listA) #O seu resultado pode ser diferente.
4
```

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

[2nd] [catalog]

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

os comandos de importação encontram-se em [2nd] [catalog]

chr(integer)

Módulo: Plano integrado

2nd [catalog]

Sintaxe: chr(integer)

Descrição: Devolve uma cadeia a partir de um número inteiro que representa o caráter unicode.

Para mais informações, consulte a documentação Python.

Exemplo:

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

class

Palavra-chave

2nd [catalog]

Descrição: Utilize class para criar uma classe. Para mais informações, consulte a documentação Python.

cls() apagar ecrã

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.cls() apagar ecrã

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

Descrição: Apaga o ecrã do Shell (Interpretador) para a representação gráfica. As teclas de atalho não são exibidas ao efetuar a representação gráfica.

Nota: plt.cls() tem um comportamento diferente de ti_system module disp_clr().

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib

Exemplo:

Consulte o programa exemplo: GRAPH.

cls() apagar ecrã

Setup.

color(r,g,b) 0-255

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.color(r,g,b) 0-255

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

Descrição: Define a cor para todos os gráficos/traçados seguintes. Os valores (r,g,b) têm de ser especificados 0-255. A cor especificadas é utilizada na representação gráfica até que color() seja executado novamente com uma cor diferente.

A cor predefinida é preto ao importar ti_plotlib.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_ plotlib Setup.

Exemplo:

Consulte o programa exemplo: COLORLIN.

complex(real,imag)

Módulo: Plano integrado

2nd [catalog]

Sintaxe: complex(real,imag)

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

Descrição: Tipo de números complexos.

Exemplo:

```
>>>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)
```

Nota: "1+2j" é a sintaxe correta. Os espaços como "1 + 2j" exibem uma exceção.

continue

Palavra-chave

2nd [catalog]

Descrição: Utilize continue em a para ou durante um circuito para terminar a iteração atual. Para mais informações, consulte a documentação Python.

cos()

Módulo: math

sin Trig
4: cos()

Sintaxe: cos(x)

Descrição: Devolve cos de x. O argumento angular está em radianos.

2nd [catalog]

Exemplo:

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

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

Exemplo alternativo:

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

Nota: O Python exibe notação científica utilizando e ou E. Alguns resultados matemáticos em Python serão diferentes dos obtidos no CE OS.

.count()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: listname.count(item)

Descrição: count()é o método que devolve o número de ocorrências de um item numa lista, énuplo, bytes, str, bytearray, ou objeto array.array.

Exemplo:

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

def function ():

Palavra-chave

2nd [catalog]

Sintaxe: def function(var, var,...)

Descrição: Definir uma função dependente de variáveis especificadas. Normalmente utilizado com a palavra-chave return.

[Fns...]>Func
1:def function():

[Fns...]>Func
2:return

Exemplo:

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

degrees()

Módulo: math

sin Trig
2:degrees()

Sintaxe: degrees(x)

Descrição: Converte o ângulo x em radianos para graus.

2nd [catalog]

Exemplo:

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

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

del

Palavra-chave

2nd [catalog]

Descrição: Utilize del para eliminar objetos, como variáveis, listas, etc.
Para mais informações, consulte a documentação Python.

disp_at(row,col,"text")

Módulo: ti_system

Sintaxe: disp_at(row,col,"text")

Descrição: Exibir texto começando numa posição de linha e coluna na área de representação gráfica.

REPL with cursor >>>| aparece depois do texto se estiver no fim do programa. Utilize disp_cursor() para controlar a apresentação do cursor.

Argumento:

row	1 - 11, números inteiros
column	1 - 32, números inteiros
"text"	é uma cadeia que se molda na área do ecrã

Argumentos opcionais para cor e fundo são aqui apresentados: disp_at(row,col,"text","align",color 0-15, background color 0-5)

Exemplo:

Programa exemplo:

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

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

disp_at(row,"text","align")

Módulo: ti_system

Sintaxe: disp_at(row,"text","align")

Descrição: Exibir o texto alinhado conforme especificado no ecrã de representação gráfica para a linha 1-11. A linha é limpa antes de ser exibida. Se utilizado em ciclo, o conteúdo é atualizado com cada exibição.

REPL with cursor >>>| aparece depois do texto se estiver no fim do programa. Utilize disp_cursor() para controlar a exibição do cursor antes de utilizar disp_at() no programa.

Argumento:

row	1 - 11, números inteiros
"text"	é uma cadeia que se molda na área do ecrã
"align"	"left" (predefinido) "center" "right"

Argumento opcional apresentado aqui: disp_at (row,"text","align","color 0-15, background color 0-15)

Exemplo:

Programa exemplo:

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

2nd [catalog]

2nd [rcl]
ti_system
7:disp_at()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

disp_clr() apagar ecrã de texto

Módulo: ti_system

Sintaxe: disp_clr() apagar ecrã de texto

Descrição: Limpar o ecrã no ambiente Shell (Interpretador). Linha 0-11, o número inteiro pode ser usado como argumento opcional para limpar uma linha de exibição do ambiente Shell (Interpretador).

Exemplo:

Programa exemplo:

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

2nd [catalog]

2nd [rcl]
ti_system
8:disp_clr()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

disp_cursor() 0=off 1=on

Módulo: ti_system

2nd [catalog]

Sintaxe: disp_cursor() 0=off 1=on

Descrição: Controle a exibição do cursor no Shell (Interpretador) quando um programa está a ser executado.

2nd [rcl]
ti_system
0:disp_cursor()

Argumento:

0 = desligado

não 0 = ligado

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

Exemplo:

Programa exemplo:

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

disp_wait() [clear]

Módulo: ti_system

Sintaxe: disp_wait() [clear]

Descrição: Pare a execução do programa neste ponto e exiba o conteúdo do ecrã até premir [clear] e o ecrã ser apagado.

Exemplo:

Programa exemplo:

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

2nd [catalog]

2nd [rcl]
ti_system
9:disp_wait()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

e

Módulo: math

Sintaxe: math.e ou e se o módulo math foi importado

Descrição: A constante e é apresentada conforme mostrado em baixo.

Exemplo:

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

Exemplo alternativo:

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

2nd [e] (acima de ÷)

[Fns…] > Modul 1:math… > Const 1:e

elif :

Palavra-chave

Para mais informações, ver if..elif..else..

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

else:

Palavra-chave

Para mais informações, ver if..elif..else..

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

escape()

Módulo: ti_system

Sintaxe: escape()

Descrição: escape() devolve True ou False.

O valor inicial é False.

Quando a tecla [clear] na CE é premida, o valor é definido para True.

Quando a função é executada, o valor é reposto em False.

Exemplo de utilização:

while not escape():

Num circuito while em execução num programa que permite terminar o circuito mas manter o script em execução.

if escape():break

Pode ser utilizado para um programa de depuração para inspecionar as variáveis utilizando [vars] do Shell (Interpretador) depois de executar o programa e utilizar esta pausa.

2nd [catalog]

Como linha de programa:

2nd [rcl]
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

os comandos de importação encontram-se em 2nde [catalog] ou no menu ti_system Modul.

eval()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: eval(x)

[Fns…] I/O 3:eval()

Descrição: Devolve a avaliação da expressão x.

Exemplo:

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

except exceção:

Palavra-chave

2nd [catalog]

Descrição: Utilize except num bloco de código try..except. Para mais informações, consulte a documentação Python.

exp()

Módulo: math

Sintaxe: exp(x)

Descrição: Devolve e**x.

Exemplo:

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

Exemplo alternativo: [Tools] > 6:New Shell

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

2nd [e^x] (acima de ln)

2nd [catalog]

[Fns…] > Modul 1:math… 4:exp()

os comandos de importação encontram-se em 2nd [catalog].

.extend()

Módulo: Plano integrado

Sintaxe: listname.extend(newlist)

Descrição: O método extend() é um método para prolongar newlist até ao fim de uma lista.

Exemplo:

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

2nd [catalog]

fabs()

Módulo: math

Sintaxe: fabs(x)

Descrição: Devolve o valor absoluto de x

Exemplo:

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

2nd [catalog]

[Fns...] > Modul 1:math... 2:fabs()

os comandos de importação encontram-se em 2nd [catalog].

Ver também a função incorporada abs().

False

Palavra-chave

Descrição: Devolve False quando a afirmação executada é False. "False" representa o valor falso de objetos do tipo bool.

Exemplo:

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

2nd [test] (acima de math)

2nd [catalog]

[Fns...] > Ops B:False

[a A #]

finally:

Palavra-chave

2nd [catalog]

Descrição: Utilize finally num bloco de código try..except..finally. Para mais informações, consulte a documentação Python.

float()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: float(x)

Descrição: Devolve x como um número de ponto flutuante.

[Fns…] > Type
2:float()

Exemplo:

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

floor()

Módulo: math

math Modul
1:math
9:floor()

Sintaxe: floor(x)

Descrição: Devolve o maior número inteiro que é igual ou inferior a x.

2nd [catalog]

Exemplo:

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

[Fns…] > Modul
1:math
9:floor()

os comandos de importação encontram-se em
2nd [catalog]

fmod(x,y)

Módulo: math

Sintaxe: fmod(x,y)

Descrição: Para mais informações, consulte a documentação Python. A utilização preferencial é quando x e y são números de ponto flutuante.

Pode não devolver o mesmo resultado que x%y.

Exemplo:

```
>>>from math import *
>>>fmod(50.0,8.0)
2.0
>>>fmod(-50.0,8.0)
-2.0
>>>-50.0 - (-6.0)*8.0 #validação da descrição
-2.0
```

Consulte também: x%y.

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

[2nd] [catalog]

[Fns…] > Modul
1:math…
7:fmod()

os comandos de importação encontram-se em [2nd] [catalog]

for i in list:

Palavra-chave

Sintaxe: for i in list:

Descrição: Utilizado para iterar através de elementos da lista.

Exemplo:

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

[Fns…] Ctl
7:for i in list:

[2nd] [catalog]

for i in range(size):

Palavra-chave

Sintaxe: for i in range(size)

Descrição: Utilizado para iterar através de um intervalo.

Exemplo:

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

[Fns…] Ctl 4:for i in range (size):

2nd [catalog]

for i in range(start,stop):

Palavra-chave

Sintaxe: for i in range(start,stop)

Descrição: Utilizado para iterar através de um intervalo.

Exemplo:

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

[Fns…] Ctl 5:for i in range (start,stop):

2nd [catalog]

for i in range(start,stop,step):

Palavra-chave

[Fns…] Ctl 6:for i in range (start,stop,step):

Sintaxe: for i in range(start,stop,step)

Descrição: Utilizado para iterar através de um intervalo.

2nd [catalog]

Exemplo:

```
>>> for i in range(1,8,2):
… print(i)
…
…
…
1
3
4
7
```

str.format() formato de cadeia

Módulo: Plano integrado

2nd [catalog]

Sintaxe:str.format()

Descrição: Formata a cadeia indicada. Para mais informações, consulte a documentação Python.

Exemplo:

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

frexp()

Módulo: math

Sintaxe: frexp(x)

Descrição: Devolve um par (y,n) quando x == y * 2**n. y é um número de ponto flutuante em que 0.5<abs(y)<1; e n é um número inteiro.

Exemplo:

```
>>>from math import *
>>>frexp(2000.0)
(0.9765625, 11)
>>>0.9765625 * 2**11 #validar descrição
2000.0
```

[math] Modul
1:math
A:frexp()

[2nd] [catalog]

[Fns…] > Modul
1:math
A:frexp()

os comandos de importação encontram-se em
[2nd] [catalog]

from PROGRAM import *

Palavra-chave

Sintaxe: from PROGRAM import *

Descrição: Utilizado para importar um programa. Importa os atributos públicos de um módulo Python para o espaço de nome atual.

Shell [Tools]
A:from
PROGRAM
import *

[2nd] [catalog]

from math import *

Palavra-chave

Sintaxe: from math import *

Descrição: Utilizado para import todas as funções e constantes do math module.

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

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

2nd [catalog]

from random import *

Palavra-chave

Sintaxe: from random import *

Descrição: Utilizado para importar todas as funções do módulo random.

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

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

2nd [catalog]

from time import *

Palavra-chave

Sintaxe: from time import *

Descrição: Utilizado para importar todos os métodos do módulo time.

Exemplo:

Consulte o programa exemplo: DASH1.

2nd [catalog]

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

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

from ti_system import *

Palavra-chave

Sintaxe: from ti_system import *

Descrição: Utilizado para importar todos os métodos do módulo ti_system.

Exemplo:

Consulte o programa exemplo: REGEQ1.

2nd [catalog]

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

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

from ti_hub import *

Palavra-chave 2nd [catalog]

Sintaxe: from ti_hub import *

Descrição: Utilizado para importar todos os métodos do módulo ti_hub. Para os dispositivos individuais de entrada e saída, utilize a funcionalidade do módulo dinâmico, selecionando o dispositivo a partir de [Fns…]>Modul>ti_hub>Import menu quando estiver no Editor.

Ver:módulo ti_hub – Adicionar importação ao Editor e adicionar o módulo ti_hub sensor ao menu Modul.

Exemplo:

Consulte o programa exemplo: DASH1.

global

Palavra-chave

2nd [catalog]

Descrição: Utilize global para criar variáveis globais dentro de uma função.

Para mais informações, consulte a documentação CircuitPython

grid(xscl,yscl,"style")

Módulo: ti_plotlib

2nd [catalog]

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

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

Descrição: Exibe uma grelha utilizando uma escala especificada para os eixos x e y. Nota: Toda a representação gráfica se dá quando plt.show_plot() é executado.

A definição da cor da grelha é o argumento opcional de (r,g,b) utilizando valores 0-255 sendo cinzento (192,192,192) o valor predefinido.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Valor predefinido para xscl ou yscl = 1.0.

"style" = "dot" (predefinido), "dash", "solid" ou "point"

Exemplo:

Consulte os programas exemplo: COLORLIN ou GRAPH.

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

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.grid(xscl,yscl,"style",(r,g,b))

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

Descrição: Exibe uma grelha utilizando uma escala especificada para os eixos x e y. Nota: Toda a representação gráfica se dá quando plt.show_plot() é executado.

A definição da cor da grelha é o argumento opcional de (r,g,b) utilizando valores 0-255 sendo cinzento (192,192,192) o valor predefinido.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Valor predefinido para xscl ou yscl = 1.0.

"style" = "dot" (predefinido), "dash", "solid" ou "point".

Se os valores xscl ou yscl forem inferiores a 1/50.º da diferença entre xmax-xmin ou ymax-ymin, é emitida uma exceção de 'Invalid grid scale value'.

Exemplo:

Consulte o programa exemplo: GRAPH.

hex(integer)

Módulo: Plano integrado 2nd [catalog]

Sintaxe: hex(integer)

Descrição: Exibe o formato hexadecimal do argumento de número inteiro. Para mais informações, consulte a documentação Python.

Exemplo:

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

"if :"	
Para mais informações, ver if..elif..else..	2nd [catalog] [Fns…] > Ctl 1:if.. 2:if..else.. 3:if..elif..else 9:elif : 0:else:

if..elif..else..

Palavra-chave

Sintaxe: ••Identificadores de indentação cinzentos disponibilizados automaticamente na aplicação Python para utilização fácil.

if :

••

elif :

••

else:

Descrição: if..elif..else é uma instrução condicional. O Editor disponibiliza automaticamente indentações como pontos cinzentos para ajudar nas indentação corretas de programação.

Exemplo: Crie e execute este programa, introduza S01, a partir do Editor

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

Interação do Shell (Interpretador)

```
>>> # Shell Reinitialized
>>> # Running S01
>>>from S01 import * #cola automaticamente
>>>f(5)
5
>>>f(0)
zero
>>>f(-5)
5
```

2nd [catalog]

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

if..else..

Palavra-chave 2nd [catalog]

Para mais informações, ver if..elif..else..

[Fns…] > Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:

.imag

Módulo: Plano integrado 2nd [catalog]

Sintaxe:var.imag

Descrição: Devolve o coeficiente da parte imaginária de uma variável específica do tipo número complexo.

Exemplo:

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

import math

Palavra-chave

Sintaxe: import math 2nd [catalog]

Descrição: O módulo math é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo "math" dentro do seu próprio espaço de nome.

import random

Palavra-chave

Sintaxe: import random

2nd [catalog]

Descrição: O módulo random é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo "random" dentro do seu próprio espaço de nome.

import ti_hub

Palavra-chave

2nd [catalog]

Sintaxe: import ti_hub

Descrição: O módulo ti_hub é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo ti_hub dentro do seu próprio espaço de nome.

Para os dispositivos individuais de entrada e saída, utilize a funcionalidade do módulo dinâmico, selecionando o dispositivo a partir de [Fns…]>Modul>ti_hub>Import menu quando estiver no Editor.

Ver:[Fns…] > Modul: módulo ti_hub.

import time

Palavra-chave

2nd [catalog]

Sintaxe: import time

Descrição: O módulo time é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo time dentro do seu próprio espaço de nome.

Ver:[Fns…] > Modul: módulos time e ti_system.

import ti_plotlib as plt

Palavra-chave

Sintaxe: import ti_plotlib as plt

Descrição: O módulo ti_plotlib é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo ti_plotlib dentro do seu próprio espaço de nome. Os atributo do módulo ti_plotlib têm de ser introduzidos como plt.attribute.

Exemplo:

Consulte o programa exemplo: COLORLIN.

2nd [catalog]

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

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

import ti_rover as rv

Palavra-chave

Sintaxe: import ti_rover as rv

Descrição: O módulo ti_rover é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo ti_rover dentro do seu próprio espaço de nome. Os atributos do módulo ti_rover têm de ser introduzidos como rv.attribute.

Exemplo:

Consulte o programa exemplo: ROVER.

2nd [catalog]

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

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

import ti_system

Palavra-chave 2nd [catalog]

Sintaxe: import ti_system

Descrição: O módulo ti_system é acedido utilizando este comando. Esta instrução importa os atributos públicos do módulo ti_system dentro do seu próprio espaço de nome.

Exemplo:

Consulte o programa exemplo: REGEQ1.

in

Palavra-chave 2nd [catalog]

Descrição: Utilize in para verificar se um valor está numa sequência ou para iterar uma sequência num circuito for.

.index(x)

Módulo: Plano integrado 2nd [catalog]

Sintaxe:var.index(x)

Descrição: Devolve o índice ou a posição de um elemento de uma lista. Para mais informações, consulte a documentação Python.

Exemplo:

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

input()

Módulo: Plano integrado 2nd [catalog]

input()

Sintaxe: input()

Descrição: Pede introdução [Fns…] I/O 2:input()

Exemplo:

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

Exemplo alternativo:

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


RunProgram A
>>> # Shell Reinitialized
>>> # Running A
>>>from A import *
len: 15(digitar15)
15.0(resultadoflutuante 15.0)
```

.insert(index,x)

Módulo: Plano integrado

2nd [list] List
8:.insert(index,x)

Sintaxe: listname.insert(index,x)

Descrição: O método insert() introduz um item x depois do index dentro de uma sequência.

2nd [catalog]

Exemplo:

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

[Fns…] > List
8:.insert(index,x)

int()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: int(x)

Descrição: Devolve x como um objeto inteiro.

[Fns…] > Type
1:int()

Exemplo:

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

is

Palavra-chave

2nd [catalog]

Descrição: Utilize is para testar se os dois objetos são o mesmo.

labels("xlabel","ylabel",x,y)

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.labels("xlabel","ylabel",x,y)

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

Descrição: Exibe as etiquetas "xlabel" e "ylabel" nos eixos do gráfico nas posições de linha x e y. Ajuste conforme necessário para a sua representação gráfica.

"xlabel" está posicionada na linha x especificada (linha 12 predefinida) e está ajustada à direita.

"ylabel" está posicionada na linha y especificada (linha 2 predefinida) e está ajustada à esquerda.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Nota: plt.labels("|","",12,2) serão coladas com as predefinições de linha x e y, 12,2, que depois podem ser modificadas no programa.

Exemplo:

Consulte o programa exemplo: GRAPH.

lambda

Palavra-chave

2nd [catalog]

Sintaxe: lambda arguments : expression

Descrição: Utilize lambda para definir uma função anónima. Para mais informações, consulte a documentação Python.

len()

Módulo: Plano integrado

2nd [list] (acima de stat) List 3:len()

Sintaxe: len(sequence)

Descrição: Devolve o número de itens no argumento. O argumento pode ser uma sequência ou uma coleção.

2nd [catalog]

Para mais informações, consulte a documentação Python.

Exemplo:

[Fns…] > List 3:len()

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

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

Módulo: ti_plotlib

2nd [catalog]

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

[Fns…]>Modul ou math 5:ti_plotlib…> Draw 7:line ou vetor

Descrição: Exibe um segmento de linha de (x1,y1) to (x2,y2)

O tamanho e o estilo são definidos utilizando pen() e color() antes de line().

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Argumentos:

x1,y1, x2,y2 são flutuantes reais.

"mode": Com a predefinição "", não é desenhada qualquer ponta de seta.
Com "arrow" é desenhada uma ponta de seta de vetor em (x2,y2).

Exemplo:

Consulte o programa exemplo: COLORLIN.

lin_reg(xlist,ylist,"disp",row)

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.lin_reg(xlist,ylist,"disp",row)

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

Descrição: Calcula e desenha o modelo de regressão linear, ax+b, de xlist,ylist. Este método deve seguir o método de dispersão. A exibição por defeito da equação é "center" na linha 11.

Argumento:

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

plt.a (declive) e plt.b (intersecção) são guardados quando lin_reg é executado.

Exemplo:

Consulte o programa exemplo: LINREGR.

list(sequence)

Módulo: Plano integrado

2nd [list] (acima de [stat]) List 2:list(sequence)

Sintaxe: list(sequence)

Descrição: Sequência Mutable dos itens do tipo guardado.

2nd [catalog]

list()" converte o seu argumento para o tipo "list". Tal como muitas outras sequências, os elementos de uma lista não têm de ser do mesmo tipo.

[Fns…] > List 2:list(sequence)

Exemplo:

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

Exemplo:

```
>>>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)

Módulo: math

Sintaxe: log(x,base)

Descrição: log(x) sem base devolve o logaritmo natural x.

Exemplo:

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

2nd log para log(x,10)

2nd ln para log(x) (natural log)

math Modul 1:math… 6:log(x,base)

2nd [catalog]

[Fns…] > Modul 1:math… 6:log(x,base)

os comandos de importação encontram-se em 2nd [catalog]

math.function

Módulo: math

2nd [catalog]

Sintaxe: math.function

Descrição: Utilize depois do comando import math para utilizar uma função no módulo math.

Exemplo:

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

max()

Módulo: Plano integrado

2nd [list] (acima de stat) List 4:max()

Sintaxe: max(sequence)

Descrição: Devolve o valor máximo na sequência Para mais informações sobre max(), consulte a documentação Python.

2nd [catalog]

Exemplo:

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

[Fns…] > List 4:max()

min()

Módulo: Plano integrado

2nd [list] (acima de stat) List 5:min()

Sintaxe: min(sequence)

Descrição: Devolve o valor mínimo na sequência Para mais informações sobre min(), consulte a documentação Python.

2nd [catalog]

Exemplo:

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

[Fns…] > List 5:min()

monotonic() elapsed time

Módulo: time

2nd [catalog]

Sintaxe: monotonic() elapsed time

Descrição: Devolve um valor de tempo a partir do ponto de execução. Utilize o valor return para comparar contra outros valores de monotonic().

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

Exemplo:

Programa exemplo:

os comandos de importação encontram-se em 2nd [catalog] ou no menu do módulo time.

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

Execute o programa EXAMPLE até a execução
parar.
>>>15.0
```

None

Palavra-chave 2nd [catalog]

Descrição: None representa a ausência de um valor.

Exemplo: [a A #]

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

nonlocal

Palavra-chave 2nd [catalog]

Sintaxe: nonlocal

Descrição: Utilize nonlocal para declarar que uma variável não é local. Para mais informações, consulte a documentação Python.

not

Palavra-chave 2nd [test] Ops 0:not

Sintaxe: not x

Descrição: Avalia para True se x for False e False se não for. Cola com espaço antes e depois de keyword not. Editar conforme necessário. [Fns…] > Ops 0:not

Exemplo:

```
>>> not 2<5 #editar o espaço antes de not
False
>>>3<8 and not 2<5
False
```

2nd [catalog]

[a A #]

oct(integer)

Módulo: Plano integrado

2nd [catalog]

Sintaxe: oct(integer)

Descrição: Devolve a representação octal do número inteiro. Para mais informações, consulte a documentação Python.

Exemplo:

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

or

Palavra-chave

2nd [test] Ops 9:or

Sintaxe: x or y

[Fns…] > Ops 9:or

Descrição: Pode devolver Verdadeiro ou Falso Devolve x se x for avaliado como True e y se não for. Cola com espaço antes e depois de or. Editar conforme necessário.

2nd [catalog]

Exemplo:

[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("character")

Módulo: Plano integrado 2nd [catalog]

Sintaxe: ord("character")

Descrição: Devolve o valor unicode do caráter. Para mais informações, consulte a documentação Python.

Exemplo:

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

pass

Palavra-chave

2nd [catalog]

Descrição: Utilize pass numa função ou definição de classe vazia como um espaço reservado para código futuro à medida que constrói o seu programa. As definições vazias não irão causar um erro quando o programa for executado.

pen("size","style")

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.pen("size","style")

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

Descrição: Define o aspeto de todas as linhas seguintes até que o próximo pen() seja executado.

Argumento:

A predefinição de pen() é "thin" e "solid."

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Exemplo:

Consulte os programas exemplo: COLORLIN ou GRAPH.

pi

Módulo: math

2nd [π] (acima de sin)

Sintaxe: math.pi ou pi se módulo math importado.

Descrição: A constante pi é apresentada conforme mostrado em baixo.

[Fns…] > Modul 1:math… > Const 2:pi

Exemplo:

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

Exemplo alternativo:

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

plot(xlist,ylist,"mark")

Módulo: ti_plotlib

Sintaxe: plt.plot(xlist,ylist,"mark")

Descrição: Um gráfico de linhas é apresentado utilizando pares ordenados de xlist e ylist especificados. O estilo e o tamanho da linha são definidos com plt.pen ().

xlist e ylist devem ser flutuantes reais e as listas devem ter a mesma dimensão.

Argumento:

"mark" é o carácter de marca como se segue:

o	ponto a ponto (default)
+	cruz
x	x
.	píxel

Exemplo:

Consulte o programa exemplo: LINREGR.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

plot(x,y,"mark")

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.plot(x,y,"mark")

Descrição: Um gráfico de pontos, (x,y) é exibido utilizando x e y especificados.

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

xlist e ylist devem ser flutuantes reais e as listas devem ter a mesma dimensão.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Argumento:

"mark" é o carácter de marca como se segue:

o	ponto a ponto (default)
+	cruz
x	x
.	píxel

Exemplo:

Consulte o programa exemplo: LINREGR.

pow(x,y)

Módulo: math

Sintaxe: pow(x,y)

Descrição: Devolve x elevado à potência de y. Converte x e y para flutuante. Para mais informações, consulte a documentação Python.

Utilize a função built-in pow(x,y) ou ** para calcular potências inteiras exatas.

Exemplo:

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

Exemplo utilizando: Plano integrado:

[Tools] > 6:New Shell

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

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

[2nd] [catalog]

[Fns…] > Modul
1:math
5:pow(x,y)

os comandos de importação encontram-se em [2nd] [catalog]

print()

Módulo: Plano integrado

Sintaxe: print(argument)

Descrição: Exibe o argumento como uma cadeia.

Exemplo:

```
>>>x=57.4
>>>print("my number is =", x)
my number is= 57.4
```

[2nd] [catalog]

[Fns…] > I/O
1:print()

radians()) degree ▶radians

Módulo: math

Sintaxe: radians(x)

Descrição: Converte o ângulo x em degrees para radians.

Exemplo:

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

sin Trig
1:radians()

2nd [catalog]

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

raise

Palavra-chave

Sintaxe: raise exception

Descrição: Utilize raise para criar uma exceção específica e parar o seu programa.

2nd [catalog]

randint(min,max)

Módulo: random

Sintaxe: randint(min,max)

Descrição: Devolve um número inteiro aleatório entre min e max.

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

Exemplo:

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

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

Exemplo alternativo:

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

[2nd] [catalog]

Os resultados variam em função de uma saída aleatória.

os comandos de importação encontram-se em [2nd] [catalog]

random()

Módulo: random

Sintaxe: random()

Descrição: Devolve um número com ponto flutuante de 0 a 1.0. Esta função não assume argumentos.

Exemplo:

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

Exemplo alternativo:

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

Os resultados variam em função de uma saída aleatória.

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

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

[2nd] [catalog]

os comandos de importação encontram-se em [2nd] [catalog]

random.function

Módulo: random

Sintaxe: random.function

Descrição: Utilize depois de importar random para aceder a uma função no módulo random.

Exemplo:

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

Os resultados variam em função de uma saída aleatória.

[2nd] [catalog]

randrange(start,stop,step)

Módulo: random

Sintaxe: randrange(start,stop,step)

Descrição: Devolve um número aleatório do início ao fim, passo a passo.

Exemplo:

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

Exemplo alternativo:

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

Os resultados variam em função de uma saída aleatória.

[math] Modul
2:random…
Random
6:randrange
(start,stop,step)

[math] Modul
2:random…
Random
6:randrange
(start,stop,step)

[2nd] [catalog]

os comandos de importação encontram-se em [2nd][catalog]

range(start,stop,step)

Módulo: Built in

[2nd] [catalog]

Sintaxe: range(start,stop,step)

Descrição: Utilize a função range para devolver uma sequência de números. Todos os argumentos são opcionais. A predefinição de start é 0, a predefinição de step é 1 e a sequência termina em stop.

Exemplo:

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

.real

Módulo: Plano integrado 2nd [catalog]

Sintaxe:var.real

Descrição: Devolve a parte real de uma variável específica de tipo de número complexo.

Exemplo:

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

var=recall_list("name") 1-6

Módulo: ti_system

Sintaxe:var=recall_list("name") 1-6

Descrição: Recuperar uma lista de SO predefinida. O comprimento da lista tem de ser inferior ou igual a 100.

Argumento: "**name**"

Para OS L1-L6

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

Para a lista personalizada SO "name"

----- Máx. 5 carateres, números ou letras, começando com letras e as letras têm de estar em maiúscula.

Exemplos:

"ABCDE"

"R12"

"L1" será L1 personalizado e não SO L1

Lembrete: Python tem precisão dupla. Python suporta mais dígitos que o SO.

Exemplo:

Programa exemplo:

Crie uma lista no SO.
LIST={1,2,3}

Execute a aplicação Python.
Crie um novo programa AA.

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

Execute o programa AA.
O Shell (Interpretador) exibe o resultado.

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

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_list()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

var=recall_RegEQ()

Módulo: ti_system

Sintaxe:var=recall_RegEQ()

Descrição: Recupera a variável RegEQ a partir do SO CE. A equação de regressão deve ser calculada no SO antes de recuperar a RegEQ na aplicação Python.

Exemplo:

Consulte o programa exemplo: REGEQ1.

2nd [catalog]

2nd [rcl]
ti_system
4:var=recall_
REGEQ()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

.remove(x)

Módulo: Plano integrado

Sintaxe: listname.remove(item)

Descrição: O método remove() remove a primeira instância de um item de uma sequência.

Exemplo:

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

2nd [list]
List
7:.remove(x)

2nd [catalog]

[Fns…] > List
7:.remove(x)

return

Módulo: Plano integrado

2nd [catalog]

Sintaxe: return expression

Descrição: Um instrução return define o valor produzido por uma função. As funções Python devolvem None por defeito. Consulte também: def function ():

[Fns…] > Func 1:def function():

Exemplo:

[Fns…] > Func 2:return

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

.reverse()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: listname.reverse()

Descrição: Inverte a ordem dos dados numa lista.

Exemplo:

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

round()

Módulo: Built in

2nd [catalog]

Sintaxe: round(number, digits)

Descrição: Utilize a função round para devolver um número com ponto flutuante arredondado aos dígitos especificados. O dígito predefinido é 0 e devolve o número inteiro mais próximo.

Exemplo:

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

scatter(xlist,ylist,"mark")

Módulo: ti_plotlib

2nd[catalog]

Sintaxe: plt.scatter(xlist,ylist,"mark")

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

Descrição: Uma sequência de pares ordenados de (xlist,ylist) será traçada com o estilo de marca especificado. O estilo e o tamanho da linha são definidos com plt.pen().

xlist e ylist devem ser flutuantes reais e as listas devem ter a mesma dimensão.

os comandos de importação encontram-se em 2nd[catalog] ou no menu ti_plotlib Setup.

Argumento:

"mark" é o carácter de marca como se segue:

o	ponto a ponto (default)
+	cruz
x	x
.	píxel

Exemplo:

Consulte o programa exemplo: LINREGR.

seed()

Módulo: random

Sintaxe: seed() ou seed(x) onde x é número inteiro

Descrição: Inicialize o gerador de número aleatório.

Exemplo:

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

Os resultados variam em função de uma saída aleatória.

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

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

[2nd] [catalog]

os comandos de importação encontram-se em [2nd] [catalog]

set(sequence)

Módulo: Plano integrado

[2nd] [catalog]

Sintaxe: set(sequence)

Descrição: Devolve uma sequência como um conjunto. Para mais informações, consulte a documentação Python.

Exemplo:

```
>>> print(set("84CE")
{'E', '8', '4', 'C'}
```

show_plot() display > [clear]

Módulo: ti_plotlib

Sintaxe: plt.show_plot() display > [clear]

Descrição: Executa a exibição do gráfico conforme configurado no programa.

show_plot() tem de ser colocado depois de todos os objetos de configuração da representação gráfica. A ordem de programação dos objetos de representação gráfica é sugerida pela ordem do menu Setup.

Para a ajuda do modelo de representação gráfica, a partir do Gestor de ficheiros, selecione [New] ([zoom]) e depois [Types] ([zoom]) para selecionar o tipo de programa "Plotting (x,y) & Text".

Depois de executar o programa, o ecrã da representação gráfica é limpo pressionando [clear] para voltar ao prompt do Shell (Interpretador).

Exemplo:

Consulte os programas exemplo: COLORLIN ou GRAPH.

2nd [catalog]

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

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

sin()

Módulo: math

Sintaxe: sin()

Descrição: Devolve seno de x. O argumento está em radianos.

Exemplo:

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

sin 3:sin()

2nd [catalog]

[Fns…] > Modul
1:math… > Trig
3:sin()

os comandos de importação encontram-se em
2nd [catalog]

sleep(seconds)

Módulo: ti_system; time

Sintaxe: sleep(seconds)

Descrição: Hibernar durante um determinado número de segundos. O argumento de segundos é flutuante.

Exemplo:

Programa exemplo:

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

Executar o programa TIME

```
>>>15.0
```

2nd [catalog]

2nd [rcl]
ti_system
A:sleep()

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

[Fns…]>Modul ou
math
3:time
2:sleep()

os comandos de importação encontram-se em
2nd [catalog] ou no menu
ti_system Modul.

.sort()

Módulo: Plano integrado

Sintaxe: listname.sort()

Descrição: O método ordena uma lista existente. Para mais informações, consulte a documentação Python.

Exemplo:

```
>>>listA=[4,3,6,2,7,4,8,9,3,5,4,6]
>>>listA.sort()
>>>print(listA) #listA atualizada para uma lista ordenada
[2,3,3,4,4,4,5,6,6,7,8,9]
```

2nd [list] (acima de stat) List A:.sort()

2nd [catalog]

[Fns…] > List A:sort()

sorted()

Módulo: Plano integrado

Sintaxe: sorted(sequence)

Descrição: Devolve uma lista ordenada a partir da sequência.

Exemplo:

```
>>>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 não alterou
[4,3,6,2,7,4,8,9,3,5,4,6]
```

2nd [list] (acima de stat) List 0:sorted()

2nd [catalog]

[Fns…] > List 0:sorted()

.split(x)

Módulo: Plano integrado

2nd [catalog]

Sintaxe:var.split(x)

Descrição: O método devolve uma lista através de um separador especificado. Para mais informações, consulte a documentação Python.

Exemplo:

```
>>> a="red,blue,green"
>>> a.split(",")
['red', 'blue', 'green']
```

sqrt()

Módulo: math

Sintaxe: sqrt(x)

Descrição: Devolve a raiz quadrada de x.

Exemplo:

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

math
Modul
1:math
3:sqrt()

2nd
[catalog]

[Fns…] >
Modul
1:math
3:sqrt()

os comandos de importação encontram-se em 2nd [catalog].

store_list("name",var) 1-6

Módulo: ti_system

Sintaxe: store_list("name",var) 1-6

Descrição: Guarda uma lista da execução de um script Python para uma variável de lista de SO "name" onde var é uma lista Python definida. O comprimento da lista tem de ser inferior ou igual a 100.

Argumento: "**name**"

Para OS L1-L6

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

Para a lista personalizada SO "name"

----- Máx. 5 carateres, números ou letras, começando com letras e as letras têm de estar em maiúscula.

Exemplos:

"ABCDE"

"R12"

"L1" será L1 personalizado e não SO L1

Lembrete: Python tem precisão dupla, portanto mais dígitos do que os suportados no SO.

Exemplo:

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

Saia da aplicação Pyton e prima 2nd [L1] (acima de [1]) e entrer no ecrã inicial para ver a lista [L1] como {1 2 3}.

2nd [catalog]

2nd [rcl]
ti_system
3:var=store_list
()

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_system Modul.

str()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: str(argument)

Descrição: Converte "argument" numa cadeia.

[Fns…]
> Type
3 :str()

Exemplo:

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

sum()

Módulo: Plano integrado

2nd [list] (acima de stat) List
9:sum()

Sintaxe: sum(sequence)

Descrição: Devolve a soma dos itens numa sequência.

2nd [catalog]

Exemplo:

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

[Fns…] > List
9:sum()

tan()

Módulo: math

Sintaxe: tan(x)

Descrição: Devolve tangente de x. O argumento está em radianos.

Exemplo:

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

[sin] 5:tan()

[Fns...] > Modul 1:math... > Trig 5:tan()

[2nd] [catalog]

os comandos de importação encontram-se em [2nd] [catalog]

text_at(row,"text","align")

Módulo: ti_plotlib

Sintaxe: plt.text_at(row,"text","align")

Descrição: Exibe "text" na área do gráfico no "align" especificado.

row	número inteiro 1 a 12
"text"	a cadeia é cortada se for muito comprida
"align"	"left" (predefinido) "center" "right"
opcional	1 limpa a linha antes do texto (predefinido) 0 a linha não é limpa

Exemplo:

[2nd] [catalog]

[Fns...]>Modul ou [math] 5:ti_plotlib...> Draw 0:text_at()

os comandos de importação encontram-se em [2nd] [catalog] ou no menu ti_plotlib Setup.

text_at(row,"text","align")

Consulte o programa exemplo: DASH1.

time.function

Módulo: Plano integrado

2nd [catalog]

Sintaxe: time.function

Descrição: Utilize depois de importar time para aceder a uma função no módulo time.

Exemplo:

Ver:[Fns...]>Modul: módulos time e ti_system.

title("title")

Módulo: ti_plotlib

2nd [catalog]

Sintaxe: plt.title("title")

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

Descrição: "title" é exibido centrado na linha superior da janela. "title" é cortado se for muito comprido

Exemplo:

Consulte o programa exemplo: COLORLIN.

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

ti_hub.function

Módulo: ti_hub [2nd] [catalog]

Sintaxe: ti_hub.function

Descrição: Utilize depois de importar ti_hub para aceder a uma função no módulo ti_hub.

Exemplo:

Ver:[Fns…]>Modul: módulo ti_hub.

ti_system.function

Módulo: ti_system [2nd] [catalog]

Sintaxe: ti_system.function

Descrição: Utilize depois de importar ti_system para aceder a uma função no módulo ti_system.

Exemplo:

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

texte>>>|

#aparece na linha 6, col 8 com prompt do
Shell (Interpretador) conforme mostrado.
```

True

Palavra-chave

Descrição: Devolve True quando a afirmação executada é True. "True" representa o valor verdadeiro de objetos do tipo bool.

Exemplo:

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

2nd [test] (acima de math)

2nd [catalog]

[Fns…] > Ops
A:True

[a A #]

trunc()

Módulo: math

Sintaxe: trunc(x)

Descrição: Devolve o valor real x truncado para um inteiro.

Exemplo:

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

math Modul
1:math…
0:trunc()

2nd [catalog]

[Fns…] > Modul
1:math…
0:trunc()

os comandos de importação encontram-se em 2nd [catalog]

try:

Palavra-chave

Descrição: Utilize o bloco de código try para testar o bloco de erros quanto a erros. Também utilizado com except e finally. Para mais informações,

2nd [catalog]

try:

consulte a documentação Python.

tuple(sequence)

Módulo: Plano integrado

2nd [catalog]

Sintaxe: tuple(sequence)

Descrição: Converte a sequência num énuplo. Para mais informações, consulte a documentação Python.

Exemplo:

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

type()

Módulo: Plano integrado

2nd [catalog]

Sintaxe: type(object)

[Fns...]>Type>6:type ()

Descrição: Devolve o tipo do objeto.

Exemplo:

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

uniform(min,max)

Módulo: random

Sintaxe: uniform(min,max)

Descrição: Devolve um número aleatório x (float) como min <= x <= max.

Exemplo:

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

Os resultados variam em função de uma saída aleatória.

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

[2nd] [catalog]

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

os comandos de importação encontram-se em [2nd] [catalog]

wait_key()

Módulo: ti_system

2nd [catalog]

Sintaxe: wait_key()

Descrição: Devolve um código chave combinado que representa a tecla premida, combinado com 2nd e/ou alpha. O método espera que uma tecla seja premida antes de voltar ao programa.

Exemplo:

Ver:[Fns…]>Modul: módulos time e ti_system.

while condition:

Palavra-chave

[Fns…] Ctl 8:while condition:

Sintaxe: while condition:

Descrição: Executa as instruções no seguinte bloco de código até que a "condition" seja avaliada como False.

2nd [catalog]

Exemplo:

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

window(xmin,xmax,ymin,ymax)

Módulo: ti_plotlib

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

Descrição: Define a janela de representação gráfica através do mapeamento do intervalo horizontal (xmin, xmax) e vertical (ymin, ymax) especificado para a área de representação gráfica (píxeis).

Este método deve ser executado antes da execução de qualquer outro comando do módulo ti_plotlib.

As vars de propriedades do ti_plotlib, xmin, xmax, ymin, ymax serão atualizadas para os valores de argumento. Os valores predefinidos são (-10, 10, -6.56, 6.56).

Exemplo:

Consulte o programa de amostra: GRAPH.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

with

Palavra-chave

Descrição: Para mais informações, consulte a documentação Python.

2nd[catalog]

xmax default 10.00

Módulo: ti_plotlib

Sintaxe: plt.xmax default 10.00

Descrição: Variável especificada para argumentos de janela definidos como plt.xmax.

Valores padrão:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exemplo:

Consulte o programa de amostra: GRAPH.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

xmin default -10.00

Módulo: ti_plotlib

Sintaxe: plt.xmin default -10.00

Descrição: Variável especificada para argumentos de janela definidos como plt.xmin.

Valores padrão:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exemplo:

Consulte o programa de amostra: GRAPH.

2nd [catalog]

[Fns...]>Modul ou
math
5:ti_plotlib...>
Properties
1:xmin

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

yield

Palavra-chave

Descrição: Utilize yield para terminar uma função. Devolve um gerador. Para mais informações, consulte a documentação Python.

2nd [catalog]

ymax default 6.56

Módulo: ti_plotlib

Sintaxe: plt.ymax default 6.56

Descrição: Variável especificada para argumentos de janela definidos como plt.ymax.

Valores padrão:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exemplo:

Consulte o programa de amostra: GRAPH.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

ymin default -6.56

Módulo: ti_plotlib

Sintaxe: plt.ymin default -6.56

Descrição: Variável especificada para argumentos de janela definidos como plt.ymin.

Valores padrão:

xmin	default -10.00
xmax	default 10.00
ymin	default -6.56
ymax	default 6.56

Exemplo:

Consulte o programa de amostra: GRAPH.

2nd [catalog]

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

os comandos de importação encontram-se em 2nd [catalog] ou no menu ti_plotlib Setup.

Símbolos

@

Operador

alpha [θ] (acima de 3)

Descrição: Decorador – Para mais informações, consulte a documentação geral Python.

2nd [catalog]

<<

Operador

2nd [catalog]

Sintaxe: x<<n

Descrição: Deslocação para a esquerda em bits n.

>>

Operador

2nd [catalog]

Sintaxe: x>>n

Descrição: Deslocação para a direita em bits n.

|

Operador

2nd [catalog]

Sintaxe: x|y

Descrição: Em bits or.

&

Operador

2nd [catalog]

Sintaxe: x&y

Descrição: Em bits and.

^

Operador 2nd [catalog]

Sintaxe: x^y

Descrição: Em bits exclusive or.

~

Operador 2nd [catalog]

Sintaxe: ~x

Descrição: Em bits not; os bits de x invertidos.

x<=y

Operador

Sintaxe: x<=y

Descrição: Comparação; x inferior ou igual a y.

Exemplo:

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

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

2nd [catalog]

[Fns…] > Ops
7:x<=y

[a A #]

x<y

Operador

Sintaxe: x<y

Descrição: Comparação; x inferior a y.

Exemplo:

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

math
1:math > Ops
6:x<y

2nd [catalog]

[Fns…] > Ops
6:x<y

[a A #]

x>=y

Operador

Sintaxe: x>=y

Descrição: Comparação; x superior ou igual a y.

Exemplo:

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

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

2nd [catalog]

[Fns…] > Ops
5:x>=y

[a A #]

x>y

Operador

Sintaxe: x>y

Descrição: Comparação; x superior a y.

Exemplo:

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

math
1:math > Ops
4:x>y

2nd [catalog]

[Fns…] > Ops
4:x>y

[a A #]

x!=y

Operador

Sintaxe: x!=y

Descrição: Comparação; x diferente de y.

Exemplo:

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

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

2nd [catalog]

[Fns…] > Ops
3:x!=y

[a A #]

x==y

Operador

Sintaxe: x==y

Descrição: Comparação; x é igual a y.

Exemplo:

```
>>>75==25+50
True
>>>1/3==0.333333
False
>>>1/3==0.3333333 #egual ao valor Python guardado
True
```

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

2nd [catalog]

[Fns…] > Ops
2:x==y

[a A #]

x=y

Operador

Sintaxe: x=y

Descrição: y é guardado na variável x

Exemplo:

```
>>>A=5.0
>>>print(A)
5.0
>>>B=2**3 #Utilize [ ^ ] no teclado para **
>>>print(B)
8
```

sto→

math
1:math > Ops
1:x=y

2nd [catalog]

[Fns…] > Ops
1:x=y

[a A #]

\

Delimitador

Descrição: Caráter de barra invertida.

2nd [catalog]

[a A #]

\t

Delimitador

Descrição: Espaço de tabulação entre as cadeias ou carateres.

2nd [catalog]

\n

Delimitador

Descrição: Nova linha para exibir a cadeia de forma clara no ecrã.

2nd [catalog]

' '

Delimitador

Descrição: Duas aspas simples.

Exemplo:

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

2nd [mem]
(acima de +)

2nd [catalog]

[a A #]

" "

Delimitador

Descrição: Duas aspas duplas.

Exemplo:

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

alpha ["]
(acima de +)

2nd [catalog]

[a A #]

Anexo

Conteúdo do módulo, palavras-chave e incorporado da TI-Python

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

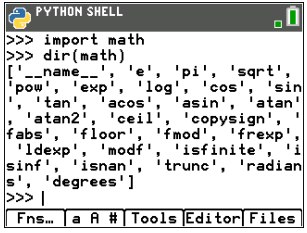
Built-ins	Built-ins	Built-ins
		'ReloadException'>
list -- <class 'list'>	globals -- <function>	KeyError -- <class 'KeyError'>
map -- <class 'map'>	hasattr -- <function>	LookupError -- <class 'LookupError'>
memoryview -- <class 'memoryview'>	hash -- <function>	MemoryError -- <class 'MemoryError'>
object -- <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>	

palavras-chave

palavras-chave	palavras-chave	palavras-chave
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>		

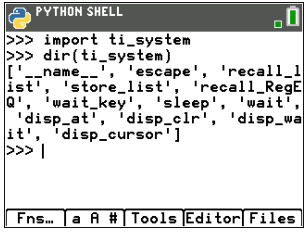
random

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

time

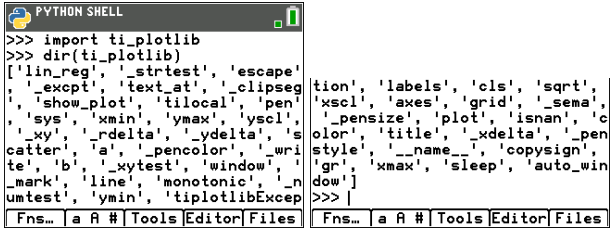
time	time	time
__name__		
monotonic		
sleep		
struc_time		

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



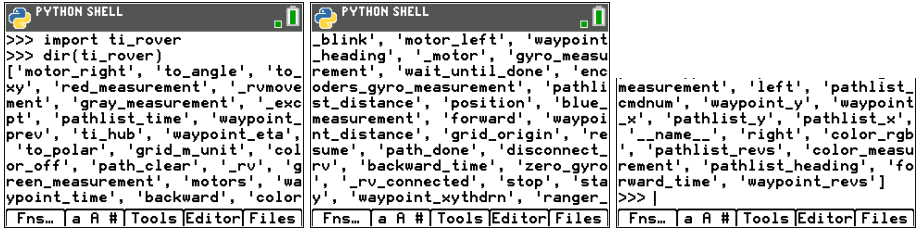
ti_plotlib	ti_plotlib	ti_plotlib
__name__	a	grid
lin_reg	_pencolor	-pensize
_strtest	_write	_sema
escape	b	-pensize
_except	_xytest	plot
text_alt	window	isnan
_clipseg	_mark	color
show-plot	line	title
tilocal	monotonic	_xdelta
pen	_ntest	_penstyle

ti_plotlib	ti_plotlib	ti_plotlib
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	
calibrate	what	
range	who	

ti_rover



ti_rover	ti_rover	ti_rover
__name__	color_blink	_rv
motor_right	motor_left	stay
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

ti_rover	ti_rover	ti_rover
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	
motors	stop	
backward		

.

Informações gerais

Ajuda online

education.ti.com/eguide

Selecione o seu país para obter mais informação sobre o produto.

Contacte a assistência técnica da TI

education.ti.com/ti-cares

Selecione o seu país para obter recursos técnicos ou assistência.

Informações da Assistência e Garantia

education.ti.com/warranty

Selecione o seu país para obter informações sobre a duração e os termos da garantia ou sobre a assistência ao produto.

Garantia Limitada. Esta garantia não afeta os seus direitos legais.