



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

Versão 5.7.0 Bundle 84CE-T

Saiba mais sobre a Tecnologia TI através da ajuda online em [education.ti.com/eguide](https://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.

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

© 2019 - 2021 Texas Instruments Incorporated

# Índice

<b>O que há de novo?</b> .....	<b>1</b>
O que há de novo na aplicação Python .....	1
<b>Aplicação Python</b> .....	<b>2</b>
Utilizar a aplicação Python .....	3
Gestão da memória do programa Python (PY AppVar) .....	3
Navegação na aplicação Python .....	4
Atividade de exemplo .....	5
Configurar uma sessão Python com os seus programas .....	7
<b>Áreas de trabalho Python</b> .....	<b>8</b>
Python File Manager .....	9
Python Editor .....	11
Shell (Interpretador) Python .....	14
<b>Suporte para edição rápida</b> .....	<b>18</b>
Utilizar o teclado Python .....	18
Utilizar o catálogo Python .....	20
Utilizar um mapa de caracteres [a A #] .....	21
<b>Menus [Fns], módulos e módulos Add-On</b> .....	<b>22</b>
[Fns...] Menus .....	22
[Fns...] Incorporado, operadores e teclados .....	22
[Fns...] Módulos .....	22
[Fns...] Módulos Add-On .....	28
Módulo Add-On [Fns...] ti_draw .....	29
Módulos Add-On [Fns...] ti_image .....	34
<b>Mensagens da aplicação Python</b> .....	<b>38</b>
Utilizar o TI-SmartView™ CE-T e a experiência Python .....	40
Utilizar o TI Connect™ CE para converter programas Python .....	41
<b>Qual é a experiência de programação Python?</b> .....	<b>42</b>
Módulos incluídos na TI-84 Plus CE-T Python Edition .....	42
<b>Programas exemplo:</b> .....	<b>49</b>
<b>Guia de referência para a experiência TI-Python</b> .....	<b>57</b>
Lista do CATÁLOGO .....	57
Lista do alfabeto .....	57

<b>Anexo</b> .....	<b>150</b>
Conteúdo do módulo, palavras-chave e incorporado da TI-Python .....	151
Mapeamento do teclado para wait_key() .....	165
<b>Informações gerais</b> .....	<b>166</b>
Ajuda online .....	166
Contacte a assistência técnica da TI .....	166
Informações da Assistência e Garantia .....	166

# O que há de novo?

## *O que há de novo na aplicação Python*

### TI-84 Plus CE-T Python Edition

---

#### Programação Python

- Acesse à aplicação Python a partir de [prgm] quando a aplicação Python estiver carregada. A aplicação Python também está listada em [2nd] [apps].
  - Mantenha-se atualizado em [education.ti.com/84cetupdate](http://education.ti.com/84cetupdate).
  - Encontre detalhes para a aplicação Python no guia de programação Python em [education.ti.com/eguide](http://education.ti.com/eguide).
- Colagem rápida de instruções de importação para módulos Add-On. Os módulos Add-On estão disponíveis nas atividades Python publicadas em [education.ti.com](http://education.ti.com).
- Os módulos Add-On New ti\_draw e ti\_image novos carregam com o CE Bundle.
  - Desenhe e utilize imagens nos programas Python.
- O menu do módulo ti\_system contém agora o método wait\_key() para facilidade de utilização.
- Os módulos ti\_hub e ti\_rover contêm o suporte mais recente ao TI-Innovator™ Hub sketch v 1.5.
  - Data Collection - recolher várias amostras de dados num único comando
  - Instruções compostas para sincronizar vários resultados
  - TI-RGB Array - controlar vários LEDs
  - Sound - utilizar um único comando para reproduzir bipes repetidos
  - Ranger - devolver "tempo de voo"

---

#### 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.

---

Para mais informações sobre a funcionalidade nova e atualizada, consulte [education.ti.com/84cetupdate](http://education.ti.com/84cetupdate).

---

# Aplicação Python

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

- [Utilizar a aplicação Python](#)
  - [Gestão da memória do programa Python \(PY AppVar\)](#)
- [Navegação na aplicação Python](#)
- [Atividade de exemplo](#)
- [Configurar uma sessão Python com os seus programas](#)

## 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](http://education.ti.com/84cetupdate) para atualizar a sua TI-84 Plus CE-T *Python Edition*.

### Gestão da memória do programa Python (PY AppVar)

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. Pode guardar AppVars Python no arquivo para a gestão da memória [2nd] [mem] 2:. Se o ecrã do gestor de ficheiros da aplicação Python não exibir um dos seus programas **PY AppVar**, pode mover um programa Python da calculadora **PY AppVar** entre a memória RAM ou de arquivo, conforme exibido. \* indica um ficheiro no arquivo. Prima [enter] para mover o ficheiro entre a memória RAM e de arquivo.



**Nota:** Se a sua calculadora for a TI-84 Plus CE *Python*, consulte [education.ti.com/84cetupdate](http://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.

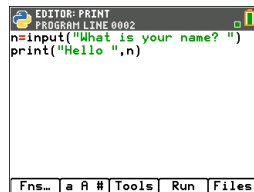
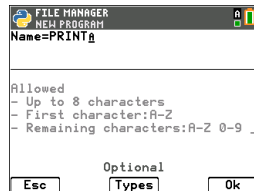
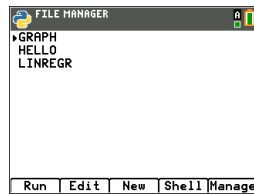
Introdução do nome do novo ficheiro

- 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!



Caracteres 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 <b>sto</b> quando o cursor for <code>_</code> .  rcl X 
Onde é que estes estão? input() print()	<b>[Fns...]</b> I/O 1:print() 2:input()
Onde estão as aspas duplas?	alpha [ " ]  mem " " 
Onde estão ( e )?	Utilize o teclado quando o cursor for <code>_</code> .  ( K ) L  

**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.



```

PYTHON SHELL
BEEP

>>> # Shell Reinitialized
>>> # Running PRINT
>>> from PRINT import *
What is your name? CE
Hello CE
>>> |
  
```

## 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](http://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](http://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.

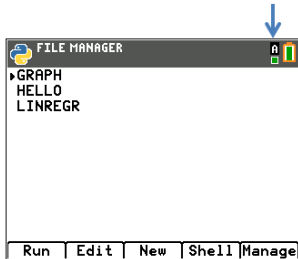
- [File Manager](#)
- [Editor](#)
- [Shell \(Interpretador\)](#)

## 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]	<b>[y=]</b>	Selecione um programa utilizando <b>[↑]</b> ou <b>[↓]</b> . A seguir, selecione [Run] para executar o seu programa.
[Edit]	<b>[window]</b>	Selecione um programa utilizando <b>[↑]</b> ou <b>[↓]</b> . A seguir, selecione [Edit] para exibir o programa no Editor para editar o programa.
[New]	<b>[zoom]</b>	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]	<b>[trace]</b>	Selecione [Shell] para exibir o prompt do Shell (Interpretador Python). O Shell (Interpretador) irá estar no estado atual.
[Manage]	<b>[graph]</b>	Selecione [Manage] para: <ul style="list-style-type: none"><li>• Ver o número de versão.</li><li>• Replicar, apagar ou mudar nome de um</li></ul>

## Teclas de atalho e menus do File Manager

Menus	Tecla premida	Descrição
		programa selecionado. <ul style="list-style-type: none"><li>• Ver o ecrã About.</li><li>• Sair da aplicação. Utilize também <b>[2nd]</b> <b>[quit]</b></li></ul>

## 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

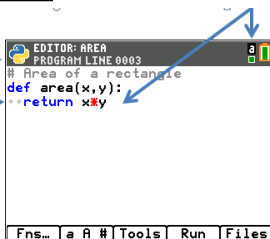
O Python Editor é exibido a partir de um programa selecionado no File Manager ou a partir do Shell (Interpretador). O Editor exibe palavras-chave, operadores, comentários, cadeias e indentações em cor. Estão disponíveis uma colagem rápida de palavras-chave e funções Python comuns, bem como uma introdução direta pelo teclado e uma introdução de caracteres [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 carácter.

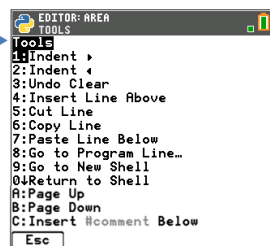
Localização do cursor na linha do programa.

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. <b>Nota:</b> [2nd] [catalog] também é útil para colagem rápida.

## Teclas de atalho e menus do Python Editor

Menus	Tecla premida	Descrição																				
[a A #]	<b>Window</b>	Selecione <b>[a A #]</b> para aceder a uma paleta de caracteres como forma alternativa de introduzir muitos caracteres.																				
[Tools]	<b>Zoom</b>	<p>Selecione [Tools] para aceder a funcionalidades para o ajudar da edição ou na sua interação com o Shell (Interpretador).</p> <table border="1"> <tbody> <tr> <td>1: Indentação ▶</td> <td>Indenta a linha do programa para a direita, o cursor desloca-se para o primeiro carácter da linha.</td> </tr> <tr> <td>2: Indentação ◀</td> <td>Reduz a indentação da linha do programa para a esquerda. O cursor desloca-se para o primeiro carácter da linha.</td> </tr> <tr> <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>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>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>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>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>8: Go to Program Line...</td> <td>Exibe o cursor no início da linha de programa especificada.</td> </tr> <tr> <td>9: Ir para Novo Shell (Interpretador)</td> <td>Exibe o Shell (Interpretador) reinicializado.</td> </tr> <tr> <td>0: Regressar a</td> <td>Exibe o Shell (Interpretador) no</td> </tr> </tbody> </table>	1: Indentação ▶	Indenta a linha do programa para a direita, o cursor desloca-se para o primeiro carácter da linha.	2: Indentação ◀	Reduz a indentação da linha do programa para a esquerda. O cursor desloca-se para o primeiro carácter da linha.	3: Anular limpar	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.	4: Inserir linha acima	Introduz uma linha por cima da linha de programa com o cursor. A linha irá indentar e exibir pontos de indentação quando apropriado.	5: Cortar linha	A linha atual do programa com o cursor está cortada. O cursor é exibido na linha de programa abaixo da linha de corte.	6: Copiar linha	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.	7: Colar linha em baixo	Cola a última linha de programa guardada na linha por baixo da posição do cursor.	8: Go to Program Line...	Exibe o cursor no início da linha de programa especificada.	9: Ir para Novo Shell (Interpretador)	Exibe o Shell (Interpretador) reinicializado.	0: Regressar a	Exibe o Shell (Interpretador) no
1: Indentação ▶	Indenta a linha do programa para a direita, o cursor desloca-se para o primeiro carácter da linha.																					
2: Indentação ◀	Reduz a indentação da linha do programa para a esquerda. O cursor desloca-se para o primeiro carácter da linha.																					
3: Anular limpar	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.																					
4: Inserir linha acima	Introduz uma linha por cima da linha de programa com o cursor. A linha irá indentar e exibir pontos de indentação quando apropriado.																					
5: Cortar linha	A linha atual do programa com o cursor está cortada. O cursor é exibido na linha de programa abaixo da linha de corte.																					
6: Copiar linha	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.																					
7: Colar linha em baixo	Cola a última linha de programa guardada na linha por baixo da posição do cursor.																					
8: Go to Program Line...	Exibe o cursor no início da linha de programa especificada.																					
9: Ir para Novo Shell (Interpretador)	Exibe o Shell (Interpretador) reinicializado.																					
0: Regressar a	Exibe o Shell (Interpretador) no																					



## Teclas de atalho e menus do Python Editor

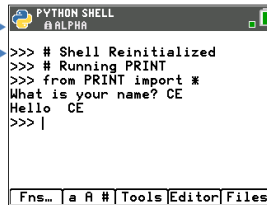
Menus	Tecla premida	Descrição	
		Shell (Interpretador)	estado atual.
		A: Subir uma página	Exibe 11 linhas de programa acima da posição atual do cursor, conforme disponível.
		B: Descer uma página	Exibe 11 linhas de programa abaixo da posição atual do cursor, conforme disponível.
		C: Inserir #comment abaixo	Inserir # numa linha nova por baixo da posição do cursor.
[Run]	<code>trace</code>	Selecione [Run] para executar o seu programa.	
[Files]	<code>graph</code>	Selecione [Files] para exibir o File Manager.	

## 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 caracteres [\[a A #\]](#). 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.



```
PYTHON SHELL
a ALPHA

>>> # Shell Reinitialized
>>> # Running PRINT
>>> from PRINT import *
What is your name? CE
Hello CE
>>> |
```

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



```
PYTHON SHELL
TOOLS

1: Run Last Program
2: Run...
3: Paste from Editor
4: Vars... [vars]
5: Clear Screen
6: New Shell
7: Go to Program Line...
8: Last Entry >>>
9: View History [2nd][*][v]
0 Tab Complete [2nd][enter]
A: from PROGRAM import *...
Esc
```

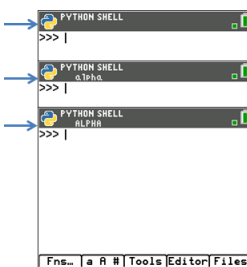
### Estados do cursor do Shell (Interpretador).

não  
alfabético

**[2nd] [alpha]** se  
necessário  
para alternar

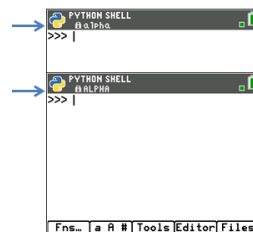
**[alpha]** alfa

**[alpha]**  
novamente  
ALFA



**[2nd] [alpha]**  
alpha  
bloqueado

**[alpha]**  
novamente  
ALPHA  
bloqueado



## Teclas de atalho e menus do Shell (Interpretador) Python

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

## Teclas de atalho e menus do Shell (Interpretador) Python

Menus	Tecla premida	Descrição
		<p>utilizando <code>ti_plotlib</code>, <code>ti_draw</code> ou <code>ti_image</code>, ao premir <code>[clear]</code> irá apagar o desenho no Shell. O histórico não irá ser apresentado no ecrã. Utilize <code>[2nd] [▲]</code> e <code>[2nd] [▼]</code> para ver o histórico conforme necessário.</p>
		<p><b>O: Tab Complete</b> <code>[2nd] [enter]</code></p> <p>Exibe os nomes das variáveis e funções disponíveis para acesso na atual sessão do Shell (Interpretador).</p> <p>Quando for introduzida uma letra de uma variável ou função disponível, prima <code>[2nd] [enter]</code> para completar automaticamente o nome se estiver disponível uma correspondência na sessão atual do Shell (Interpretador).</p>
		<p><b>A: from PROGRAM import *...</b></p> <p>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.</p> <p>Quando executado novamente na mesma sessão do Shell (Interpretador), a execução aparece como sem execução.</p> <p>Este comando também pode ser colado de <code>[2nd] [catalog]</code>.</p>
[Editor]	<code>[trace]</code>	Selecione [Editor] para exibir o Editor com os últimos programas no Editor. Se o Editor estiver vazio, pode exibir o Gestor de ficheiros.
[Files]	<code>[graph]</code>	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 os módulos `ti_plotlib`, `ti_draw` ou `ti_image` para desenhar para o Shell, prima `[clear]` para apagar o desenho e voltar ao prompt do Shell na parte de cima do ecrã. Para ver o histórico do Shell, utilize `[2nd] [▲]` e `[2nd] [▼]` para ver o histórico conforme necessário.

## **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.

# Suporte para edição rápida

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

Sugestões para introduções rápidas

- [Utilizar o teclado Python](#)
- [Utilizar o catálogo Python](#)
- [Utilizar um mapa de caracteres \[a A #\]](#)

## Utilizar o teclado Python

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 caracteres especiais por linha de teclado

The diagram illustrates various keyboard shortcuts for navigating and editing within the Python application. It features a central keyboard layout with arrows pointing to specific keys and their functions. The functions are detailed in text boxes around the keyboard.

**Navegação na aplicação**

- [2nd] acesso a tecla.
- [2nd] [quit] sair da aplicação.
- [del] retroceder na linha de edição.
- [del] eliminar do File Manager.

[alpha] alterna o estado do cursor: não alfabético, alfabético e ALFABÉTICO. [2nd] [alpha] bloqueia um estado alfabético. Selecionar letras do teclado.

[2nd] [link] cola \

[sto >] cola =

[2nd] [off] desliga CE. A aplicação fecha. A sessão Python é reinicializada como uma sessão nova quando a aplicação é iniciada. [on] liga CE; desliga o escurecimento automático; liga CE a partir de APD\*. Sessão Python retida a partir de escurecimento automático e APD. [on] interrompe um programa em execução no Shell.

**Teclas de setas**

- Navegação na linha do editor.
- Shell prompt e navegação no histórico.
- Brilho do ecrã.
- [2nd] [-<] ou [->] para o início ou fim da linha.

[clear] limpa uma linha de edição ou o ecrã Sobre.

[clear] não limpa menus. ([Esc] na aplicação.) [clear] limpa uma representação no Shell.

**Parêntesis e pontuação**

[ ] [ ] [ ]

[2nd] [ ] ou [ ]

[2nd] [ ] ou [ ]

[ , ]

[2nd] [ L3 ] cola #

[alpha] [ @ ] cola @

[alpha] [ \* ] cola aspas duplas

[ 2nd ] [ mem ] cola aspas simples

[alpha] [ space ] cola um espaço

[ . ] cola um ponto final ou um ponto decimal

[2nd] [ ans ] com \_ sublinhado

[alpha] [ ? ] cola ?

[2nd] [ entry ] separador Completo (Shell-Tools)

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

The diagram shows a TI-84 Plus calculator interface with several callouts pointing to specific keys and functions:

- [X,T,θ,n] X ou x** and **[2nde] [lists] Menu Lista**: Points to the **alpha** key.
- [math] Menu Módulos** and **[2nde] [test] Menu Operadores**: Points to the **math** key.
- [x<sup>^-</sup>1] Colar \*\*1**: Points to the **x<sup>-1</sup>** key.
- [2nd] [rc] Exibe ti\_system menu importar módulo ti\_system**: Points to the **2nd** key.
- [2nd] [catalog] Exibe o catálogo específico Python.**: Points to the **2nd** key.
- [2nd] [i] Exibe o tipo de complexo imaginário ja+b.**: Points to the **2nd** key.
- [var] Exibe variáveis disponíveis no Shell depois de um programa ser executado.**: Points to the **vars** key.
- [2nd] [π] [sin]; [cos]; [tan]; [2nd][sin]; [2nd][cos]; [2nd][tan] Exibe o menu Trig; importar o módulo de matemática**: Points to the **2nd** key.

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

The diagram shows a TI-84 Plus calculator interface with several callouts pointing to specific keys and functions:

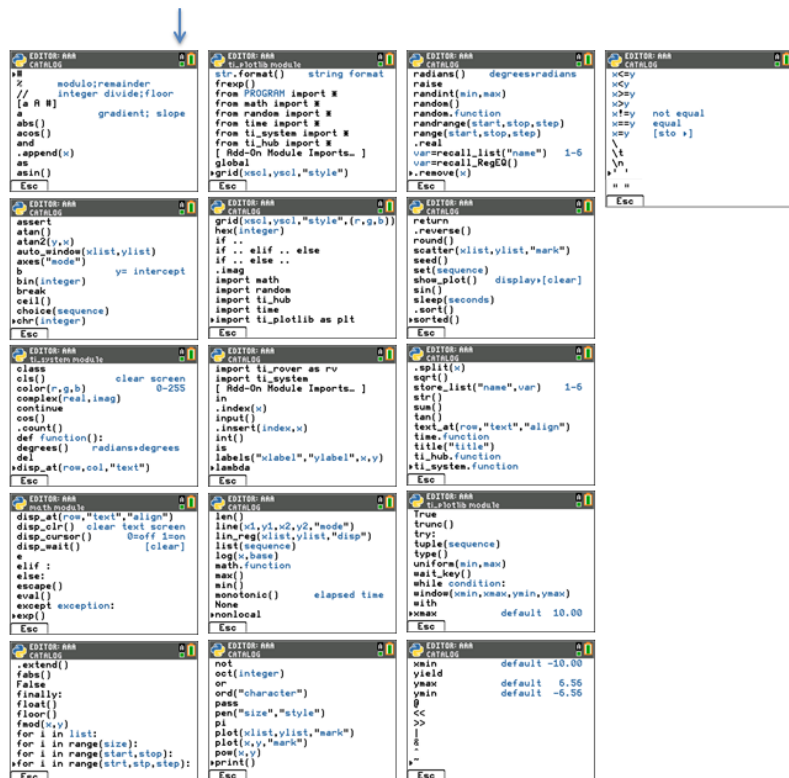
- [x^2] cola \*\*2**, **[2nd] [√] cola sqrt()**, and **[2nd] [EE] cola E**: Points to the **x<sup>2</sup>** key.
- [log] cola log(.10)** and **[2nd] [10^x] cola 10\*\***: Points to the **log** key.
- [ln] cola log()base e** and **[2nd] [e^x] cola exp()**: Points to the **ln** key.
- [sto >] cola =**: Points to the **sto >** key.
- [var] Exibe variáveis disponíveis no Shell depois de um programa ser executado.**: Points to the **vars** key.
- [clear] Limpa a área de desenho no Shell para métodos de desenho ti\_plotlib.**: Points to the **clear** key.
- [^] cola \*\***: Points to the **^** key.
- [+] cola /** and **[2nde] [e] cola e**: Points to the **+** key.
- [\*] cola \***: Points to the **\*** key.
- [-] cola -**: Points to the **-** key.
- [+] cola +**: Points to the **+** key.
- [enter]**: Points to the **enter** key.
- No File Manager, executa o programa selecionado.**
- No Editor, divide uma linha de programa.**
- Utilize [2nd] [enter] para introduzir uma linha em baixo.**

## Utilizar o catálogo Python

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.

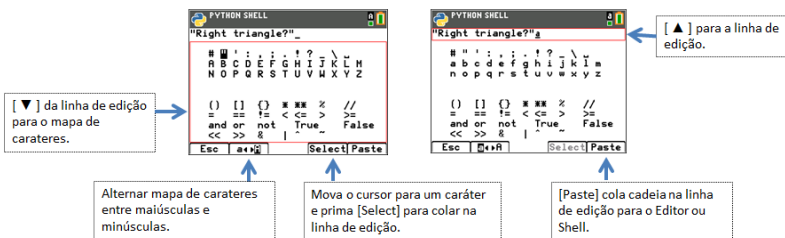
Quando estiver no catálogo, seleccione **[alpha]** e uma tecla de letra para exibir a listagem a iniciar nessa letra.





## Utilizar um mapa de caracteres [a A #]

O separador de atalho [a A #] para uma paleta 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 caracteres, o teclado está limitado.

# Menus [Fns], módulos e módulos Add-On

- [\[Fns...\] Menus](#)
- [\[Fns...\] Incorporado, operadores e teclados](#)
- [\[Fns...\] Módulos](#)
- [\[Fns...\] Módulos Add-On](#)

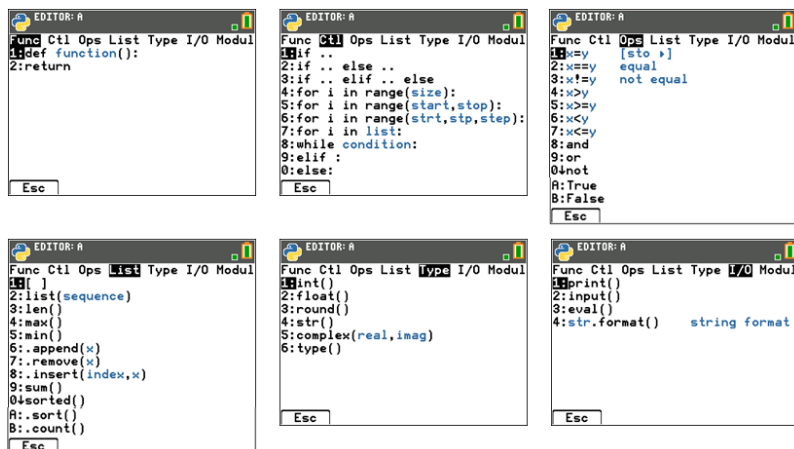
---

## [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 e dos módulos Add-On. 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 Utilizar o catálogo Python e Utilizar o teclado Python para métodos de introdução alternativos.

---

## [Fns...] Incorporado, operadores e teclados



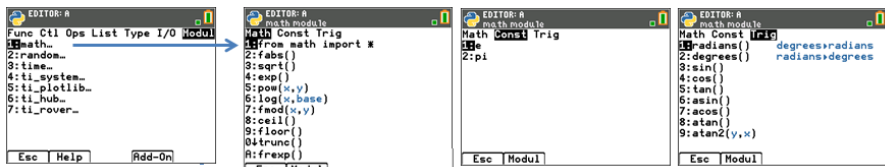
---

## [Fns...] Módulos

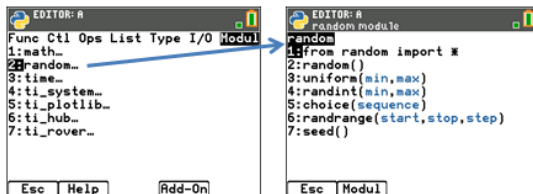
Ao utilizar uma função ou constante Python a partir de um módulo, utilize sempre uma instrução de importação para indicar a localização do módulo da função, método ou constante.

Ver [Qual é a experiência de programação Python?](#)

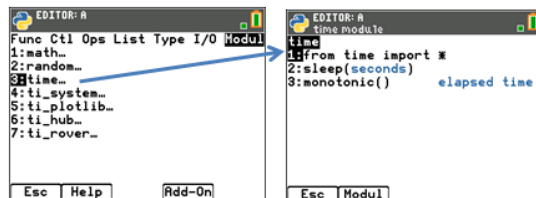
## [Fns...]>Modul: módulo math



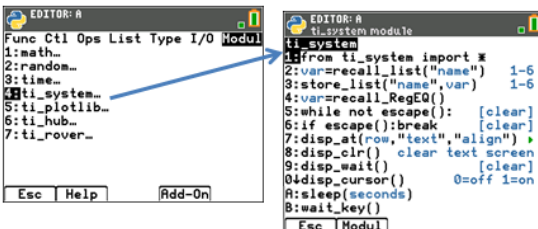
## [Fns...]>Modul: módulo random



## [Fns...]>Modul: módulo time

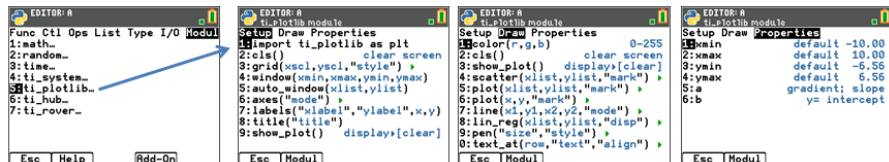


## [Fns...]>Modul: módulo ti\_system



Consultar: Mapeamento do teclado para `wait_key()`

## [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]`. Para ver o histórico do Shell, prima `[2nd]` `[↑]` e `[2nd]` `[↓]`.
- 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](http://education.ti.com).

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

```

EDITOR: #
Hub Modules
Func Ctrl Ops List Type I/O Modul
1: math.
2: random.
3: time.
4: ti_system.
5: ti_plotlib.
6: ti_hub.
7: ti_rover.
Esc Help Add-On

```

```

EDITOR: #
Hub Modules
Import Commands Ports Advanced
1: Hub Built-in devices.
2: Input devices.
3: Output devices.
4: Collect data.
Esc | Modul

```

```

EDITOR: #
Import Commands Ports Advanced
1: from ti_system import *
2: sleep(seconds)
3: disp_at(row,"text","align")
4: disp_clr() clear text screen
5: disp_wait() [clear]
6: disp_cursor() Onoff ison
7: while not escape(): [clear]
Esc | Modul

```

```

EDITOR: #
Import Commands Ports Advanced
1: OUT 1
2: OUT 2
3: OUT 3
4: IN 1
5: IN 2
6: IN 3
7: BB 1
8: BB 2
9: BB 3
0: BB 4
Esc | Modul

```

```

EDITOR: #
Import Commands Ports Advanced
1: from ti_hub import *
2: connect("obj","arg")
3: disconnect("obj","arg")
4: set("obj","arg")
5: read("obj","arg")
6: calibrate("obj","arg")
7: range("obj","arg")
8: version()
9: begin()
0: start()
Esc | Modul

```

```

EDITOR: #
País = Color → Modulo menu
Hub Built-in devices
1: Color RGB LED Output
2: Light Red LED Output
3: Sound Sound Output
4: Brightness Light Sensor Input
Esc | Import

```

```

EDITOR: #
País = DHT → Modulo menu
Input devices
1: DHT Digital Humidity & Temp
2: Ranger
3: Light Level
4: Temperature
5: Moisture
6: Magnetic
7: Vernier TI-SensorLink Input
8: Analog in
9: Digital in
0: Potentiometer
R: Thermistor
B: Loudness
C: Color Input
D: BB Port Breadboard Port
E: Hub Time Time Count from Hub
F: TI-RGB Array Input/Output
G: var.release()
Esc | Import

```

```

EDITOR: #
País = LED → Modulo menu
Output devices
1: LED
2: RGB
3: TI-RGB Array Input/Output
4: Speaker Speaker Output
5: Power
6: Continuous Servo
7: Analog out
8: Vibration Motor
9: Relay
0: Servo
R: Squarewave
B: Digital out
C: BB Port Breadboard Port
D: var.release()
Esc | Import

```

```

Collect data...
EDITOR: #
Sensors Collect Option Advanced
1: Brightness Light Sensor Input
2: DHT Digital Humidity & Temp
3: Ranger Distance (m) Light Sensor
4: Light Level Degrees C
5: Temperature
6: Moisture
7: Magnetic
8: Vernier TI-SensorLink Input
9: Analog in
R: Potentiometer
B: Thermistor
C: Loudness Breadboard Port
D: BB Port Breadboard Port
Esc | Modul

```

```

EDITOR: #
Sensors Collect Option Advanced
1: collect()
2: set_sensors(sensors_ 1-4
3: set_time(time) t>0,<=100s
4: set_rate(rate) r>0,<=10/s
5: set_max(max) True/False
6: start()
7: varc.measurements(ansr,opt)
8: varc.measurements("ansr")
9: varc.measurements("time")
Esc | Modul

```

```

EDITOR: #
Sensors Collect Option Advanced
1: DHT Temperature
2: DHT Humidity
3: Magnetic Level
4: Ranger Time of Flight
Esc | Modul

```

```

EDITOR: #
Sensors Collect Option Advanced
1: start(false)
2: while not c.done():
3: sleep(seconds)
Esc | Modul

```

## Módulo ti\_hub – Adicionar importação ao Editor e adicionar o módulo ti\_hub sensor ao menu Modul.

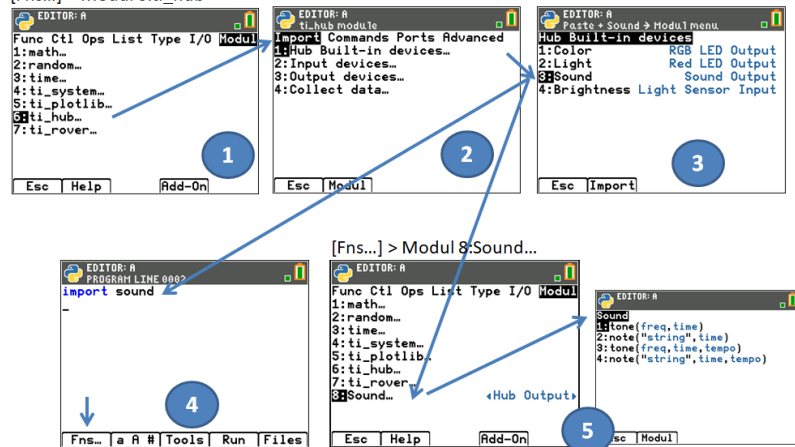
### Exemplo do ecrã: Importar som

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

[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\_rovers

Os métodos ti\_rovers 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](http://education.ti.com).

EDITOR: A  
ti\_rovers module  
Drive I/O Settings Commands  
1: Import ti\_rovers as rv  
2: forward(distance) unit  
3: backward(distance) unit  
4: left(angle) degrees  
5: right(angle) degrees  
6: stop() seconds  
7: resume() seconds  
8: stay(time) seconds  
9: to\_xy(x,y) degrees  
10: to\_polar(r,theta) < degrees  
A: rotate\_angle(angle) degrees  
B: forward\_time(time) seconds  
C: backward\_time(time) seconds  
D: left(angle,"unit") seconds  
E: right(angle,"unit") seconds  
F: forward\_time(T,S,"unit") seconds  
G: backward\_time(T,S,"unit") seconds  
H: forward(D,"unit",S,"unit") seconds  
I: backward(D,"unit",S,"unit") seconds  
A: disconnect\_rv() Disconnect  
Esc Modul

EDITOR: A  
ti\_rovers module  
Drive I/O Settings Commands  
1: units/s  
2: m/s  
3: revs/s  
4: units  
5: m  
6: revs  
7: degrees  
8: radians  
9: grads  
A: counterclockwise  
Esc Modul

EDITOR: A  
ti\_system module  
Drive I/O Settings Commands  
1: from ti\_system import \*  
2: sleep(seconds)  
3: disp\_at(row,"text","align")  
4: disp\_clr() clear text screen  
5: disp\_wait() [clear]  
6: disp\_cursor() 0=off 1=on  
7: while not escape(): [clear]  
8: wait\_until\_done()  
9: while not path\_done():  
0: position(x,y)  
A: position(x,y,heading,"unit")  
B: grid\_origin()  
C: grid\_m\_unit(scale\_value)  
D: path\_clear()  
E: zero\_gyro()  
Esc Modul

EDITOR: A  
ti\_rovers module  
Drive I/O Settings Commands  
1: Inputs:  
2: Outputs:  
3: Path:  
Esc Modul

EDITOR: A  
ti\_rovers module  
Inputs  
1: ranger\_measurement() meters  
2: color\_measurement() 1-9  
3: red\_measurement() 0-255  
4: green\_measurement() 0-255  
5: blue\_measurement() 0-255  
6: gray\_measurement() 0-255  
7: encoders\_gyro\_measurement() degrees  
8: gyro\_measurement() degrees  
9: ranger\_time() seconds  
Esc I/O

EDITOR: A  
ti\_rovers module  
Outputs  
1: color\_rgb(r,g,b) 0-255  
2: color\_blink(freq,time)  
3: color\_off()  
4: motor\_left(speed,time) ±255  
5: motor\_right(speed,time) ±255  
6: motors("ldir",L,"rdir",R,T)  
Esc I/O




EDITOR: A  
ti\_rovers module  
Path  
1: waypoint\_xythdrn()  
2: waypoint\_prev()  
3: waypoint\_eta()  
4: path\_done()  
5: pathlist\_x()  
6: pathlist\_y()  
7: pathlist\_time()  
8: pathlist\_heading()  
9: pathlist\_distance()  
0: pathlist\_revs()  
A: pathlist\_cmdnum()  
B: waypoint\_x()  
C: waypoint\_y()  
D: waypoint\_time()  
E: waypoint\_heading()  
F: waypoint\_distance()  
G: waypoint\_revs()  
Esc I/O

### 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.

## Suporte Python para TI-Innovator Sketch v1.5

 <p>ti_hub module</p>	<p>Built-in... &gt; Sound Module</p> <pre> EDITOR: 0 Sound 1:tone(freq,time) 2:note("string",time) 3:tone(freq,time,tempo) 4:note("string",time,tempo) </pre>	<p>Output &gt; Speaker Module</p> <pre> EDITOR: 0 Speaker 1:var:speaker("port") 2:var:tone(freq,time) 3:var:note("string",time) 4:var:tone(freq,time,tempo) 5:var:note("string",time,tempo) </pre>	<p>Input/Output &gt; TI-RGB Module</p> <pre> EDITOR: 0 TI-RGB Array 1:var:rgb_array() 2:var:set(led_position,r,g,b) 3:var:set_all(r,g,b) 0-255 4:var:all_off() 5:var:pattern(val) 0-65535 6:var:measurement() mA current 7:var:set(led_list,r,g,b) 0-255 </pre>
 <p>ti_hub module</p>	<p>Input &gt; Ranger Module</p> <pre> EDITOR: 0 Ranger 1:var:ranger("port") 2:var:measurement() 3:var:measurement_time() </pre>	<p>Input &gt; Magnetic Module (fix)</p> <pre> EDITOR: 0 Magnetic 1:var:magnetic("port") 2:var:magnet_cause() 3:var:measurement() 4:var:trigger(val) 0-16383 </pre>	<p>ti_rover &gt; I/O &gt; Inputs</p> <pre> EDITOR: 0 TI-Rover module Inputs 1:ranger_measurement() meters 1-9 2:color_measurement() 0-255 3:red_measurement() 0-255 4:green_measurement() 0-255 5:blue_measurement() 0-255 6:gray_measurement() 0-255 7:encoders_gyro_measurement() 8:gyro_measurement() degrees 9:ranger_time() seconds </pre> 

## [Fns...] Módulos Add-On

Os módulos Add-On melhoram a experiência do módulo da aplicação Python com funcionalidades adicionais e fácil acesso aos métodos Python adicionais a partir dos menus da aplicação Python.

Poderá notar um módulo Add-On a carregar, utilizando o TI Connect™ CE, como parte de uma atividade Python publicada em [education.ti.com](http://education.ti.com), como `ce_turtl`, `ce_chart`, `ce_box`, `ce_quivr` e `microbit`, dependendo da sua região. Irá necessitar da versão mais recente dos módulos Add-On publicados atualmente. Alguns módulos Add-On serão carregados na sua calculadora, como `ti_draw` e `ti_image`, quando fizer a atualização para o CE Bundle mais recente.

A aplicação irá exibir os menus do módulo Add-On no menu [Fns...] > Modul apenas se o programa no Editor iniciar com uma instrução de importação adequada.

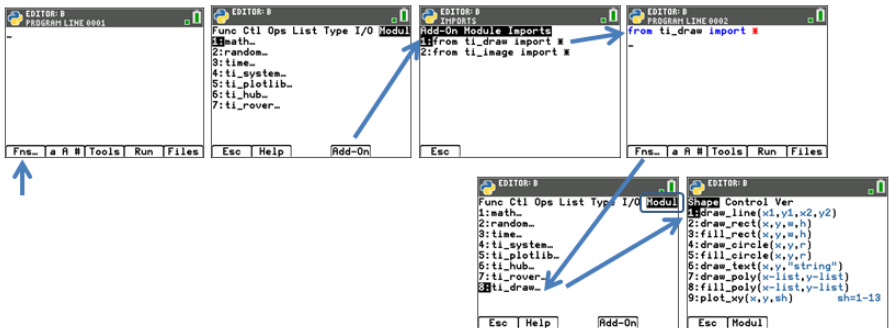
## Colar uma afirmação de importação do módulo Add-On para o Editor

Passos:

1. Criar um novo programa.
2. No Editor, seleccionar [Fns...] > Modul.
3. Seleccionar [Add-On] e quando um módulo Add-On estiver carregado na calculadora, é exibido um menu de afirmação de importação para os módulos.



4. Selecionar a afirmação de importação para colar para o Editor.
5. Selecionar [Fns...] > Modul para localizar os menus para o módulo Add-On importado.



### Factos:

- [Add-On Modules Imports...] é também listado em [2nd][catalog].
- Os módulos Add-On são ficheiros “AppVar” da calculadora guardados no arquivo e aparecem em [mem] como uma AppVar. É recomendado manter esses ficheiros na memória de arquivo para uma melhor experiência do módulo da aplicação Python.
- É executado um programa Python na aplicação Python a partir do gestor de ficheiros ou do Editor quando o programa “PY AppVar” estiver na RAM. Se um programa PY AppVar Python for colocado na memória de arquivo, não estará disponível para executar ou editar na aplicação Python.

### Módulo Add-On [Fns...] ti\_draw

O módulo `ti_draw` está incluído no CE Bundle mais recente. Utilize a funcionalidade [Fns...] > Modul [Add-On] para colar a afirmação de importação no seu programa. O menu `ti_draw` será então exibido no menu [Fns...] > Modul, tal como aqui ilustrado.

```
EDITOR: D
PROGRAM LINE 0002
from ti_draw import *
```

```
EDITOR: D
Func Ctl Ops List Type I/O Modul
1: math...
2: random...
3: time...
4: ti_system...
5: ti_plotlib...
6: ti_hub...
7: ti_rover...
8: ti_draw...
```

ti\_draw menus

```
EDITOR: D
Shape Control Ver
1: draw_line(x1,y1,x2,y2)
2: draw_rect(x,y,w,h)
3: fill_rect(x,y,w,h)
4: draw_circle(x,y,r)
5: fill_circle(x,y,r)
6: draw_text(x,y,"string")
7: draw_poly(x-list,y-list)
8: fill_poly(x-list,y-list)
9: plot_xy(x,y,sh) sh=1-13
```

```
EDITOR: D
Shape Control Ver
1: clear()
2: clear_rect(x,y,w,h)
3: set_color(r,g,b) 0-255
4: set_pen("size","style")
5: set_window(xmn,xmx,ymn,ymx)
6: show_draw() [clear]
```

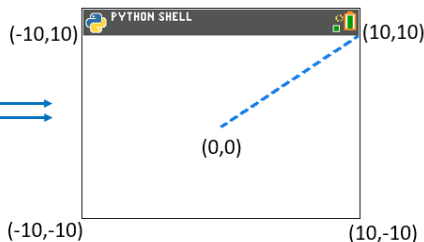
```
EDITOR: D
Shape Control Ver
1: -- version 1.0 --
2: ©2021 Texas Instruments
3: All Rights Reserved.
```

### Informação do programa ao utilizar ti\_draw:

- Após a afirmação de importação, utilize o método clear() para limpar a área de desenho Shell, se necessário.
- Os programas devem conter o comando show\_draw() para exibir o desenho ao executar o programa.
- Utilizando os métodos draw\_rect(), draw\_circle() ou draw\_poly() desenha o contorno da construção enquanto os métodos fill\_rect(), fill\_circle() e fill\_poly() preenchem o interior da forma especificada (dependendo do tamanho da caneta).
- Prima [clear] para limpar o desenho e voltar ao prompt do Shell. Note: O histórico do Shell pode ser visualizado com [2nd] [▲] e [2nd] [▼].
- Leia as informações dos menus Shape e Control na tabela em baixo. Os desenhos criados com os métodos do menu Shape dependem dos métodos do menu Control como set\_color() e set\_pen().
- Os **argumentos de coordenada** são coordenadas de píxel de ecrã ou definidos pelo método set\_window().

- Métodos ti\_draw utilizando coordenadas set\_window()

```
EDITOR: LINE#ND#H
PROGRAM LINE 0007
from ti_draw import *
clear()
set_color(25,125,225)
set_pen("medium","dashed")
set_window(-10,10,-10,10)
draw_line(0,0,10,10)
show_draw()_
```

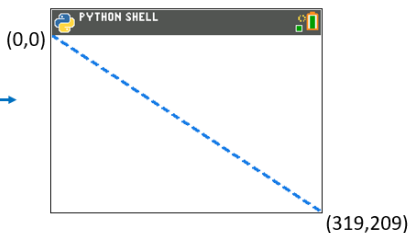


- Métodos ti\_draw utilizando coordenadas de píxel de ecrã

```

EDITOR: LINE
PROGRAM LINE 0004
from ti_draw import *
clear()
set_color(25,125,225)
set_pen("medium","dashed")
draw_line(0,0,319,209)
show_draw()



```



	Menu Shape	Descrição
1:	<code>draw_line(x1,y1,x2,y2)</code>	Desenha um segmento de reta com extremos nos pontos especificados $(x1,y1)$ e $(x2,y2)$ .
2:	<code>draw_rect(x,y,w,h)</code>	Desenha um retângulo com o canto superior esquerdo em $(x,y)$ com uma largura de $w$ pixels e altura de $h$ .
3:	<code>fill_rect(x,y,w,h)</code>	Preenche o interior de um retângulo com o canto superior esquerdo em $(x,y)$ com uma largura de $w$ pixels e altura de $h$ .
4:	<code>draw_circle(x,y,r)</code>	Desenha uma circunferência com o centro localizado em $(x,y)$ e um raio de $r$ pixels.
5:	<code>fill_circle(x,y,r)</code>	Desenha um círculo com o centro localizado em $(x,y)$ e um raio de $r$ pixels e preenchido com a cor especificada (com <code>set_color</code> ou preto se não definido).
6:	<code>draw_text(x,y,"string")</code>	Desenha uma cadeia de texto no ecrã com o canto superior esquerdo no texto a começar em $(x,y)$ .
7:	<code>draw_poly(x-list,y-list)</code>	Desenha um conjunto de linhas poligonais. As linhas são desenhadas utilizando o tamanho e cor de caneta atuais.
8:	<code>fill_poly(x-list,y-list)</code>	$x$ -list e $y$ -list têm de ter comprimento igual aos argumentos de lista numa lista de vértices $(x,y)$ . O polígono é desenhado ligando cada par de vértices e preenchendo a região com a cor de caneta atual.
9:	<code>poly_xy(x,y,sh) sh=1-13</code>	Utilizando os argumentos $x$ e $y$ como localização do ponto central será desenhado o valor de forma solicitado ( $sh$ ) em baixo. As formas são desenhadas com a cor de caneta atual.

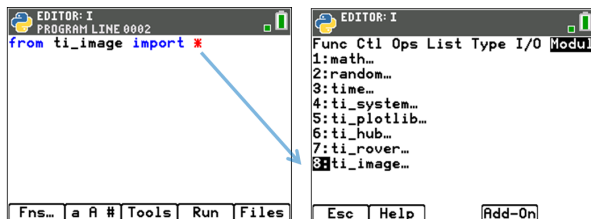
	Menu Shape	Descrição
		<p><b>Forma Descrição</b></p> <p>1 Círculo de raio 2</p> <p>2 Circunferência de raio 2</p> <p>3 Quadrado preenchido 3x3</p> <p>4 Fronteira do quadrado 3x3</p> <p>5 Marca x é desenhada</p> <p>6 Marca + é desenhada</p> <p>7 Píxel único</p> <p>8 Círculo com raio de 4 píxeis</p> <p>9 Círculo aberto com raio 4 de píxeis</p> <p>10 Círculo com raio de 6 píxeis</p> <p>11 Circunferência com raio 6 de píxeis</p> <p>12 Círculo com raio de 8 píxeis</p> <p>13 Circunferência com raio 8 de píxeis</p>

	Menu Control	Descrição
1:	clear()	Limpa a área de desenho no Shell. Este método tem de ser executado antes de desenhar para garantir que a área de desenho do Shell está limpa, para visualizar os resultados esperados.
2:	clear_rect(x,y,w,h)	<p>Preenche o interior de um retângulo com o canto superior esquerdo em (x,y) com uma largura de w e altura de h.</p> <p>A cor de preenchimento predefinida é branco. Depois de colar o método no Editor, o método pode aceitar um quinto argumento opcional para especificar uma cor diferente através da utilização de um énumlo a especificar o valor (r,g,b). Um terno válido (r,g,b) contém valores inteiros no intervalo de 0 a 255.</p>
3:	set_color(r,g,b) 0-255	define a cor da caneta de desenho utilizando o terno (r,g,b).
4:	set_pen("size","style")	Define a caneta de desenho para o "tamanho" e "estilo" para todos os

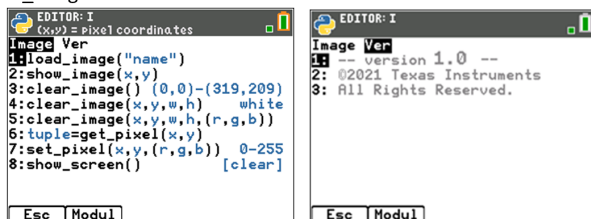
	Menu Control	Descrição
		<p>desenhos subsequentes até ser especificada uma alteração.</p> <p>Ao importar <code>ti_draw</code>, o tamanho é "fino", "médio" ou "espesso" e o estilo é "sólido", "ponteadado" ou "tracejado". Se não for especificado, os argumentos predefinidos são "fino" e "sólido". O assistente de argumentos &gt; irá ajudá-lo a preencher as cadeias de argumentos corretas.</p> <p><b>Nota:</b> Ao importar o módulo <code>ti_plotlib</code>, o argumento de estilo do método <code>pen()</code> é "sólido", "ponto" ou "traço".</p>
5:	<code>set_window</code> <code>(xmin,xmx,ymn,ymx)</code>	<p>Define a área de desenho com intervalos de coordenadas <code>[xmin,xmax]</code> e <code>[ymin,ymax]</code> com <code>(0,0)</code> no ponto médio dos intervalos. Note: Ser os valores de argumento estiverem fora da área de desenho especificada, não será dado qualquer erro.</p> <p>Se <code>set_window(xmin,xmax,ymin,ymax)</code> não for executado num programa, o tamanho em píxeis da janela é predefinido com <code>(xmin,xmax,ymin,ymax) = (0,319,0,209)</code> com <code>(0,0)</code> na coordenada de píxel do canto superior esquerdo da área.</p>
6:	<code>show_draw()</code> [ <code>clear</code> ]	<p>Tem de estar incluído para exibir o desenho. Prima [<code>clear</code>] para limpar o desenho e voltar ao prompt do Shell. Para ver o histórico do Shell, prima [<code>2nd</code>]  e [<code>2nd</code>] .</p>

## Módulos Add-On [Fns...] ti\_image

O módulo `ti_image` está incluído no CE Bundle mais recente. Utilize a funcionalidade [Fns...] > Modul [Add-On] para colar a afirmação de importação no seu programa. O menu `ti_image` será então exibido no menu [Fns...] > Modul, tal como aqui ilustrado.





### ti\_image menus



### Informação do programa ao utilizar `ti_image`:



- O módulo `ti_image` pode ser utilizado para
  - exibir a sua AppVar de imagem Python guardada na memória de arquivo na sua CE.
  - exibir retângulos preenchidos com cor especificados no local com coordenada de píxeis.
  - definir ou obter uma cor de píxel
  - limpar o interior de uma área de ecrã retangular.
  - limpar a área de desenho do ecrã no Shell a partir de coordenadas de píxeis (0,0) a (319,209).
- Após a afirmação de importação, utilize o método `clear_image()` para limpar a área de desenho Shell, se necessário.
- A AppVar de imagem Python é um ficheiro de imagem Python especial (\*.8xv).
  - Atualmente, é publicada uma biblioteca de imagens de currículo em <https://resources.t3europa.eu/t3europa-home?country=15&q=images&cHash=d50a2b65ab1b875dfa3ec11bca12154c>
  - Quando utilizar uma AppVar de imagem Python, a melhor prática é
    - guardar a AppVar de imagem Python na memória de arquivo. [2nd][mem]
    - saber as dimensões de píxeis da imagem utilizada durante a codificação.

- saber o nome exato da sua AppVar de imagem Python. Tem de introduzir o nome com a ortografia correta e com maiúsculas e minúsculas corretas. Não serão emitidos quaisquer erros para nomes de AppVar de imagem Python mal escritos.
- Mantenha-se atualizado com o TI Connect™ CE e o TI-SmartView™ CE mais recente em [education.ti.com/84cetupdate](http://education.ti.com/84cetupdate)
- Os argumentos de coordenada (x,y) são píxeis de coordenada APENAS nos métodos ti\_image e intervalo de (0,0) a (319,209). Leia mais informações sobre cada método na tabela em baixo. Alguns métodos podem ser colados no Editor em vários formatos quando são oferecidos argumentos opcionais.
- Prima [clear] para limpar o desenho e voltar ao prompt do Shell. O histórico do Shell pode ser visualizado com [2nd]  e [2nd] .

	Menu Control	Descrição
1:	<code>load_image("name")</code>	<p>Carrega uma AppVar de imagem Python "name" para utilização no programa</p> <p>A imagem Python "name" tem de estar exatamente no mesmo caso e ortografia qua a AppVar imagem Python. Note: Não é gerada qualquer mensagem de erro gerada se o nome AppVar NÃO for especificado exatamente como indicado.</p> <p>A imagem Python "name" será a imagem utilizada para ser exibida em <code>show_image(x,y)</code>.</p> <p><b>Melhores práticas:</b></p> <ul style="list-style-type: none"> <li>• Saber as dimensões de píxeis da imagem Python.</li> <li>• Dica de memória: As AppVars de imagem Python devem ser guardadas na memória de arquivo.</li> </ul>
2:	<code>show_image(x,y)</code>	<p>Exibe a imagem especificada em <code>load_image("name")</code>.</p> <p>Exibe a imagem com o canto superior esquerdo de píxeis (x,y) da área de desenho no Shell. As coordenadas dos píxeis do ecrã (x,y) variam de superior esquerdo como (0,0) para inferior direito (319,209).</p> <p>Se nenhum nome de imagem tiver sido</p>

	Menu Control	Descrição
		<p>especificado usando <code>load_image()</code>, é reportado um erro quando o programa é executado. Se "name" for introduzido incorretamente, não será exibido qualquer erro.</p> <p>Use o método <code>show_screen()</code> para reter a imagem no ecrã até <code>[clear]</code> para voltar ao Shell. Para ver o histórico do Shell, prima <code>[2nd] [↩]</code> e <code>[2nd] [⏏]</code>.</p>
3:	<code>clear_image() (0,0)-(319,209)</code>	<p>O método <code>clear_image()</code> sem argumentos é usado para limpar a área de desenho do Shell. A área de desenho é exibida como um ecrã branco.</p> <p>As coordenadas dos píxeis variam de superior esquerdo como (0,0) para inferior direito (319,209).</p> <p>Depois de toda a área de desenho ser "limpa" com este método, use o método <code>load_image("name")</code> e <code>show_image(x,y)</code> para exibir a imagem "name" conforme necessário.</p> <p>Quando utilizar também os métodos do módulo <code>ti_draw</code>, note que a cor <code>set_pen()</code> será definida para preto quando o método <code>ti_image</code>, <code>clear_image()</code>, for executado.</p>
4:	<code>clear_image(x,y,w,h) white</code>	<p>Dada uma coordenada de píxeis (x,y) para o canto superior esquerdo de um retângulo com w píxeis de largura e h píxeis de altura, este método irá "limpar" a área retangular interior para branco.</p>
5:	<code>clear_image(x,y,w,h,(r,g,b))</code>	<p>Dada uma coordenada de píxeis (x,y) para o canto superior esquerdo de um retângulo com w píxeis de largura e h píxeis de altura, este método irá "limpar" a área retangular interior para a cor RGB especificada no terno (r,g,b).</p>
6:	<code>tuple=get_pixel(x,y)</code>	<p>Devolve os valores RGB do píxel na coordenada de píxeis (x,y) como um énpulo (r,g,b).</p>
7:	<code>set_pixel(x,y,(r,g,b))</code>	<p>Define a cor do píxel na coordenada de píxeis (x,y) para a cor RGB especificada em (r,g,b).</p>



	Menu Control	Descrição
8:	<code>show_screen()</code> [clear]	<p>Este método deve ser utilizado para manter a exibição do desenho no ecrã quando se utiliza o módulo <code>ti_image</code>.</p> <p>Quando [clear] é pressionada após cada instância de <code>show_screen()</code>, o programa irá continuar a ser executado até, por fim, limpar o ecrã do prompt do Shell.</p> <p>Para ver o histórico do Shell, utilize [2nd]  e [2nd] . Consulte Shell &gt; [Tools] para mais opções do Shell.</p>

# 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

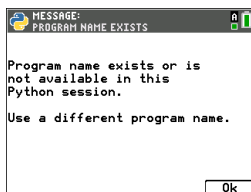
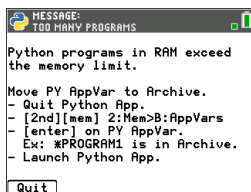
Irá ser questionado para se certificar de que pretende sair da aplicação. Sair da aplicação irá parar a sua sessão Python. Quando executar novamente a aplicação Python, os seus programas e módulos Python AppVar irão sincronizar. O Shell (Interpretador) irá reiniciar.

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.



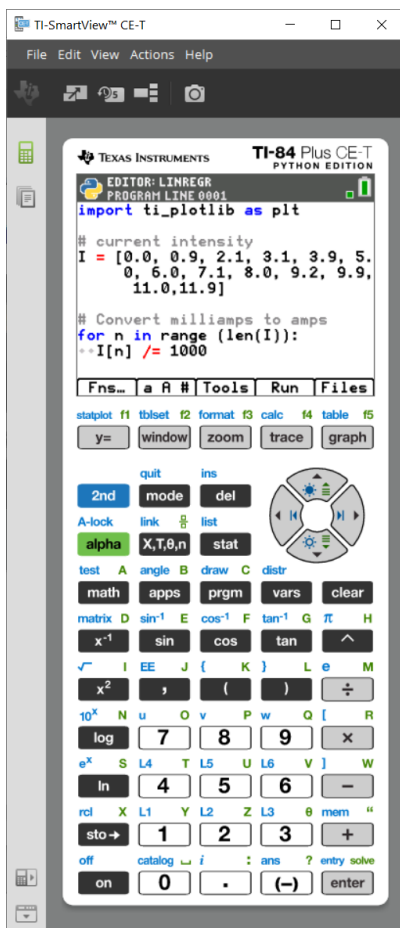
## Utilizar o TI-SmartView™ CE-T e a experiência Python

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](http://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 noutra 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

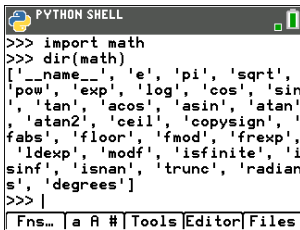
Para a lista de módulos incluídos nesta versão, consulte:

[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.



```
PYTHON SHELL
>>> import math
>>> dir(math)
['__name__', 'e', 'pi', 'sqrt',
'pow', 'exp', 'log', 'cos', 'sin',
'tan', 'acos', 'asin', 'atan',
'atan2', 'ceil', 'copysign', 'fabs',
'floor', 'fmod', 'frexp', 'ldexp',
'modf', 'isfinite', 'isinf', 'isnan',
'trunc', 'radians', 'degrees']
>>> |
Fns... a A # Tools Editor Files
```

módulo math



```
PYTHON SHELL
>>> import random
>>> dir(random)
['__name__', 'seed', 'getrandbit',
's', 'randrange', 'randint', 'choice',
'random', 'uniform']
>>> |
Fns... a A # Tools Editor Files
```

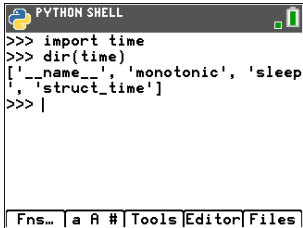
módulo random

---



---

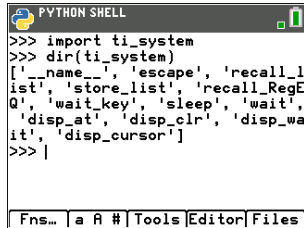
Estes ecrãs apresentam o conteúdo dos módulos para `time` e `ti_system`.



```
PYTHON SHELL
>>> import time
>>> dir(time)
['__name__', 'monotonic', 'sleep',
 'struct_time']
>>> |
```

Fns... a A # Tools Editor Files

`time`



```
PYTHON SHELL
>>> import ti_system
>>> dir(ti_system)
['__name__', 'escape', 'recall_list', 'store_list', 'recall_RegE',
 'wait_key', 'sleep', 'wait', 'disp_at', 'disp_clr', 'disp_wait',
 'disp_cursor']
>>> |
```

Fns... a A # Tools Editor Files

`ti_system`

---

---

Estes ecrãs apresentam o conteúdo do módulo para `ti_plotlib`.



```
PYTHON SHELL
>>> import ti_plotlib
>>> dir(ti_plotlib)
['lin_reg', 'strtest', 'escape',
 '_excpt', 'text_at', '_clipseg',
 'show_plot', 'tilocal', 'pen',
 'sys', 'xmin', 'ymax', 'yscl',
 '_xy', '_rdelta', '_ydelta', 'scatter',
 'a', '_pencolor', 'write', 'b',
 '_xytest', 'window', '_mark',
 'line', 'monotonic', '_numtest',
 'ymin', 'tiplotlibException',
 'labels', 'cls', 'sqrt', 'xscl',
 'axes', 'grid', '_sema', '_pensize',
 'plot', 'isnan', 'color', 'title',
 '_xdelta', '_penstyle', '__name__',
 'copysign', 'gr', 'xmax', 'sleep',
 'auto_window']
>>> |
Fns... | a A # |Outils|Éditer|Script
```

`ti_plotlib`

---

---

Este ecrã apresenta o conteúdo do módulo para ti\_hub.



```
PYTHON SHELL
>>> import ti_hub
>>> dir(ti_hub)
['_name_', 'connect', 'disconnect', 'set', 'read', 'calibrate', 'range', 'version', 'about', 'isti', 'what', 'who', 'begin', 'wait', 'sleep', 'start', 'last_error', 'get', 'send', 'tihubException']
>>> |
Fns... | a R # | Tools | Editor | Files
```

ti\_hub

---

---

Estes ecrãs apresentam o conteúdo do módulo para `ti_rover`.



```
PYTHON SHELL
>>> import ti_rover
>>> dir(ti_rover)
['motor_right', 'to_angle', 'to_xy', 'red_measurement', 'rvmove
ment', 'gray_measurement', 'exc
pt', 'pathlist_time', 'waypoint_
prev', 'ti_hub', 'waypoint_eta',
'to_polar', 'grid_unit', 'col
or_off', 'path_clear', 'rv', 'g
reen_measurement', 'motors', 'wa
ypoint_time', 'backward', 'color
_blink', 'motor_left', 'waypoint
_heading', 'motor', 'gyro_mesu
rement', 'wait_until_done', 'enc
oders_gyro_measurement', 'pathli
st_distance', 'position', 'blue_
measurement', 'forward', 'waypoi
nt_distance', 'grid_origin', 're
sume', 'path_done', 'disconnect_
rv', 'backward_time', 'zero_gyro
', 'rv_connected', 'stop', 'sta
y', 'waypoint_xythdrn', 'ranger_
measurement', 'left', 'pathlist_
cmdnum', 'waypoint_y', 'waypoint
_x', 'pathlist_y', 'pathlist_x',
'__name__', 'right', 'color_rgb
', 'pathlist_revs', 'color_mesu
rement', 'pathlist_heading', 'fo
rward_time', 'waypoint_revs']
>>> |
Fns... a A # Tools Editor Files
```

`ti_rover`

---

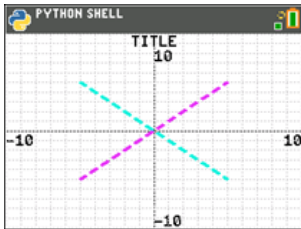
## Programas exemplo:

Utilize os seguintes programas exemplo para se familiarizar com os métodos da seção 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  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)

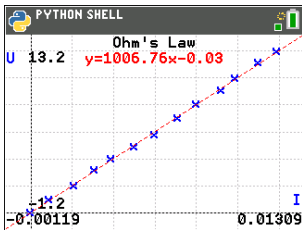
```
import ti_plotlib as plt

# 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  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).

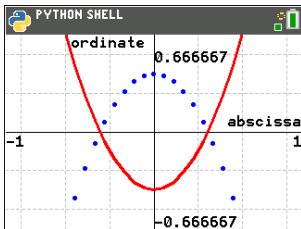
def f(x):
**return 3*x**2-.4

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 tiplotlib 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)
```



The screenshot shows a text editor window titled "EDITOR: DASH1" with "PROGRAM LINE 0001" at the top. The code is color-coded: keywords in blue, strings in green, and comments in red. The code is identical to the one shown in the previous block. At the bottom of the editor window, there is a menu bar with the following items: "Fns...", "a", "A", "#", "Tools", "Run", and "Files".

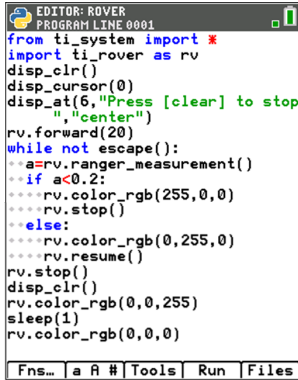


---

## 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)
```

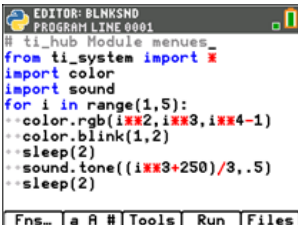


The screenshot shows a window titled "EDITOR: ROVER" with "PROGRAM LINE 0001" at the top. The code is displayed in a monospaced font with syntax highlighting: keywords in blue, comments in red, strings in green, and code in black. The code is identical to the one shown in the previous block. At the bottom of the window, there is a menu bar with the following items: "Fns...", "a A #", "Tools", "Run", and "Files".

---

## BLNKSND - Programa exemplo para o TI-Innovator™ Hub

Consultar: [\[Fns...\]>Modul: módulo ti\\_hub](#)



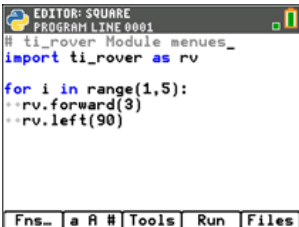
```
EDITOR: BLNKSND
PROGRAM LINE 0001
# ti_hub Module menus_
from ti_system import *
import color
import sound
for i in range(1,5):
  --color.rgb(i**2,i**3,i**4-1)
  --color.blink(1,2)
  --sleep(2)
  --sound.tone((i**3+250)/3,.5)
  --sleep(2)
```

Fns... a A # Tools Run Files

---

## SQUARE - Programa exemplo para o TI-Innovator™ Rover

Consultar: [\[Fns...\]>Modul módulo ti\\_rover](#)



```
EDITOR: SQUARE
PROGRAM LINE 0001
# ti_rover Module menus_
import ti_rover as rv
for i in range(1,5):
  rv.forward(3)
  rv.left(90)
```

Fns... a A # Tools Run Files

## STOP\_GO - Sample ti\_draw, ti\_image, time Program

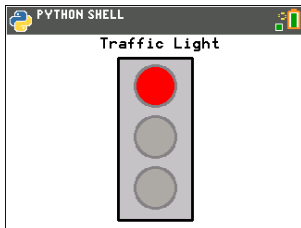
See: [\[Fns...\]>Modul \[Add-On\]](#)

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

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

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

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



# Guia de referência para a experiência TI-Python

A aplicação Python contém menus de funções, classes, controles, 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

## A

### #

**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 paleta de caracteres [a A #].

Inclui caracteres acentuados como ç à â è é ê ë ì ï ô õ ù û

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

## a **gradiente; declive**

**Módulo:** ti\_plotlib

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** plt.a **gradiente; declive**

[Fns...]>Modul  
ou [\[math\]](#)  
5:ti\_plotlib...>  
Properties  
5:a

**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](#).

os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#) ou no menu ti\_plotlib Setup.

## abs()

**Módulo:** Built-in

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** abs(x)

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

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

**Exemplo:**

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

## acos()

**Módulo:** math

[\[2nd\]](#) [7:acos\(\)](#)

**Sintaxe:** acos(x)

**Descrição:** Devolve o arco cosseno de x em radianos.

[\[2nd\]](#) [\[catalog\]](#)

**Exemplo:**

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

[Fns...]  
Modul  
1:math... >  
Trig  
7:acos()

**Exemplo alternativo:** [Tools] > 6:New Shell

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

os comandos  
de  
importação  
encontram-  
se em  
[\[2nd\]](#) [\[catalog\]](#)

## and

**Palavra-chave**

[\[2nd\]](#) [\[test\]](#)

**Sintaxe:** x and y

Ops 8:and

**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.

[Fns...] > Ops  
8:and

**Exemplo:**

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

[\[2nd\]](#) [\[catalog\]](#)

[a A #]



## **.append(x)**

**Módulo:** Built-in

[2nd] [list]

**Sintaxe:** listname.append(item)

List  
6: .append(x)

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

[2nd] [catalog]

**Exemplo:**

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

[Fns...] > List  
6:..append(x)

## **as**

**Palavra-chave**

[2nd] [catalog]

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

## **asin()**

**Módulo:** math

[sin] 6:asin()

**Sintaxe:** asin()

**Descrição:** Devolve o arco seno de x em radianos.

[2nd] [catalog]

**Exemplo:**

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

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

**Exemplo alternativo:**

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

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

[\[2nd\]](#) [catalog]

**Sintaxe:** plt.auto\_window(xlist,ylist)

[Fns...]>Modul  
ou [math](#)

**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().

5:ti\_plotlib...>  
Setup  
5:auto\_window  
( )

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

**Exemplo:**

Consulte o programa de amostra: [LINREGR.](#)

os comandos de  
importação  
encontram-se  
em [\[2nd\]](#)  
[catalog] ou no  
menu  
ti\_plotlib Setup.

## axes("mode")

**Módulo:** ti\_plotlib

**[2nd]** [catalog]

**Sintaxe:** plt.axes("mode")

[Fns...]>Modul

ou **[math]**

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

5:ti\_plotlib...>

Setup

**Argumento:**

6:axes()

**Opções de argumento "mode":**

---

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

---

os comandos de

importação

encontram-se

em **[2nd]** [catalog]

ou no menu

ti\_plotlib Setup.

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 [LINREG](#).

**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

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

**Sintaxe:** ceil(x)

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

[2nd](#) [\[catalog\]](#)

**Exemplo:**

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

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

os comandos de  
importação  
encontram-se em  
[2nd](#) [\[catalog\]](#)

### choice(sequence)

**Módulo:** random

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

**Sintaxe:** choice(sequence)

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

[2nd](#) [\[catalog\]](#)

**Exemplo:**

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

[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 carácter unicode.

Para mais informações, consulte a documentação Python.

**Exemplo:**

```
>>> chr(40)
'('
>>> chr(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()
```

**Exemplo:**

Consulte o programa exemplo: [GRAPH](#).

os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#) ou no menu ti\_plotlib

**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.

**Exemplo:**Consulte o programa exemplo: [COLORLIN](#).os comandos de  
importação  
encontram-se  
em [\[2nd\]](#) [\[catalog\]](#)  
ou no menu ti\_  
plotlib Setup.**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

**Sintaxe:** cos(x)

4: cos()

**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, énpulo, bytes, str, bytearray, ou objeto array.array.

**Exemplo:**

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

## D

### 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():

**Exemplo:**

[Fns...]>Func  
2:return

```
>>> 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

[2nd] [catalog]

**Sintaxe:** disp\_at(row,col,"text")

[2nd] [rc]

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

ti\_system  
7:disp\_at()

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

[Fns...]>Modul  
ou [math]  
4:ti\_system  
7:disp\_at()

### Argumento:

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

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

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()
```

## disp\_at(row,"text","align")

**Módulo:** ti\_system

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** disp\_at(row,"text","align")

[\[2nd\]](#) [\[rc\]](#)

**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.

ti\_system  
7:disp\_at()

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.

[Fns...]>Modul  
ou [\[math\]](#)  
4:ti\_system  
7:disp\_at()

**Argumento:**

os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#) ou no menu ti\_system Modul.

---

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()
```

## disp\_clr() apagar ecrã de texto

**Módulo:** ti\_system

[2nd] [catalog]

**Sintaxe:** disp\_clr() [apagar ecrã de texto](#)

[2nd] [rc1]

**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).

ti\_system  
8:disp\_clr()

**Exemplo:**

Programa exemplo:

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

[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**

[\[2nd\]](#) [\[rc\]](#)

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

`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

[2nd] [catalog]

**Sintaxe:** disp\_wait() [clear]

[2nd] [rc]

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

ti\_system  
9:disp\_wait()

**Exemplo:**

[Fns...]>Modul  
ou [math]  
4:ti\_system  
9:disp\_wait()

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.

## E

### e

**Módulo:** math

[\[2nd\]](#) [\[e\]](#) (acima de [☺](#))

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

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

[Fns...] > Modul  
1:math...  
> Const 1:e

**Exemplo:**

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

**Exemplo alternativo:**

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

### elif :

**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:



## 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:

## escape()

**Módulo:** ti\_system

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** escape()

Como linha de programa:

**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 repostado 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.

[\[2nd\]](#) [\[rc1\]](#)

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

[2nd](#) [e<sup>x</sup>]  
(acima de  
[ln](#))

**Sintaxe:** exp(x)

**Descrição:** Devolve e\*\*x.

**Exemplo:**

[2nd](#) [catalog]

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

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

**Exemplo alternativo:** [Tools] > 6:New Shell

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

os  
comandos  
de  
importação  
encontram-  
se em  
[2nd](#)  
[\[catalog\]](#).

## .extend()

**Módulo:** Plano integrado

[2nd](#) [catalog]

**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]
```

## F

### fabs()

**Módulo:** math

[2nd]

**Sintaxe:** fabs(x)

[catalog]

**Descrição:** Devolve o valor absoluto de x

**Exemplo:**

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

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

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

Ver  
também a  
função  
incorporada  
abs().

### False

**Palavra-chave**

[2nd] [test] (acima  
de [math])

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

[2nd] [catalog]

**Exemplo:**

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

[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

**Sintaxe:** floor(x)

1:math  
9:floor()

**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

[math](#) Modul

**Sintaxe:** fmod(x,y)

1:math  
7:fmod()

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

[2nd](#) [catalog]

Pode não devolver o mesmo resultado que x%y.

**Exemplo:**

[Fns...] > Modul  
1:math...  
7:fmod()

```
>>>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
```

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

**Consulte também:** x%y.

## for i in list:

**Palavra-chave**

[Fns...] Ctl

**Sintaxe:** for i in list:

7:for i in list:

**Descrição:** Utilizado para iterar através de elementos da lista.

[2nd](#) [catalog]

**Exemplo:**

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

## for i in range(size):

**Palavra-chave**

[Fns...] Ctl

**Sintaxe:** for i in range(size)

4:for i in range  
(size):

**Descrição:** Utilizado para iterar através de um intervalo.

[2nd](#) [catalog]

**Exemplo:**

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

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

**Palavra-chave**

[Fns...] Ctl

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

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

**Descrição:** Utilizado para iterar através de um intervalo.

[2nd](#) [catalog]

**Exemplo:**

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

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

**Palavra-chave**

[Fns...] Ctl

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

6: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

[math](#) Modul

**Sintaxe:** frexp(x)

1:math  
A:frexp()

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

[2nd](#) [catalog]

**Exemplo:**

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

[Fns...] > Modul  
1:math  
A:frexp()

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

## from PROGRAM import \*

**Palavra-chave**

Shell [Tools]

**Sintaxe:** from PROGRAM import \*

A:from  
PROGRAM  
import \*

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

[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**

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** from time import \*

[\[math\]](#) Modul  
3:time...

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

1:from time  
import \*

**Exemplo:**

[Fns...]>Modul

Consulte o programa exemplo: [DASH1](#).

3:time...

1:from time

import \*

## from ti\_system import \*

**Palavra-chave**

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** from ti\_system import \*

[\[math\]](#) Modul  
4:ti\_system...

**Descrição:** Utilizado para importar todos os métodos do módulo ti\_system.

1:from system  
import \*

**Exemplo:**

[Fns...]>Modul

Consulte o programa exemplo: [REGEQ1](#).

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](#).

## G

### global

**Palavra-chave**

[2nd](#) [catalog]

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

Para mais informações, consulte a documentação `CircuitPython`

### `grid(xsc1,ysc1,"style")`

**Módulo:** `tiplotlib`

[2nd](#) [catalog]

**Sintaxe:** `plt.grid(xsc1,ysc1,"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.

Valor predefinido para `xsc1` ou `ysc1` = 1.0.

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

**Exemplo:**

Consulte os programas exemplo: [COLORLIN](#) ou [GRAPH](#).

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

## `grid(xsc1,ysc1,"style",(r,g,b))`

**Módulo:** `tiplotlib`

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** `plt.grid(xsc1,ysc1,"style",(r,g,b))`

[Fns...]>Modul

ou [\[math\]](#)

5:tiplotlib...>

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

`tiplotlib`

Setup.

Valor predefinido para `xsc1` ou `ysc1` = 1.0.

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

Se os valores `xsc1` ou `ysc1` 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](#).

## H

### 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...] &gt; Ctl

1:if..

2:if..else..

3:if..elif..else

9:elif :

0:else:



## if..elif..else..

**Palavra-chave**

[2nd] [catalog]

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

[Fns...] > Ctl

if :

1:if..

\*\*

2:if..else..

elif :

3:if..elif..else

\*\*

9:elif :

else:

0: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
```

## 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 acessado 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 acessado 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 acessado 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**

**2nd** [catalog]

**Sintaxe:** import ti\_plotlib as plt

**math** Modul  
5:ti\_plotlib...  
1: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.

[Fns...]>Modul  
5:ti\_plotlib...  
1:import ti\_plotlib  
as plt

**Exemplo:**

Consulte o programa exemplo: [COLORLIN](#).

## import ti\_rover as rv

**Palavra-chave**

**2nd** [catalog]

**Sintaxe:** import ti\_rover as rv

**math** Modul  
7:ti\_rover...  
1: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.

[Fns...]>Modul  
7:ti\_rover...  
1:import ti\_rover  
as rv

**Exemplo:**

Consulte o programa exemplo: [ROVER](#).

## import ti\_system

**Palavra-chave**

[2nd] [catalog]

**Sintaxe:** import ti\_system

**Descrição:** O módulo ti\_system é acessado 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 (digit ar15)
15.0 (resultado flutuante 15.0)
```

## `.insert(index,x)`

**Módulo:** Plano integrado

[2nd](#) [\[list\]](#) List

**Sintaxe:** listname.insert(index,x)

8:.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.

**Nota:** plt.labels(" | ", "", 12, 2) serão coladas com as predefinições de linha x e y, 12, 2, que depois podem ser modificadas no programa.os comandos de importação encontram-se em **[2nd]** [catalog] ou no menu ti\_plotlib Setup.**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:** tiplotlib

[2nd] [catalog]

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

[Fns...]>Modul ou  
[math]

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

5:tiplotlib...> Draw  
7:line ou vetor

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

**Argumentos:**

x1,y1, x2,y2 são flutuantes reais.

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

"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.

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.

[2nd](#) [\[catalog\]](#)

**Exemplo:**

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

[Fns...] > List  
2:list(sequence)

**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

[\[2nd\]](#) [\[log\]](#) para log(x,10)

**Sintaxe:** log(x,base)

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

[\[2nd\]](#) [\[ln\]](#) para log(x) (natural log)

**Exemplo:**

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

[\[math\]](#) Modul  
1:math...  
6:log(x,base)

[\[2nd\]](#) [\[catalog\]](#)

[Fns...] > Modul  
1:math...  
6:log(x,base)

os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#)

## M

### 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:momonic  
( )

**Exemplo:**

Programa exemplo:

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

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

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

## N

### 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
```

[a A #]

## O

### 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 carácter. 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:** `tiplotlib`[\[2nd\]](#) [\[catalog\]](#)**Sintaxe:** `plt.pen("size", "style")`

[Fns...]&gt;Modul ou

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

[math](#)

5:tiplotlib...&gt;

Draw

9:pen()

**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 tiplotlib 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

[2nd] [catalog]

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

[Fns...]>Modul  
ou [math]  
5:ti\_plotlib...>  
Draw  
5:Connected  
Plot with Lists

**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
- 

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

**Exemplo:**

Consulte o programa exemplo: [LINREGR](#).

## `plot(x,y,"mark")`

**Módulo:** ti\_plotlib

`[2nd]` `[catalog]`

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

`[Fns...]>Modul` ou

`[math]`

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

`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

[math](#) Modul

**Sintaxe:** pow(x,y)

1:math  
5:pow(x,y)

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

[2nd](#) [catalog]

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

**Exemplo:**

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

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

**Exemplo utilizando:** Plano integrado:

[Tools] > 6:New Shell

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

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

## print()

**Módulo:** Plano integrado

[2nd](#) [catalog]

**Sintaxe:** print(argument)

**Descrição:** Exibe o argumento como uma cadeia.

[Fns...] > I/O  
1:print()

**Exemplo:**

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

## R

### `radians()` [degree](#) ▶ [radians](#)

**Módulo:** `math`

[\[sin\]](#) Trig  
1:`radians()`

**Sintaxe:** `radians(x)`

**Descrição:** Converte o ângulo `x` em degrees para radians.

[\[2nd\]](#) [\[catalog\]](#)

**Exemplo:**

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

[\[Fns...\]](#) > Modul  
1:`math...` > Trig  
1:`radians()`

### `raise`

**Palavra-chave**

[\[2nd\]](#) [\[catalog\]](#)

**Sintaxe:** `raise exception`

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

## randint(min,max)

**Módulo:** random

[math](#) Modul

**Sintaxe:** randint(min,max)

2:random

4:randint

(min,max)

**Descrição:** Devolve um número inteiro aleatório entre min e 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

[\[math\]](#) Modul

**Sintaxe:** random()

2:random...

Random

2: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
```

[Fns...] > Modul

2:random...

Random

2:random()

**Exemplo alternativo:**

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

[\[2nd\]](#) [catalog]

Os resultados variam em função de uma saída aleatória.

os comandos de importação encontram-se em [\[2nd\]](#) [catalog]

## random.function

**Módulo:** random

[\[2nd\]](#) [catalog]

**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.

## randrange(start,stop,step)

**Módulo:** random

[math](#) Modul

**Sintaxe:** randrange(start,stop,step)

2:random...

Random

6: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
```

[math](#) Modul

2:random...

Random

6:randrange

(start,stop,step)

**Exemplo alternativo:**

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

[2nd](#) [catalog]

Os resultados variam em função de uma saída aleatória.

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 stop 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`

`[2nd]` `[catalog]`

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

`[2nd]` `[rc]`

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

`ti_system`

`4:var=recall_list()`

**Argumento:** "name"

`[Fns...]>Modul` ou

`[math]`

Para OS L1-L6

`4:ti_system`

`4:var=recall_list()`

---

1-6

"1" - "6"

'1' - '6'

---

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

Para a lista personalizada SO "name"

----- Máx. 5 caracteres, 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) exhibe o resultado.

`[1.0, 2.0, 3.0]`

## `var=recall_RegEQ()`

**Módulo:** ti\_system

[2nd] [catalog]

**Sintaxe:** `var=recall_RegEQ()`

[2nd][rci]

**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.

ti\_system  
4:var=recall\_  
REGEQ()

**Exemplo:**

Consulte o programa exemplo: [REGEQ1](#).

[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

[2nd] [list]

**Sintaxe:** `listname.remove(item)`

List  
7:remove(x)

**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] [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:**

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

[Fns...] > Func  
2: return

## .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:** `tiplotlib`[\[2nd\]](#)[\[catalog\]](#)**Sintaxe:** `plt.scatter(xlist,ylist,"mark")`[Fns...]>Modul  
ou [\[math\]](#)  
5:tiplotlib...>  
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.

**Argumento:**

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

- 
- o      [ponto a ponto](#)  
          [\(default\)](#)
  - +      [cruz](#)
  - x      [x](#)
  - .      [píxel](#)
- 

os comandos de importação encontram-se em [\[2nd\]](#)[\[catalog\]](#) ou no menu `tiplotlib Setup`.**Exemplo:**Consulte o programa exemplo: [LINREGR](#).

## seed()

**Módulo:** random

[\[math\]](#) Modul  
2:random...  
Random  
7:seed()

**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
```

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

[\[2nd\]](#) [catalog]

Os resultados variam em função de uma saída aleatória.

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

**[2nd]** [catalog]

**Sintaxe:** plt.show\_plot() display > [clear]

[Fns...]>Modul

ou **[math]**

5:ti\_plotlib...>

Setup

9:show\_plot

**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.

[Fns...]>Modul

ou [math]

5:ti\_plotlib... >

Draw

9:show\_plot()

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).

os comandos de importação encontram-se em **[2nd]** [catalog] ou no menu ti\_plotlib Setup.

**Exemplo:**

Consulte os programas exemplo: [COLORLIN](#) ou [GRAPH](#).

## sin()

**Módulo:** math

[sin](#) 3:sin()

**Sintaxe:** sin()

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

[2nd](#) [\[catalog\]](#)

**Exemplo:**

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

```
[Fns...] > Modul
1:math... > Trig
3:sin()
```

os comandos de importação encontram-se em [2nd](#) [\[catalog\]](#)

## sleep([seconds](#))

**Módulo:** ti\_system; time

[2nd](#) [\[catalog\]](#)

**Sintaxe:** sleep([seconds](#))

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

[2nd](#) [\[rc\]](#)  
ti\_system  
A:sleep()

**Exemplo:**

Programa exemplo:

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

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

Executar o programa TIME  
>>>15.0

```
[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

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

**Sintaxe:** listname.sort()

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

[2nd](#) [\[catalog\]](#)

**Exemplo:**

[Fns...] >  
List  
A:sort()

```
>>>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]
```

## sorted()

**Módulo:** Plano integrado

[2nd](#) [\[list\]](#) (acima  
de [\[stat\]](#)) List  
O:sorted()

**Sintaxe:** sorted(sequence)

**Descrição:** Devolve uma lista ordenada a partir da sequência.

[2nd](#) [\[catalog\]](#)

**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]
```

[Fns...] > List  
O: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

[\[math\]](#)

**Sintaxe:** `sqrt(x)`

Modul

1:math

**Descrição:** Devolve a raiz quadrada de x.

3:sqrt()

**Exemplo:**

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

[\[2nd\]](#)

[\[catalog\]](#)

[Fns...] >

Modul

1:math

3:sqrt()

os comandos de importação encontram-se em

[\[2nd\]](#)

[\[catalog\]](#).

## store\_list("name",var) 1-6

**Módulo:** ti\_system

[2nd] [catalog]

**Sintaxe:** store\_list("name",var) 1-6

[2nd][rcI]

**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.

ti\_system  
3:var=store\_list  
( )

**Argumento:** "name"

[Fns...]>Modul  
ou [math]  
4:ti\_system  
3:var=store\_list  
( )

Para OS L1-L6

---

1-6

"1" - "6"

'1' - '6'

---

Para a lista personalizada SO "name"

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

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

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 Python e prima [2nd][L1] (acima de [ 1 ]) e [enter] no ecrã inicial para ver a lista [L1] como {1 2 3}.

## 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[sin](#) 5:tan()**Sintaxe:** tan(x)**Descrição:** Devolve tangente de x. O argumento está em radianos.[Fns...] > Modul  
1:math... > Trig  
5:tan()**Exemplo:**

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

[2nd](#) [catalog]

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

**text\_at(row,"text","align")****Módulo:** tiplotlib[2nd](#) [catalog]**Sintaxe:** plt.text\_at(row,"text","align")[Fns...]>Modul ou  
[math](#)  
5:tiplotlib...>  
Draw  
0:text\_at()**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

os comandos de  
importação  
encontram-se em  
[2nd](#) [catalog] ou no  
menu  
tiplotlib Setup.

**Exemplo:**

## `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

[2nd](#) [\[test\]](#)  
(acima de [math](#))

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

[2nd](#) [\[catalog\]](#)

### Exemplo:

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

[Fns...] > Ops  
A:True

[a A #]

## trunc()

### Módulo: math

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

### Sintaxe: trunc(x)

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

[2nd](#) [\[catalog\]](#)

### Exemplo:

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

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

os comandos  
de importação  
encontram-se  
em  
[2nd](#) [\[catalog\]](#)

## try:

### Palavra-chave

[2nd](#) [\[catalog\]](#)

**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,

## try:

consulte a documentação Python.

## tuple(sequence)

Módulo: Plano integrado

[2nd](#) [catalog]

Sintaxe: tuple(sequence)

**Descrição:** Converte a sequência num éuplo. 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'>
```

## U

### uniform(min,max)

**Módulo:** random

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

**Sintaxe:** uniform(min,max)

**Descrição:** Devolve um número aleatório x (float) como  $\text{min} \leq x \leq \text{max}$ .

**Exemplo:**

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

[2nd](#) [\[catalog\]](#)

Os resultados variam em função de uma saída aleatória.

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

os comandos  
de  
importação  
encontram-se  
em  
[2nd](#) [\[catalog\]](#)

## W

### `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`.

**Ver:** Mapeamento do teclado para `wait_key()`.

### `while condition:`

**Palavra-chave**

[Fns...] Ctl

**Sintaxe:** `while condition:`

8: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

[\[2nd\]](#) [\[catalog\]](#)

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

[Fns...]>Modul ou

**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).

[math](#)

5:ti\_plotlib...>

Setup

4:window()

Este método deve ser executado antes da execução de qualquer outro comando do módulo ti\_plotlib.

os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#) ou no menu ti\_plotlib Setup.

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](#).

## with

**Palavra-chave**

[\[2nd\]](#) [\[catalog\]](#)

**Descrição:** Para mais informações, consulte a documentação Python.

**xmax default 10.00****Módulo:** ti\_plotlib[\[2nd\]](#) [catalog]**Sintaxe:** plt.xmax **default 10.00**[Fns...]>Modul ou  
[math](#)**Descrição:** Variável especificada para argumentos de janela definidos como plt.xmax.5:ti\_plotlib...>  
Properties  
2:xmax**Valores padrão:**xmin **default -10.00**xmax **default 10.00**ymin **default -6.56**ymax **default 6.56**os comandos de  
importação  
encontram-se em  
[\[2nd\]](#) [catalog] ou no  
menu  
ti\_plotlib Setup.**Exemplo:**Consulte o programa de amostra: [GRAPH](#).

## xmin default -10.00

**Módulo:** ti\_plotlib

**[2nd] [catalog]**

**Sintaxe:** plt.xmin default -10.00

**[Fns...]>Modul ou  
[math]**

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

**5:ti\_plotlib...>  
Properties  
1:xmin**

**Valores padrão:**

---

xmin default -10.00

xmax default 10.00

ymin default -6.56

ymax default 6.56

---

os comandos de importação encontram-se em **[2nd] [catalog]** ou no menu ti\_plotlib Setup.

**Exemplo:**

Consulte o programa de amostra: [GRAPH](#).



**yield****Palavra-chave**[\[2nd\]](#) [\[catalog\]](#)

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

**ymax default 6.56****Módulo:** ti\_plotlib[\[2nd\]](#) [\[catalog\]](#)**Sintaxe:** plt.ymax [default 6.56](#)[Fns...]>Modul ou [math](#)

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

5:ti\_plotlib...>  
Properties  
4:ymax**Valores padrão:**xmin [default -10.00](#)xmax [default 10.00](#)ymin [default -6.56](#)ymax [default 6.56](#)os comandos de importação encontram-se em [\[2nd\]](#) [\[catalog\]](#) ou no menu ti\_plotlib Setup.**Exemplo:**Consulte o programa de amostra: [GRAPH](#).

## ymin default -6.56

**Módulo:** ti\_plotlib

**[2nd]** [catalog]

**Sintaxe:** plt.ymin default -6.56

[Fns...]>Modul ou  
**[math]**

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

5:ti\_plotlib...>  
Properties  
3:ymin

**Valores padrão:**

---

xmin default -10.00

xmax default 10.00

ymin default -6.56

ymax default 6.56

---

os comandos de importação encontram-se em **[2nd]** [catalog] ou no menu ti\_plotlib Setup.

**Exemplo:**

Consulte o programa de amostra: [GRAPH](#).

## 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:**  $\sim x$

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

## $x \leq y$

**Operador**

`math`

**Sintaxe:**  $x \leq y$

1:math > Ops

7:x<=y

**Descrição:** Comparação; x inferior ou igual a y.

**Exemplo:**

`2nd` [catalog]

```
>>>2<=5
```

```
True
```

```
>>>3<=0
```

```
False
```

[Fns...] > Ops

7:x<=y

[a A #]

## $x < y$

**Operador**

`math`

**Sintaxe:**  $x < y$

1:math > Ops

6:x<y

**Descrição:** Comparação; x inferior a y.

**Exemplo:**

`2nd` [catalog]

```
>>>6<10
```

```
True
```

```
>>>12<-15
```

```
False
```

[Fns...] > Ops

6:x<y

[a A #]

## **x>=y**

### **Operador**

**math**

**Sintaxe:** `x>=y`

1:math > Ops  
5:x>=y

**Descrição:** Comparação; x superior ou igual a y.

### **Exemplo:**

**2nd** [catalog]

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

[Fns...] > Ops  
5:x>=y

[a A #]

## **x>y**

### **Operador**

**math**

**Sintaxe:** `x>y`

1:math > Ops  
4:x>y

**Descrição:** Comparação; x superior a y.

### **Exemplo:**

**2nd** [catalog]

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

[Fns...] > Ops  
4:x>y

[a A #]

## x!=y

### Operador

`math`

1:math > Ops

3:x!=y

### Sintaxe: x!=y

**Descrição:** Comparação; x diferente de y.

### Exemplo:

`2nd` [catalog]

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

[Fns...] > Ops

3:x!=y

[a A #]

## x==y

### Operador

`math`

1:math > Ops

2:x==y

### Sintaxe: x==y

**Descrição:** Comparação; x é igual a y.

### Exemplo:

`2nd` [catalog]

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

[Fns...] > Ops

2:x==y

[a A #]

## x=y

**Operador**

sto →

**Sintaxe:** x=y

**Descrição:** y é guardado na variável x

math

1:math > Ops

1:x=y

**Exemplo:**

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

2nd [catalog]

[Fns...] > Ops

1:x=y

[a A #]

## \

**Delimitador**

2nd [catalog]

**Descrição:** Caráter de barra invertida.

[a A #]

## \t

**Delimitador**

2nd [catalog]

**Descrição:** Espaço de tabulação entre as cadeias ou caracteres.

## \n

**Delimitador**

2nd [catalog]

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



' '

**Delimitador**

[2nd](#) [\[mem\]](#)  
(acima de [+](#))

**Descrição:** Duas aspas simples.

**Exemplo:**

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

[2nd](#) [\[catalog\]](#)

[a A #]

" "

**Delimitador**

[alpha](#) [\["\]](#)  
(acima de [+](#))

**Descrição:** Duas aspas duplas.

**Exemplo:**

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

[2nd](#) [\[catalog\]](#)

[a A #]

## Anexo

[Conteúdo do módulo, palavras-chave e incorporado da TI-Python](#)

[Mapeamento do teclado para `wait\_key\(\)`](#)

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

### Built-ins

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

```
PYTHON SHELL
>>> import math
>>> dir(math)
['_name__', 'e', 'pi', 'sqrt',
'pow', 'exp', 'log', 'cos', 'sin',
'tan', 'acos', 'asin', 'atan',
'atan2', 'ceil', 'copysign',
'fabs', 'floor', 'fmod', 'frexp',
'ldexp', 'modf', 'isfinite', 'i
sinf', 'isnan', 'trunc', 'radian
s', 'degrees']
>>> |
```

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

## random

```
PYTHON SHELL
>>> import random
>>> dir(random)
['__name__', 'seed', 'getrandbit
s', 'randrange', 'randint', 'cho
ice', 'random', 'uniform']
>>> |
```

Fns... a A # Tools Editor Files

random	random	random
<code>__name__</code>	<code>randint -- &lt;function&gt;</code>	
<code>seed -- &lt;function&gt;</code>	<code>choice -- &lt;function&gt;</code>	
<code>getrandbits -- &lt;function&gt;</code>	<code>random -- &lt;function&gt;</code>	
<code>randrange -- &lt;function&gt;</code>	<code>uniform -- &lt;function&gt;</code>	



## time

```
PYTHON SHELL
>>> import time
>>> dir(time)
['__name__', 'monotonic', 'sleep',
 'struct_time']
>>> |
```

Fns... a A # Tools Editor Files

time	time	time
__name__		
monotonic		
sleep		
struc_time		

---

## ti\_system

```
PYTHON SHELL
>>> import ti_system
>>> dir(ti_system)
['__name__', 'escape', 'recall_l
ist', 'store_list', 'recall_RegE
Q', 'wait_key', 'sleep', 'wait',
'disp_at', 'disp_clr', 'disp_wa
it', 'disp_cursor']
>>> |
```

Fns... a A # Tools|Editor|Files

ti_system	ti_system	ti_system
<code>__name__</code>	<code>recall_RegEQ</code>	<code>disp_at</code>
<code>escape</code>	<code>wait_key</code>	<code>disp_clr</code>
<code>recall_list</code>	<code>sleep</code>	<code>disp_wait</code>
<code>store_list</code>	<code>wait</code>	<code>disp_cursor</code>

## ti\_plotlib

```
PYTHON SHELL
>>> import ti_plotlib
>>> dir(ti_plotlib)
['lin_reg', '_strtest', 'escape',
 '_except', 'text_at', '_clipseg',
 'show_plot', 'tilocal', 'pen',
 '_sys', 'xmin', 'ymax', 'yscl',
 '_xy', '_rdelta', '_ydelta', 's
catter', 'a', '_pencolor', '_wri
te', 'b', '_xytest', 'window', '_
_mark', 'line', 'monotonic', '_n
umtest', 'ymin', 'tiplotlibExcep
tion', 'labels', 'cls', 'sqrt',
'xscl', 'axes', 'grid', '_sema',
'_pensize', 'plot', 'isnan', 'c
olor', 'title', '_xdelta', '_pen
style', '__name__', 'copysign',
'gr', 'xmax', 'sleep', 'auto_win
dow']
>>> |
Fns... a A # |Outils|Éditer|Script
```

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

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

---

## ti\_hub

```
PYTHON SHELL
>>> import ti_hub
>>> dir(ti_hub)
['__name__', 'connect', 'disconnect', 'set', 'read', 'calibrate', 'range', 'version', 'about', 'isti', 'what', 'who', 'begin', 'wait', 'sleep', 'start', 'last_error', 'get', 'send', 'tihubException']
>>> |
Fns... a A # |Outils|Éditer|Script
```

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

## ti\_rover

```
PYTHON SHELL
>>> import ti_rover
>>> dir(ti_rover)
['motor_right', 'to_angle', 'to_xy', 'red_measurement', 'gray_measurement', 'rvmovement', 'excpt', 'ranger_time', 'waypoint_prev', 'ti_hub', 'pathlist_time', 'to_polar', 'waypoint_eta', 'color_off', 'grid_m_unit', 'path_clear', 'green_measurement', 'waypoint_time', 'motors', 'backward', 'color_blink', 'motor_left', 'waypoint_heading', 'motor', 'gyro_measurement', 'wait_until_done', 'encoders_gyro_measurement', 'pathlist_distance', 'position', 'blue_measurement', 'forward', 'waypoint_distance', 'grid_origin', 'resume', 'path_done', 'disconnect_rv', 'backward_time', 'zero_gyro', 'rv_connected', 'stop', 'rv', 'stay', 'waypoint_xythdrn', 'ranger_measurement', 'left', 'pathlist_cmdnum', 'waypoint_y', 'waypoint_x', 'pathlist_y', 'pathlist_x', '_name_', 'right', 'color_rgb', 'pathlist_revs', 'color_measurement', 'pathlist_heading', 'forward_time', 'waypoint_revs']
>>> |
Fns... | a A # |Outils|Éditer|Script
```

ti_rover	ti_rover	ti_rover
<code>__name__</code>	color_blink	<code>_rv</code>
motor_right	motor_left	stay

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

<b>ti_rover</b>	<b>ti_rover</b>	<b>ti_rover</b>
motors	stop	
backward		

---



## Mapejamento do teclado para wait\_key()

statplot #1 y=	73 48 73	tblset #2 window	72 75 72	format #3 zoom	46 87 46	calc #4 trace	90 59 90	table #5 graph	68 74 68
2nd	----	quit mode	69 64 69	ms del	10 11 10	<  ←	2 14 2	>  →	1 15 1
A-lock alpha	----	link # X,T,θ,n	180 65 180	list stat	49 58 49	☀ ▲	3 ----- 3	☀ ▼	4 ----- 4
test A math	50 51 154	angle B apps	44 57 155	draw C prgm	45 47 156	distr vars	53 56 53	clear	9 9 9
matrix D x <sup>-1</sup>	182 55 157	sin <sup>-1</sup> E sin	183 184 158	cos <sup>-1</sup> F cos	185 186 159	tan <sup>-1</sup> G tan	187 188 160	π H ^	132 181 161
√ I x <sup>2</sup>	189 190 162	EE J ,	139 152 163	( K (	133 236 164	) L )	134 237 165	÷ M ÷	131 239 166
10 <sup>x</sup> N log	193 194 167	u O 7	149 249 168	v P 8	150 250 169	w Q 9	151 251 170	× R ×	130 135 171
e <sup>x</sup> S ln	191 192 172	L4 T 4	146 246 173	L5 U 5	147 247 174	L6 V 6	148 248 175	W -	129 136 176
rcI X sto→	138 12 177	L1 Y 1	143 243 178	L2 Z 2	144 244 179	L3 θ 3	145 245 204	mem # +	128 54 203
off on	break ----	catalog 0	142 62 153	i : .	141 238 198	ans ? (-)	140 197 202	entry solve enter	5 13 5

## Informações gerais

### ***Ajuda online***

[education.ti.com/eguide](http://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](http://education.ti.com/ti-cares)

Selecione o seu país para obter recursos técnicos ou assistência.

### ***Informações da Assistência e Garantia***

[education.ti.com/warranty](http://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.