TI-Nspire™ Python 編程手冊

請透過 education.ti.com/eguide 的線上說明，瞭解更多有關 TI 技術的資訊。

重要資訊

除伴隨程式的授權中明確陳述之外，德州儀器概不提供有關任何程式或書籍資料的明示或暗示保證，包括但不限於任何可售性和特別目的適合性的暗示保證，並且僅按「原樣」提供此等資料。無論任何情況，德州儀器皆不負責與購買或使用這類資料有關或所致的任何特殊、附屬、附帶或衍生損害賠償，且無論行動的形式，德州儀器的唯一責任不會超過程式授權中載明的金額。此外，德州儀器不承擔任何種類的賠償責任，不管是否有任何其他當事人因使用這些材料而索賠。

© 2020 Texas Instruments Incorporated

「Python」和 Python 標誌為 Python 軟體基金會的商標或註冊商標，由 Texas Instruments Incorporated 在取得基金會許可的情況下使用。

產品實物可能與所提供的圖片資料略有不同。

目錄

「Python 編程」新手入門

您可以將 Python 與 TI-Nspire™ 產品搭配使用，以便:

- 將 Python 程式新增至 TNS 檔案
- 使用範本建立 Python 程式
- 與其他 TI-Nspire™ 應用程式互動並共用資料
- 與 TI-Innovator™ 套裝 (Hub) 和 TI-Innovator™ Rover 自走車互動。

TI-Nspire™ Python 實作以 MicroPython 為基礎，後者是專為在微控制器上執行所設計的 Python 3 標準資料庫小型子集合。原始的 MicroPython 實作已經改編供 TI 使用。

注意: 某些數字答案可能不同於「計算工具」的結果，原因是基礎數學實作中存在差異。

以下 TI-Nspire™ 產品提供 Python:

計算機	桌面軟體
TI-Nspire™ CX II	TI-Nspire™ CX Premium 教師版軟體
TI-Nspire™ CX II CAS	TI-Nspire™ CX CAS Premium 教師版軟體
TI-Nspire™ CX II-T	TI-Nspire™ CX 學生版軟體
TI-Nspire™ CX II-T CAS	TI-Nspire™ CX CAS Student Software
TI-Nspire™ CX II-C	
TI-Nspire™ CX II-C CAS	

注意: 在多數情形下，計算機與軟體檢視之間的功能相同，但是您可能會看到一些差異。本手冊將假設您使用計算機裝置或是軟體中的計算機檢視。

Python 模組

TI-Nspire™ Python 包括下列模組:

標準模組	TI 模組
數學 (math)	TI PlotLib (ti_plotlib)
隨機 (random)	TI Hub (ti_hub)
複數數學 (cmath)	TI Rover (ti_rover)
時間 (time)	TI System (ti_system)
	TI Draw (ti_draw)
	TI Image (ti_image)

注意: 如果您現有的 Python 程式建立在其他 Python 開發環境中，您可能需要對其進行編輯才能在 TI-Nspire™ Python 解決方案中執行。模組可以在程式中使用與 TI 模組不同的方法、引數以及方法順序。一般而言，在使用任何版本的 Python 以及 Python 模組時，都請留意相容性的問題。

在將 Python 程式從非 TI 平台轉移至 TI 平台，或是從某個 TI 產品轉移至其他產品時，請記得:

- 使用核心語言功能以及標準函式庫(math、random 等) 的程式無需更改即可移轉。
- 使用平台特定函式庫的程式(例如 PC 版 matplotlib 或是 TI 模組) 將需要進行編輯，之後才能在不同的平台上執行。即使在 TI 平台也可能如此。

與任何 Python 版本一樣，您需要加入匯入資料才能使用包含在給定模組中的任何函數、方法或是常數。舉例來說，如果要從數學模組執行 cos() 函數，請使用下列指令:

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

如需查看包含項目與說明的功能表的清單，請參閱功能表概覽一節。

Python 工作空間

共有兩個工作空間供您在 Python 編程時使用:Python 編輯器以及 Python Shell。

Python 編輯器	Python Shell
• 建立、編輯以及儲存 Python 程式 • 語法突顯與自動縮排 • 行內提示以指導函數引數 • 工具提示可顯示有效值的範圍 • var 鍵可列出目前程式中定義的全域使用者變數與函數 • 小鍵盤快速鍵	• 執行 Python 程式 • 方便測試小型程式碼片段 • 與 Shell 歷史記錄互動以便選取先前的輸入與輸出並重複使用 • var 鍵可列出在特定問題中執行的最後程式中所定義的全域使用者變數

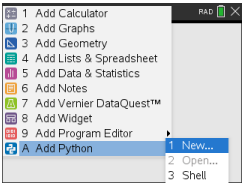
注意: 可以在問題中新增多個 Python 程式與 Shell。

Python 編輯器

Python 編輯器是您可以在其中建立、編輯與儲存 Python 程式的地方。

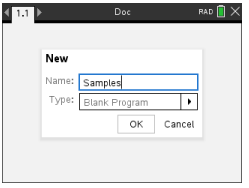
新增 Python 編輯器頁面

若要在目前的問題中加入新的 Python 編輯器頁面，請按 menu 然後選擇 **新增 Python > 新增**。

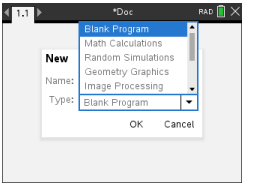


您可以建立空白程式，或是選取一個範本。

空白程序



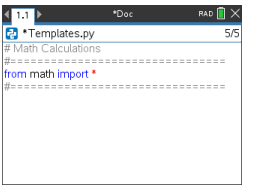
範本



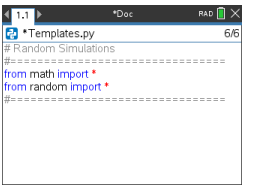
建立程式之後，系統將會顯示 Python 編輯器。如果您選取了範本，系統將會自動新增必要的匯入語句(請見下方) 。

注意：您可以將多個程式儲存在單一 TNS 檔案中，如同其他應用程式一樣。如果 Python 程式的用途是作為模組使用，則可以將 TNS 檔案儲存在 PyLib 資料夾中。該模組隨後可以用於其他程式和文件。

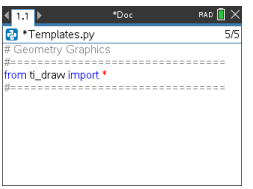
數學計算



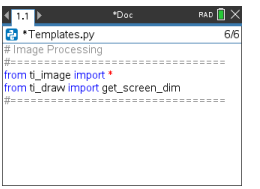
隨機模擬



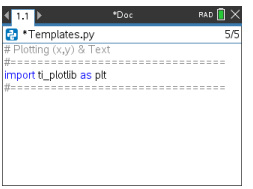
幾何作圖



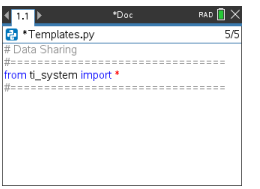
影像處理



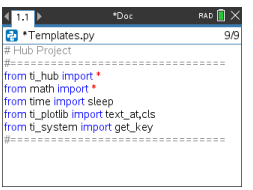
繪圖 (x,y) 與文字



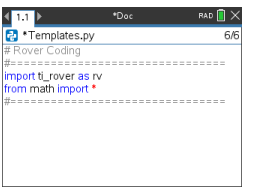
資料分享



TI-Innovator 套裝 (Hub) 專案



TI-Rover 編碼



開啟 Python 程式

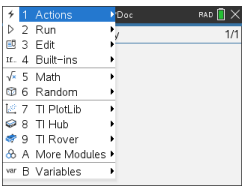
若要開啟現有的 Python 程式，請按下 doc 並選取**插入 > 新增 Python > 開啟**。如此將會顯示已經在 TNS 檔案中儲存的程式清單。

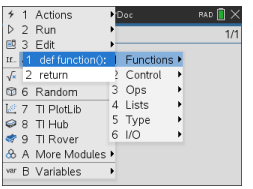
如果已經刪除用於建立程式的編輯器頁面，則仍然可以在 TNS 檔案中使用程式。

使用 Python 編輯器

按下 menu 可顯示「文件工具」功能表。您可以利用這些功能表選項為您的程式新增、移動以及複製程式碼區塊。

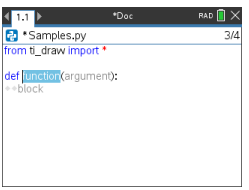
[文件工具]功能表



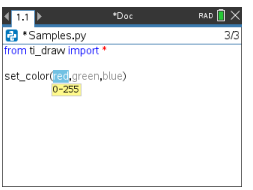


從模組功能表選取的項目會自動將程式碼範本新增至編輯器，並且包含每個函數部分的行內提示。您可以從某個引數瀏覽至下個引數，只需按下 tab (向前) 或 shift+tab (向後) 即可。在需要時將會出現工具提示以及彈出清單，以協助您選取適當的值。

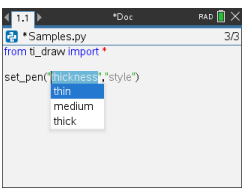
行內提示



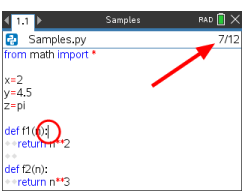
工具提示



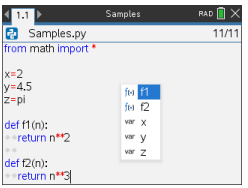
彈出清單



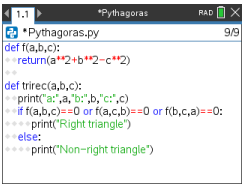
程式名稱右邊的數字可反映游標目前的行數以及程式中的總行數。



按下 var 然後從清單中選取，即可插入在目前游標位置上方各行中定義的全域函數與變數。



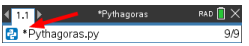
在您將程式碼新增至程式時，編輯器會以不同顏色顯示關鍵字、運算元、備註、字串以及縮排，以協助識別不同的元素。



儲存並執行程式

當您完成程式之後，請按 menu 然後選取**執行 > 檢查語法並儲存**。如此將會檢查 Python 程式的語法並將其儲存在 TNS 檔案中。

注意: 如果您的程式中有尚未儲存的變更，將會在程式名稱旁邊顯示一個星號。



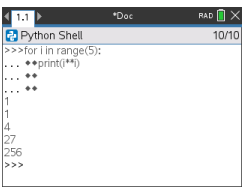
若要執行程式，請按下 menu，然後選取**執行 > 執行**。如此將會在下個 Python Shell 頁面執行目前的程式，如果下個頁面不是 Shell，則會在新頁面中執行目前程式。

注意: 執行程式可自動檢查語法並儲存程式。

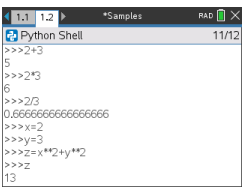
Python Shell

Python Shell 是執行您的 Python 程式、其他 Python 程式碼片段或是簡單指令的解譯器。

Python 程式碼

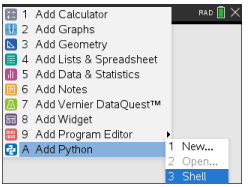


簡單指令

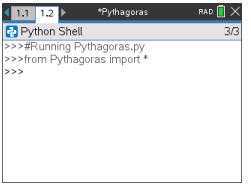


新增 Python Shell 頁面

若要在目前的問題中加入新的 Python Shell 頁面，請按下 menu 並選取**新增 Python > Shell**。



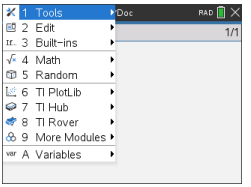
Python Shell 也可以透過 Python 編輯器啟動，啟動方式是按 menu 然後選取**執行 > 執行**。

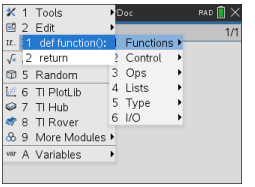


使用 Python Shell

按下 menu 可顯示「文件工具」功能表。您可以利用這些功能表選項新增、移動以及複製程式碼區塊。

[文件工具]功能表

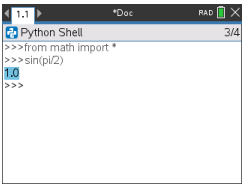




注意：如果您從其中一個可用模組使用任何方法，請務必先在任何 Python 編碼環境中執行匯入模組語句。

與 Shell 輸出的互動類似於「計算工具」應用程式，您可以在其中選取並複製先前的輸入與輸出，以便用於 Shell、編輯器或是其他應用程式中的其他地方。

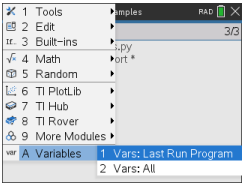
使用向上箭頭選取，然後複製並貼至想要的位置



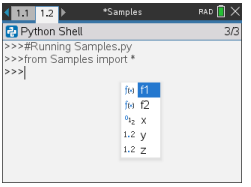
您可以從上次執行的程式插入全域函數與變數，方法是按下 var 或是 ctrl+L 然後從清單中選取，或是按 menu 然後選擇**變數 > 變數:上次執行的程式**。

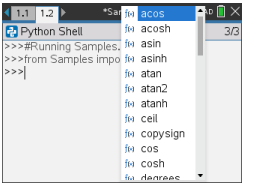
若要從上次執行的程式以及任何匯入模組中選擇全域函數與變數的清單，請按 menu 然後選擇**變數 > 變數:全部**。

「變數」功能表



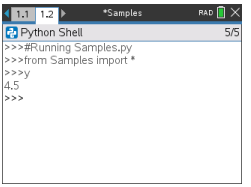
上次執行的程式變數　　*所有變數*

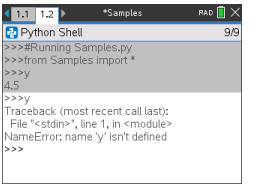




相同問題中的所有 Python Shell 頁面都有相同的狀態(使用者定義和匯入的變數定義)。當您在該問題中儲存或執行 Python 程式，或按 menu 並選擇**工具 > 重新初始化 Shell** 時，Shell 歷史記錄將顯示灰色背景，表示先前狀態不再有效。

儲存或重新初始化之前　　*儲存或重新初始化之後*





注意: menu **工具 > 清除歷史記錄**選項會清除 Shell 中任何先前活動的畫面，但變數仍然可用。

訊息

當您在 Python 作業階段中時，系統可能會顯示錯誤和其他資訊訊息。如果程式執行時在 Shell 中出現錯誤，則系統會顯示程式列號。按 ctrl menu 並選擇**前往 Python 編輯器**。在編輯器中，按 menu，然後選擇**編輯 > 前往指定行**。輸入行號並按 enter。游標將顯示在發生錯誤的行的第一個字元上。

中斷執行中的程式

當程式或函數在執行中時，系統會顯示忙碌中的游標 ⧖。

- 停止程式或函數:
  - Windows®:按 **F12** 鍵。
  - Mac®:按 **F5** 鍵。
  - 計算機:按 on 鍵。

Python 功能表概覽

本節列出 Python 編輯器以及 Shell 的所有功能表與功能表項目，並且提供這些內容的簡要說明。

注意: 對於有鍵盤快速鍵的功能表項目，Mac® 使用者應該將使用 **Ctrl** 的地方用 ⌘ (**Cmd**) 代替。如需查看 TI-Nspire™ 計算機與軟體快速鍵的完整清單，請參閱 TI-Nspire™ 科技電子手冊。

「動作」功能表

注意: 此功能表僅適用於編輯器。

項目	說明
新增	開啟**新增**對話方塊並在對話方塊中輸入名稱，然後選擇新程式的類型。
開啟	開啟可以在目前文件中使用的程式清單。
建立副本	開啟**建立副本**對話方塊，您可以將目前程式以不同名稱儲存在此對話方塊中。
重新命名	開啟**重新命名**對話方塊，您可以在其中變更目前程式的名稱。
關閉	關閉目前程式。
設定	開啟**設定**對話方塊，您可以在其中變更編輯器與 Shell 的字型大小。

「執行」功能表

注意：此功能表僅適用於編輯器。

項目	快速鍵	說明
運行	Ctrl+R	檢查語法、儲存程式並在 Python Shell 中執行。
檢查語法並儲存	Ctrl+B	檢查語法並儲存程式。
前往 Shell	不適用	將焦點移至與目前程式相關的 Shell，或在編輯器旁邊開啟新的 Shell 頁面。

「工具」功能表

注意: 此功能表僅適用於 Shell。

項目	快速鍵	說明
重新執行最後一個程式	Ctrl+R	重新執行與目前 Shell 有關的最後程式。
前往 Python 編輯器	不適用	開啟與目前 Shell 有關的編輯器頁面。
運行	不適用	開啟可以在目前文件中使用的程式清單。 在選取之後，將會執行選擇的程式。
清除歷史記錄	不適用	清除目前 Shell 中的歷史記錄，但是不要重新初始化 Shell。
重新初始化 Shell	不適用	在目前問題中重設所有開啟 Shell 頁面的狀態。 將無法再使用所有定義的變數以及匯入的函數。
dir()	不適用	在匯入語句之後使用時，顯示指定模組中的函數目錄。
從程式匯入 *	不適用	開啟可以在目前文件中使用的程式清單。 在選取之後，系統會將匯入語句貼至 Shell。

編輯功能表

注意: Ctrl+A 可選取所有程式碼的各行或輸出，以剪下或刪除(僅編輯器)，或是複製並貼上(編輯器與 Shell)。

項目	快速鍵	說明
縮排	TAB*	縮排目前行或是選取行的文字。 * 如果有不完整的行內提示，TAB 將會瀏覽至下一提示。
取消縮排	Shift+TAB**	取消縮排目前行或是選取行的文字。 ** 如果有不完整的行內提示，Shift+TAB 將會瀏覽至前一提示。
註解/取消註解	Ctrl+T	在目前行的開頭新增/移除註解符號。
插入多行字串	不適用	(僅編輯器) 插入多行字串範本。
尋找	Ctrl+F	(僅編輯器) 開啟**尋找**對話方塊，並在目前程式中搜尋輸入的字串。
取代	Ctrl+H	(僅編輯器) 開啟**取代**對話方塊，並在目前的程式中搜尋輸入的字串。
前往指定行	Ctrl+G	(僅編輯器) 開啟**前往指定行**對話方塊，並跳至目前程式中的指定行。
行首	Ctrl+8	將游標移至目前行的開頭。
行尾	Ctrl+2	將游標移至目前行的結尾。
跳至最上方	Ctrl+7	將游標移至程式中第一行的開頭。
跳至最下方	Ctrl+1	將游標移至程式中最後一行的結尾。

「內建項目」功能表

函數

項目	說明
def function():	定義依賴指定變數的函數。
return	定義函數所產生的值。

控制

項目	說明
if..	條件陳述式。
if..else..	條件陳述式。
if..elif..else..	條件陳述式。
for index in range(size):	反覆查看某個範圍。
for index in range(start,stop):	反覆查看某個範圍。
for index in range(start,stop,step):	反覆查看某個範圍。
for index in list:	反覆查看列表元素。
while..	執行程式碼區塊中的陳述式，直到條件評估為 False 為止。
elif:	條件陳述式。
else:	條件陳述式。

運算

項目	說明
x=y	設定變數值。
x==y	貼上等於 (==) 比較運算元。
x!=y	貼上不等於 (!=) 比較運算元。
x>y	貼上大於 (>) 比較運算元。
x>=y	貼上大於或等於 (>) 比較運算元。
x<y	貼上小於 (<) 比較運算元。
x<=y	貼上小於或等於 (<=) 比較運算元。

項目	說明
與	貼上和 (and) 邏輯運算元。
或	貼上或 (or) 邏輯運算元。
非	貼上非 (not) 邏輯運算元。
真	貼上 True 布林值。
假	貼上 False 布林值。

清單

項目	說明
[]	貼上括弧 ([])。
list()	將序列轉換為「列表」類型。
len()	傳回列表的元素數目。
max()	傳回列表中的最大值。
min()	傳回列表中的最小值。
.append()	此方法會將元素附加至列表。
.remove()	此方法會移除列表中元素的第一個實例。
range(start,stop,step)	傳回一組數字。
for index in range(start,stop,step)	用於反覆查看某個範圍。
.insert()	此方法會在指定位置新增元素。
.split()	此方法會傳回以指定分隔符號隔開的元素列表。
sum()	傳回列表中元素的和。
sorted()	傳回已排序列表。
.sort()	此方法會對列表進行適當排序。

類型

項目	說明
int()	傳回整數部份。
float()	傳回浮點值。
round(x,ndigits)	傳回四捨五入到指定小數點位數的浮點數。

項目	說明
str()	傳回字串。
complex()	傳回複數。
type()	傳回物件類型。

I/O

項目	說明
print()	將引數顯示為字串。
input()	提示使用者輸入。
eval()	計算以字串形式顯示的表達式。
.format()	設定指定字串格式的方法。

「數學」功能表

注意: 在建立使用此模組的新程式時，建議使用**數學計算**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from math import *	從數學模組匯入所有方法(函數)。
fabs()	傳回實數的絕對值。
sqrt()	傳回實數的平方根。
exp()	傳回 e**x。
pow(x,y)	傳回 x 的 y 乘冪。
log(x,base)	傳回 $\log_{base}(x)$。 沒有底數的 log(x) 傳回自然對數 x。
fmod(x,y)	傳回 x 與 y 的模組值。當 x 和 y 為浮點值時使用。
ceil()	傳回大於或等於實數的最小整數。
floor()	傳回小於或等於實數的最大整數。
trunc()	將實數截斷為整數。
frexp()	傳回一對 (y,n)，其中 x == y * 2**n。

Const

項目	說明
e	Returns value for the constant e.
pi	Returns value for the constant pi.

Trig

項目	說明
radians()	將角度單位從度數轉換為徑度。
degrees()	將角度單位從徑度轉換為度數。
sin()	傳回引數正弦的徑度。
cos()	傳回引數餘弦的徑度。
tan()	傳回引數正切的徑度。

項目	說明
asin()	傳回引數反正弦的徑度。
acos()	傳回引數反餘弦的徑度。
atan()	傳回引數反正切的徑度。
atan2(y,x)	傳回 y/x 反正切的徑度。

「隨機」功能表

注意: 在建立使用此模組的新程式時，建議使用**隨機模擬**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from random import *	從隨機模組匯入所有方法。
random()	傳回從 0 到 1.0 的浮點數。
uniform(min,max)	傳回隨機亂數 x(浮動)，使得最小值 <= x <= 最大值。
randint(min,max)	傳回最小值與最大值之間的隨機整數。
choice(sequence)	自非空白序列傳回隨機元素。
randrange(start,stop,step)	從開始到停止逐步傳回隨機亂數。
seed()	初始化隨機亂數產生器。

TI PlotLib 功能表

注意: 在建立使用此模組的新程式時，建議使用**繪製 (x,y) 與文字** 程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
import ti_plotlib as plt	從「plt」命名空間中的 ti_plotlib 模組匯入所有方法(函數)。因此，從功能表貼上的所有函數名稱最前面都會有「plt」。

設定

項目	說明
cls()	清除繪圖畫布。
grid(x-scale,y-scale,"style")	顯示在 x 軸和 y 軸使用指定比例的格線。
window(xmin,xmax,ymin,ymax)	透過將指定的水平間隔 (xmin, xmax) 與鉛直間隔 (ymin, ymax) 對應至分配的繪圖區域(像素)以定義繪圖視窗。
auto_window(x-list,y-list)	自動調整繪圖視窗，讓資料範圍能夠納入 auto_window() 之前在程式中指定的 x-list 與 y-list 中。
axes("mode")	在繪圖區域的指定視窗中顯示軸。
labels("x-label","y-label",x,y)	在繪圖軸上的列位置 x 和 y 顯示「x-label」和「y-label」標籤。
title("title")	在視窗最上方的一行置中顯示「title」。
show_plot()	顯示緩衝的繪圖輸出。 use_buffer() 和 show_plot() 函數可用於在畫面中顯示多個物件時可能會造成延遲的情況(大多數情況不一定會延遲)。
use_buffer()	啟用螢幕外緩衝區以加快繪圖速度。

繪圖:

項目	說明
color(red,green,blue)	設定下列所有函數繪圖/繪圖的顏色。
cls()	清除繪圖畫布。

項目	說明
show_plot()	依照程式中的設定執行繪圖的顯示。
scatter(x-list,y-list,"mark")	從 (x-list,y-list) 繪製擁有指定標記樣式的有序數對。
plot(x-list,y-list,"mark")	使用有序數對從指定的 x-list 和 y-list 繪製線條。
plot(x,y,"mark")	使用座標 x 和 y 以指定的標記樣式繪製一個點。
line(x1,y1,x2,y2,"mode")	繪製從 (x1,y1) 到 (x2,y2) 的線段。
lin_reg(x-list,y-list,"display")	計算並繪製 x-list,y-list 的線性回歸模型 ax+b。
pen("size","style")	在下個 pen() 執行之前，設定所有下列線條的外觀。
text_at(row,"text","align")	在繪圖區域以指定的「align」顯示「text」。

屬性

項目	說明
xmin	視窗引數的指定變數定義為 plt.xmin。
xmax	視窗引數的指定變數定義為 plt.xmax。
ymin	視窗引數的指定變數定義為 plt.ymin。
ymax	視窗引數的指定變數定義為 plt.ymax。
公尺	在程式中執行 plt.linreg() 之後，斜率 m 以及截距 b 的計算值會儲存在 plt.m 和 plt.b 中。
b	在程式中執行 plt.linreg() 之後，斜率 a 以及 y-截距 b 的計算值會儲存在 plt.a 和 plt.b 中。

「TI 套裝 (Hub)」功能表

注意: 在建立使用此模組的新程式時，建議使用**套裝 (Hub) 專案**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from ti_hub import *	從 ti_hub 模組匯入所有方法。

套裝 (Hub) 內建裝置 > 色彩輸出

項目	說明
rgb(red,green,blue)	設定 RGB LED 的色彩。
blink(frequency,time)	設定選取色彩的閃爍頻率以及持續期間。
off()	關閉 RGB LED。

套裝 (Hub) 內建裝置 > 光源輸出

項目	說明
on()	開啟 LED。
off()	關閉 LED。
blink(frequency,time)	設定 LED 的閃爍頻率以及持續期間。

套裝 (Hub) 內建裝置 > 聲音輸出

項目	說明
tone(frequency,time)	以指定頻率於指定持續期間播放音調。
note("note",time)	在指定時間播放指定的筆記。 筆記是使用筆記名稱和 octave 指定。範例:A4, C5。 筆記名稱為 C、CS、D、DS、E、F、FS、G、GS、A、AS 和 B。 octave 號碼範圍為從 1 至 9(包含首尾)。

套裝 (Hub) 內建裝置 > 亮度感應器輸入

項目	說明
measurement()	讀取內建亮度(光源等級)感應器並傳回讀數。 預設範圍為 0 到 100。可以使用 range() 函數改變範圍大小。
range(min,max)	設定從光源等級感應器對應讀數的範圍。 如果兩者資料皆遺失，或是將值設定為「無」，則會將亮度範圍設定在 0 到 100 之間。

新增輸入裝置

此功能表有 ti_hub 模組支援的感應器清單(輸入裝置)。所有功能表項目都將貼上物件名稱，並且應該會有一個變數以及和感應器搭配使用的連接埠。每個感應器都有一個 measurement() 方法，可傳回感應器數值。

項目	說明
DHT(數位濕度與溫度)	傳回清單，內容包含目前溫度、濕度、感應器類型以及上次快取的讀取狀態。
距離感應器	從指定超音波距離感應器傳回目前的距離測量結果。
亮度等級	從外部光源等級(亮度)感應器傳回光源等級。
溫度	從外部溫度感應器傳回溫度讀數。 預設配置可支援 IN 1、IN 2 或是 IN 3 連接埠的 Seeed 溫度感應器。 若要使用 TI-Innovator™ 套裝 (Hub) 麵包板套組中的 TI LM19 溫度感應器，請編輯使用中的 BB 針腳連接埠，並且使用選用引數「TIANALOG」。 範例 :mylm19=temperature("BB 5","TIANALOG")
濕度	傳回濕度感應器讀數。
磁性	偵測磁場是否存在。 確定磁場是否存在的閾值透過 trigger() 函數設定。 閾值的預設值為 150。
威尼爾	從指令中指定的威尼爾類比感應器讀取數值。 指令支援下列威尼爾感應器: • **temperature** - 不銹鋼溫度感應器。 • **lightlevel** - TI 光源感應器。 • **pressure** - 原始氣壓感應器 • **pressure** - 較新的氣壓感應器。 • **pH** - pH 感應器。

項目	說明
	• **force10** - ±10 N 設定，雙範圍力感應器。 • **force50** - ±50 N 設定，雙範圍力感應器。 • **accelerometer** - 低重力加速度計。 • **generic** - 允許未直接支援上述感應器的其他感應器設定，以及使用上方的 calibrate() API 設定方程式係數。
類比輸入	支援使用類比輸入通用裝置。
數位輸入	傳回連接至 DIGITAL 物件的數位針腳目前的狀態，或是上次為物件設定的數位輸出值的快取狀態。
電位計	支援電位計感應器。 可以使用 range() 函數變更感應器的範圍。
電熱調節器	讀取熱敏電阻感應器。 預設係數的作用是在與 10KΩ 固定電阻搭配使用時，能夠與 TI-Innovator™ 套裝 (Hub) 的麵包板套組中包括的熱敏電阻感應器相匹配。 可以使用 calibrate() 函數為熱敏電阻感應器設定一組新的校正係數和基準電阻。
響度	支援聲音響度感應器。
色彩輸入	提供 I2C 連接的色彩輸入感應器平台。 除了 I2C 連接埠，還可使用 bb_port 針腳以控制色彩感應器上的 LED。 • **color_number():** 傳回 1 到 9 的數值，代表感應器偵測到的色彩。 數字代表以下對應的顏色: 1: 紅色 2:綠色 3: 藍色 4: 青色 5: 洋紅色 6: 黃色 7: 黑色 8: 白色 9: 灰色 • **red():** 傳回 0 到 255 的數值，代表所偵測的紅色等級的強度。 • **green():** 傳回 0 到 255 的數值，代表所偵測的綠色等級的強度。 • **blue():** 傳回 0 到 255 的數值，代表所偵測的藍色等級的強度。

項目	說明
	• **gray():** 傳回 0 到 255 的數值，代表所偵測的灰色等級，其中 0 為黑色而 255 為白色。
BB 連接埠	為使用所有 10 個 BB 連接埠針腳作為合併的數位輸入/輸出連接埠提供支援。 初始化函數包含選用的「mask」參數，可允許使用由 10 個針腳組成的子集合。 • **read_port():** 讀取 BB 連接埠輸入針腳上的目前值。 • **write_port(value):** 將輸出針腳值設定為指定值，此值需介於 0 和 1023 之間。請注意，如果有提供遮罩，此值也會根據 var=bbport(mask) 運算中的遮罩值進行調整。
套裝 (Hub) 時間	提供內部毫秒計時器的存取權限。
TI-RGB Array	提供用於對 TI-RGB 陣列進行編程的函數。 初始化函數可接受選用「LAMP」參數，以便為需要外部電源的 TI-RGB 陣列啟用高亮度模式。 • **set(led_position, r,g,b):** 將特定的 led_position (0-15) 設定為指定的 r,g,b 值，其中 r,g,b 為 0 到 255 之間的值。 • **set_all(r,g,b):** 將陣列中的所有 RGB LED 設定為相同的 r,g,b 值。 • **all_off():** 關閉陣列中的所有 RGB。 • **measurement():** 傳回 RGB 陣列正透過使用 TI-Innovator™使用的近似目前繪圖 (單位為毫安) 。 • **pattern(pattern):** 使用引數值作為 0 到 65535 範圍內的二進制值，數值 1 表示開啟像素。LED 已開啟，RED 的 pwm 等級值為 255。

新增輸出裝置

此功能表有 ti_hub 模組支援的輸出裝置清單。所有功能表項目將貼上物件名稱，並且應該會有一個變數以及和裝置搭配使用的連接埠。

項目	說明
LED	用於控制外部連接 LED 的函數。
RGB	支援控制外部 RGB LED。
TI-RGB Array	提供用於對 TI-RGB 陣列進行編程的函數。
喇叭	支援外接喇叭與 TI-Innovator™ 套裝 (Hub) 的函數。 這些函數與上述用於「sound」的函數相同。
功率	用於控制外部電源與 TI-Innovator™ 套裝 (Hub) 的函數。

項目	說明
	• **set(value):** 將功率等級設定為指定值(0 與 100 之間)。 • **on():** 將功率等級設定為 100。 • **off():** 將功率等級設定為 0。
連續作用的伺服	用於控制連續作用的伺服馬達的函數。 • **set_cw(speed,time):** 伺服將以指定的速度 (0-255) 沿順時針方向旋轉，並以指定的持續時間(以秒為單位)旋轉。 • **set_ccw(speed,time):** 伺服將以指定的速度 (0-255) 沿反時針方向旋轉，並以指定的持續時間(以秒為單位)旋轉。 • **stop():** 停止連續作用的伺服。
類比輸出	用於使用類比輸入通用裝置的函數。
震動馬達	用於控制震動馬達的函數。 • **set(val):** 將震動馬達強度設定為「val」(0-255)。 • **off():** 關閉震動馬達。 • **on():** 將震動馬達開啟至最高強度。
繼電器	用於控制繼電器的控制平台。 • **on():** 將繼電器設定為「開啟」狀態。 • **off():** 將繼電器設定為「關閉」狀態。
伺服	用於控制伺服馬達的函數。 • **set_position(pos):** 將掃頻伺服位置設定在 -90 到 +90 的範圍。 • **zero():** 將掃頻伺服位置設定在零位置。
方波	用於產生方波的函數。 • **set(frequency,duty,time):** 以 50% 的預設工作週期設定輸出方波(如果沒有指定工作)以及「frequence」指定的輸出頻率。頻率可以介於 1 到 500 Hz 之間。工作週期(如果有指定)可以介於 0 到 100% 之間。 • **off():** 關閉方波。
數位輸出	用於控制數位輸出的平台。 • **set(val):** 將數位輸出設定為「val」指定的數值(0 或 1)。 • **on():** 將數位輸出的狀態設定為高 (1)。 • **off():** 將數位輸出的狀態設定為低 (0)。
BB 連接埠	提供用於對 TI-RGB 陣列進行編程的函數。 請參閱上述詳細資料。

指令

項目	說明
sleep(seconds)	在指定的秒數期間暫停執行程式。 從「time」模組匯入。
text_at(row,"text","align")	以指定的「align」將指定的「text」顯示在繪圖區域。 ti_plotlib 模組的部分。
cls()	清除 Shell 畫面以進行繪製。 ti_plotlib 模組的部分。
while get_key() != "esc":	在「while」迴圈中執行指令，直到按下「esc」鍵為止。
get_key()	傳回代表按鍵已按下的字串。 「1」鍵會傳回「1」，「esc」鍵會傳回「esc」等。 呼叫時沒有包含任何參數 - get_key() - 會立即傳回。 呼叫時包含參數 - get_key(1) - 會等到按下按鍵後再傳回。 ti_system 模組的部分。

連接埠

這些是 TI-Innovator™ 套裝 (Hub) 中可用的輸入與輸出連接埠。

項目
OUT 1
OUT 2
OUT 3
IN 1
IN 2
IN 3
BB 1
BB 2
BB 3
BB 4
BB 5

項目
BB 6
BB 7
BB 8
BB 9
BB 10
I2C

「TI Rover」功能表

注意: 在建立使用此模組的新程式時，建議使用 **Rover 編碼**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
將 ti_rover 匯入為 rv	從「rv」命名空間中的 ti_rover 模組匯入所有方法(函數)。因此，從功能表貼上的所有函數名稱最前面都會有「rv」。

驅動

項目	說明
forward(distance)	以指定距離(格線單位)將 Rover 向前移動。
backward(distance)	以指定距離(格線單位)將 Rover 向後移動。
left(angle_degrees)	以指定角度(度數)將 Rover 向左轉動。
right(angle_degrees)	以指定角度(度數)將 Rover 向右轉動。
stop()	立即停止任何目前的動作。
stop_clear()	立即停止所有目前動作並清除所有待處理的指令。
resume()	繼續處理指令。
stay(time)	Rover 在指定的時間(秒)內維持在原位(選用)。 如果沒有指定時間，Rover 會維持 30 秒不動。
to_xy(x,y)	將 Rover 移至虛擬格線上的座標位置 (x,y)。
to_polar(r,theta_degrees)	將 Rover 移至虛擬格線上的極座標位置 (r, theta)。 角度單位為「度」。
to_angle(angle,"unit")	將 Rover 旋轉至虛擬格線中的指定角度。 角度是相對於在虛擬格線中向下指向 x 軸的零角度而言。

驅動 > 帶有選項的驅動

項目	說明
forward_time(time)	在指定的時間內將 Rover 向前移動。
backward_time(time)	在指定的時間內將 Rover 向後移動。

項目	說明
forward(distance,"unit")	以預設速度將 Rover 向前移動至指定距離。 可以使用格線單位、公尺或輪旋轉數指定距離。
backward(distance,"unit")	以預設速度將 Rover 向後移動至指定距離。 可以使用格線單位、公尺或輪旋轉數指定距離。
left(angle,"unit")	以指定角度將 Rover 向左轉動。 角度可以是度數、徑度或梯度。
right(angle,"unit")	以指定角度將 Rover 向右轉動。 角度可以是度數、徑度或梯度。
forward_time(time,speed,"rate")	在指定時間以指定速度將 Rover 向前移動。 可以使用格線單位/秒、公尺/秒或輪旋轉數/秒指定速度。
backward_time(time,speed,"rate")	在指定時間以指定速度將 Rover 向後移動。 可以使用格線單位/秒、公尺/秒或輪旋轉數/秒指定速度。
forward(distance,"unit",speed,"rate")	以指定速度將 Rover 向前移動指定距離。 可以使用格線單位、公尺或輪旋轉數指定距離。 可以使用格線單位/秒、公尺/秒或輪旋轉數/秒指定速度。
backward(distance,"unit",speed,"rate")	以指定速度將 Rover 向後移動指定距離。 可以使用格線單位、公尺或輪旋轉數指定距離。 可以使用格線單位/秒、公尺/秒或輪旋轉數/秒指定速度。

輸入

項目	說明
ranger_measurement()	讀取 Rover 前方的超音波距離感應器，並傳回目前的距離(公尺)。
color_measurement()	傳回 1 到 9 的數值，指出 Rover 色彩輸入感應

項目	說明
	器「看見」的主要色彩。 1 = 紅色 2 = 綠色 3 = 藍色 4 = 青色 5 = 洋紅色 6 = 黃色 7 = 黑色 8 = 灰色 9 = 白色
red_measurement()	傳回介於 0 和 255 之間的值，指出色彩輸入感應器看見的感知紅色水準。
green_measurement()	傳回介於 0 和 255 之間的值，指出色彩輸入感應器看見的感知綠色水準。
blue_measurement()	傳回介於 0 和 255 之間的值，指出色彩輸入感應器看見的感知藍色水準。
gray_measurement()	傳回介於 0 和 255 之間的值，指出色彩輸入感應器看見的感知灰色水準。
encoders_gyro_measurement()	傳回包含左輪與右輪編碼器計數以及目前陀螺儀航向的值的清單。
gyro_measurement()	以度為單位傳回代表目前陀螺儀讀數的值 (包括漂移量)。

輸出

項目	說明
color_rgb(r,g,b)	將 Rover RGB LED 的色彩設定為指定的紅色、綠色與藍色值。
color_blink(frequency,time)	設定選取色彩的閃爍頻率以及持續期間。
color_off()	關閉 Rover RGB LED。
motor_left(speed,time)	在指定持續期間將左邊馬達馬力設定為指定值。 速度範圍為 -255 到 255，0 表示停止。正速度值是以逆時針方向旋轉，負速度值是順時針方向旋轉。 選用時間參數 (如果有指定) 的有效範圍為 0.05 到 655.35 秒。如果沒有

項目	說明
	指定，則會使用 5 秒的預設時間。
motor_right(speed,time)	在指定持續期間將左邊馬達馬力設定為指定值。 速度範圍為 -255 到 255，0 表示停止。正速度值是以逆時針方向旋轉，負速度值是順時針方向旋轉。 選用時間參數 (如果有指定) 的有效範圍為 0.05 到 655.35 秒。如果沒有指定，則會使用 5 秒的預設時間。
motors("ldir",left_val,"rdir",right_val,time)	將左輪與右輪設定為指定的速度等級，作為選用的時間量 (秒)。 速度 (left_val, right_val) 值的範圍為 0 到 255，0 表示停止。ldir 和 rdir 參數可指定各自輪的 CW 或 CCW 旋轉。 選用時間參數 (如果有指定) 的有效範圍為 0.05 到 655.35 秒。如果沒有指定，則會使用 5 秒的預設時間。

路徑

項目	說明
waypoint_xythdrn()	讀取 x-coord、y-coord、時間、航向、行經距離、輪旋轉數、目前定位點的指令數目。傳回包含所有這些元素值的列表。
waypoint_prev	讀取 x-coord、y-coord、時間、航向、行經距離、輪旋轉數、先前定位點的指令數目。
waypoint_eta	傳回驅動至定位點的預估時間。
path_done()	傳回 0 或 1 的值，取決於 Rover 是否正在移動 (0) 或是已完成所有移動 (1)。
pathlist_x()	傳回從開始到目前定位點 X 值 (含) 的 X 值列表。
pathlist_y()	傳回從開始到目前定位點 Y 值 (含) 的 Y 值列表。
pathlist_time()	傳回從開始到目前定位點時間值 (含) 的時間 (單位為秒) 列表。
pathlist_heading()	傳回從開始到目前定位點航向值 (含) 的航向列表。
pathlist_distance()	傳回從開始到目前定位點距離值 (含) 的行經距離列表。
pathlist_revs()	傳回從開始到目前定位點旋轉數值 (含) 的行經旋轉數列表。

項目	說明
pathlist_cmdnum()	傳回路徑的指令數目列表。
waypoint_x()	傳回目前定位點的 x 座標。
waypoint_y()	傳回目前定位點的 y 座標。
waypoint_time()	傳回從前一個定位點行經到目前定位點所花費的時間。
waypoint_heading()	傳回目前定位點的絕對航向。
waypoint_distance()	傳回前一個定位點與目前定位點之間的行經距離。
waypoint_revs()	傳回前一個定位點與目前定位點之間行經所需的旋轉數。

設定

項目	說明
units/s	將速度以每秒格線單位數顯示的選項。
m/s	將速度以每秒公尺數顯示的選項。
revs/s	將速度以每秒輪旋轉數顯示的選項。
單位	將距離以格線單位顯示的選項。
公尺	將距離以公尺顯示的選項。
revs	將距離以輪旋轉數顯示的選項。
degrees	顯示旋轉度數的選項。
radians	顯示旋轉徑度的選項。
gradians	顯示旋轉梯度的選項。
clockwise	指定輪方向的選項。
counter-clockwise	指定輪方向的選項。

指令

這些指令是其他模組以及 TI Rover 模組的函數集合。

項目	說明
sleep(seconds)	在指定的秒數期間暫停執行程式。 已從 time 模組匯入。
text_at(row,"text","align")	在繪圖區域以指定的「align」顯示「text」。

項目	說明
	已從 ti_plotlib 模組匯入。
cls()	清除 Shell 畫面以進行繪製。 已從 ti_plotlib 模組匯入。
while get_key() != "esc":	在「while」迴圈中執行指令，直到按下「esc」鍵為止。
wait_until_done()	暫停程式，直到 Rover 完成目前指令為止。 此指令對於將非 Rover 指令與 Rover 動作同步很有用。
while not path_done()	以「while」迴圈執行指令，直到 Rover 完成所有移動為止。 path_done() 函數傳回 0 或 1 的值，具體取決於 Rover 是否正在移動 (0) 或是已完成所有移動 (1)。
position(x,y)	將虛擬格線上的 Rover 位置設定為指定的 x,y 座標。
position(x,y,heading,"unit")	將虛擬格線上的 Rover 位置設定為指定的 x,y 座標，同時如果有提供航向 (以指定的角度單位)，則會設定相對於虛擬 x 軸的虛擬航向。 0 到 360 的正角是假設從正向 x 軸以逆時針方向旋轉。0 到 -360 的負角是假設從正向 x 軸以順時針方向旋轉。
grid_origin()	將 RV 設定為位於目前格線的 (0,0) 原點。
grid_m_unit(scale_value)	將每個單位的虛擬格線間隔公尺數 (m/unit) 設定為指定值。0.1 為預設 m/unit，並且轉換為 1 單位 = 100 mm 或 10 cm 或 1 dm 或是 0.1 m。 有效 scale_value 的範圍是從 0.01 到 10.0。
path_clear()	清除任何預先存在的路徑或是定位點資訊。
zero_gyro()	將 Rover 陀螺儀重設為 0.0 角度，並且清除左輪與右輪編碼器計數。

「複數數學」功能表

此子功能表位於**更多模組**下。

項目	說明
from cmath import *	從 cmath 模組匯入所有方法。
complex(real,imag)	傳回複數。
rect(modulus,argument)	將極座標轉換為複數的直角座標形式。
.real	傳回複數的實部。
.imag	傳回複數的虛部。
polar()	將直角座標形式轉換為複數的極座標。
phase()	傳回複數的相位。
exp()	傳回 e**x。
cos()	傳回複數的餘弦。
sin()	傳回複數的正弦。
log()	傳回複數的自然對數。
log10()	傳回以 10 為底的複數自然對數。
sqrt()	傳回複數的平方根。

「時間」功能表

此子功能表位於**更多模組**下。

項目	說明
from time import *	從 time 模組匯入所有方法。
sleep(seconds)	在指定的秒數期間暫停執行程式。
clock()	傳回目前的處理器時間用作浮點數(以秒數表示)。
localtime()	將自 2000 年 1 月 1 日起以秒數表示的時間轉換成包含年、月、月日、小時、分、秒、平日、年日、以及日光節約時間 (DST) 旗標的九個元組。 如果沒有提供選用(秒)引數，則會使用即時時鐘。
ticks_cpu()	傳回處理器特有的增加毫秒計數器以及任意參考點。 如需持續測量不同系統之間的時間，請使用 ticks_ms()。
ticks_diff()	測量連續呼叫 ticks_cpu() 或 ticks_ms() 之間的期間。 此函數不應用於測量任意的長期時間。

「TI 系統」功能表

此子功能表位於**更多模組**下。

注意: 在建立使用此模組的新程式時，建議使用**資料分享**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from ti_system import *	從 ti_system 模組匯入所有方法(函數)。
recall_value("name")	調用名為「name」的預定義 OS 變數(值)。
store_value("name",value)	將 Python 變數(值)儲存至名為「name」的 OS 變數。
recall_list("name")	調用名為「name」的預定義 OS 清單。
store_list("name",list)	將 Python 列表(列表)儲存至名為「name」的 OS 列表變數。
eval_function("name",value)	以指定值計算預定義的 OS 函數。
get_platform()	在計算機傳回「hh」，在桌面傳回「dt」。
get_key()	傳回代表按鍵已按下的字串。 「1」鍵會傳回「1」，「esc」鍵會傳回「esc」等。 呼叫時沒有包含任何參數 - get_key() - 會立即傳回。 呼叫時包含參數 - get_key(1) - 會等到按下按鍵後再傳回。
get_mouse()	傳回滑鼠座標作為兩個元素元組， 如果在畫布外部，則傳回畫布像素位置或是 (-1,-1)。
while get_key() != "esc":	在「while」迴圈中執行指令，直到按下「esc」鍵為止。
clear_history()	清除 Shell 歷史記錄。
get_time_ms()	傳回以毫秒為精度的毫秒時間。 此功能可以用於計算持續期間，而非確定實際的時鐘時間。

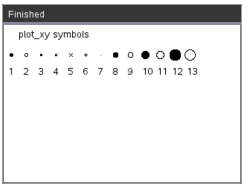
「TI Draw」功能表

此子功能表位於**更多模組**下。

注意: 在建立使用此模組的新程式時，建議使用**幾何作圖**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from ti_draw import *	從 ti_draw 模組匯入所有方法。

形狀

項目	說明
draw_line()	繪製一條從指定的 x1,y1 座標開始，延伸至 x2,y2 的直線。
draw_rect()	繪製一個從指定的 x,y 座標開始，已指定寬度與高度的矩形。
fill_rect()	繪製一個從指定的 x,y 座標開始，已指定寬度與高度，並且填滿指定色彩(如果未定義則使用 set_color 或黑色)的矩形。
draw_circle()	繪製一個從指定的 x,y 中心座標開始，已指定半徑的圓形。
fill_circle()	繪製一個從指定的 x,y 中心座標開始，已指定半徑，並且填滿指定色彩(如果未定義則使用 set_color 或黑色)的圓形。
draw_text()	繪製一個從指定的 x,y 座標開始的文字字串。
draw_arc()	繪製一個從指定的 x,y 座標開始，已指定寬度、高度與角度的弧形。
fill_arc()	繪製一個從指定的 x,y 座標開始，已指定寬度、高度與角度，並且填滿指定色彩(如果未定義則使用 set_color 或黑色)的弧形。
draw_poly()	繪製一個使用指定 x-list,y-list 值的多邊形。
fill_poly()	繪製一個使用指定 x-list,y-list 值，並且填滿指定色彩(如果未定義則使用 set_color 或黑色)的多邊形。
plot_xy()	繪製一個使用指定 x,y 座標，以及代表不同形狀與符號(範圍在 1-13 內)的指定數字的形狀(請見下方)。 

控制

項目	說明
clear()	清除整個畫面。可以與 x,y,width,height 參數搭配使用以清除現有矩形。
clear_rect()	清除從指定的 x,y 座標開始，已指定寬度與高度的矩形。
set_color()	依照程式內容設定形狀顏色，直到設定其他顏色為止。
set_pen()	繪製形狀時設定指定的邊界粗細與樣式(使用填滿指令時不適用)。
set_window()	設定視窗大小，可在視窗中繪製任何形狀。 此函數可用於調整視窗大小，以符合資料或是變更繪圖畫布的原點 (0,0)。
get_screen_dim()	傳回視窗畫面範圍的 xmax 和 ymax。
use_buffer()	啟用螢幕外緩衝區以加快繪圖速度。
paint_buffer()	顯示緩衝的繪圖輸出。 use_buffer() 和 paint_buffer() 函數可用於在畫面中顯示多個物件時可能會造成延遲的情況。

筆記

- 畫面左上角的預設配置為 (0,0)。右邊的正向 x 軸點以及下方的正向 y 軸點。可使用 set_window() 函數修改此內容。
- ti_draw 模組中的函數僅可用於計算機以及桌面的計算機檢視。

「TI 圖片」功能表

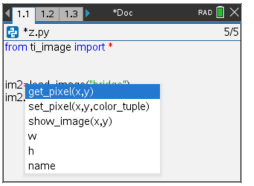
此子功能表位於**更多模組**下。

注意: 在建立使用此模組的新程式時，建議使用**圖片處理**程式類型。如此將可確保所有相關模組皆已匯入。

項目	說明
from ti_image import *	從 ti_image 模組匯入所有方法。
new_image(width,height,(r,g,b))	建立用於 Python 程式，且包含指定寬度與高度的新圖片。 新圖片的顏色由 (r,g,b) 值定義。
load_image("name")	載入用於 Python 程式，由「名稱」指定的圖片。 圖片必須是「筆記」或「函數繪圖」應用程式中的 TNS 文件的一部分。 「name」提示將會顯示圖片名稱 (如果先前已經將其命名) 或是指示其插入順序的數字。
copy_image(image)	建立由「image」變數指定的圖片副本。

圖片物件的方法

如需有關圖片物件的其他函數，可以輸入變數名稱，後面加上「.」(點號)，即可在編輯器以及 Shell 中使用。



- **get_pixel(x,y):** 在 (x,y) 平面座標值所定義的位置取得像素 (r,g,b) 值。

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

- **set_pixel(x,y,color_tuple):** 將位置 (x,y) 的像素設定為 color_tuple 中指定的色彩。

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

  將 (100,100) 的像素設定為 (0,0,255) 色。

- **show_image(x,y):** 顯示圖片，其左上角的位置在 (x,y)。
- **w, h, name:** 設定圖片的寬度、高度以及名稱參數。

範例

```
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)
```

「變數」功能表

注意: 這些目錄並未包括任何其他 TI-Nspire™ 應用程式中所定義的變數。

項目	說明
變數: 目前程式	(限編輯器) 顯示目前程式中定義的全域函數與變數清單
變數:上次執行的程式	(限 Shell) 顯示上次執行程式中定義的全域函數與變數目錄
變數:所有	(限 Shell) 顯示上次執行程式以及任何匯入模組中的全域函數與變數目錄

附錄

Python 關鍵字

下列關鍵字建立在 TI-Nspire™ Python 實作中。

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 鍵盤對應

在編輯器或是 Shell 中輸入編碼時，小鍵盤的作用是貼上適當的 Python 運算或是開啟功能表，以便輕鬆輸入函數、關鍵字、方法、運算元等。

鍵	對應
var	開啟 [變數] 功能表
sto→	貼上「=」符號
del	刪除游標左側的字元
clear	無動作
=	貼上「=」符號
|≠≥>	貼上選取的符號: • > • < • != • >= • <= • == • 與 • 或 • 非 • | • & • ~
trig	貼上選取的函數: • sin • cos • tan • atan2 • asin • acos • atan
?	顯示提示
:=	貼上「:=」
^	貼上「**」
√	無動作
x²	貼上「**2」

鍵	對應
√	貼上「sqrt()」
×	貼上乘號 (*)
"	貼上一個雙引號 (")
÷	貼上除號 (/)
a/b	無動作
e^x	貼上「exp()」
ln	貼上「log()」
10^x	貼上「10**」
log	貼上「log(value,base)」
(	貼上「(」
)	貼上「)」
[]	貼上「[ ]」
{}	貼上「{ }」
−	貼上減號 (-)
≈	在目前行的後面新增一行
EE	貼上「E」
?!	貼上選取的符號: • ? • ! • $ • ° • ' • % • " • : • ; • _ • \ • #
π	貼上「pi」
⇦	現有旗標行為
↵	在目前行的後面新增一行

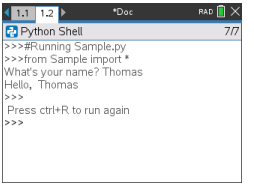
Python 程式範例

利用下列範例程式以便進一步熟悉 Python 方法。您也可以在**範例**資料夾的 **Python.tns 新手入門**檔案中找到這些範例程式。

注意: 如果您複製了包含標籤縮排指標 (••) 的任何範例編碼，並將其貼至 TI-Nspire™ 軟體，則您需要以實際的標籤縮排取代這些實例。

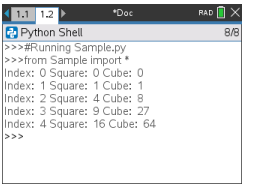
您好

```
# 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")
```



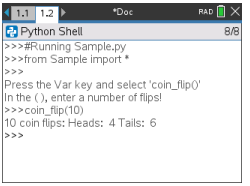
迴圈範例

```
# 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)
```



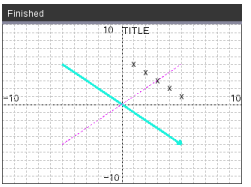
正面還是反面

```
# 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!")
```



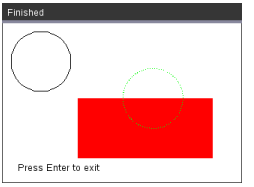
繪製

```
# 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")
```



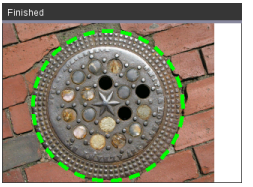
繪畫

```
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")
```



圖片

```
# 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)
```



套裝

此程式使用 Python 來控制可程式化的微控制器 TI-Innovator™ 套裝 (Hub)。在沒有附加 TI-Innovator™ 套裝 (Hub) 的情況下執行程式將會出現錯誤訊息。

如需有關 TI-Innovator™ 套裝 (Hub) 的更多資訊，請參訪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()
```

一般資訊

線上說明

education.ti.com/eguide

選擇您的國家/地區以取得更多產品資訊。

連絡 TI 技術支援部門

education.ti.com/ti-cares

選擇您的國家/地區以取得技術和其他支援資源。

服務與保固資訊

education.ti.com/warranty

選擇您的國家/地區，即可瞭解保固期間與條款或產品服務的相關資訊。

這保證不會影響您的法定權利。

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243