Phyton TI-Nspire™
Guía de programación

Obtenga más información acerca de la tecnología de TI en la ayuda en línea en education.ti.com/eguide.

Información importante

Excepto por lo que se establezca expresamente en contrario en la Licencia que se incluye con el programa, Texas Instruments no otorga ninguna garantía, ni expresa ni implícita, incluso pero sin limitarse a cualquier garantía implícita de comerciabilidad e idoneidad con un propósito en particular, en relación con cualquier programa o material impreso, y hace dichos materiales disponibles únicamente "tal y como se encuentran". En ningún caso Texas Instruments será responsable en relación con ninguna persona por daños especiales, colaterales, incidentales o consecuenciales en conexión con o que surjan de la compra o el uso de estos materiales, y la responsabilidad única y exclusiva de Texas Instruments, independientemente de la forma de acción, no excederá la cantidad estipulada en la licencia del programa. Asimismo, Texas Instruments no será responsable de ninguna reclamación de ningún tipo en contra del uso de estos materiales por parte de cualquier otra parte.

"Python" y los logotipos de Python son marcas comerciales o marcas comerciales registradas de Python Software Foundation, utilizadas por Texas Instruments Incorporated con permiso de la Fundación.

Los productos reales pueden ser ligeramente distintos de las imágenes proporcionadas.

Índice de contenido

Primeros pasos con la programación de Python

Con el uso de Python con productos TI-Nspire™ puede:

- añadir programas de Python a archivos TNS
- crear programas de Python utilizando plantillas
- interactuar y compartir datos con otras aplicaciones de TI-Nspire™
- interactuar con TI-Innovator™ Hub y TI-Innovator™ Rover

La implementación de TI-Nspire™ Python se basa en MicroPython, que es un pequeño subconjunto de la biblioteca estándar de Python 3 diseñada para ejecutarse en microcontroladores. La implementación original de MicroPython ha sido adaptada para su uso por TI.

Nota: Algunas respuestas numéricas pueden variar de los resultados de la Calculadora debido a diferencias en las implementaciones de matemáticas subyacentes.

Python está disponible en estos productos TI-Nspire™:

Dispositivos portátiles	Software de escritorio
TI-Nspire™ CX II	TI-Nspire™ CX Premium Teacher Software
TI-Nspire™ CX II CAS	TI-Nspire™ CX CAS Premium Teacher Software
TI-Nspire™ CX II-T	TI-Nspire™ CX Student Software
TI-Nspire™ CX II-T CAS	TI-Nspire™ CX CAS Student Software
TI-Nspire™ CX II-C	
TI-Nspire™ CX II-C CAS	

Nota: En la mayoría de los casos, la funcionalidad es idéntica entre la unidad portátil y las vistas del software, pero puede ver algunas diferencias. Esta guía asume que está utilizando el dispositivo portátil o la vista Unidad portátil en el software.

Módulos de Python

TI-Nspire™ Python incluye los siguientes módulos:

Módulos estándar	Módulos de TI
Matemáticas (matemáticas)	TI PlotLib (ti_plotlib)
Aleatorio (aleatorio)	TI Hub (ti_hub)
Matemáticas complejas (cmath)	TI Rover (ti_rover)
Tiempo (tiempo)	TI System (ti_system)
	TI Draw (ti_draw)
	TI Image (ti_image)

Nota: Si tiene programas de Python creados en otros entornos de desarrollo de Python, es posible que necesite editarlos para ejecutarlos en el ambiente TI-Nspire™ Python. Los módulos pueden utilizar diferentes métodos, argumentos y clasificación de

métodos en un programa comparado con los módulos TI. En general, tenga en cuenta la compatibilidad al utilizar cualquier versión de Phyton y de los módulos Python.

Al transferir programas de Python desde una plataforma que no sea TI a una plataforma TI O de un producto TI a otro, recuerde:

- Los programas que utilizan funciones de idioma principal y bibliotecas estándar (matemáticas, aleatorias, etc.) pueden portarse sin cambios.
- Los programas que utilizan bibliotecas específicas de plataforma como matplotlib para módulos PC o TI requerirán modificaciones antes de ejecutarse en una plataforma diferente. Esto puede ser cierto incluso entre plataformas TI.

Al igual que con cualquier versión de Python, deberá incluir las importaciones para utilizar cualquier función, método o constantes contenidas en un módulo determinado. Por ejemplo, para ejecutar la función cos() del módulo matemático, utilice los comandos siguientes:

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

Para ver una lista de menús con sus elementos y descripciones, consulte la sección Mapa de menú.

Instalación de un programa de Python como un módulo

Para guardar su programa de Python como un módulo:

- En el Editor, seleccione **Acciones > Instalar como módulo Python**.
- En Shell, seleccione **Herramientas > Instalar como módulo Python**.

Después de la selección, ocurre lo siguiente:

- Se verifica la sintaxis de Python.
- El archivo se guarda y se mueve a la carpeta PyLib.
- Aparece un cuadro de diálogo que confirma que el archivo se ha instalado como un módulo.
- El archivo está cerrado y el módulo está listo para su uso.
- El nombre del módulo se añadirá al menú **Más módulos** con un elemento del menú **de <module> importación***.

Si piensa compartir este módulo con otros, se recomienda que siga estos lineamientos:

- Almacenar solo un módulo por archivo TNS.
- El nombre del módulo coincide con el nombre del archivo TNS (por ejemplo, el módulo "my_program" está en el archivo "my_program.tns").
- Agregue una página de Notas antes del Editor Python que describa la intención del módulo, la versión y las funciones.
- Utilice la función ver() para mostrar el número de versión del módulo.

- (Opcional) Agregue una función de ayuda para mostrar la lista de métodos en la función.

Espacios de trabajo de Python

Hay dos espacios de trabajo para la programación de Python: Python Editor y Python Shell.

Python Editor	Python Shell
• Cómo crear, editar y guardar programas de Python • Resaltar sintaxis y sangría automática • Indicaciones en línea para guiar con argumentos de función • Sugerencias para mostrar el rango de valores válidos • La tecla var enumera las variables y funciones de usuario globales definidas en el programa actual • Accesos directos del teclado	• Ejecución de programas de Python • Conveniente para probar fragmentos de código pequeños • Interacción con el historial de Shell para seleccionar entradas y salidas anteriores para su reutilización • La tecla var enumera las variables de usuario globales definidas en el último programa ejecutadas en el problema dado

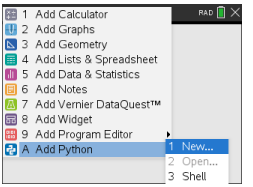
Nota: Se pueden añadir varios programas de Python y Shell a un problema.

Python Editor

En Python Editor puede crear, editar y guardar programas de Python.

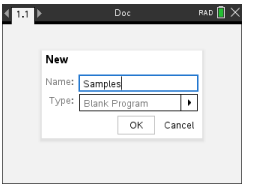
Cómo agregar una página a Python Editor

Para agregar una nueva página a Python Editor en el problema actual, presione menu y seleccione **Agregar a Phython > Nuevo**.

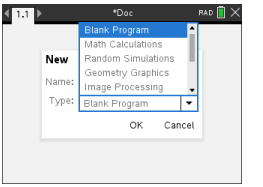


Puede crear un programa en blanco o seleccionar una plantilla.

Programa en blanco



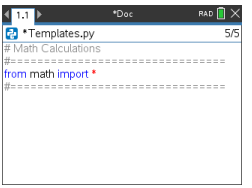
Plantilla



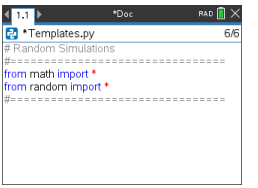
Después de crear el programa, se muestra Python Editor. Si seleccionó una plantilla, se agregarán automáticamente las declaraciones de importación necesarias (consulte a continuación).

Nota: Puede tener varios programas en un solo archivo TNS igual que en otras aplicaciones. Si el programa Python está diseñado para utilizarlo como módulo, el archivo TNS se puede guardar en la carpeta PyLib. Ese módulo se puede utilizar entonces en otros programas y documentos.

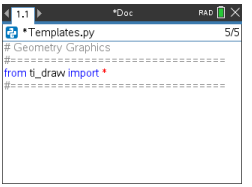
Cálculos matemáticos



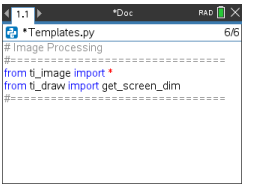
Simulaciones aleatorias



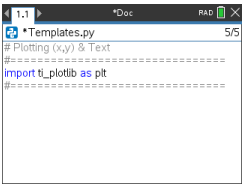
Gráficos de Geometría



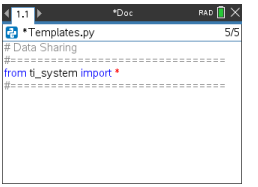
Procesamiento de imágenes



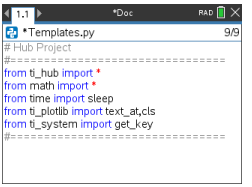
Gráficas (x,y) y texto



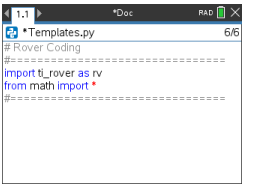
Uso compartido de datos



Proyecto de TI-Innovator Hub



Codificación de TI-Rover



Cómo abrir un programa de Python

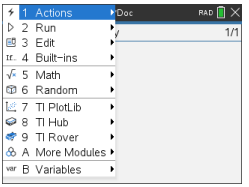
Para abrir un programa de Python existente, presione doc y seleccione **Insertar >Agregar a Phyton > Abrir**. Aparecerá una lista de programas guardados en el archivo TNS.

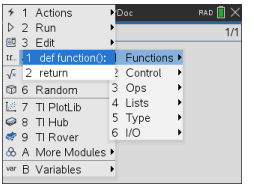
Si se ha eliminado la página Editor que se utilizó para crear el programa, el programa todavía está disponible en el archivo TNS.

Cómo trabajar en Python Editor

Cuando se presiona menu, se muestra el menú Herramientas de documentos. Con estas opciones de menú puede agregar, mover y copiar bloques de código para su programa.

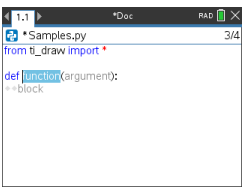
Menú de herramientas de documentos



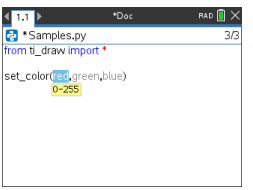


Los elementos seleccionados de los menús del módulo agregarán automáticamente una plantilla de código al Editor con indicaciones en línea para cada parte de la función. Puede desplazarse de un argumento a otro si presiona tab (hacia adelante) o shift+tab (hacia atrás). La información rápida o las listas emergentes aparecerán cuando estén disponibles para ayudarle a seleccionar los valores adecuados.

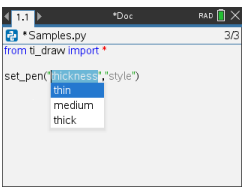
Indicaciones en línea



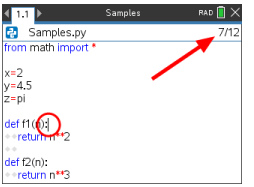
Información rápida



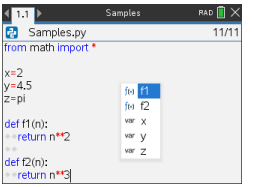
Listas emergentes



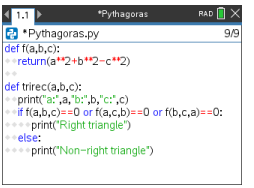
Los números a la derecha del nombre del programa reflejan el número de la línea actual del cursor y el número total de líneas del programa.



Las funciones y variables globales definidas en las líneas antes de la posición actual del cursor pueden insertarse presionando var y seleccionando de la lista.



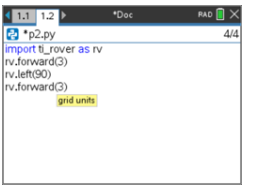
Al agregar código a su programa, el Editor muestra palabras clave, operadores, comentarios, cadenas y sangrías en distintos colores para ayudar a identificar los distintos elementos.



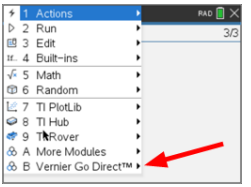
Los elementos de menú pegados se colocan en la línea actual o en la siguiente en función del contexto.

Anterior a v6.2.0

v6.2.0 y posterior

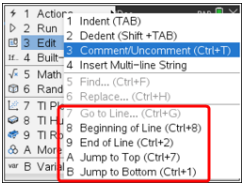


El menú principal muestra el menú del módulo basado en el programa que se está editando, proporcionando un acceso más rápido a esas opciones. Solo se muestra un módulo a la vez.

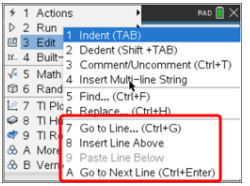


El menú Edtar muestra los elementos del menú y los accesos directos más utilizados.

Anterior a v6.2.0



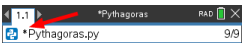
v6.2.0 y posterior



Cómo guardar y ejecutar programas

Cuando haya terminado con su programa, presione menu y seleccione **Ejecutar > Comprobar sintaxis & Guardar**. Esto comprobará la sintaxis del programa Python y lo guardará en el archivo TNS.

Nota: Si tiene cambios sin guardar en su programa, aparecerá un asterisco junto al nombre del programa.



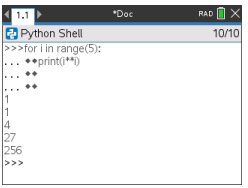
Para ejecutar el programa, presione menu y seleccione **Ejecutar > Ejecutar**. Esto ejecutará el programa actual en la siguiente página de Python Shell o uno nuevo si la página siguiente no es Shell.

Nota: Cuando se ejecuta el programa, se comprueba automáticamente la sintaxis y guarda el programa.

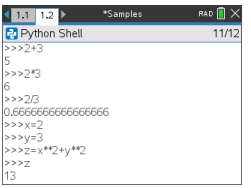
Python Shell

Python Shell es el intérprete que ejecuta sus programas de Python, otras piezas de código Python o comandos sencillos.

Código Python

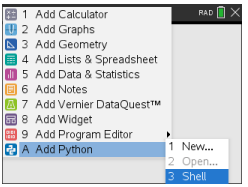


Comando sencillos

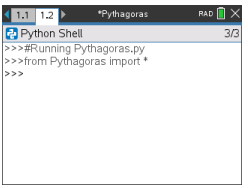


Cómo añadir una página de Python Shell

Para añadir una nueva página de Python Shell en el problema actual, presione menu y seleccione **Agregar Phyton > Shell**.



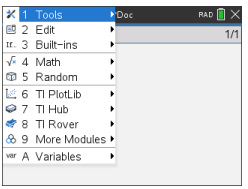
El Python Shell también puede iniciarse desde Python Editor al ejecuta un programa si presiona menu y seleccionar **Ejecutar > Ejecutar**.

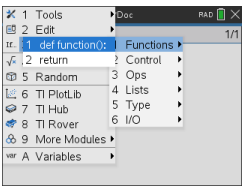


Cómo trabajar en el Phyton Shell

Presionar menu muestra el menú Herramientas de documentos. Con estas opciones de menú puede añadir, mover y copiar bloques de código.

Menú de Herramientas de documentos

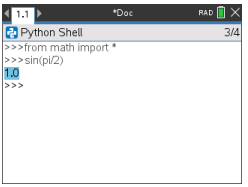




Nota: Si utiliza algún método de uno de los módulos disponibles, asegúrese de ejecutar primero una declaración de módulo de importación como en cualquier entorno de codificación de Python.

La interacción con la salida del Shell es similar a la aplicación Calculadora, donde puede seleccionar y copiar entradas y salidas anteriores para usarlas en cualquier otra parte del Shell, Editor u otras aplicaciones.

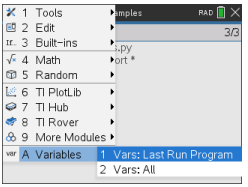
Flecha arriba para seleccionar y luego copiar y pegar en la ubicación deseada



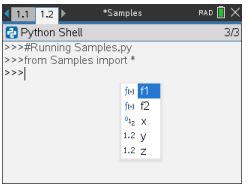
Las funciones y variables globales del último programa en ejecución se pueden insertar presionado var o ctrl+L y seleccione de la lista o presione menu y seleccione **Variables > Vars: Programa de última ejecución**.

Para elegir entre una lista de funciones y variables globales del programa de última ejecución y de cualquier módulo importado, presione menu y seleccione **Variables > Vars: Todos**.

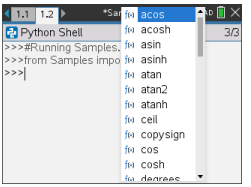
Menú Variables



Variables del programa de última ejecución

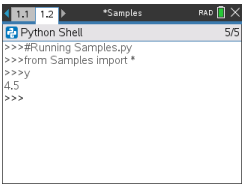


Todas las variables

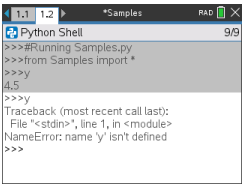


Todas las páginas de Python Shell del mismo problema comparten el mismo estado (definiciones de variables definidas por el usuario e importadas). Cuando guarde o ejecute un programa Python en ese problema, o presione menu y seleccione **Herramientas > Reinicializar Shell**, el historial de Shell tendrá un fondo gris que indica que el estado anterior ya no es válido.

Antes de guardar o reinicializar



Después de guardar o reinicializar



Nota: La opción menu **Herramientas > Borrar historial** borra de la pantalla cualquier actividad anterior en Shell, pero las variables siguen estando disponibles.

Mensajes

Se pueden mostrar errores y otros mensajes informativos mientras esté en una sesión de Python. Si se muestra un error en el Shell cuando se ejecuta un programa, aparecerá un número de línea de programa. Presione ctrl menu y seleccione **Ir a Python Editor**. En el Editor, presione menu y a continuación seleccione **Editar > Ir a línea**. Introduzca el número de línea y presione enter. El cursor se mostrará en el primer carácter de la línea en la que se produjo el error.

Cómo interrumpir un programa o una función que se está ejecutando

Mientras un programa o una función se está ejecutando, se muestra el puntero ocupado ⧖.

▶ Para detener un programa o función,

- Windows®: Presione la tecla **F12**.
- Mac®: Presione la tecla **F5**.
- Dispositivo portátil: Presione la tecla on.

Mapa del menú de Python

En esta sección se enumeran todos los menús y elementos del menú para Python Editor y Shell y una breve descripción para cada uno.

Nota: Para los elementos del menú que tienen accesos directos de teclado, los usuarios de Mac® deben sustituir ⌘ (**cmd**) en cualquier lugar donde se utiliza **Ctrl**. Para obtener una lista completa de los accesos directos de la unidad portátil y el software TI-Nspire™, consulte la guía electrónica de tecnología TI-Nspire™.

Menú Acciones.

Nota: Esto se aplica únicamente al Editor.

Elemento	Descripción
Nuevo	Abre el **Nuevo** cuadro de diálogo en el que se introduce un nombre y selecciona un tipo para su nuevo programa.
Abrir	Abre una lista de programas disponibles en el documento actual.
Crear copia	Abre el cuadro de diálogo **Crear copia** en el que puede guardar el programa actual con otro nombre.

Elemento	Descripción
Cambiar nombre	Abre el cuadro de diálogo **Cambiar nombre** en el que puede cambiar el nombre del programa actual.
Cerrar	Cierra el programa actual.
Configuración	Abre el cuadro de diálogo **Ajustes** en el que puede cambiar el tamaño de fuente tanto para Editor como para Shell.
Instalar como módulo Python	Verifica la sintaxis de Python del archivo TNS actual y lo mueve a la carpeta PyLib.

Menú Ejecutar

Nota: Esto se aplica únicamente al Editor.

Elemento	Accesos directos	Descripción
Ejecutar	Ctrl+R	Comprueba la sintaxis, guarda el programa y se ejecuta en un Shell de Python.
Check Syntax & Save	Ctrl+B	Comprueba la sintaxis y guarda el programa.
Ir a Shell	N/A	Cambia el enfoque al Shell relacionado con el programa actual o abre una nueva página de Shell junto al Editor.

Menú Herramientas

Nota: Esto se aplica únicamente al Shell.

Elemento	Accesos directos	Descripción
Reiniciar último programa	Ctrl+R	Reinicia el último programa relacionado con el Shell actual.
Ir al editor de Python	N/A	Abre la página Editor relacionada con el Shell actual.
Ejecutar	N/A	Abre una lista de programas disponibles en el documento actual. Después de la selección, se ejecuta el programa elegido.
Borrar historial	N/A	Borra el historial en el Shell actual pero no reinicializa el Shell.
Reinicializar Shell	N/A	Restablece el estado de todas las páginas de Shell abiertas en el problema actual.

Elemento	Accesos directos	Descripción
		Todas las variables definidas e importadas ya no están disponibles.
dir()	N/A	Muestra la lista de funciones del módulo especificado cuando se utiliza después de la declaración de importación.
Desde el PROGRAM import *	N/A	Abre una lista de programas disponibles en el documento actual. Después de la selección, la declaración de importación se pega en el Shell.
Instalar como módulo Python	N/A	Habilitado solo para módulos en formato binario. Mueve el archivo TNS actual a la carpeta PyLib.

Menú Editar

Nota: Ctrl+A selecciona todas las líneas de código o salida para cortar o eliminar (solo Editor) o copiar y pegar (Editor y Shell).

Elemento	Acceso directo	Descripción
Sangría	TAB*	Aplica una sangría en el texto de la línea actual o en líneas seleccionadas. * Si hay indicaciones de línea incompletas, TAB pasará a la siguiente indicación.
Quitar sangría	Mayús+TAB**	Quita la sangría del texto en la línea actual o líneas seleccionadas. ** Si hay indicaciones de línea incompletas, Mayús+TAB se desplazará a la indicación anterior.
Comentar/sin comentar	Ctrl+T	Agrega/quita el símbolo de comentario hacia/desde el inicio de la línea actual.
Insertar cadena de varias líneas	N/A	(Solo Editor) Inserta la plantilla de cadena de varias líneas.
Buscar	Ctrl+F	(Solo Editor) Abre el cuadro de diálogo **Buscar** y busca la cadena introducida en el programa actual.
Reemplazar	Ctrl+H	(Solo Editor) Abre el cuadro de diálogo **Reemplazar** y busca la cadena

Elemento	Acceso directo	Descripción
		introducida en el programa actual.
Ir a la línea	Ctrl+G	(Solo Editor) Abre el cuadro de diálogo **Ir a línea** y salta a la línea especificada en el programa actual.
Inicio de la línea	Ctrl+8	Mueve el cursor hasta el inicio de la línea actual.
Fin de línea	Ctrl+2	Mueve el cursor hasta el final de la línea actual.
Saltar al principio	Ctrl+7	Mueve el cursor hasta el inicio de la primera línea del programa.
Saltar al fondo	Ctrl+1	Mueve el cursor hasta el final de la última línea del programa.
Borrar una línea	Ctrl+del	Borra la línea donde se encuentra el cursor.

Menú integrado

Funciones

Elemento	Descripción
def function():	Define una función dependiente de las variables especificadas.
devolver	Define el valor producido por una función.

Control

Elemento	Descripción
if..	Declaración condicional.
if..else	Declaración condicional.
if..elif..else..	Declaración condicional.
para índice en rango(size):	Se repite en un rango.
para índice en rango (start, stop):	Se repite en un rango.
para índice en rango (start, stop, step):	Se repite en un rango.
para índice en la lista:	Itera los elementos de la lista.
while...	Ejecuta las declaraciones en un bloque de código hasta que una condición se evalúa en False.
elif:	Declaración condicional.
else:	Declaración condicional.

Operadores

Elemento	Descripción
x=y	Establece valor variable.
x==y	Pega operador de comparación igual a (==) .
x!=y	Pega no es igual (!=) al operador de comparación.
x>y	Pega operador de comparación mayor que (>).
x>=y	Pega operador de comparación mayor o igual a (>=).
x<y	Pega operador de comparación inferior a (<).

Elemento	Descripción
x<=y	Pega operador de comparación inferior o igual a (<=) .
y	Pega operador lógico y (y).
o	Pega operador lógico o (o).
no	Pega operador lógico no (no).
Verdadero	Pega el valor booleano real.
Falso	Pega el valor booleano falso.

Listas

Elemento	Descripción
[]	Pega los corchetes ([]).
list()	Convierte una secuencia en tipo "list".
len()	Devuelve el número de elementos de la lista.
max()	Devuelve el valor máximo en la lista.
min()	Devuelve el valor mínimo en la lista.
.append()	El método agrega un elemento a una lista.
.remove()	El método quita la primera instancia de un elemento de una lista.
rango (start,stop,step)	Devuelve un conjunto de números.
para índice en rango(start,stop,stop)	Se utiliza para iterar por un rango.
.insert()	El método agrega un elemento en la posición especificada.
.split()	El método devuelve una lista con elementos separados con un delimitador especificado.
sum()	Devuelve la suma de los elementos de una lista.
sorted()	Devuelve una lista ordenada.
.sort()	El método clasifica una lista en su sitio.

Tipo

Elemento	Descripción
int()	Devuelve una parte entera.
float()	Devuelve un valor flotante.
round(x,ndigits)	Devuelve un número de punto flotante redondeado al número de dígitos especificado.
str()	Devuelve una cadena.
complex()	Devuelve un número complejo.
type()	Devuelve el tipo de objeto.

E/S

Elemento	Descripción
print()	Muestra argumento como cadena.
input()	Solicita al usuario que introduzca una entrada.
eval()	Evalúa una expresión representada como una cadena.
.format()	El método formatea la cadena especificada.

Menú Matemáticas

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Cálculos matemáticos**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
desde math import*	Importa todos los métodos (funciones) del módulo matemático.
fabs()	Devuelve el valor absoluto de un número real.
sqrt()	Devuelve la raíz cuadrada de un número real.
exp()	Devuelve e**x.
pow(x,y)	Devuelve x elevado a la potencia y.
log(x,base)	Devuelve el logaritmo$_{base}$(x). log(x) sin base devuelve el logaritmo natural x.
fmod(x,y)	Devuelve el valor del módulo de x e y. Utililizar cuando x e y sean flotantes.

Elemento	Descripción
ceil()	Devuelve el entero mayor o igual a un número real.
floor()	Devuelve el entero menor o igual a un número real.
trunc()	Trunca un número real a un entero.
frexp()	Devuelve un par (y,n) donde x == y * 2**n.

Const.

Elemento	Descripción
e	Returns value for the constant e.
pi	Returns value for the constant pi.

Trigonometría

Elemento	Descripción
radians()	Convierte el ángulo en grados a radianes.
degrees()	Convierte el ángulo en radianes a grados.
sin()	Devuelve el seno del argumento en radianes.
cos()	Devuelve el coseno del argumento en radianes.
tan()	Devuelve la tangente del argumento en radianes.
asin()	Devuelve el arcoseno del argumento en radianes.
acos()	Devuelve el arcocoseno del argumento en radianes.
atan()	Devuelve la tangente de arco del argumento en radianes.
atan2(y,x)	Devuelve la tangente de arco de y/x en radianes.

Menú aleatorio

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Simulaciones aleatorias**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
de random import *	Importa todos los métodos del módulo aleatorio.
random()	Devuelve un número de punto flotante de 0 a 1.0.

Elemento	Descripción
uniform(min,max)	Devuelve un número aleatorio x (flotante) de manera que sea mín. <= x <= máx.
randint(min,max)	Devuelve un entero aleatorio entre mín. y máx.
choice(sequence)	Devuelve un elemento aleatorio de una secuencia no vacía.
randrange(start,stop,step)	Devuelve un número aleatorio de principio a fin paso a paso
seed()	Inicializa el generador de números aleatorios.

TI PlotLib Menu

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Trazado (x,y) y Texto**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
importar ti_plotlib como plt	Importa todos los métodos (funciones) del módulo ti_plotlib en el espacio de nombre “plt”. Como resultado, todos los nombres de funciones pegados de los menús estarán precedidos por "plt.".

Configurar

Elemento	Descripción
cls()	Borra el lienzo de trazado.
grid(x-scale,y-scale,"style")	Muestra una cuadrícula utilizando una escala especificada para los ejes x e y.
window(xmin,xmax,ymin,ymax)	Define la ventana de trazado mediante la asignación del intervalo horizontal especificado (xmin, xmax) y el intervalo vertical (ymin, ymax) al área de trazado asignado (píxeles).
auto_window(list-x,list-y)	Autoescala la ventana de trazado para ajustar los rangos de datos dentro de la lista x y la lista y especificada en el programa antes de la ventana auto_window().
axes("mode")	Muestra ejes en la ventana especificada en el área de trazado.
labels("x-label","y-label",x,y)	Muestra etiquetas "x-label" y "y-label" en los ejes del gráfico en las posiciones de la fila x e y.

Elemento	Descripción
title("título")	Muestra "título" centrado en la línea superior de la ventana.
show_plot()	Muestra la salida de dibujo almacenada en búfer. Las funciones use_buffer() y show_plot() son útiles en casos en los que la visualización de varios objetos en la pantalla podría provocar retrasos (no es necesario en la mayoría de los casos).
use_buffer()	Activa un búfer de fuera de pantalla para acelerar el dibujo.

Dibujar

Elemento	Descripción
color(red,green,blue)	Establece el color para los siguientes gráficos/trazos.
cls()	Borra el lienzo de trazado.
show_plot()	Ejecuta la visualización del gráfico como se establece en el programa.
scatter(x-list,y-list,"mark")	Traza una secuencia del par ordenado de (list-x,list-y) con el estilo de marca especificado.
plot(x-list,y-list,"mark")	Traza una línea utilizando pares clasificados de la lista-x y de la lista-y especificada.
plot(x,y,"mark")	Traza un punto utilizando coordenadas x e y con el estilo de marca especificado.
line(x1,y1,x2,y2,"mode")	Traza un segmento de línea de (x1,y1) a (x2,y2).
lin_reg(x-list,y-list,"display")	Calcula y dibuja el modelo de regresión lineal, ax+b, de la lista-x y de lla lista-y.
pen("size","style")	Establece la apariencia de todas las líneas siguientes hasta que se ejecuta el siguiente pen().
text_at(row,"text","align")	Muestra "text" en el área de trazado en la "align" especificada.

Propiedades

Elemento	Descripción
xmin	Variable especificada para argumentos de ventana definidos como plt.xmin.

Elemento	Descripción
xmax	Variable especificada para argumentos de ventana definidos como plt.xmax.
ymin	Variable especificada para argumentos de ventana definidos como plt.ymin.
ymax	Variable especificada para argumentos de ventana definidos como plt.ymax.
m	Después de que plt.linreg() se ejecuta en un programa, los valores calculados de pendiente, m e intersección, b, se almacenan en plt.m y plt.b.
b	Después de que plt.linreg() se ejecuta en un programa, los valores calculados de pendiente, a e intersección, b, se almacenan en plt.a y plt.b.

Menú de TI Hub

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Proyecto Hub**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
desde ti_hub import *	Importa todos los métodos del módulo ti_hub.

Dispositivos integrados Hub > Salida de color

Elemento	Descripción
rgb (red,gree,blue)	Establece el color del LED RGB.
blink(frequency, time)	Establece la frecuencia y duración de parpadeo del color seleccionado.
off()	Apaga el LED RGB.

Dispositivos integrados Hub > Salida de luz

Elemento	Descripción
on()	Enciende el LED.
off()	Apaga el LED.
blink(frequency, time)	Establece la frecuencia y duración de parpadeo del LED.

Dispositivos integrados Hub > Salida de sonido

Elemento	Descripción
tone(frequency, time)	Reproduce un tono de la frecuencia especificada durante el tiempo especificado.
note("note",time)	Reproduce la nota especificada durante el tiempo especificado. La nota se especifica utilizando el nombre de la nota y un octavo. Por ejemplo: A4, C5. Los nombres de las notas son C, CS, D, DS, E, F, FS, G, GS, A, AS y B. Los números de las octavas van del 1 al 9.
tone(frequency, time, tempo)	Reproduce un tono de la frecuencia especificada durante el tiempo especificado y el tempo. El tempo define el número de pitidos por segundo y va de 0 a 10 (inclusive).
note("nota",time,tempo)	Reproduce la nota especificada durante el tiempo especificado y el tempo. La nota se especifica utilizando el nombre de la nota y un octavo. Por ejemplo: A4, C5. Los nombres de las notas son C, CS, D, DS, E, F, FS, G, GS, A, AS y B. Los números de las octavas van del 1 al 9. Los números de tempo van del 0 al 10 (inclusive).

Dispositivos integrados Hub > Entrada de brillo

Elemento	Descripción
measurement()	Lee el sensor de BRILLO (nivel de luz) integrado y devuelve una lectura. El rango predeterminado es de 0 a 100. Esto se puede cambiar utilizando la función range().
range (min,max.)	Establece el rango para asignar las lecturas desde el sensor de nivel de luz. Si faltan ambos, o se establece en un valor de Ninguno, entonces se establece el rango de brillo predeterminado de 0 a 100.

Agregar dispositivo de entrada

Este menú tiene una lista de los sensores (dispositivos de entrada) compatibles con el módulo ti_hub. Todos los elementos del menú pegarán el nombre del objeto y esperan una variable y un puerto utilizado con el sensor. Cada sensor tiene un método de measurement() que devuelve el valor del sensor.

Elemento	Descripción
DHT (humedad y temperatura digitales)	Devuelve una lista que consiste en la temperatura actual, la humedad, el tipo de sensor y el último estado de lectura en caché.
Medidor de rango	Devuelve la medida de distancia actual desde el dispositivo de rango ultrasónico especificado. • **measurement_time()**: muestra el tiempo que tarda la señal ultrasónica en llegar al objeto (el "tiempo de vuelo").
Nivel de luz	Devuelve el nivel de brillo desde el sensor de nivel de luz externo (brillo).
Temperatura	Devuelve la lectura de la temperatura actual del sensor de temperatura externo. La configuración predeterminada es compatible con el sensor de temperatura Seeed en los puertos IN 1, IN 2 o IN 3. Para utilizar el sensor de temperatura TI LM19 del paquete de placas universal TI-Innovator™Hub, edite el puerto al pin BB en uso y utilice un argumento opcional "TIANALOG". Ejemplo: mylm19=temperature("BB 5","TIANALOG")
Humedad	Devuelve la lectura del sensor de humedad.
Magnético	Detecta la presencia de un campo magnético. El valor de umbral para determinar la presencia del campo se establece a través de la función trigger(). El valor predeterminado del umbral es 150.
Vernier	Lee el valor del sensor analógico Vernier especificado en el comando.

Elemento	Descripción
	El comando admite los siguientes sensores Vernier: • **temperatura** - Sensor de temperatura de acero inoxidable. • **nivel de luz** - Sensor de nivel de luz TI. • **presión** - Sensor de presión de gas original • **presión** - Sensor de presión de gas más reciente. • **pH** - Sensor de pH. • **fuerza10** - Ajuste ±10 N, sensor de fuerza doble. • **fuerza50** - Ajuste ±50 N, sensor de fuerza doble. • **acelerómetro** - Acelerómetro de bajo G. • **genérico** - Permite la configuración de otros sensores no admitidos directamente arriba, y uso de la API calibrate() anterior para establecer coeficientes de ecuación.
Entrada analógica	Es compatible con el uso de dispositivos genéricos de entrada analógica.
Entrada digital	Devuelve el estado actual de la clavija digital conectada al objeto DIGITAL o el estado en caché del último valor de salida digital SET (establecido) al objeto.
Potenciómetro	Es compatible con un sensor de potenciómetro. El rango del sensor puede cambiarse con la función range().
Termistor	Lee sensores de tipo termistor. Los coeficientes predeterminados están diseñados para coincidir con el termistor incluido en el paquete de placas universal de la unidad TI-Innovator™, cuando se utilizan con una resistencia fija de 10 KΩ. Se puede configurar un nuevo conjunto de coeficientes de calibración y resistencia de referencia para el termistor mediante la función calibrate().

Elemento	Descripción
Intensidad de sonido	Admite sensores de sonido.
Entrada de color	Proporciona interconexión a un sensor de entrada de color conectado a I2C. El pin bb_port se utiliza además del puerto I2C para controlar el LED en el sensor de color. • **color_number():** Devuelve un valor de 1 a 9 que representa el color que el sensor detecta. Los números representan los colores según la siguiente asignación: 1: Rojo 2: Verde 3: Azul 4: Turquesa 5: Magenta 6: Amarillo 7: Negro 8: Blanco 9: Gris • **red():** Devuelve un valor de 0 a 255 que representa la intensidad del nivel de color ROJO que se está detectando. • **green():** Devuelve un valor de 0 a 255 que representa la intensidad del nivel de color VERDE que se está detectando. • **blue():** Devuelve un valor de 0 a 255 que representa la intensidad del nivel de color AZUL que se está detectando. • **gray():** Devuelve un valor de 0 a 255 que representa el nivel de gris detectado, donde 0 es negro y 255 es blanco.
Puerto BB	Proporciona soporte para utilizar las 10 clavijas del puerto BB como puerto de entrada/salida digital combinado.

Elemento	Descripción
	Las funciones de inicialización tienen un parámetro de "mask" opcional que permite el uso del subconjunto de 10 clavijas. • **read_port():** Lee los valores actuales en las clavijas de entrada del puerto BB. • **write_port(value):** Establece los valores del pin de salida en el valor especificado, donde el valor es entre 0 y 1023. Tenga en cuenta que el valor también se ajusta con respecto al valor de la máscara en la operación var=bbport(mask), si se proporcionó una máscara.
Tiempo de hub	Proporciona acceso al temporizador de milisegundos interno.
Arreglo TI-RGB	Proporciona funciones para programar la matriz TI-RGB. La función de inicialización acepta un parámetro opcional "LAMP" para habilitar un modo de brillo alto para la matriz TI-RGB que requiere una fuente de alimentación externa. • **set(led_position, r,g,b):** Establece una led_position específica (0-15) en el valor r,g,b especificado, donde r,g,b son valores de 0 a 255. • **set(led_list,red,green,blue):** Establece los LED definidos en la "led_list" en el color especificado por "red", "green", "blue". La "led_list" es una lista de Python que incluye índices de los LED de 0 a 15. Por ejemplo, el set ([0,2,4,6,15], 0, 0, 255) pondrá los LED 0, 2, 4, 6 y 15 en color azul. • **set_all(r,g,b):** Establece todos los LED RGB en la matriz al mismo valor r,g,b. • **all_off():** Apaga todos los RGB de la matriz. • **measurement():** Devuelve el valor aproximado de consumo de corriente que la matriz RGB utiliza de TI-Innovator™ en miliamperios.

Elemento	Descripción
	• **pattern(pattern):** Utilizando el valor del argumento como un valor binario en el rango 0 a 65535 activa los píxeles en los que el valor de la representación sería 1. Los LED se activan como ROJO con un valor de nivel de pwm de 255. • **pattern(value,red,green,blue):** Establece los LED definidos por el "pattern" en el color especificado por "red", "green", "blue".

Agregar dispositivo de salida

Este menú tiene una lista de los dispositivos de salida compatibles con el módulo ti_ hub. Todos los elementos del menú pegan el nombre del objeto y esperan una variable y un puerto utilizado con el dispositivo.

Elemento	Descripción
LED	Funciones para controlar los LED conectados externamente.
RGB	Soporte para controlar LED RGB externos.
Arreglo TI-RGB	Proporciona funciones para programar la matriz TI-RGB.
Bocina	Funciones para apoyar un altavoz externo con TI-Innovator™ Hub. Las funciones son las mismas que las de "sound" anteriores
Potencia	Funciones para controlar la energía externa con el IQ-Innovator™ Hub. • **set(value):** Establece el nivel de potencia al valor especificado, entre 0 y 100. • **on():** Establece el nivel de potencia a 100. • **off():** Establece el nivel de potencia a 0.
Servo continuo	Funciones para el control de servomotores continuos. • **set_cw(speed,time):** El servo girará en el sentido de las agujas del reloj a la velocidad especificada (0-255) y durante la duración específica en segundos. • **set_ccw(speed,time):** El servo girará en sentido contrario a las agujas del reloj a la velocidad especificada (0-255) y durante la duración específica en segundos. • **stop():** Detiene el servo continuo.
Salida analógica	Funciones para el uso de dispositivos genéricos de entrada analógica.

Elemento	Descripción
Motor de vibración	Funciones para controlar los motores de vibración. • **set(val):** Ajusta la intensidad del motor de vibración a "val" (0-255). • **off():** Apaga el motor de vibración. • **on():** Enciende el motor de vibración en el nivel más alto.
Relevador	Controla las interconexiones para controlar los relevadores. • **on():** Establece el relevador en estado ON. • **off():** Establece el relevador en el estado OFF
Servo	Funciones para controlar los servomotores. • **set_position(pos):** Establece la posición del servo de barrido en un rango de -90 a +90. • **zero():** Ajusta el servo de barrido a la posición cero.
Squarewave	Funciones para generar una onda cuadrada. • **set(frequency,duty,time):** Establece la onda cuadrada de salida con un ciclo de trabajo predeterminado del 50% (si no se especifica el ciclo) y una frecuencia de salida especificada por "frequence". La frecuencia puede ser de 1 a 500 Hz. El ciclo de trabajo, si se especifica, puede ser de 0 a 100%. • **off():** Apaga la onda cuadrada.
Salida digital	Interconexiones para controlar una salida digital. • **set(val):** Establece la salida digital al valor especificado por "val" (0 o 1). • **on():** Establece el estado de la salida digital a high (1). • **off():** Establece el estado de la salida digital a low (0).
Puerto BB	Proporciona funciones para programar la matriz TI-RGB. Consulte los detalles anteriores.

Comandos

Elemento	Descripción
sleep(seconds)	Pausa la ejecución de un programa durante una cantidad especificada de segundos. Importado desde el módulo "time".
text_at(row,"text","align")	Muestra el "text" especificado en el área de trazado en la "align" especificada. Parte del módulo ti_plotlib.
cls()	Borra la pantalla de Shell para representación gráfica. Parte del módulo ti_plotlib.

Elemento	Descripción
while get_key() != "esc":	Ejecuta los comandos en el bucle "while" hasta que se presiona la tecla "esc".
get_key()	Devuelve una cadena que representa la tecla presionada. La tecla "1" devuelve "1", "esc" devuelve "esc", etc. Cuando - get_key() - se llama sin parámetros, vuelve inmediatamente. Cuando se llama con un parámetro - get_key(1) - espera hasta que se presione una tecla. Parte del módulo ti_system.

Puertos

Estos son los puertos de entrada y salida disponibles en el TI-Innovator™ Hub.

Elemento
OUT 1
OUT 2
OUT 3
IN 1
IN 2
IN 3
BB 1
BB 2
BB 3
BB 4
BB 5
BB 6
BB 7
BB 8
BB 9
BB 10
Puerto I2C

TI Rover Menu

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Codificar.Rover**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
import ti_rover como rv	Importa todos los métodos (funciones) del módulo ti_rover en el espacio de nombre "rv". Como resultado, todos los nombres de función pegados de los menús estarán precedidos de "rv".

Unidad

Elemento	Descripción
forward(distance)	Mueve a Rover la distancia especificada en las unidades de cuadrícula.
backward(distance)	Mueve a Rover hacia atrás la distancia especificada en unidades de cuadrícula.
left(angle_degrees)	Voltea a Rover a la izquierda en el ángulo especificado en grados.
right(angle_degrees)	Voltea Rover a la derecha en el ángulo especificado en grados.
stop()	Detiene de inmediato cualquier movimiento actual.
stop_clear()	Detiene inmediatamente cualquier movimiento actual y borra todos los comandos pendientes.
resume()	Reanuda el procesamiento de comandos.
stay(time)	El Rover permanece en su lugar por un tiempo en segundos especificado (opcional). Si no se especifica tiempo, el Rover permanecerá en su sitio durante 30 segundos.
to_xy(x,y)	Mueve a El Rover a la posición de coordenadas (x,y) en la cuadrícula virtual.
to_polar(r,theta_degrees)	Mueve el Rover a la posición de coordenadas polares (r, theta) en la cuadrícula virtual. El ángulo se especifica en grados.
to_angle(angle,"unit")	Gira el Rover al ángulo especificado en la cuadrícula virtual. El ángulo es relativo a un ángulo cero que señala hacia abajo al eje x en la cuadrícula virtual.

Unidad > Unidad con opciones

Elemento	Descripción
forward_time(time)	Mueve el Rover hacia delante durante el tiempo especificado.
backward_time(time)	Mueve el Rover hacia atrás durante el tiempo especificado.
forward(distance,"unit")	Mueve el Rover hacia delante a la velocidad predeterminada para la distancia especificada. La distancia puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.
backward(distance,"unit")	Mueve el Rover hacia atrás a la velocidad predeterminada para la distancia especificada. La distancia puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.
left(angle,"unit")	Voltea el Rover en el ángulo especificado. El ángulo puede estar en grados, radianes o gradientes.
right(angle,"unit")	Voltea el Rover a la derecha en el ángulo especificado. El ángulo puede estar en grados, radianes o gradientes.
forward_time(time,speed,"rate")	Mueve el Rover hacia delante durante el tiempo especificado a la velocidad especificada. La velocidad puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.
backward_time(time,speed,"rate")	Mueve el Rover hacia atrás durante el tiempo especificado a la velocidad especificada. La velocidad puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.
forward(distance,"unit",speed,"rate")	Mueve a Rover hacia delante a la distancia especificada a la velocidad especificada. La distancia puede especificarse en

Elemento	Descripción
	unidades de cuadrícula, metros o revoluciones de rueda. La velocidad puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.
backward(distance,"unit",speed,"rate")	Mueve el Rover hacia atrás a la distancia especificada a la velocidad especificada. La distancia puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda. La velocidad puede especificarse en unidades de cuadrícula, metros o revoluciones de rueda.

Entradas

Elemento	Descripción
ranger_measurement()	Lee el sensor de distancia ultrasónico en la parte delantera del Rover, devolviendo la distancia actual en metros.
color_measurement()	Devuelve un valor de 1 a 9, indicando el color predominante "visto" por el sensor de entrada de color de Rover. 1 = rojo 2 = verde 3 = azul 4 = cían 5 = magenta 6 = amarillo 7 = negro 8 = gris 9 = blanco
red_measurement()	Devuelve un valor entre 0 y 255 que indica el nivel de rojo percibido visto por el sensor de entrada de color.
green_measurement()	Devuelve un valor entre 0 y 255 que indica el nivel de verde percibido visto por el sensor de entrada de color.
blue_measurement()	Devuelve un valor entre 0 y 255 que indica el nivel de azul percibido visto por el sensor de entrada de

Elemento	Descripción
	color.
gray_measurement()	Devuelve un valor entre 0 y 255 que indica el nivel de grises percibido visto por el sensor de entrada de color.
encoders_gyro_measurement()	Devuelve una lista de valores que contiene los contadores de la rueda izquierda y derecha, así como el rumbo de giroscopio actual.
gyro_measurement()	Devuelve un valor que representa la lectura actual del giroscopio, incluida la deriva, en grados.
ranger_time()	Muestra el tiempo que tarda la señal ultrasónica del medidor de rango TI- Rover en llegar al objeto (el "tiempo de vuelo").

Salidas

Elemento	Descripción
color_rgb(r,g,b)	Establece el color del LED RGB del Rover en los valores de color rojo, verde y azul específicos.
color_blink(frequency,time)	Establece la frecuencia y duración de parpadeo del color seleccionado.
color_off()	Apaga el LED de RGB del Rover.
motor_left(speed,time)	Establece la potencia del motor izquierdo al valor especificado durante la duración especificada. La velocidad está en el rango -255 a 255 y la parada es 0. Los valores de velocidad positiva están en sentido contrario a las agujas del reloj y los valores de velocidad negativa son en sentido horario. El parámetro de tiempo opcional, si se especifica, tiene un intervalo válido de 0.05 a 655.35 segundos. Si no se epecifica, se usa un valor predeterminado de 5 segundos.
motor_right(speed,time)	Establece la potencia del motor izquierdo al valor especificado durante la duración especificada. La velocidad está en el rango -255 a

Elemento	Descripción
	255 y la parada es 0. Los valores de velocidad positiva están en sentido contrario a las agujas del reloj y los valores de velocidad negativa son en sentido horario. El parámetro de tiempo opcional, si se especifica, tiene un intervalo válido de 0.05 a 655.35 segundos. Si no se epecifica, se usa un valor predeterminado de 5 segundos.
motors("ldir",left_val,"rdir",right_val,time)	Establece la rueda izquierda y derecha a los niveles de velocidad especificados, durante una cantidad de tiempo opcional en segundos. Los valores de velocidad (left_val, right_val) están en el rango 0 a 255 y la parada es 0. Los parámetros ldir y rdir especifican la rotación en el sentido horario o sentido contrario de las ruedas respectivas. El parámetro de tiempo opcional, si se especifica, tiene un intervalo válido de 0.05 a 655.35 segundos. Si no se epecifica, se usa un valor predeterminado de 5 segundos.

Ruta

Elemento	Descripción
waypoint_xythdrn()	Lee las coordenadas x e y, el tiempo, el rumbo, la distancia recorrida, el número de revoluciones de rueda, el número de comandos del punto de paso actual. Muestra una lista con todos estos valores como elementos.
waypoint_prev	Lee las coordenadas x e y, el tiempo, el rumbo, la distancia recorrida, el número de revoluciones de rueda, el número de comandos del punto de paso anterior.
waypoint_eta	Devuelve el tiempo estimado para conducir hasta un punto de referencia.
path_done()	Devuelve un valor de 0 o 1 dependiendo de si el Rover se mueve (0) o se ha terminado con todo el movimiento (1).
pathlist_x()	Muestra una lista de valores X desde el inicio hasta incluir el valor X actual del punto de paso.

Elemento	Descripción
pathlist_y()	Muestra una lista de valores Y desde el inicio hasta incluir el valor Y actual del punto de paso.
pathlist_time()	Muestra una lista de tiempos en segundos desde el inicio hasta incluir el valor del tiempo actual del punto de paso.
pathlist_heading()	Muestra una lista de los rumbos desde el inicio hasta incluir el valor del rumbo actual del punto de paso.
pathlist_distance()	Muestra una lista de distancias recorridas desde el inicio hasta incluir el valor de la distancia actual del punto de paso.
pathlist_revs()	Muestra una lista del número de revoluciones recorridas desde el inicio incluyendo el valor de las revoluciones del punto de paso.
pathlist_cmdnum()	Muestra una lista de números de comandos para la trayectoria
waypoint_x()	Devuelve la coordenada x del punto de paso actual.
waypoint_y()	Devuelve la coordenada y del punto de paso actual.
waypoint_time()	Muestra el tiempo invertido en el recorrido desde el punto de paso anterior hasta el actual
waypoint_heading()	Devuelve el rumbo absoluto del punto de paso actual.
waypoint_distance()	Devuelve la distancia recorrida entre el punto de paso anterior y el actual
waypoint_revs()	Devuelve el número de revoluciones necesarias para realizar el recorrido entre el punto de paso anterior y el actual.

Configuración

Elemento	Descripción
unidades/s	Opción para velocidad en unidades de cuadrícula por segundo.
m/s	Opción para velocidad en metros por segundo.
revs/s	Opción para velocidad en revoluciones de rueda por segundo.
unidades	Opción para distancia en unidades de cuadrícula.
m	Opción para distancia en metros.
revs	Opción para la distancia en revoluciones de ruedas.

Elemento	Descripción
grados	Opción para voltear en grados.
radianes	Opción para voltear en radianes.
gradianes	Opción para voltear en gradientes.
en el sentido de las manecillas del reloj	Opción para especificar la dirección de la rueda.
en sentido contrario a las manecillas del reloj	Opción para especificar la dirección de la rueda.

Comandos

Estos comandos son la recopilación de funciones de otros módulos así como del módulo TI Rover.

Elemento	Descripción
sleep(seconds)	Pausa la ejecución de un programa durante una cantidad especificada de segundos. Importado desde el módulo de tiempo.
text_at(row,"text","align")	Muestra "text" en el área de trazado en la "align" especificada. Importado desde el módulo ti_plotlib.
cls()	Borra la pantalla de Shell para representación gráfica. Importado desde el módulo ti_plotlib.
while get_key() != "esc":	Ejecuta los comandos en el bucle "while" hasta que se presiona la tecla "esc".
wait_until_done()	Detiene el programa hasta que el Rover finaliza el comando actual. Esta es una forma útil de sincronizar comandos que no sean del Rover con el movimiento de Rover.
while not path_done()	Ejecuta los comandos en el bucle "while" hasta que el Rover termine con todo el movimiento. La función path_done() devuelve un valor de 0 o 1 dependiendo de si el Rover se mueve (0) o se ha terminado con todo el movimiento (1).
position(x,y)	Establece la posición de Rover en la cuadrícula virtual a la coordenada x,y especificada.
position(x,y,heading,"unit")	Establece la posición del Rover en la cuadrícula virtual a la coordenada x,y especificada, y el rumbo virtual, con respecto al eje x virtual, se establece si se

Elemento	Descripción
	proporciona un rumbo (en las unidades para ángulos especificados). Se supone que los ángulos positivos de 0 a 360 están en sentido contrario a las manecillas del reloj desde el eje x positivo. Se supone que los ángulos negativos de 0 a -360 son en sentido horario desde el eje x positivo.
grid_origin()	Establece el RV en el punto de origen de la cuadrícula actual de (0,0).
grid_m_unit(scale_value)	Establece el espaciado de la cuadrícula virtual en metros por unidad (m/unidad) al valor especificado. 0.1 es la m/unidad predeterminada y se traduce en 1 unidad = 100 mm o 10 cm o 1 dm o 0.1 m. El rango de scale-value válido es de 0.01 a 10.0.
path_clear()	Borra cualquier trayectoria preexistente o información de punto de paso.
zero_gyro()	Restablece el giroscopio del Rover a un ángulo 0.0 y limpia los recuentos de codificador de la rueda izquierda y derecha.

Menú matemático complejo

Este submenú está situado en **Más módulos**.

Elemento	Descripción
desde cmath import.*	Importa todos los métodos del módulo cmath.
complex (real,imag)	Devuelve un número complejo.
rect(modulus,argument)	Convierte coordenadas polares de un número complejo a la forma rectangular.
.real	Devuelve la parte real del número complejo.
.imag	Devuelve la parte imaginaria de un número complejo.
polar()	Convierte la forma rectangular de un número complejo en coordenadas polares.
phase()	Devuelve la fase de un número complejo.
exp()	Devuelve e**x.
cos()	Devuelve el coseno de un número complejo.
sin()	Devuelve el seno de un número complejo.
log)()	Devuelve logaritmo natural de un número complejo.
log10()	Devuelve el logaritmo base 10 de un número complejo.

Elemento	Descripción
sqrt()	Devuelve la raíz cuadrada de un número complejo.

Menú Tiempo

Este submenú está situado en **Más módulos**.

Elemento	Descripción
desde time import *	Importa todos los métodos desde el módulo de tiempo.
sleep(seconds)	Pausa la ejecución de un programa durante una cantidad especificada de segundos.
clock()	Devuelve el tiempo del procesador actual como un número flotante expresado en segundos.
localtime()	Convierte un tiempo expresado en segundos desde el 1 de enero de 2000 a registro de nueve elementos que contiene: año, mes, día, hora, minutos, segundo, día de semana, día de año y bandera de horario de verano (DST). Si no se proporciona el argumento opcional (seconds), se utiliza el reloj de tiempo real.
ticks_cpu()	Devuelve un contador incremental de milisegundos específico del procesador con un punto de referencia arbitrario. Para medir el tiempo de forma constante en distintos sistemas, utilice ticks_ms().
ticks_diff()	Mide el periodo entre llamadas consecutivas a ticks_cpu() o ticks_ms(). Esta función no debe utilizarse para medir periodos de tiempo arbitrariamente largos.

TI System Menu

Este submenú está situado en **Más módulos**.

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Intercambio de datos**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
desde ti_system import *	Importa todos los métodos (funciones) del módulo ti_system.
recall_value("name")	Recuerda la variable del SO predefinida (valor) denominada "name".
store_value("name",value)	Almacena la variable de Python (value) en una variable del SO denominada "name".

Elemento	Descripción
recall_list("name")	Recuerda una lista del SO predefinida denominada "name".
store_list("name",list)	Almacena la lista de Python (list) en una variable de lista del SO denominada "name".
eval_function("name",value)	Evalúa una función del SO predefinida con el valor especificado.
get_platform()	Devuelve "hh" para unidad portátil y "dt" para escritorio.
get_key()	Devuelve una cadena que representa la tecla presionada. La tecla "1" devuelve "1", "esc" devuelve "esc", etc. Cuando - get_key() - se llama sin parámetros, vuelve inmediatamente. Cuando se llama con un parámetro - get_key(1) - espera hasta que se presione una tecla.
get_mouse()	Devuelve coordenadas del ratón como una tupla de dos elementos, o la posición de píxel en el lienzo o (-1,-1) si es externa al lienzo.
while get_key() != "esc":	Ejecuta los comandos en el bucle "while" hasta que se oprime la tecla "esc".
clear_history()	Borra la historia de Shell.
get_time_ms()	Devuelve tiempo en milisegundos con una precisión de milisegundos. Esta funcionalidad se puede utilizar para calcular una duración en lugar de determinar la hora real del reloj.

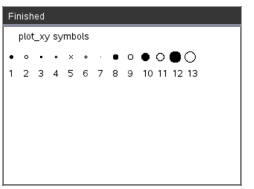
Menú de dibujo de TI

Este submenú está situado en **Más módulos**.

Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Gráficos geométricos**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
de ti_draw import *	Importa todos los métodos del módulo ti_draw.

Forma

Elemento	Descripción
draw_line()	Dibuja una línea que comienza desde la coordenada x1,y1 especificada a x2,y2.
draw_rect()	Dibuja un rectángulo que comienza en la coordenada x,y especificada con el ancho y la altura especificados.
fill_rect()	Dibuja un rectángulo que comienza en la coordenada x,y especificada con el ancho y la altura especificados y lo rellena con el color especificado (usando el color definido o negro si no está definido).
draw_circle()	Dibuja un círculo que comienza en la coordenada de centro x,y especificada con el radio especificado.
fill_circle()	Dibuja un círculo que comienza en la coordenada de centro x, y especificado con el radio especificado y lo rellena con el color especificado (usando el color definido o negro si no está definido).
draw_text()	Dibuja una cadena de texto que comienza en la coordenada x, y especificada.
draw_arc()	Dibuja un arco que comienza en la coordenada x,y especificada con el ancho, altura y ángulos especificados.
fill_arc()	Dibuja un arco que comienza en la coordenada x,y especificada con el ancho, la altura y los ángulos especificados relleno con el color especificado (usando set_color o negro si no se ha definido).
draw_poly()	Dibuja un polígono utilizando los valores de la lista x y los de la lista y especificados.
fill_poli()	Dibuja un polígono utilizando los valores de la lista x y los de la lista y especificados relleno con el color especificado (usando set-color o negro si no se ha definido).
plot_xy()	Dibuja una forma utilizando las coordenadas x, y especificadas y el número especificado del 1 al 13 que representa diferentes formas y símbolos (ver a continuación). 

Control

Elemento	Descripción
clear()	Borra toda la pantalla Se puede utilizar con parámetros de x,y,ancho y altura para borrar un rectángulo existente.
clear_rect()	Borra el rectángulo en la coordenada x,y especificada con el ancho y la altura especificados.
set_color()	Establece el color de las formas que siguen en el programa hasta que se establece otro color.
set_pen()	Establece el grosor y el estilo especificados del borde al dibujar formas (no aplicable al utilizar comandos de relleno).
set_window()	Establece el tamaño de la ventana en la que se dibuja cualquier forma. Esta función es útil para cambiar el tamaño de la ventana para que coincida con los datos o para cambiar el origen (0,0) del lienzo de dibujo.
get_screen_dim()	Devuelve la xmax y ymax de las dimensiones de la pantalla.
use_buffer()	Activa un búfer de fuera de pantalla para acelerar el dibujo.
paint_buffer()	Muestra la salida de dibujo almacenada en búfer. Las funciones use_buffer() y paint_buffer() son útiles en casos en los que la visualización de varios objetos en la pantalla podría provocar retrasos.

Notas

- La configuración predeterminada tiene (0,0) en la esquina superior izquierda de la pantalla. El eje x positivo señala hacia la derecha y el eje y positivo señala la parte inferior. Esto se puede modificar mediante la función set_window().
- Las funciones en el módulo ti_draw solo está disponible en el dispositivo portátil y en la vista de dispositivo portátil en la computadora de escritorio

Menú de TI Image

Este submenú está situado en **Más módulos**.

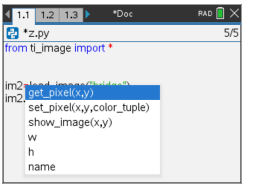
Nota: Al crear un nuevo programa que utilice este módulo, se recomienda utilizar el tipo de programa **Procesamiento de Image**. Esto garantizará que se importen todos los módulos relevantes.

Elemento	Descripción
desde ti_image import *	Importa todos los métodos del módulo ti_image.
new_image(width,height,(r,g,b))	Crea una nueva imagen con el ancho y la altura

Elemento	Descripción
	especificados para su uso en el programa Python. El color de la nueva imagen se define por los valores (r,g,b).
load_image("name")	Carga la imagen especificada por el "name" para su uso en el programa Python. La imagen debe ser parte del documento TNS, ya sea en una aplicación de Notas o de Gráficos. La indicación de "name" mostrará los nombres de imagen (si han sido nombradas anteriormente) o un número que indique su orden de inserción.
copy_image(image)	Crea una copia de la imagen especificada por la variable "image".

Métodos del objeto de imagen

Las funciones adicionales relacionadas con los objetos de imagen están disponibles en Editor y Shell escribiendo el nombre de la variable seguido de un . (punto).



- **get_pixel(x,y):** Obtiene el valor (r,g,b) del píxel en la ubicación definida por el par de coordenadas (x,y).

```
px_val = get_pixel(100,100)
print(px_val)
```

- **set_pixel(x,y,color_tuple):** Establece el píxel en la ubicación (x,y) al color especificado en el color_tuple.

```
set_pixel(100,100,(0,0,255))
```

  Establece el píxel en (100,100) al color (0,0,255).

- **show_image(x,y):** Muestra la imagen con la esquina superior izquierda en la ubicación (x,y).
- **w, h, name:** Obtiene los parámetros de anchura, altura y nombre de la imagen.

Ejemplo

```
from ti_image import *

# An image has been previously inserted into the TNS document in a Notes
application and named "bridge"
im1=load_image("bridge")
px_val = im1.get_pixel(100,100)
print(px_val)

# Set the pixel at 100,100 to blue (0,0,255)
im1.set_pixel(100,100,(0,0,255))
```

```
new_px = im1.get_pixel(100,100)
print(new_px)

# Print the width, height and name of the image
print(im1.w, im1.h, im1.name)
```

Menú Variables

Nota: Estas listas no incluyen variables definidas en otras aplicaciones de TI-Nspire™.

Elemento	Descripción
Vars: Programa actual	(Sólo Editor) Muestra una lista de funciones y variables globales definidas en el programa actual
Vars: Último programa ejecutado	(Solo Shell) Muestra una lista de funciones y variables globales y variables definidas en el último programa ejecutado
Vars: Todos	(Solo Shell) Muestra una lista de funciones y variables globales tanto de la ejecución del último programa como de cualquier módulo importado

Apéndice

Palabras clave de Python

Las siguientes palabras clave están integradas en la implementación de TI-Nspire™ Python.

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

Asignación de teclas de Python

Al introducir código en el Editor o en Shell, el teclado está diseñado para pegar las operaciones de Python apropiadas o abrir los menús para facilitar la introducción de funciones, palabras clave, métodos, operadores, etc.

Tecla	Asignación
var	Abre el menú de variables
sto→	Pega = señal
del ←	Elimina el carácter a la izquierda del cursor
clear	Ninguna acción
=	Pega = señal
|≠≥▸	Pega los símbolos seleccionados: • >= • <=

Tecla	Asignación
	• != • >= • <= • == • y • o • no • | • & • ~
trig	Pega la función seleccionada: • sin • cos • tan • atan2 • asin • acos • atan
?	Muestra sugerencias
:=	Pega :=
^	Pega **
n/d	Ninguna acción
x²	Pega **2
√	Pega raíz cuadrada()
×	Pega el signo de multiplicación (*)
"□"	Pega comillas dobles (")
÷	Pega signo de división (/)
□/□	Ninguna acción
e^x	Pega exp()
ln	Pega log()
10^x	Pega 10**
log	Pega logaritmo(valor, base)
(	Pega (

Tecla	Asignación
)	Pega)
[ ]	Pega [ ]
{ }	Pega { }
(-)	Pega el signo de sustracción (-)
≈	Agrega una línea nueva después de la linea actual
EE	Pega E
?!	Pega los símbolos seleccionados: • ? • ! • $ • ° • ' • % • " • : • ; • _ • \ • #
π	Pega "pi"
⇥	Comportamiento de bandera existente
↵	Agrega una línea nueva después de la linea actual

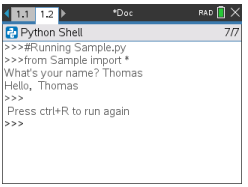
Muestras de programas de Python

Utilizar los siguientes programas de muestra para familiarizarse con los métodos de Python. También están disponibles en el archivo **Getting Started Phyton.tns** localizado en la carpeta **Ejemplos**.

Nota: Si copia y pega cualquier código de muestra que contenga indicadores de tabulador de sangría (••) en el software TI-Nspire™, deberá reemplazar esas instancias con las sangrías de tabulación reales.

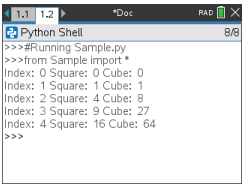
Hola

```
# This program asks for your name and uses
# it in an output message.
# Run the program here by typing "Ctrl R"

name=input("What's your name? ")
print("Hello, ", name)
print("\n Press ctrl+R to run again")
```


1.1 1.2 *Doc RAD
Python Shell 7/7
>>>#Running Sample.py
>>>from Sample import *
What's your name? Thomas
Hello, Thomas
>>>
Press ctrl+R to run again
>>>

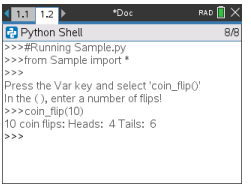
Ejemplo de bucle

```
# This program uses a "for" loop to calculate
# the squares and cubes of the first 5 numbers
# 0,1,2,3,4
# Note: Python starts counting at 0

for index in range(5):
••square = index**2
••cube = index**3
••print("Index: ", index, "Square: ", square,
••••"Cube: ", cube)
```



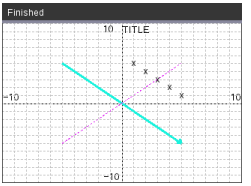
Cara o cruz

```
# Use random numbers to simulate a coin flip
# We will count the number of heads and tails
# Run the program here by typing "Ctrl R"

# Import all the functions of the "random" module
from random import *

# n is the number of times the die is rolled
def coin_flip(n):
    heads = tails = 0
  for i in range(n):
# Generate a random integer - 0 or 1
# "0" means head, "1" means tails
    side=randint(0,1)
    if (side == 0):
      heads = heads + 1
    else:
      tails = tails + 1
# Print the total number of heads and tails
  print(n, "coin flips: Heads: ", heads, "Tails: ", tails)

print("\nPress the Var key and select 'coin_flip()'")
print("In the ( ), enter a number of flips!")
```



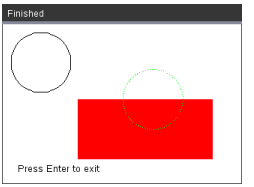
Trazar

```
# Plotting example
import ti_plotlib as plt

# Set up the graph window
plt.window(-10,10,-10,10)
plt.axes("on")
plt.grid(1,1,"dashed")
# Add leading spaces to position the title
plt.title("                TITLE")

# Set the pen style and the graph color
plt.pen("medium","solid")
plt.color(28,242,221)
plt.line(-5,5,5,-5,"arrow")

plt.pen("thin","dashed")
plt.color(224,54,243)
plt.line(-5,-5,5,5,"")

# Scatter plot from 2 lists
plt.color(0,0,0)
xlist=[1,2,3,4,5]
ylist=[5,4,3,2,1]
plt.scatter(xlist,ylist, "x")
```



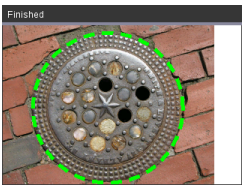
Dibujar

```
from ti_draw import *

# (0,0) is in top left corner of screen
# Let's draw some circles and squares
# Circle with center at (50,50) and radius 40
draw_circle(50,50,40)

# Set color to red (255,0,0) and fill a rectangle of
# of width 180, height 80 with top left corner at
# (100,100)
set_color(255,0,0)
fill_rect(100,100,180,80)

# Set color to green and pen style to "thin"
# and "dotted".
# Then, draw a circle with center at (200,100)
# and radius 40
set_color(0,255,0)
set_pen("thin","dotted")
draw_circle(200,100,40)

set_color(0,0,0)
draw_text(20,200,"Press Enter to exit")
```



Image

```
# Image Processing
#================================
from ti_image import *
from ti_draw import *
#================================

# Load and show the 'manhole_cover' image
# It's in a Notes app
# Draw a circle on top
im1=load_image("manhole_cover")
im1.show_image(0,0)
set_color(0,255,0)
set_pen("thick","dashed")
draw_circle(140,110,100)
```



Hub

Este programa utiliza Python para controlar el TI-Innovator™ Hub, un microcontrolador programable. La ejecució del programa sin conectar un TI-Innovator™ Hub mostrará un mensaje de error.

Para obtener más información sobre TI-Innovator™ Hub, visite education.ti.com.

```
#========== Import Section ==========
from ti_hub import *
from math import *
from random import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#======== End of Import Section =======

print("Connect the TI-Innovator Hub and hit 'enter'")
input()
print("Blinking the RGB LED for 4 seconds")
# Set the RGB LED on the Hub to purple
color.rgb(255,0,255)

# Blink the LED 2 times a second for 4 seconds
color.blink(2,4)

sleep(5)

print("The brightness sensor reading is: ", brightness.measurement())

# Generate 10 random colors for the RGB LED
# Play a tone on the Hub based on the random
# color
print("Generate 10 random colors on the Hub & play a tone")
for i in range(10):
  r=randint(0,255)
  b=randint(0,255)
  g=randint(0,255)
  color.rgb(r,g,b)
  sound.tone((r+g+b)/3,1)
  sleep(1)

color.off()
```

Información general

Ayuda en línea

education.ti.com/eguide

Seleccione su país para obtener más información del producto.

Comuníquese con Asistencia de TI

education.ti.com/ti-cares

Seleccione su país para obtener recursos técnicos y otro tipo de ayuda.

Información sobre el servicio y la garantía

education.ti.com/warranty

Seleccione su país para obtener información acerca de la duración de los términos de la garantía o sobre el servicio para productos.

Garantía limitada. Esta garantía no afecta a sus derechos legales.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243