TI-Nspire™ Python
编程指导手册

欲详细了解 TI 技术，可访问 education.ti.com/eguide 以查看在线帮助。

重要信息

除非在程序附带的《许可证》中明示声明，否则 Texas Instruments 不对任何程序或书面材料做出任何明示或暗示担保，包括但不限于对某个特定用途的适销性和适用性的暗示担保，并且这些材料均以“原样”提供。任何情况下，Texas Instruments 对因购买或使用这些材料而蒙受特殊、附带、偶然或连带损失的任何人都不承担任何责任。无论采用何种赔偿方式，Texas Instruments 的唯一且排他性义务不得超出本程序许可证规定的数额。此外，对于任何其他方因使用这些材料而提起的任何类型的索赔，Texas Instruments 概不负责。

“Python”和 Python 徽标是 Python 软件基金会的商标或注册商标，在获得基金会许可的情况下，由 Texas Instruments Incorporated 使用。

实际产品可能与提供的图像有所差异。

表的内容

Python 编程快速入门

在 TI-Nspire™ 产品中使用 Python 时，您可以:

- 将 Python 程序添加到 TNS 文件
- 使用模板创建 Python 程序
- 与其他 TI-Nspire™ 应用程序交互和共享数据
- 与 TI-Innovator™ Hub 和 TI-Innovator™ Rover 进行交互

TI-Nspire™ Python 实现基于 MicroPython，它是专为在微控制器上运行而设计的 Python 3 标准库的小子集。原始 MicroPython 实现已改编以供 TI 使用。

注: 由于底层数学实现的差异，某些数字答案可能与计算器结果有所不同。

Python 可在这些 TI-Nspire™ 产品上使用:

手持设备	台式设备软件
TI-Nspire™ CX II	TI-Nspire™ CX 高级教师版软件
TI-Nspire™ CX II CAS	TI-Nspire™ CX CAS 高级教师版软件
TI-Nspire™ CX II-T	TI-Nspire™ CX 学生软件
TI-Nspire™ CX II-T CAS	TI-Nspire™ CX CAS 学生软件
TI-Nspire™ CX II-C	
TI-Nspire™ CX II-C CAS	

注: 大多数情况下，手持设备和软件视图的功能相同，但也可能会发现一些差异。本指南假定您正在使用软件中的手持设备或手持设备视图。

Python 模块

TI-Nspire™ Python 包括以下模块:

标准模块	TI 模块
Math (math)	TI PlotLib (ti_plotlib)
Random (random)	TI Hub (ti_hub)
Complex Math (cmath)	TI Rover (ti_rover)
Time (time)	TI System (ti_system)
	TI Draw (ti_draw)
	TI Image (ti_image)

注: 如果已在其他 Python 开发环境中创建了现有 Python 程序，则可能需要对其进行编辑以在 TI-Nspire™ Python 解决方案上运行。与 TI 模块相比，模块可以在程序中使用不同的方法、参数和方法排序。一般来讲，使用任何版本的 Python 和 Python 模块时都要注意兼容性。

将 Python 程序从非 TI 平台传送到 TI 平台或从一个 TI 产品传送到另一个产品时，请记住:

- 可以移植使用核心语言功能和标准 libs(数学、随机等) 的程序，而无需进行任何更改。
- 使用特定于平台的库(例如用于 PC 或 TI 模块的 matplotlib) 的程序将需要进行编辑，才能在其他平台上运行。即使在 TI 平台之间也是如此。

与任何版本的 Python 一样，需要包含导入才能使用给定模块中包含的任何函数、方法或常数。例如，要从数学模块执行 cos() 函数，使用以下命令:

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

有关其项目和描述的菜单列表，请参阅菜单层次图部分。

作为模块安装 Python 程序

将 Python 程序另存为模块:

- 在 Editor 中，选择 **Actions > Install as Python Module(操作 > 作为 Python 模块安装)** 。
- 在 Shell 中，选择 **Tools > Install as Python Module(工具 > 作为 Python 模块安装)** 。

选择后，会出现以下情况:

- 将检查 Python 语法。
- 文件将被保存并移动到 PyLib 文件夹。
- 将出现一个对话框，确认文件已作为模块安装。
- 文件已关闭，模块可供使用。
- 模块名称将被添加到 **More Modules(更多模块)** 菜单中，并包含 **from <module> import *(导入)** 菜单项。

如果您打算与他人分享本模块，建议您遵循以下指南:

- 每个 TNS 文件仅存储一个模块。
- 模块名称与 TNS 文件的名称匹配(例如，“my_program”模块位于“my_program.tns”文件中) 。
- 在 Python Editor 之前添加一个记事本页面，描述模块的目的、版本和功能。
- 使用 ver() 函数以显示模块的版本编号。
- (可选) 添加帮助函数以在函数中显示方法列表。

Python 工作区

Python 编程有两个工作区:Python 编辑器和 Python Shell。

Python 编辑器	Python Shell
• 创建、编辑和保存 Python 程序 • 句法高亮显示和自动缩进 • 使用函数参数指导的内联提示 • 显示有效值范围的工具提示 • var 键列出当前程序中定义的全局用户变量和函数 • 键盘快捷键	• 运行 Python 程序 • 便于测试小代码片段 • 与 Shell 历史记录进行交互，以选择之前的输入和输出以供重新使用 • var 键列出在指定问题中上一个程序中定义的全局用户变量

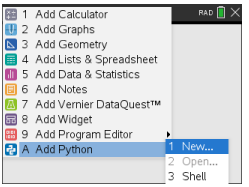
注: 可以将多个 Python 程序和 Shell 添加到问题中。

Python 编辑器

您可以在 Python 编辑器中创建、编辑和保存 Python 程序。

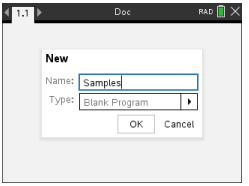
添加 Python 编辑器页面

要在当前问题中添加新的 Python 编辑器页面，按 menu 键并选择**添加 Python > 新**。

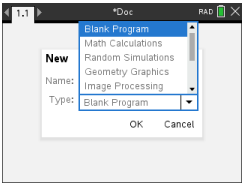


您可以创建一个空白程序，也可以选择一个模板。

空白程序



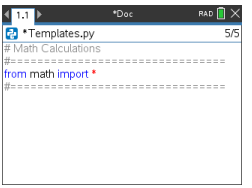
模板



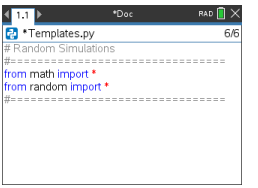
创建程序后，将显示 Python 编辑器。如果选择了模板，则会自动添加必要的 import 语句(参见下文) 。

注：就像其他应用程序一样，您可以在单个 TNS 文件中拥有多个程序。如果打算将 Python 程序用作模块，则可以将 TNS 文件保存在 PyLib 文件夹中。然后，该模块便可用于其他程序和文档。

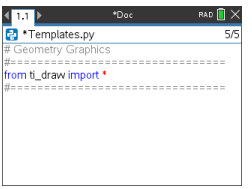
数学计算



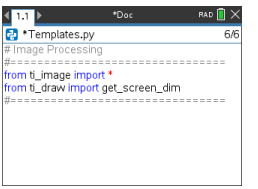
随机模拟



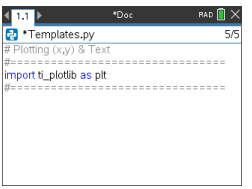
几何图形



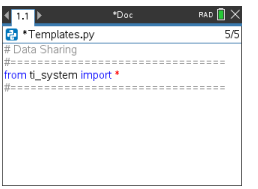
图片处理



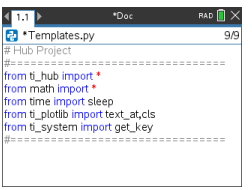
绘图 (x,y) & 文本



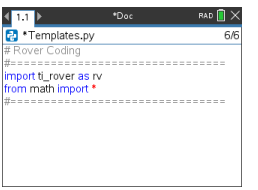
数据共享



TI-Innovator Hub 项目



TI-Rover 编码



打开 Python 程序

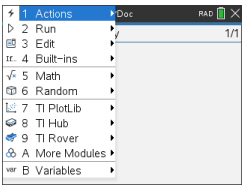
要打开现有 Python 程序，按 doc 键并选择**插入 > 添加 Python > 打开**。这将显示已保存在 TNS 文件中的程序列表。

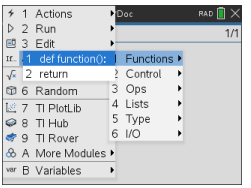
如果用于创建程序的编辑器页面已被删除，则该程序在 TNS 文件中仍然可用。

在 Python 编辑器中工作

按 menu 键显示“文档工具”菜单。通过这些菜单选项，您可以添加、移动和复制程序的代码块。

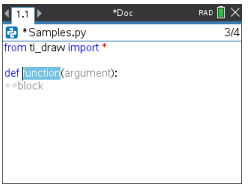
文档工具菜单



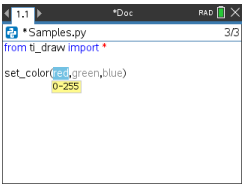


从模块菜单中所选的项目将自动将代码模板添加到编辑器中，并为函数的每个部分提供内联提示。按 tab 键(向前)或 shift+tab 键(向后)可以从一个参数导航到下一个参数。在可用的情况下，将显示工具提示或弹出列表，以帮助您选择适当的值。

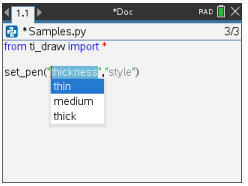
内联提示



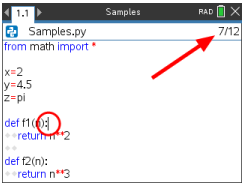
工具提示



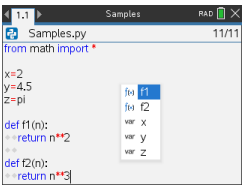
弹出列表



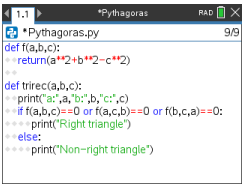
程序名右侧的数字反映光标的当前行编号和程序中的行总数。



可通过按 var 键并从列表中选择，来插入当前光标位置上方各行中定义的全局函数和变量。



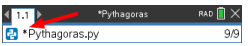
在向程序中添加代码时，编辑器会以不同的颜色显示关键字、运算符、注释、字符串和缩进，以帮助识别不同的元素。



保存和运行程序

完成程序后，按 menu 键并选择**运行 > 检查句法 & 保存**。这将检查 Python 程序的句法并将其保存到 TNS 文件中。

注: 如果程序中有未保存的更改，则程序名称旁边将显示星号。



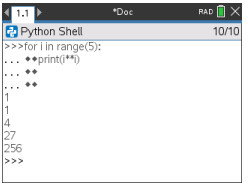
要运行程序，按 menu 键并选择**运行 > 运行**。这将在下一个 Python Shell 页面或新页面(如果下一页不是 Shell) 中运行当前程序。

注: 运行程序会自动检查句法并保存程序。

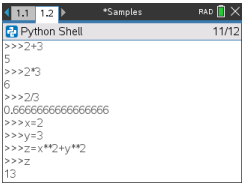
Python Shell

Python Shell 是执行 Python 程序、其他 Python 代码或简单命令的解释器。

Python 代码

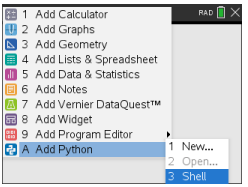


简单命令

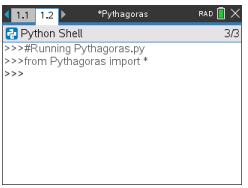


添加 Python Shell 页面

要在当前问题中添加新的 Python Shell 页面，按 menu 键并选择 **添加 Python > Shell**。



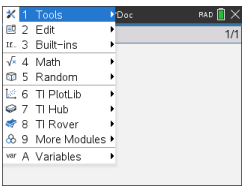
还可以通过按 menu 键并选择**运行 > 运行**执行程序，来从 Python 编辑器中启动 Python Shell。

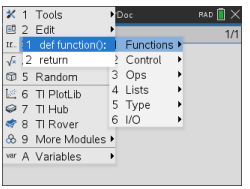


在 Python Shell 中工作

按 menu 键显示“文档工具”菜单。通过这些菜单选项，您可以添加、移动和复制代码块。

文档工具菜单

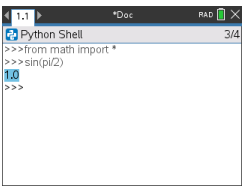




注: 如果使用某个可用模块中的任何方法，请确保像在任何 Python 编码环境中一样，首先执行导入模块语句。

与 Shell 输出的交互类似于计算器应用程序，您可以在其中选择和复制之前的输入和输出，以便在 Shell、编辑器或其他应用程序中的其他位置使用。

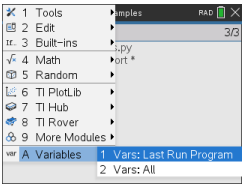
使用向上箭头进行选择，然后复制并粘贴到所需位置



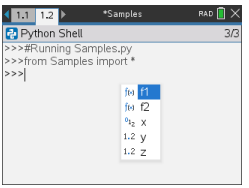
要插入上次运行程序的全局函数和变量，请按 var 或 ctrl+L 并从列表中选择或按 menu 并选择 **变量 > Vars:上次运行程序**。

要从上次运行程序和任何导入模块的全局函数和变量列表中进行选择，请按 menu 键并选择 **变量 > Vars:全部**。

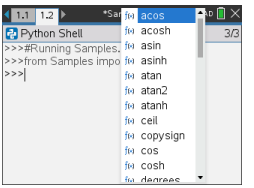
变量菜单



上次运行程序变量

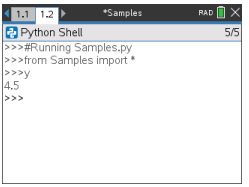


所有变量

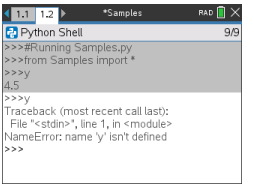


同一问题中的所有 Python Shell 页面共享相同的状态(用户定义的和导入的变量定义)。当您在该问题中保存或运行 Python 程序，或按 menu 并选择**工具 > 重新初始化 Shell** 时，Shell 历史记录将显示灰色背景，表示之前的状态不再有效。

保存或重新初始化之前



保存或重新初始化之后



注: menu **工具 > 清除历史记录**选项清除 Shell 中之前任何活动的屏幕，但变量仍然可用。

消息

在 Python 会话中时，可能会显示错误和其他信息性消息。如果程序执行时在 Shell 中出现错误，则会显示程序行编号。按 ctrl menu 并选择**转至 Python 编辑器**。在编辑器中，按 menu 然后选择**编辑 > 转至行**。输入行编号并按 enter。光标将显示在出错行的第一个字符上。

中断正在运行的程序

当程序或函数正在运行时，会显示忙指针 ⏲。

▶ 要停止程序或函数，

- Windows®:按 **F12** 键。
- Mac®:按 **F5** 键。
- 手持设备:按 on 键。

Python 菜单层次图

本节列出了 Python 编辑器和 Shell 的所有菜单和菜单项以及对其每一个的简短说明。

注：对于带有键盘快捷键的菜单项，Mac® 用户应在任何使用 **Ctrl** 的位置替换为 ⌘ (**Cmd**)。有关 TI-Nspire™ 手持设备和软件快捷键的完整列表，请参阅 TI-Nspire™ 技术电子指南。

"操作"菜单

注: 仅适用于编辑器。

项目	说明
新建	打开**新建**对话框，在此输入名称并为新程序选择一个类型。
打开	打开当前文档中可用程序的列表。
创建副本	打开**创建副本**对话框，可以在此使用其他名称保存当前程序。
重命名	打开**重命名**对话框，可以在此重命名当前程序。
关闭	关闭当前程序。
设置	打开**设置**对话框，可以在此更改编辑器和 Shell 的字体大小。
作为 Python 模块安装	检查当前 TNS 文件的 Python 语法，并将其移动到 PyLib 文件夹。

运行菜单

注: 仅适用于编辑器。

项目	快捷键	说明
运行	Ctrl+R	在 Python Shell 中检查句法、保存和执行程序。
检查句法并保存	Ctrl+B	检查句法并保存程序。
转至 Shell	无	将焦点转移至与当前程序相关的 Shell，或在编辑器旁边打开一个新的 Shell 页面。

工具菜单

注: 仅适用于 Shell。

项目	快捷键	说明
重新运行上次运行的程序	Ctrl+R	重新运行与当前 Shell 相关的最后一个程序。
转到 Python 编辑器	无	打开与当前 Shell 相关的编辑器页面。
运行	无	打开当前文档中可用程序的列表。 选择后，将运行所选程序。
清空历史记录	无	清除当前 Shell 中的历史记录，但不会重新初始化 Shell。
重新初始化 Shell	无	重置当前问题中所有打开的 Shell 页面的状态。 所有定义的变量和导入的函数都不再可用。
dir()	无	显示在 import 语句之后使用的指定模块中的函数列表。
From PROGRAM import *	无	打开当前文档中可用程序的列表。 选择后，import 语句被粘贴在 Shell 中。
作为 Python 模块安装	无	仅对二进制格式的模块启用。将当前 TNS 文件移动到 PyLib 文件夹。

编辑菜单

注: Ctrl+A 选择要剪切或删除(仅限编辑器) 、或复制和粘贴(编辑器和 Shell) 的所有代码行或输出。

项目	快捷键	说明
缩进	TAB*	缩进当前行或所选行上的文本。 * 如果内联提示不完整，TAB 将导航至下一个提示。
取消缩进	Shift+TAB**	取消缩进当前行或所选行上的文本。 ** 如果内联提示不完整，Shift+TAB 将导航至上一个提示。
注释/取消注释	Ctrl+T	在/从当前行的开头添加/移除注释符号。
插入多行字符串	无	(仅限编辑器) 插入多行字符串模板。
查找	Ctrl+F	(仅限编辑器) 打开**查找**对话框，在当前程序中搜索输入的字符串。
替换	Ctrl+H	(仅限编辑器) 打开**替换**对话框，在当前程序中搜索输入的字符串。
转至行	Ctrl+G	(仅限编辑器) 打开**转至行**对话框，跳转到当前程序中的指定行。
行首	Ctrl+8	将光标移至当前行的开头。
行尾	Ctrl+2	将光标移至当前行的末尾。
跳至顶部	Ctrl+7	将光标移至程序中第一行的开头。
跳至底部	Ctrl+1	将光标移至程序中最后一行的末尾。

内置项菜单

函数

项目	说明
def function():	定义依赖于指定变量的函数。
return	定义函数产生的值。

控件

项目	说明
if..	条件语句。
if..else..	条件语句。
if..elif..else..	条件语句。
for index in range(size):	迭代一个范围。
for index in range(start,stop):	迭代一个范围。
for index in range(start,stop,step):	迭代一个范围。
for index in list:	迭代列表元素。
while..	执行代码块中的语句，直到条件的计算结果为 False。
elif:	条件语句。
else:	条件语句。

Ops

项目	说明
x=y	设置变量值。
x==y	粘贴等于 (==) 比较运算符。
x!=y	粘贴不等于 (!=) 比较运算符。
x>y	粘贴大于 (>) 比较运算符。
x>=y	粘贴大于或等于 (>=) 比较运算符。
x<y	粘贴小于 (<) 比较运算符。
x<=y	粘贴小于或等于 (<=) 比较运算符。

项目	说明
和	粘贴 and (and) 逻辑运算符。
或	粘贴 or (or) 逻辑运算符。
not	粘贴 not (not) 逻辑运算符。
真	粘贴 True 布尔值。
假	粘贴 False 布尔值。

列表

项目	说明
[]	粘贴括号 ([])。
list()	将序列转换为“列表”类型。
len()	返回列表的元素数。
max()	返回列表中的最大值。
min()	返回列表中的最小值。
.append()	该方法将元素附加到列表中。
.remove()	该方法从列表中移除元素的第一个实例。
range(start,stop,step)	返回一组数字。
for index in range(start,stop,step)	用于迭代范围。
.insert()	该方法在指定位置添加元素。
.split()	该方法将返回列表，其中的元素由指定分隔符分隔。
sum()	返回列表元素的和。
sorted()	返回已排序的列表。
.sort()	该方法对列表进行适当排序。

类型

项目	说明
int()	返回整数部分。
float()	返回浮点值。
round(x,ndigits)	返回四舍五入到指定位数的浮点数。

项目	说明
str()	返回一个字符串。
complex()	返回一个复数。
type()	返回对象的类型。

I/O

项目	说明
print()	将参数显示为字符串。
input()	提示用户输入。
eval()	计算表示为字符串的表达式。
.format()	该方法格式化指定的字符串。

数学菜单

注: 创建使用此模块的新程序时，建议使用**数学计算**程序类型。这将确保导入所有相关模块。

项目	说明
from math import *	从 math 模块导入所有方法(函数)。
fabs()	返回实数的绝对值。
sqrt()	返回实数的平方根。
exp()	返回 e**x。
pow(x,y)	返回 x 的 y 次幂。
log(x,base)	返回 $\log_{底数}(x)$。 无底数的 log(x) 返回自然对数 x。
fmod(x,y)	返回 x 和 y 的模块值。当 x 和 y 是浮点数时使用。
ceil()	返回大于或等于实数的最小整数。
floor()	返回小于或等于实数的最大整数。
trunc()	将实数截断为整数。
frexp()	返回一对 (y,n)，其中 x == y * 2**n。

Const

项目	说明
e	Returns value for the constant e.
pi	Returns value for the constant pi.

Trig

项目	说明
radians()	将以度为单位的角度转换为弧度。
degrees()	将以弧度为单位的角度转换为度。
sin()	返回参数的正弦值(以弧度为单位)。
cos()	返回参数的余弦值(以弧度为单位)。
tan()	返回参数的正切值(以弧度为单位)。

项目	说明
asin()	返回参数的反正弦值(以弧度为单位)。
acos()	返回参数的反余弦值(以弧度为单位)。
atan()	返回参数的反正切值(以弧度为单位)。
atan2(y,x)	返回 y/x 的反正切值(以弧度为单位)。

随机菜单

注: 创建使用此模块的新程序时，建议使用**随机模拟**程序类型。这将确保导入所有相关模块。

项目	说明
from random import *	从随机模块导入所有方法。
random()	返回从 0 到 1.0 的浮点数。
uniform(min,max)	返回随机数字 x(浮点)，使得最小值 <= x <= 最大值。
randint(min,max)	返回介于最小值和最大值之间的随机整数。
choice(sequence)	从非空序列返回随机元素。
randrange(start,stop,step)	从开始到结束逐步返回随机数字。
seed()	初始化随机数字生成器。

TI PlotLib 菜单

注: 创建使用此模块的新程序时，建议使用**Plotting (x,y) &文本**程序类型。这将确保导入所有相关模块。

项目	说明
将 ti_plotlib 导入为 plt	从“plt”命名空间中的 ti_plotlib 模块导入所有方法(函数)。因此，从菜单中粘贴的所有函数名称都将以“plt”开头。

设置

项目	说明
cls()	清除绘图画布。
grid(x-scale,y-scale,"style")	使用 x 和 y 轴的指定比例显示网格。
window(xmin,xmax,ymin,ymax)	通过将指定的水平间隔 (xmin, xmax) 和垂直间隔 (ymin, ymax) 映射到分配的绘图区域(像素)来定义绘图窗口。
auto_window(x-list,y-list)	自动缩放绘图窗口以适合在 auto_window() 之前的程序中指定的 x-list 和 y-list 内的数据范围。
axes("mode")	显示绘图区域指定窗口上的坐标轴。
labels("x-label","y-label",x,y)	在行位置 x 和 y 处的绘图轴上显示“x-label”和“y-label”的标签。
title("title")	在窗口顶行居中显示“标题”。
show_plot()	显示缓冲绘图输出。 在屏幕上显示多个对象可能导致延迟的情况下，use_buffer() 和 show_plot() 函数非常有用(多数情况下不需要)。
use_buffer()	启用离屏缓冲区，以加速绘图。

绘制

项目	说明
color(red,green,blue)	设置以下所有图形/绘图的颜色。
cls()	清除绘图画布。

项目	说明
show_plot()	执行程序中设置的绘图显示。
scatter(x-list,y-list,"mark")	使用指定的标记样式绘制 (x-list,y-list) 中的一系列有序数对。
plot(x-list,y-list,"mark")	使用指定的 x-list 和 y-list 中的有序数对绘制一条线。
plot(x,y,"mark")	使用带有指定标记样式的坐标 x 和 y 绘制一个点。
line(x1,y1,x2,y2,"mode")	绘制一条从 (x1,y1) 到 (x2,y2) 的线段。
lin_reg(x-list,y-list,"display")	计算并绘制 x-list、y-list 的线性回归模型 ax+b。
pen("size","style")	设置以下所有行的外观，直到执行下一个 pen() 为止。
text_at(row,"text","align")	在绘图区以指定的“对齐”显示“文本”。

属性

项目	说明
xmin	定义为 plt.xmin 的窗口参数的指定变量。
xmax	定义为 plt.xmax 的窗口参数的指定变量。
ymin	定义为 plt.ymin 的窗口参数的指定变量。
ymax	定义为 plt.ymax 的窗口参数的指定变量。
m	在程序中执行 plt.linreg() 之后，斜率 m 和截距 b 的计算值存储在 plt.m 和 plt.b 中。
b	在程序中执行 plt.linreg() 之后，斜率 a 和截距 b 的计算值存储在 plt.a 和 plt.b 中。

TI Hub 菜单

注: 创建使用此模块的新程序时，建议使用 **Hub 项目**程序类型。这将确保导入所有相关模块。

项目	说明
from ti_hub import *	从 ti_hub 模块导入所有方法。

Hub 内置设备 > 颜色输出

项目	说明
rgb(red,green,blue)	设置 RGB LED 的颜色。
blink(frequency,time)	设置所选颜色的闪烁频率和持续时间。
off()	关闭 RGB LED。

Hub 内置设备 > 亮度输出

项目	说明
on()	打开 LED。
off()	关闭 LED。
blink(frequency,time)	设置 LED 的闪烁频率和持续时间。

Hub 内置设备 > 声音输出

项目	说明
tone(frequency,time)	在指定的时间内播放指定频率的声音。
note("note",time)	在指定的时间内播放指定的音符。 使用音名和一个八度指定音符。例如:A4、C5。 音名为 C、CS、D、DS、E、F、FS、G、GS、A、AS 和 B。 八度编号范围为 1 到 9(含 1 和 9) 。
tone(frequency,time,tempo)	在指定的时间和节拍内播放指定频率的声音。 节拍定义了 0 到 10(含 0 和 10) 之间的每秒蜂鸣音次数。
note("note",time,tempo)	在指定的时间和节拍内播放指定的音符。

项目	说明
	使用音名和一个八度指定音符。例如:A4、C5。 音名为 C、CS、D、DS、E、F、FS、G、GS、A、AS 和 B。 八度编号范围为 1 到 9(含 1 和 9)。 节拍编号范围为 0 到 10(含 0 和 10)。

Hub 内置设备 > 亮度输入

项目	说明
measurement()	读取内置 BRIGHTNESS(光级)传感器并返回读数。 默认范围为 0 到 100。可以使用 range() 函数对其进行更改。
range(min,max)	设置从光级传感器映射读数的范围。 如果两者都缺失,或设置为“无”值,则设置 0 到 100 的默认亮度范围。

添加输入设备

此菜单包含 ti_hub 模块支持的传感器(输入设备)列表。所有菜单项都会粘贴对象的名称,并预计有变量和与传感器一起使用的端口。每个传感器都有返回传感器值的 measurement() 方法。

项目	说明
DHT (Digital Humidity & Temp)	返回包含当前温度、湿度、传感器类型和最后缓存读取状态的列表。
测距仪	返回指定超声波测距仪的当前距离测量结果。 • **measurement_time()**——返回超声波信号到达对象所需的时间(“飞行时间”)。
亮度级	返回外部光级(亮度)传感器的亮度级别。
温度	返回外部温度传感器的温度读数。 默认配置是在 IN 1、IN 2 或 IN 3 端口中支持 Seeed 温度传感器。 要使用 TI-Innovator™ Hub 试验板包的 TI LM19 温度传感器,请编辑使用中的 BB 插针端口,并使用可选参数“TIANALOG”。 示例:mylm19=temperature("BB 5","TIANALOG")
湿度	返回湿度传感器读数。
磁性	检测磁场的存在。

项目	说明
	通过 trigger() 函数设置确定磁场存在的阈值。 阈值的默认值为 150。
Vernier	读取命令中指定的 Vernier 模拟传感器的值。 该命令支持以下 Vernier 传感器: • **温度** - 不锈钢温度传感器。 • **lightlevel** - TI 光级传感器。 • **压力** - 原装气压传感器 • **压力** - 新型气压传感器。 • **pH** - pH 传感器。 • **force10** - 设置为 ±10 N、双力传感器。 • **force50** - 设置为 ±50 N、双力传感器。 • **加速计** - 低重力加速计。 • **通用** - 允许设置上述不直接支持的其他传感器，并使用上述 calibrate() API 设置方程系数。
Analog In	支持使用模拟输入通用设备。
Digital In	返回连接到 DIGITAL 对象的数字插针的当前状态，或最近一次为对象设置的数字输出值的缓存状态。
电位计	支持电位计传感器。 传感器的范围可通过 range() 函数更改。
热敏电阻器	读取热敏电阻器传感器。 当与 10KΩ 固定电阻器配合使用时，默认系数专为与 TI-Innovator™ Hub 试验板包中包含的热敏电阻器匹配。 可使用 calibrate() 函数为热敏电阻器配置一组新的校准系数和参考电阻。
响度	支持声音响度传感器。
颜色输入	提供与 I2C 连接的颜色输入传感器的接口。 除了 I2C 端口外，还使用 bb_port 插针来控制颜色传感器上的 LED。 • **color_number():** 返回介于 1 到 9 之间的值，该值代表传感器正在检测的颜色。 数字代表以下映射对应的颜色: 1: 红色 2: 绿色 3: 蓝色 4: 青色

项目	说明
	5: 品红色 6: 黄色 7: 黑色 8: 白色 9: 灰色 • **red():** 返回介于 0 到 255 之间的值，该值表示检测到的红色等级强度。 • **green():** 返回介于 0 到 255 之间的值，该值表示检测到的绿色等级强度。 • **blue():** 返回介于 0 到 255 之间的值，该值表示检测到的蓝色等级强度。 • **gray():** 返回介于 0 到 255 之间的值，表示检测到的灰色等级，其中 0 为黑色，255 为白色。
BB 端口	支持将 10 个 BB 端口插针全部用作组合数字输入/输出端口。 初始化函数包含可选的“掩码”参数，允许使用 10 个插针的子集。 • **read_port():** 读取 BB 端口输入插针上的当前值。 • **write_port(value):** 将输出插针值设置为指定值，其中值介于 0 到 1023 之间。请注意，如果提供了掩码，则该值也会根据 var=bbport (mask) 运算中的掩码值进行调整。
Hub 时间	提供对内部毫秒计时器的访问。
TI-RGB Array	提供用于对 TI-RGB 阵列进行编程的函数。 初始化函数接受可选的“LAMP”参数，为需要外部电源的 TI-RGB 阵列启用高亮度模式。 • **set(led_position, r,g,b):** 将特定的 led_position (0-15) 设置为指定的 r、g、b 值，其中 r、g、b 为 0 到 255 之间的值。 • **set(led_list,red,green,blue):** 将“led_list”中定义的 LED 设置为“red”、“green”、“blue”中指定的颜色。“led_list”是一个 Python 列表，包括 0 到 15 之间的 LED 索引。例如，set ([0,2,4,6,15], 0, 0, 255) 会将 LED 0, 2, 4, 6 和 15 设置为蓝色。 • **set_all(r,g,b):** 将阵列中的所有 RGB LED 设置为相同的 r、g、b 值。 • **all_off():** 关闭阵列中的所有 RGB。

项目	说明
	• **measurement():** 以毫安为单位从 TI-Innovator™ 返回 RGB 阵列正在使用的近似电流消耗。 • **pattern(pattern):** 使用参数值作为 0 到 65535 范围内的二进制值，打开表示中值为 1 的像素。LED 呈红色亮起，pwm 电平值为 255。 • **pattern(value,red,green,blue):** 将“pattern”定义的 LED 设置为“red”、“green”、“blue”指定的颜色。

添加输出设备

此菜单有 ti_hub 模块支持的输出设备列表。所有菜单项都会粘贴对象的名称，并预计有变量和与设备一起使用的端口。

项目	说明
LED	控制外部连接的 LED 的函数。
RGB	支持控制外部 RGB LED。
TI-RGB Array	提供用于对 TI-RGB 阵列进行编程的函数。
扬声器	通过 TI-Innovator™ Hub 支持外部扬声器的函数。 函数与上述的“声音”的函数相同。
功率	通过 TI-Innovator™ Hub 控制外部电源的函数。 • **set(value):** 将功率等级设置为介于 0 到 100 之间的指定值。 • **on():** 将功率等级设置为 100。 • **off():** 将功率等级设置为 0。
连续伺服	控制连续伺服电机的函数。 • **set_cw(speed,time):** 伺服电机将以指定的速度 (0-255) 在特定持续时间 (以秒为单位) 内顺时针旋转。 • **set_ccw(speed,time):** 伺服电机将以指定的速度 (0-255) 在特定持续时间 (以秒为单位) 内逆时针旋转。 • **stop():** 停止连续伺服。
模拟输出	使用模拟输入通用设备的函数。
振动电机	控制振动电机的函数。 • **set(val):** 将振动电机强度设置为 "val" (0-255)。 • **off():** 关闭振动电机。 • **on():** 以最高等级打开振动电机。
继电器	控制用于控制继电器的接口。

项目	说明
	• **on():** 将继电器设置为 ON 状态。 • **off():** 将继电器设置为 OFF 状态。
伺服	控制伺服电机的函数。 • **set_position(pos):** 将扫描伺服位置设置在 -90 到 +90 范围内。 • **zero():** 将扫描伺服设置为零位置。
方波	生成方波的函数。 • **set(frequency,duty,time):** 设置输出方波，默认占空比为 50% (如果未指定占空比)，且输出频率由“frequence”指定。频率可以是 1 到 500 Hz。如有指定，则占空比可为 0 到 100%。 • **off():** 关闭方波。
数字输出	控制数字输出的接口。 • **set(val):** 将数字输出设置为“val”指定的值 (0 或 1)。 • **on():** 将数字输出的状态设置为高 (1)。 • **off():** 将数字输出的状态设置为低 (0)。
BB 端口	提供用于对 TI-RGB 阵列进行编程的函数。 请参阅上述详情。

命令

项目	说明
sleep(seconds)	将程序暂停指定的秒数。 从“时间”模块导入。
text_at(row,"text","align")	在绘图区以指定的“对齐”显示指定的“文本”。 ti_plotlib 模块的一部分。
cls()	清除用于绘图的 Shell 屏幕。 ti_plotlib 模块的一部分。
while get_key() != "esc":	在“while”循环中运行命令，直到按下“esc”键。
get_key()	返回表示按下的按键的字符串。 “1”键返回“1”，“esc”返回“esc”，依此类推。 在没有任何参数的情况下调用 get_key()，它会立即返回。 在使用参数调用 get_key(1) 时，它会等待直到按下一个键。 ti_system 模块的一部分。

Ports

这些是 TI-Innovator™ Hub 上可用的输入和输出端口。

项目
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
I2C

TI Rover 菜单

注: 创建使用此模块的新程序时，建议使用 **Rover 编码**程序类型。这将确保导入所有相关模块。

项目	说明
将 ti_rover 导入为 rv	从“rv”命名空间中的 ti_rover 模块导入所有方法(函数)。因此，从菜单中粘贴的所有函数名称都将以“rv.”开头。

驱动器

项目	说明
forward(distance)	将 Rover 向前移动指定的距离(以网格单元为单位)。
backward(distance)	将 Rover 向后移动指定的距离(以网格单位为单位)。
left(angle_degrees)	将 Rover 向左转指定的角度(以度为单位)。
right(angle_degrees)	将 Rover 向右转指定的角度(以度为单位)。
stop()	立即停止当前的任何运动。
stop_clear()	立即停止当前的任何运动并清除所有等待中的命令。
resume()	恢复处理命令。
stay(time)	Rover 在原地停留指定的时间(以秒为单位，可选)。 如果未指定时间，则 Rover 将停留 30 秒。
to_xy(x,y)	将 Rover 移动到虚拟网格上的坐标位置 (x,y)。
to_polar(r,theta_degrees)	将 Rover 移动到虚拟网格上的极坐标位置 (r, theta)。 角度以度为单位指定。
to_angle(angle,"unit")	将 Rover 旋转到虚拟网格中指定的角度。 该角度相对于虚拟网格中指向 x 轴下方的零角度。

驱动器 > 带选项的驱动器

项目	说明
forward_time(time)	将 Rover 向前移动指定的时间。
backward_time(time)	将 Rover 向后移动指定的时间。
forward(distance,"unit")	将 Rover 以默认速度向前移动指定距离。 可以以网格单元、米或车轮转数为单位来指定距离。
backward(distance,"unit")	将 Rover 以默认速度向后移动指定距离。 可以以网格单元、米或车轮转数为单位来指定距离。
left(angle,"unit")	将 Rover 向左转指定的角度。 角度可以以度、弧度或百分度为单位。
right(angle,"unit")	将 Rover 向右转指定的角度。 角度可以以度、弧度或百分度为单位。
forward_time(time,speed,"rate")	将 Rover 以指定速度向前移动指定时间。 可以用网格单元/秒、米/秒或车轮转数/秒来指定速度。
backward_time(time,speed,"rate")	将 Rover 以指定速度向后移动指定时间。 可以用网格单元/秒、米/秒或车轮转数/秒来指定速度。
forward(distance,"unit",speed,"rate")	将 Rover 以指定速度向前移动指定距离。 可以以网格单元、米或车轮转数为单位来指定距离。 可以用网格单元/秒、米/秒或车轮转数/秒来指定速度。
backward(distance,"unit",speed,"rate")	将 Rover 以指定速度向后移动指定距离。 可以以网格单元、米或车轮转数为单位来指定距离。 可以用网格单元/秒、米/秒或车轮转数/秒来指定速度。

输入

项目	说明
ranger_measurement()	读取 Rover 正面的超声波测距传感器，返回当前距离(以米为单位)。
color_measurement()	返回介于 1 到 9 之间的值，表示 Rover 颜色输入传感器“看到”的主要颜色。 1 = 红色 2 = 绿色 3 = 蓝色 4 = 青色 5 = 品红色 6 = 黄色 7 = 黑色 8 = 灰色 9 = 白色
red_measurement()	返回介于 0 到 255 之间的值，该值表示颜色输入传感器看到的感知红色等级。
green_measurement()	返回介于 0 到 255 之间的值，该值表示颜色输入传感器看到的感知绿色等级。
blue_measurement()	返回介于 0 到 255 之间的值，该值表示颜色输入传感器看到的感知蓝色等级。
gray_measurement()	返回介于 0 到 255 之间的值，该值表示颜色输入传感器看到的感知灰色等级。
encoders_gyro_measurement()	返回值列表，其中包含左、右轮编码器计数以及当前陀螺仪航向。
gyro_measurement()	返回表示当前陀螺仪读数(包括漂移)的值(以度为单位)。
ranger_time()	返回 TI-Rover 测距仪发出的超声波信号到达对象所需的时间(“飞行时间”)。

输出

项目	说明
color_rgb(r,g,b)	将 Rover RGB LED 的颜色设置为特定的红色、绿色、蓝色值。
color_blink(frequency,time)	设置所选颜色的闪烁频率和持续时间。

项目	说明
color_off()	关闭 Rover RGB LED。
motor_left(speed,time)	将指定持续时间内的左电机功率设置为指定值。 速度在 -255 到 255 范围内，0 为停止。正速度值为逆时针旋转，负速度值为顺时针旋转。 可选时间参数(如有指定)的有效范围为 0.05 到 655.35 秒。如未指定，则使用默认值 5 秒。
motor_right(speed,time)	将指定持续时间内的左电机功率设置为指定值。 速度在 -255 到 255 范围内，0 为停止。正速度值为逆时针旋转，负速度值为顺时针旋转。 可选时间参数(如有指定)的有效范围为 0.05 到 655.35 秒。如未指定，则使用默认值 5 秒。
motors("ldir",left_val,"rdir",right_val,time)	将左、右轮设置为可选时间长度(以秒为单位)内指定的速度等级。 速度 (left_val, right_val) 值在 0 到 255 范围内，0 为停止。ldir 和 rdir 参数指定相应车轮的 CW 或 CCW 旋转。 可选时间参数(如有指定)的有效范围为 0.05 到 655.35 秒。如未指定，则使用默认值 5 秒。

路径

项目	说明
waypoint_xythdrn()	读取当前路径点的 x-coord、y-coord、时间、航向、运动的距离、车轮转数、命令编号。返回所有这些值都作为元素的列表。
waypoint_prev	读取上一个路径点的 x-coord、y-coord、时间、航向、运动的距离、车轮转数、命令编号。
waypoint_eta	返回驱动器到路径点的估算时间。
path_done()	返回值是 0 还是 1，具体取决于 Rover 是正在移动 (0) 还是完成所有移动 (1)。
pathlist_x()	返回从起点到当前路径点 X 值(含)的 X 值列表。
pathlist_y()	返回从起点到当前路径点 Y 值(含)的 Y 值列表。

项目	说明
pathlist_time()	返回从起点到当前路径点时间值(含)的时间(以秒为单位)列表。
pathlist_heading()	返回从起点到当前路径点航向值(含)的航向列表。
pathlist_distance()	返回从起点到当前路径点距离值(含)的运动距离列表。
pathlist_revs()	返回从起点到当前路径点转数值(含)的运动转数列表。
pathlist_cmdnum()	返回路径的命令编号列表。
waypoint_x()	返回当前路径点的 x 坐标。
waypoint_y()	返回当前路径点的 y 坐标。
waypoint_time()	返回从上一个路径点运动到当前路径点花费的时间。
waypoint_heading()	返回当前路径点的绝对航向。
waypoint_distance()	返回前一个和当前路径点之间的运动距离。
waypoint_revs()	返回在上一个和当前路径点之间运动所需的转数。

设置

项目	说明
单元/秒	以网格单元/秒为单位的速度选项。
m/s	以米/秒为单位的速度选项。
revs/s	以车轮转数/秒为单位的速度选项。
单位	以网格单元为单位的距离选项。
m	以米为单位的距离选项。
revs	以车轮转数为单位的距离选项。
度数	以度数为单位的旋转选项。
弧度	以弧度为单位的旋转选项。
百分度	以百分度为单位的旋转选项。
顺时针	指定轮方向的选项。
逆时针	指定轮方向的选项。

命令

这些命令是其他模块以及 TI Rover 模块的函数集合。

项目	说明
sleep(seconds)	将程序暂停指定的秒数。 从时间模块导入。
text_at(row,"text","align")	在绘图区以指定的“对齐”显示“文本”。 从 ti_plotlib 模块导入。
cls()	清除用于绘图的 Shell 屏幕。 从 ti_plotlib 模块导入。
while get_key() != "esc":	在“while”循环中运行命令，直到按下“esc”键。
wait_until_done()	暂停程序，直到 Rover 完成当前命令。 这是一种将非 Rover 命令与 Rover 运动同步的有用方法。
while not path_done()	运行“while”循环中的命令，直到 Rover 完成所有移动。 path_done() 函数返回的值是 0 还是 1，具体取决于 Rover 是正在移动 (0) 还是完成所有移动 (1)。
position(x,y)	将 Rover 在虚拟网格上的位置设置为指定的 x,y 坐标。
position(x,y,heading,"unit")	将 Rover 在虚拟网格上的位置设置为指定的 x,y 坐标，并且如果已提供航向，则设置相对于虚拟 x 轴的虚拟航向 (以指定角度为单位)。 假设从 0 到 360 的正角相对于正 x 轴为逆时针方向。假设从 0 到 -360 的负角相对于正 x 轴为顺时针方向。
grid_origin()	将 RV 设置为位于当前网格原点 (0,0)。
grid_m_unit(scale_value)	将以米/单元 (m/unit) 为单位的虚拟网格间距设置为指定值。0.1 米/单元为默认值，且转换格式为 1 个单元 = 100 毫米或 10 厘米或 1 分米或 0.1 米。 有效 scale_value 的范围为 0.01 至 10.0。
path_clear()	清除任何先已存在的路径或路径点信息。
zero_gyro()	重置 Rover 陀螺仪至 0.0 角度，并清除左、右轮编码器计数。

复数运算菜单

此子菜单位于**更多模块**项下方。

项目	说明
from cmath import *	从 cmath 模块导入所有方法。
complex(real,imag)	返回一个复数。
rect(modulus,argument)	将极坐标转换为复数的矩形形式。
.real	返回复数的实部。
.imag	返回复数的虚部。
polar()	将矩形形式转换为复数的极坐标。
phase()	返回复数的相位。
exp()	返回 e**x。
cos()	返回复数的余弦值。
sin()	返回复数的正弦值。
log()	返回复数的自然对数。
log10()	返回以 10 为底的复数对数。
sqrt()	返回复数的平方根。

时间菜单

此子菜单位于**更多模块**项下方。

项目	说明
from time import *	从时间模块导入所有方法。
sleep(seconds)	将程序暂停指定的秒数。
clock()	以浮点数形式返回当前处理器时间(以秒为单位)。
localtime()	将自 2000 年 1 月 1 日起以秒表示的时间转换为包含年、月、月-日、小时、分钟、秒、工作日、年-日和夏令时(DST)标志的九元组。 如果未提供可选(秒)参数，则使用实时时钟。
ticks_cpu()	返回带有任意参考点的处理器特定的递增毫秒计数器。 要在不同系统之间一致地度量时间，请使用 ticks_ms()。
ticks_diff()	测量连续调用 ticks_cpu() 或 ticks_ms() 之间的时间间隔。 此函数不应用于测量任意长的时间段。

TI 系统菜单

此子菜单位于**更多模块**项下方。

注: 创建使用此模块的新程序时，建议使用**数据共享**程序类型。这将确保导入所有相关模块。

项目	说明
from ti_system import *	从 ti_system 模块导入所有方法(函数)。
recall_value("name")	调用名为“name”的预定义操作系统变量(值)。
store_value("name",value)	将 Python 变量(值)存储到名为“name”的操作系统变量中。
recall_list("name"))	调用名为“name”的预定义操作系统列表。
store_list("name",list)	将 Python list (list) 存储到名为“name”的操作系统列表变量中。
eval_function("name",value)	以指定值计算预定义的操作系统函数。
get_platform()	返回手持设备的“hh”和台式设备的“dt”。
get_key()	返回表示按下的按键的字符串。 “1”键返回“1”，“esc”返回“esc”，依此类推。 在没有任何参数的情况下调用 get_key()，它会立即返回。 在使用参数调用 get_key(1) 时，它会等待直到按下一个键。
get_mouse()	返回鼠标坐标为两个元素的元组、 画布像素位置或 (-1,-1)(如果在画布之外)。
while get_key() != "esc":	在“while”循环中运行命令，直到按下“esc”键。
clear_history()	清除 Shell 历史记录。
get_time_ms()	以毫秒精度返回时间(以毫秒为单位)。 此函数可用于计算持续时间，而不是确定实际时钟时间。

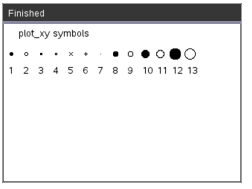
TI 绘图菜单

此子菜单位于**更多模块**项下方。

注: 创建使用此模块的新程序时，建议使用**几何图形**程序类型。这将确保导入所有相关模块。

项目	说明
from ti_draw import *	从 ti_draw 模块导入所有方法。

形状

项目	说明
draw_line()	从指定的 x1、y1 坐标到 x2、y2 坐标绘制一条线。
draw_rect()	从指定的 x、y 坐标开始，以指定的宽度和高度绘制矩形。
fill_rect()	从指定的 x、y 坐标开始，以指定的宽度和高度绘制矩形，并用指定的颜色进行填充(如果未定义，则使用 set_color 或黑色)。
draw_circle()	从指定的 x、y 中心坐标开始，以指定的半径绘制圆。
fill_circle()	从指定的 x、y 中心坐标开始，以指定的半径绘制圆，并用指定的颜色进行填充(如果未定义，则使用 set_color 或黑色)。
draw_text()	从指定的 x、y 坐标开始绘制文本字符串。
draw_arc()	从指定的 x、y 坐标开始，以指定的宽度、高度和角度绘制圆弧。
fill_arc()	从指定的 x、y 坐标开始，以指定的宽度、高度和角度绘制圆弧，并用指定的颜色进行填充(如果未定义，则使用 set_color 或黑色)。
draw_poly()	使用指定的 x-list、y-list 值绘制多边形。
fill_poly()	使用指定的 x-list、y-list 值绘制多边形，并用指定的颜色进行填充(如果未定义，则使用 set_color 或黑色)。
plot_xy()	使用指定的 x、y 坐标和 1-13 中指定的数字(代表不同的图形和符号)绘制图形(参见下文)。 

控件

项目	说明
clear()	清除整个屏幕。可与 x、y、宽度、高度参数一起使用以清除现有矩形。
clear_rect()	清除指定 x、y 坐标处具有指定宽度和高度的矩形。
set_color()	设置程序中后续图形的颜色，直到设置另一种颜色。
set_pen()	设置绘制图形时边框的指定厚度和样式(使用填充命令时不适用)。
set_window()	设置将在其中绘制任何图形的窗口的大小。 此函数用于调整窗口大小以匹配数据或更改绘图画布的原点(0,0)。
get_screen_dim()	返回屏幕尺寸的 xmax 和 ymax。
use_buffer()	启用离屏缓冲区，以加速绘图。
paint_buffer()	显示缓冲绘图输出。 在屏幕上显示多个对象可能导致延迟的情况下，use_buffer() 和 paint_buffer() 函数非常有用。

记事本

- 默认配置在屏幕左上角有(0,0)。指向右侧的正 x 轴和指向底部的正 y 轴，这可以通过使用 set_window() 函数进行修改。
- ti_draw 模块中的函数仅可在手持设备上及桌面版软件的手持设备视图中使用。

TI 图像菜单

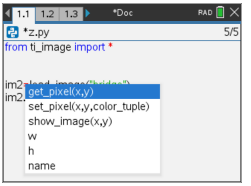
此子菜单位于**更多模块**项下方。

注: 创建使用此模块的新程序时，建议使用**图像处理**程序类型。这将确保导入所有相关模块。

项目	说明
from ti_image import *	从 ti_image 模块导入所有方法。
new_image(width,height,(r,g,b))	创建具有指定宽度和高度的新图像，以用于 Python 程序。 新图像的颜色由 (r,g,b) 值定义。
load_image("name")	加载由“name”指定的图像，以用于 Python 程序。 图像必须是“记事本”或“图形”应用程序中 TNS 文档的一部分。 “名称”提示将显示图像名称(如果先前已命名)或表示其插入顺序的数字。
copy_image(image)	创建由“图像”变量指定的图像副本。

图像对象的方法

通过输入后跟 . (dot) 的变量名称，使与图像对象相关的其他函数在编辑器和 Shell 中可用。



- **get_pixel(x,y):** 获取 (x,y) 坐标对所定义的位置处像素的 (r,g,b) 值。

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

- **set_pixel(x,y,color_tuple):** 将位置 (x,y) 处的像素设置为 color_tuple 中指定的颜色。

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

  将 (100,100) 处的像素设置为 (0,0,255) 颜色。

- **show_image(x,y):** 显示左上角位于位置 (x,y) 处的图像。
- **w, h, name:** 获取图像的宽度、高度和名称参数。

示例

```
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)
```

变量菜单

注: 这些列表不包括在任何其他 TI-Nspire™ 应用程序中定义的变量。

项目	说明
Vars: 当前程序	(仅限编辑器) 显示当前程序中定义的全局函数和变量列表
Vars:上次运行的程序	(仅限 Shell) 显示上次运行的程序中定义的全局函数和变量列表
Vars:全部	(仅限 Shell) 显示上次运行程序和任何导入模块的全局函数和变量列表

附录

Python 关键字

TI-Nspire™ Python 实现中内置以下关键字。

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

Python 键位映射

在编辑器或 Shell 中输入代码时，键盘的设计可以用于粘贴适当的 Python 操作或打开菜单，以便轻松输入函数、关键字、方法、运算符等。

键位	映射
var	打开“变量”菜单
sto→	粘贴 = 符号
del	删除光标左侧的字符
clear	不执行任何动作
=	粘贴 = 符号
|≠≥>	粘贴所选符号： • > • < • != • >= • <= • == • 和 • 或 • not • | • & • ~
trig	粘贴所选函数： • sin • cos • tan • atan2 • asin • acos • atan
?	显示提示
:=	粘贴:=
^	粘贴 **
√	不执行任何动作
x²	粘贴 **2

键位	映射
√	粘贴 sqrt()
×	粘贴乘号 (*)
""	粘贴一个双引号 (")
÷	粘贴除号 (/)
a/b	不执行任何动作
e^x	粘贴 exp()
ln	粘贴 log()
10^x	粘贴 10**
log	粘贴 log(value,base)
(	粘贴 (
)	Pastes)
[]	粘贴 [ ]
{}	粘贴 { }
−	粘贴减号 (-)
≈	在当前行之后添加新行
EE	粘贴 E
?!•	粘贴所选符号: • ? • ! • $ • ° • ' • % • " • : • ; • _ • \ • #
π•	粘贴 "pi"
⇐	现有标志行为
↵	在当前行之后添加新行

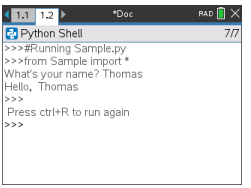
Python 程序样本

使用以下程序样本熟悉 Python 方法。也可在位于**示例**文件夹中的**快速入门 Python.tns** 文件中找到它们。

注: 如果将任何包含制表符缩进指示符的样本代码 (••) 复制并粘贴到 TI-Nspire™ 软件中，则需要将这些实例替换为实际的制表符缩进。

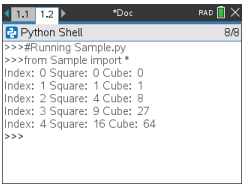
您好

```
# 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")
```



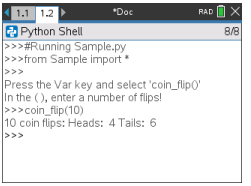
循环示例

```
# 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)
```



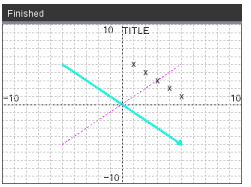
正面或反面

```
# 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!")
```



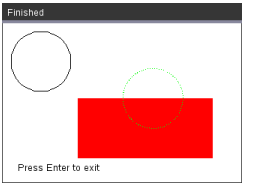
绘制

```
# 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")
```



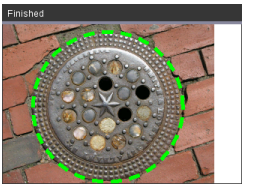
绘制

```
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")
```



图像

```
# 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

此程序使用 Python 控制 TI-Innovator™ Hub(可编程微控制器) 。在不连接 TI-Innovator™ Hub 的情况下运行程序将显示错误消息。

有关 TI-Innovator™ Hub 的更多信息，请访问 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()
```

一般信息

在线帮助

education.ti.com/eguide

选择您的国家，获取更多产品信息。

联络 TI 支持部门

education.ti.com/ti-cares

选择您的国家，获取技术和其他支持资源。

维修和保修信息

education.ti.com/warranty

选择您所在的国家/地区，了解有关保修期限和条款或产品服务的信息。

保修期内不会影响您的法定权利。

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243