

NEXIOT

nexvic

VIC7000

Python Script
Function Introduction

v 1.2.0.3



黃翊凱

Garry Huang

應用工程師

- 開啟 Python 腳本

開啟設定視窗，在腳本欄位內選擇 Python，就可以將腳本切換成 Python 模式

在右邊的欄位選擇 Python 輸出，可以看到 Python 腳本執行的輸出

The image shows two screenshots of the nexVIC software interface. The left screenshot displays the 'System Settings' dialog box. In the 'Script' section, the 'Script' dropdown is set to 'Python', and the 'Script Watchdog Timeout(ms)' dropdown is set to 'Built-In'. A red arrow points to the 'Script' dropdown. The right screenshot shows the main nexVIC interface. The 'Script' button in the top toolbar is highlighted. The script editor displays a Python script:

```
1 from vic import *  
2  
3  
4 def mainLoop(info, arr):  
5     pass  
6  
7
```

 The 'Python Output' tab in the right-hand panel is selected, and a red box highlights the output area. A red arrow points to the 'Python Output' tab.

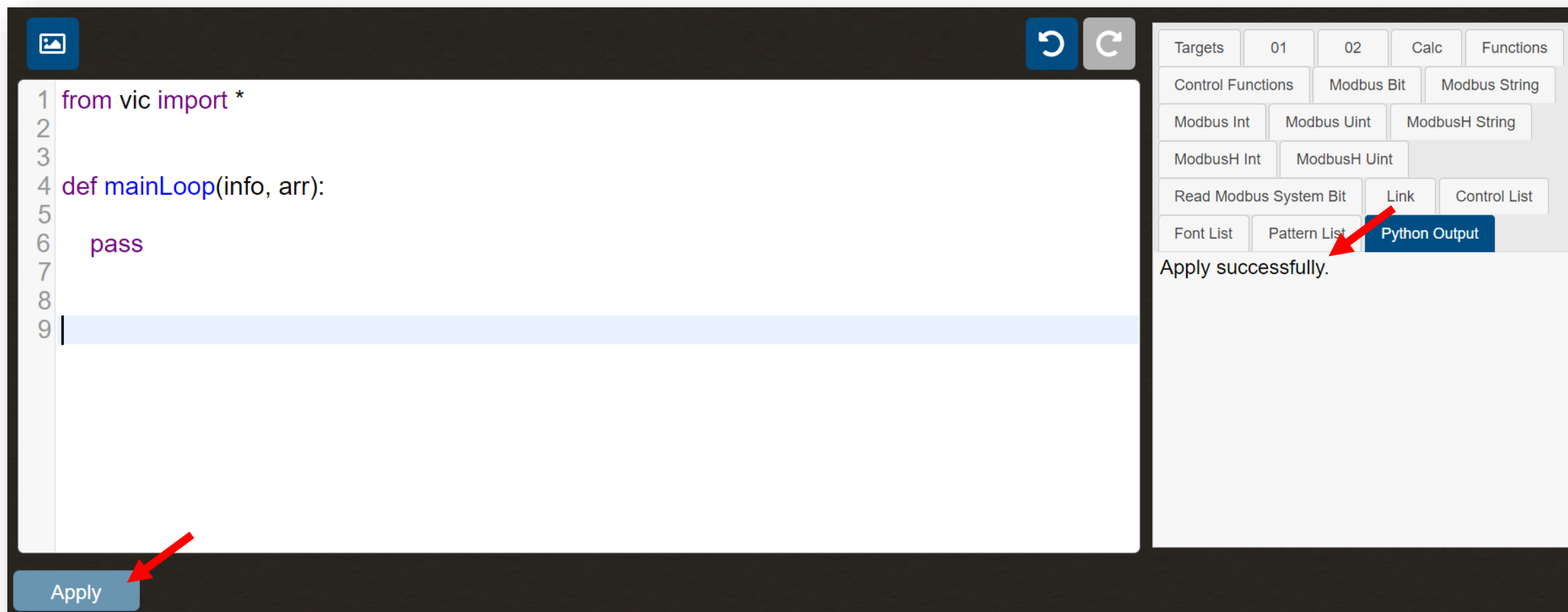
- **mainLoop**

每辨識一張圖，就會執行一次 mainLoop 函式

```
1 from vic import *  
2  
3  
4 def mainLoop(info, arr):  
5     pass  
6  
7
```

- 儲存變更

編輯完腳本後，要點擊左下方的儲存變更，腳本才會生效，快捷鍵為 ctrl + s



- 引用辨識項目

在右邊的欄位中，選擇目標值和其他頁面編號，可以引用每一頁設定的辨識項目

這裡列出的辨識項目和專案內設定的辨識項目相同

The screenshot displays the Nexvic software interface, specifically the 'Targets' and 'Settings' panels. The 'Targets' panel on the left lists various recognition items: TARGET01, TARGET02, TARGET_COLOR01, TARGET_COLOR02, TARGET_PATTERN01, and TARGET_PATTERN02. The 'Settings' panel on the right shows the configuration for these items, including the 'Input Parameters' section. Red and orange boxes highlight specific elements in the interface.

Targets Panel:

- Buttons: Targets, 01, 02, Calc, Functions
- Control Functions: Modbus Bit, Modbus String
- Modbus Int, Modbus Uint, ModbusH String
- ModbusH Int, ModbusH Uint, Read Modbus System Bit
- Link, Control List, Font List, Pattern List
- Python Output

Settings Panel:

- Buttons: Add Page, 01, 02
- Recognition Method: Matching target
- Settings: Name, Value
- Input Parameters: Name, Value

Over View Panel:

- Buttons: OCR01, OCR02, COLOR01, COLOR02, PATTERN01, PATTERN02

- 辨識項目

- TARGET、OCR

此變數為當前圖片的文字辨識結果

變數形式為 TA.TARGETxx 和 PAGExx.OCRxx，xx為頁面與辨識項目編號

TA.TARGET01
PAGE01.OCR01

- 辨識項目

- TARGET_COLOR、COLOR

此變數為當前圖片的顏色比對結果

變數形式為 TA.TARGET_COLORxx 和 PAGExx.COLORxx

xx為頁面與辨識項目編號，還可細分為

VALUE：顏色比對結果輸出

MATCH：是否比對成功

RED：當前圖片上顏色比對區域的 RGB 色域紅色值

GREEN：當前圖片上顏色比對區域的RGB 色域綠色值

BLUE：當前圖片上顏色比對區域的RGB 色域藍色值

```
TA.TARGET_COLOR01
TA.TARGET_COLOR01.VALUE
TA.TARGET_COLOR01.MATCH
TA.TARGET_COLOR01.RED
TA.TARGET_COLOR01.GREEN
TA.TARGET_COLOR01.BLUE
PAGE01.COLOR01
PAGE01.COLOR01.VALUE
PAGE01.COLOR01.MATCH
PAGE01.COLOR01.RED
PAGE01.COLOR01.GREEN
PAGE01.COLOR01.BLUE
```

- 辨識項目

- TARGET_PATTERN、PATTERN

此變數為當前圖片的顏色比對結果

變數形式為 TA.TARGET_PATTERNxx 和 PAGExx.PATTERNxx

xx為頁面與辨識項目編號，還可細分為

VALUE：樣式比對結果輸出

FOUND：是否有找到樣式

SCORE：樣式比對成功時的比對分數

X：當前圖片上找到樣式的 X 座標

Y：當前圖片上找到樣式的 Y 座標

WIDTH：當前圖片上找到樣式的寬度

HEIGHT：當前圖片上找到樣式的高度

BASE_X：設定比對樣式的圖片上的樣式 X 座標

BASE_Y：設定比對樣式的圖片上的樣式 Y 座標

```
TA.TARGET_PATTERN01
TA.TARGET_PATTERN01.VALUE
TA.TARGET_PATTERN01.FOUND
TA.TARGET_PATTERN01.SCORE
TA.TARGET_PATTERN01.X
TA.TARGET_PATTERN01.Y
TA.TARGET_PATTERN01.WIDTH
TA.TARGET_PATTERN01.HEIGHT
TA.TARGET_PATTERN01.BASE_X
TA.TARGET_PATTERN01.BASE_Y
PAGE01.PATTERN01
PAGE01.PATTERN01.VALUE
PAGE01.PATTERN01.FOUND
PAGE01.PATTERN01.SCORE
PAGE01.PATTERN01.X
PAGE01.PATTERN01.Y
PAGE01.PATTERN01.WIDTH
PAGE01.PATTERN01.HEIGHT
PAGE01.PATTERN01.BASE_X
PAGE01.PATTERN01.BASE_Y
```


- 運算

- **CALC**：為變數，可用於儲存腳本執行結果，運算值會被儲存進資料庫，且可透過外部連線傳輸

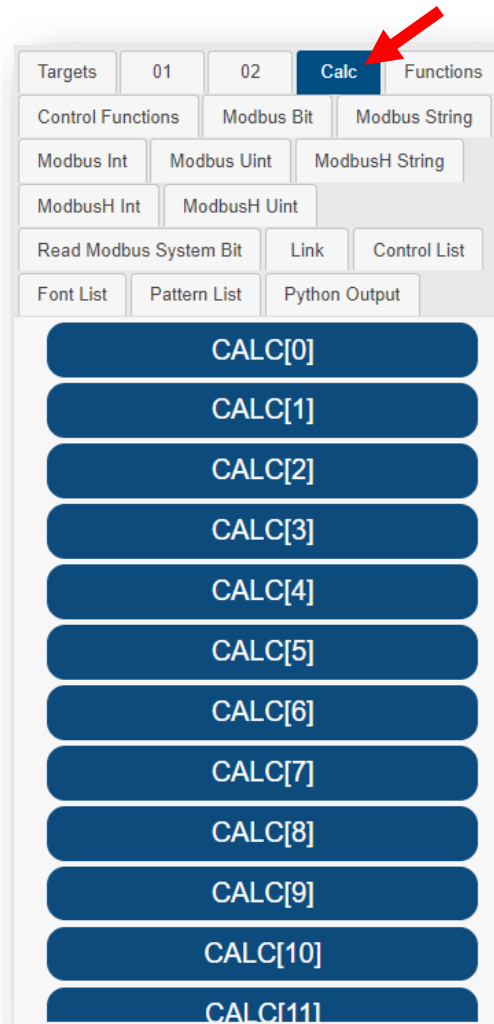
使用語法為CALC[x]，x 為 0 ~ 99

CALC[0]

CALC[1]

CALC[98]

CALC[99]



- **Modbus 資料欄位**

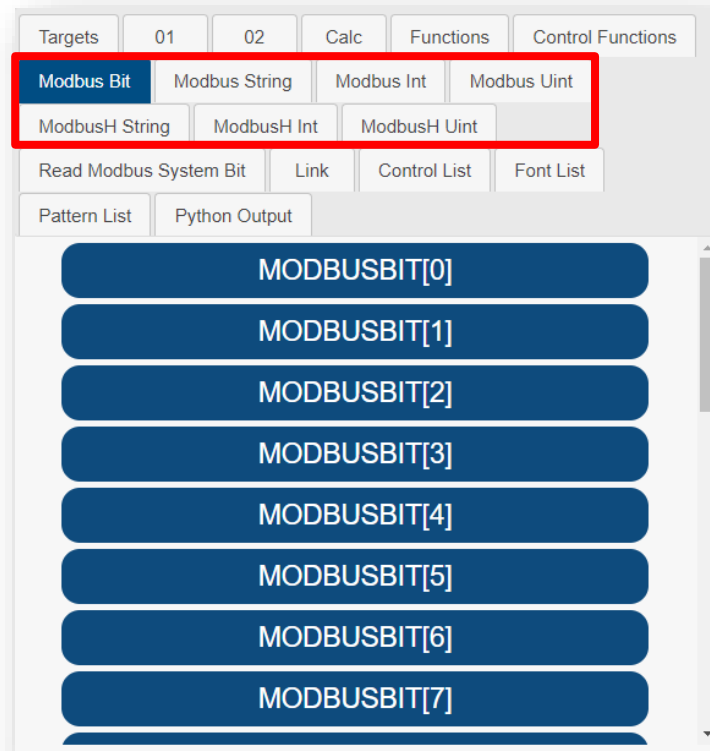
- MODBUSBIT、MODBUSSTR、MODBUSINT、MODBUSUINT、MODBUSHSTR、MODBUSHINT、MODBUSHUINT：為變數，可自定義要存放的位元、字串、有號整數、無號整數在 MODBUS 欄位中

使用語法為 MODBUSBIT[x]、MODBUSSTR[x]、MODBUSINT[x]、MODBUSUINT[x]、

MODBUSHSTR[y]、MODBUSHINT[z]、MODBUSHUINT[z]

其中 x 為 0 ~ 99，y 為 0 ~ 199，z 為 0 ~ 499

```
MODBUSBIT[0]
MODBUSSTR[0]
MODBUSINT[0]
MODBUSUINT[0]
MODBUSHSTR[0]
MODBUSHINT[0]
MODBUSHUINT[0]
```



- **Modbus 系統資訊欄位**

- **MODBUSSYSBIT**：存放代表系統資訊的欄位，使用語法為 MODBUSSYSBIT[x]

x = 0：程式是否運行中，1 = 運行中，0 = 停止

x = 1：是否有使用者登入，1 = 有人登入，0 = 無人登入

x = 2：系統硬碟容量 5G 檢測，1 = 低於 5G，0 = 高於 5G

x = 3：系統硬碟容量 10G 檢測，1 = 低於 10G，0 = 高於 10G

x = 4：是否停止存圖，1 = 停止，0 = 運行中

x = 7：控制是否運行中，1 = 運行中，0 = 停止

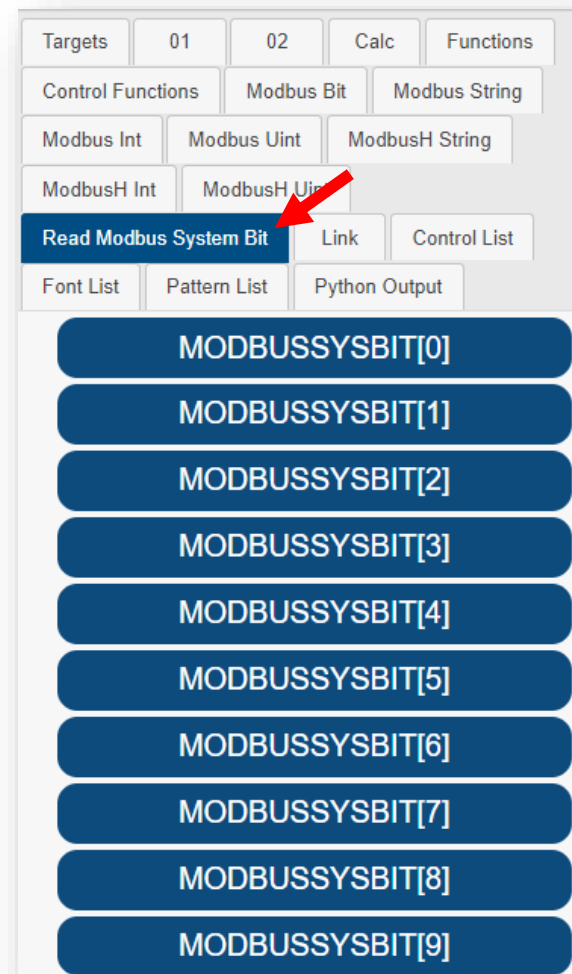
x = 8：序列埠控制是否被啟用，1 = 取用，0 = 停用

x = 9：序列埠目前狀態，1 = 連線，0 = 斷線

x = 16：資料硬碟容量 10G 檢測，1 = 低於 5G，0 = 高於 10G

x = 17：資料硬碟容量 20G 檢測，1 = 低於 10G，0 = 高於 20G

x = 18：錄製硬碟容量 1G 檢測，1 = 低於 1G，0 = 高於 1G



- **Modbus 系統資訊欄位**

- **MODBUSSYSBIT**：存放代表系統資訊的欄位，使用語法為 MODBUSSYSBIT[x]

x = 5、6：OCR辨識結果，0 = 辨識成功，1 = NG，2 = 比對失敗

x = 19：是否有使用者登入，1 = 有人登入，0 = 無人登入

x = 20：額外儲存路徑容量 1G 檢測，1 = 低於 1G，0 = 高於 1G

x = 21：目前錄製狀態，1 = 錄製中，0 = 停止錄製

x = 6	x = 5	Sum	Result
0	0	0	辨識成功
0	1	1	辨識 NG
1	0	2	比對失敗

- 函式

在右邊的欄位選擇函式，這裡會列出 VIC 的功能函式及變數，可直接點擊要引用的函式或變數

PYTHON_MAIN_LOOP：在腳本區域輸入腳本基本內容

SEND.EMAIL：透過 SMTP 發送電子郵件

SEND.LINE：發送 LINE 通知

SEND.WECHAT：發送 WeChat 通知

SEND.WECHA_P：發送 WeChat 通知

LOG：記錄訊息在系統日誌中

RT.PAGE_NO：當前辨識頁面的編號

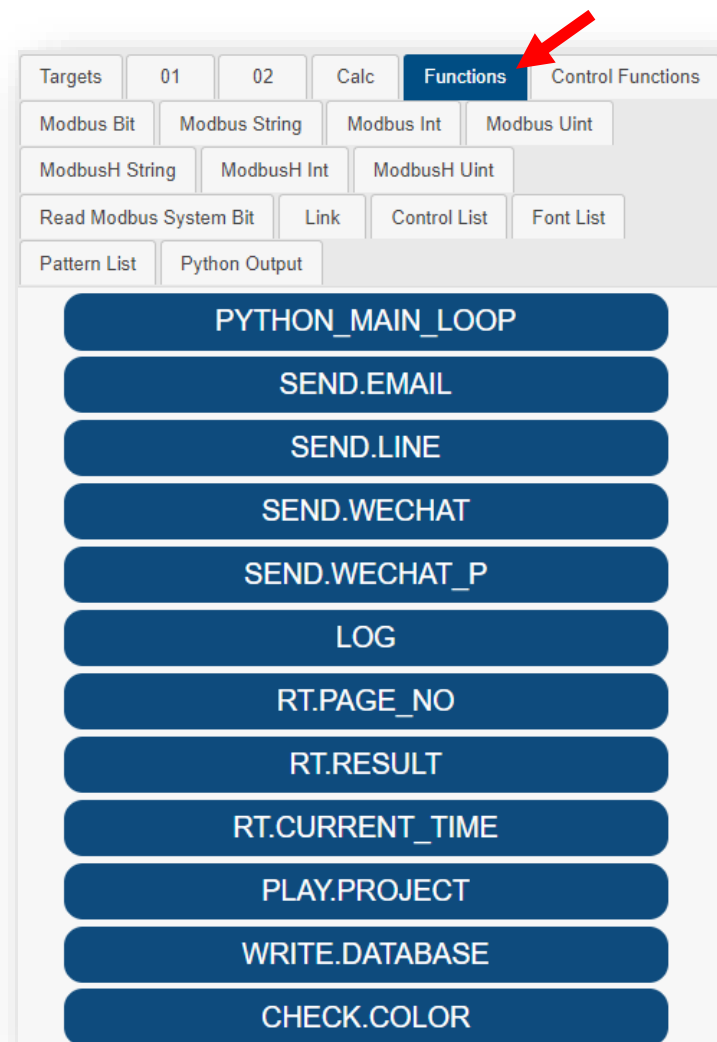
RT.RESULT：當前辨識結果

RT.CURRENT_TIME：執行函式的當前時間

PLAY.PROJECT：設定專案啟動/停止

WRITE.DATABASE：設定辨識資料是否寫入資料庫

CHECK.COLOR：偵測顏色是否閃爍



- 函式

在右邊的欄位選擇函式，這裡會列出 VIC 的功能函式及變數，可直接點擊要引用的函式或變數

MODBUS_MASTER_READ_BIT：從 Modbus slave/server 讀取位元資料

MODBUS_MASTER_WRITE_BIT：寫入位元資料到 Modbus slave/server

MODBUS_MASTER_READ_STR：從 Modbus slave/server 讀取字串資料

MODBUS_MASTER_WRITE_STR：寫入字串資料到 Modbus slave/server

MODBUS_MASTER_READ_UINT：從 Modbus slave/server 讀取整數資料

MODBUS_MASTER_WRITE_UINT：寫入整數資料到 Modbus slave/server

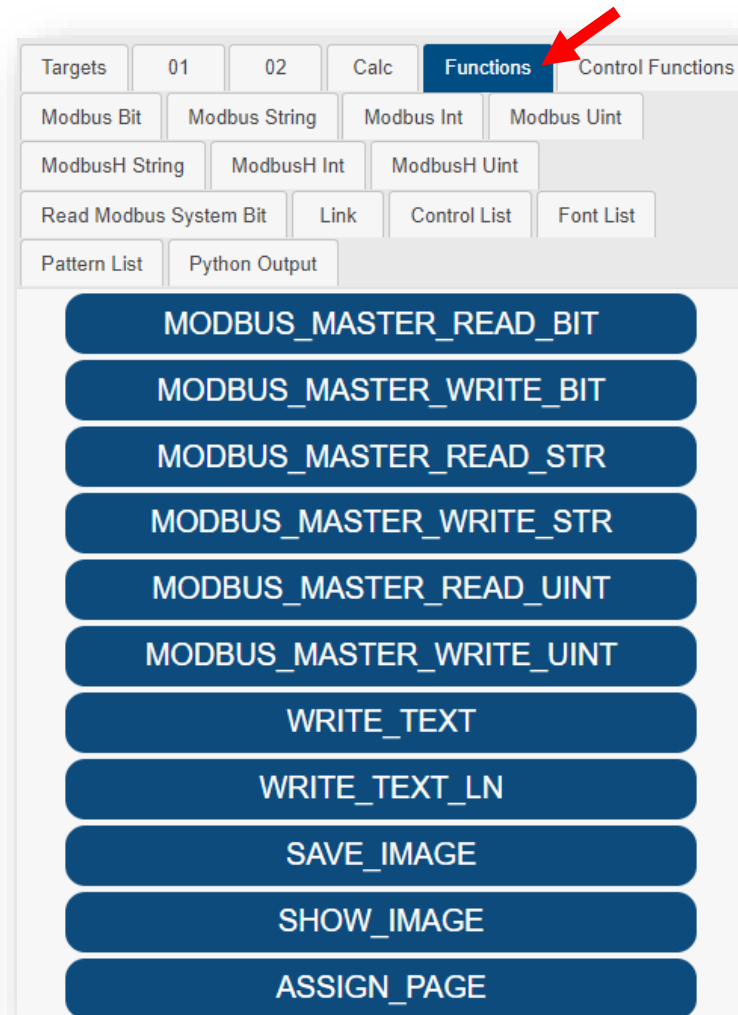
WRITE_TEXT：寫入文字到本地文字檔

WRITE_TEXT_LN：寫入文字到本地文字檔，寫入後換行

SAVE_IMAGE：儲存當前辨識的圖片到本地

SHOW_IMAGE：在腳本圖像中顯示當前辨識的圖片

ASSIGN_PAGE：指定當前辨識的頁面



- 函式

在右邊的欄位選擇函式，這裡會列出 VIC 的功能函式及變數，可直接點擊要引用的函式或變數

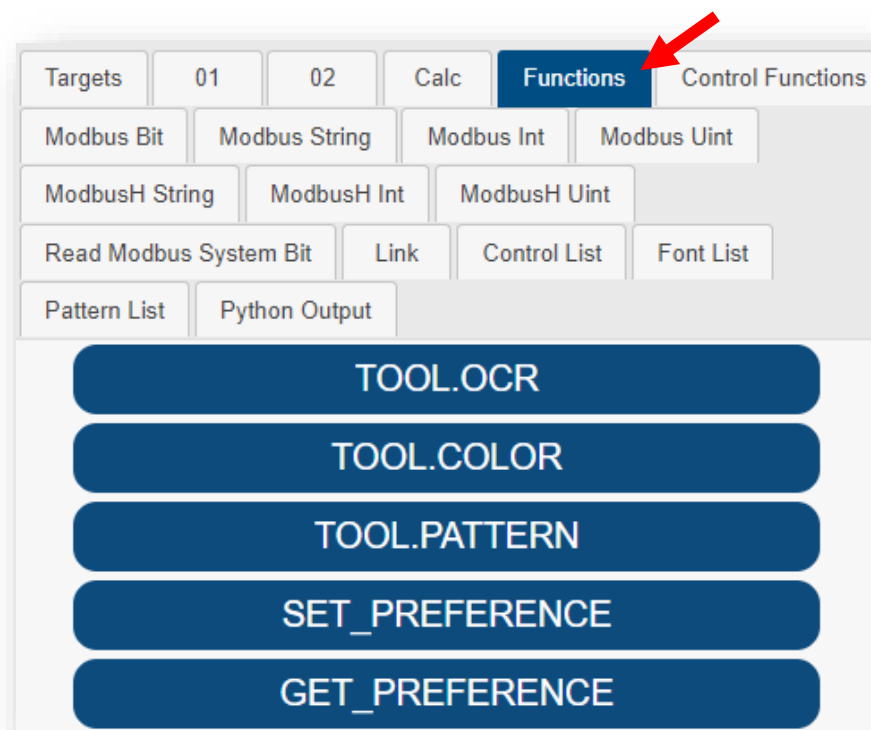
TOOL.OCR：在當前辨識的圖像上進行文字辨識

TOOL.COLOR：在當前辨識的圖像上進行顏色比對

TOOL.PATTERN：在當前辨識的圖像上進行樣式比對

SET_PREFERENCE：設定系統變數的值

GET_PREFERENCE：取得系統變數的值



- 控制函式

在右邊的欄位選擇控制函式，這裡會列出 VIC 控制函式，可直接點擊要引用的控制函式

PLAY.CONTROL：設定控制啟動/停止

MOUSE_MOVE：控制滑鼠游標移動到特定位置

MOUSE_CLICK：控制滑鼠游標移動到特定位置，並點擊一次

MOUSE_DOUBLE_CLICK：控制滑鼠游標移動到特定位置，並連點兩次

MOUSE_DRAG：控制滑鼠游標從一點移動另一點，同時進行拖曳

MOUSE_OCR_CLICK：控制滑鼠在符合比對條件的文字的位置點擊一次

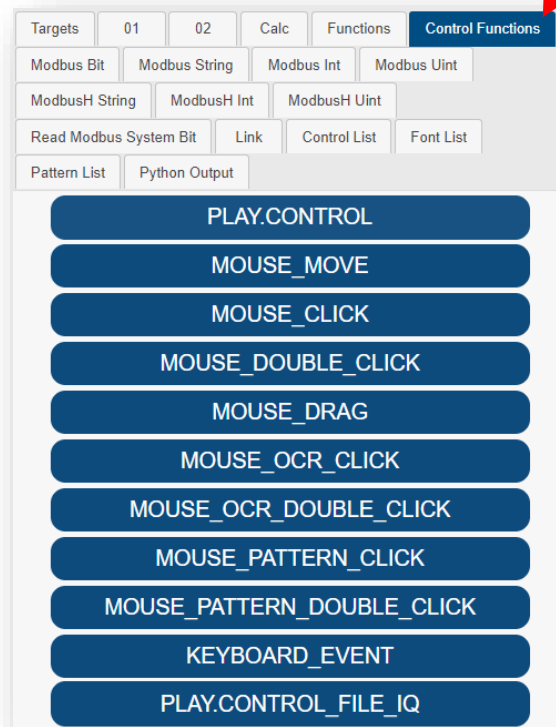
MOUSE_OCR_DOUBLE_CLICK：控制滑鼠在符合比對條件的文字的位置連點兩次

MOUSE_PATTERN_CLICK：控制滑鼠在符合比對條件的樣式的位置點擊一次

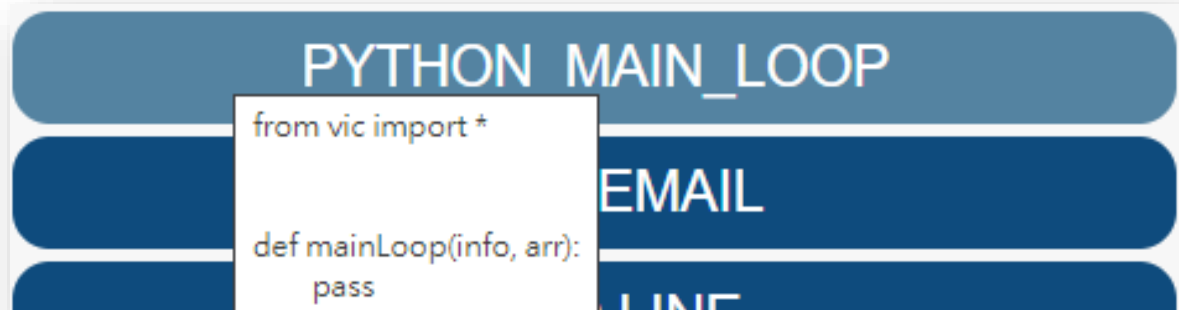
MOUSE_PATTERN_DOUBLE_CLICK：控制滑鼠在符合比對條件的樣式的位置連點兩次

KEYBOARD_EVENT：控制鍵盤按下設定的字串內容

PLAY.CONTROL_FILE_IQ：套用佇列和輸入屬性，執行特定控制檔



- 函式
 - **PYTHON_MAIN_LOOP**：在腳本區域輸入 Python 腳本基本內容
輸入內容包括匯入 vic 函式庫和定義 mainLoop 函式



```
from vic import *
```

```
def mainLoop(info, arr):  
    pass
```

- 函式

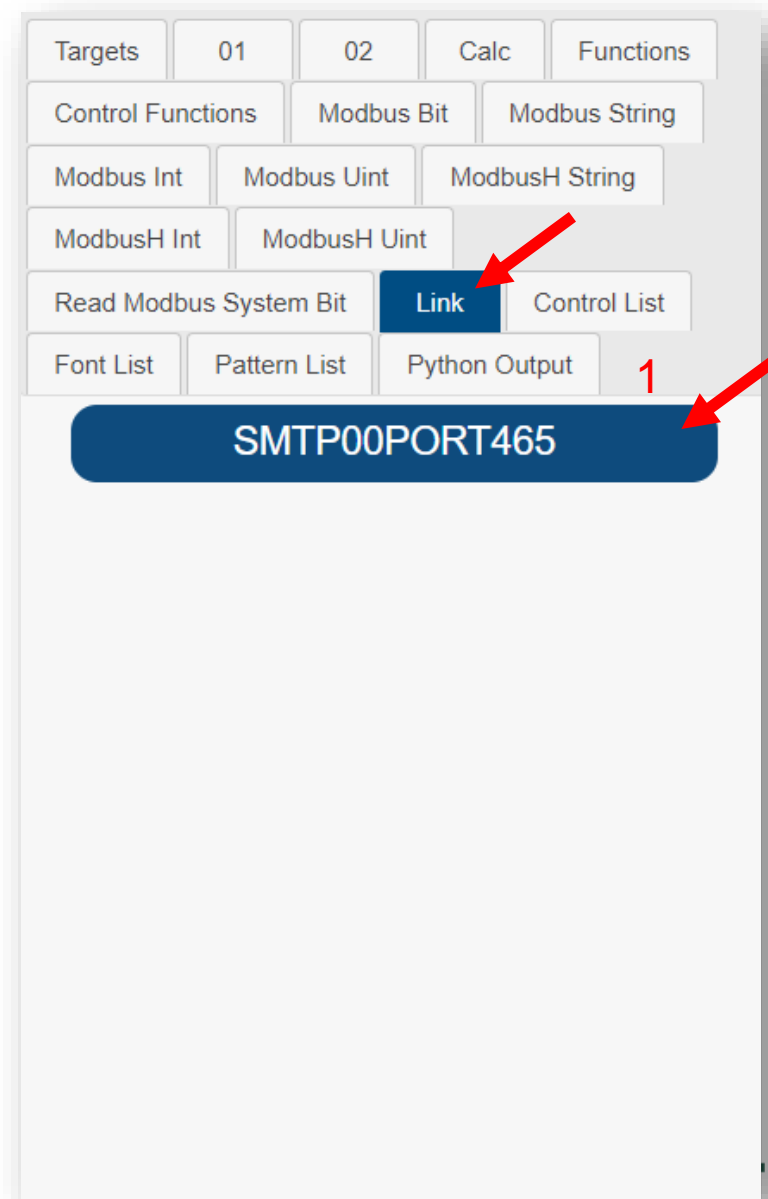
- **SEND.EMAIL**：透過 SMTP 發送電子郵件

1. id：要使用的 SMTP 連線，要先在連結頁面內建立連線可從連結中選擇要使用的連線
2. to：收件人信箱
3. subject：信件主旨
4. body：信件內容
5. send image：是否傳送當下畫面(1：傳送，0：不傳送)
6. sec：執行間隔時間

SEND.EMAIL

`SEND.EMAIL(id, to, subject, body, send image, sec)`

SEND.EMAIL(, , , , ,)



- 函式

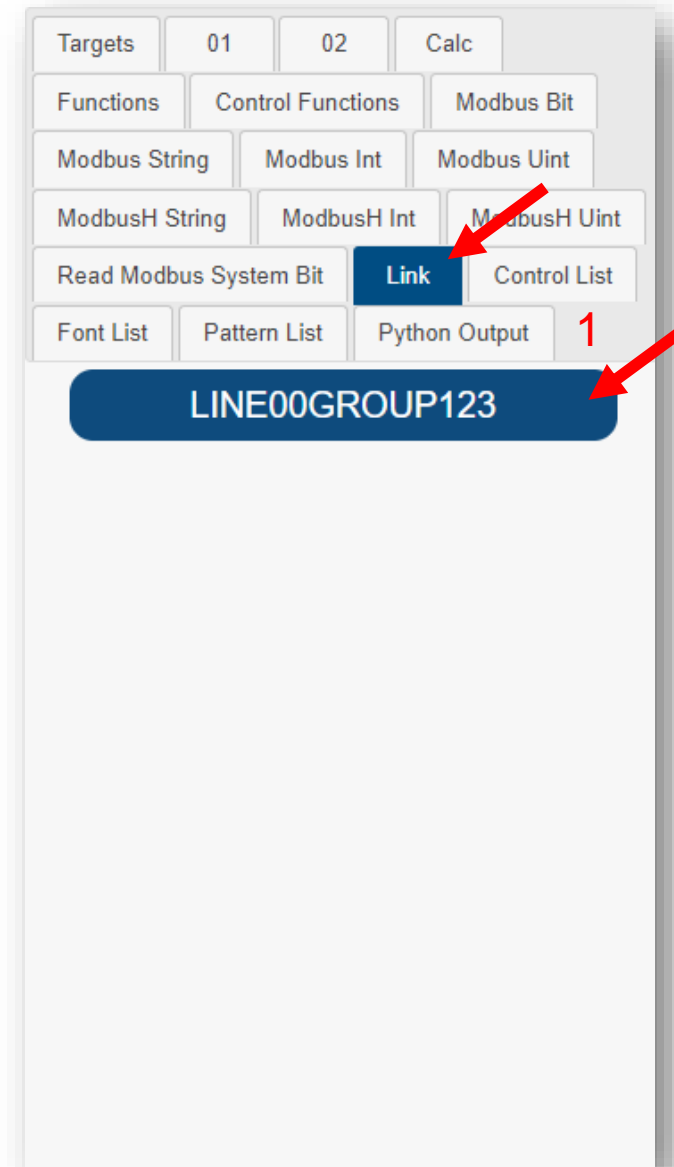
- **SEND.LINE**：發送 LINE 通知

1. id：要使用的 LINE Notify 連線，要先在連結頁面內建立連線
可從連結中選擇要使用的連線
2. message：要傳送的訊息
3. send image：是否傳送當下畫面(1：傳送，0：不傳送)
4. sec：執行間隔時間

SEND.LINE

SEND.LINE(id, message, send image, sec)

SEND.LINE(, , ,)



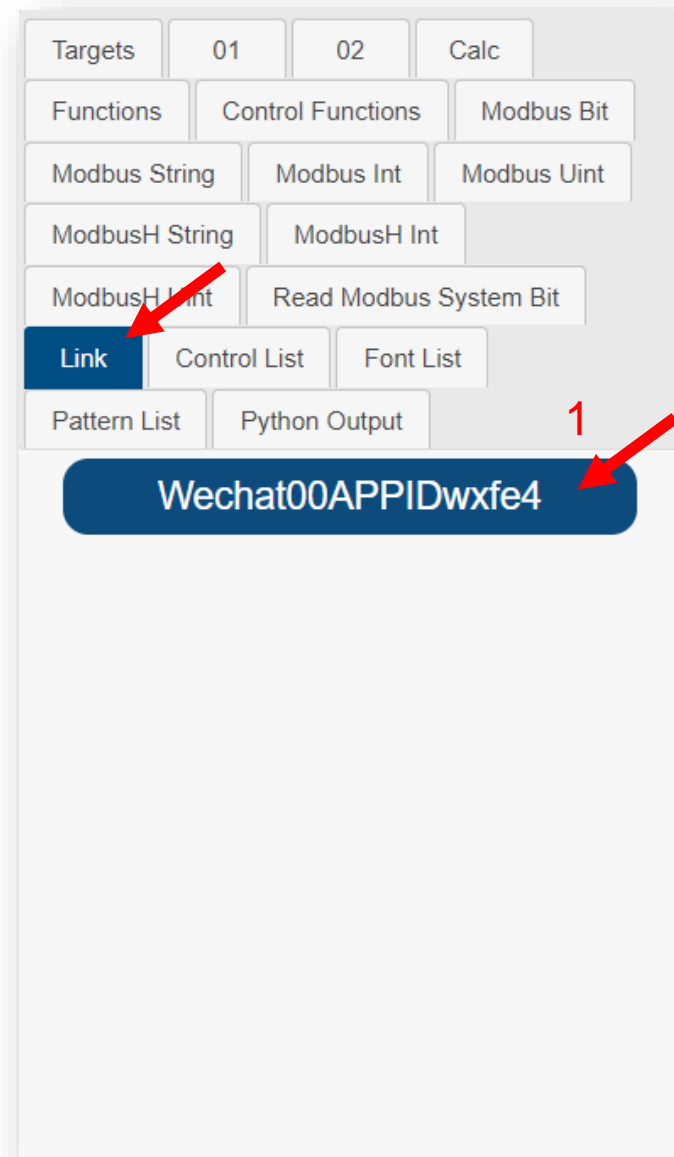
- 函式

- **SEND.WECHAT**：發送 WeChat 通知給所有關注此公眾號的用戶
 1. id：要使用的 WeChat 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
 2. templateID：要傳送的模板的ID
 3. keyword1：要傳送的模板內的{{keyword1.DATA}}
 4. keyword2：要傳送的模板內的{{keyword2.DATA}}
 5. keyword3：要傳送的模板內的{{keyword3.DATA}}
 6. sec：執行間隔時間

SEND.WECHAT

SEND.WECHAT(id, templateID, keyword1, keyword2, keyword3, sec)

SEND.WECHAT(, , , , ,)

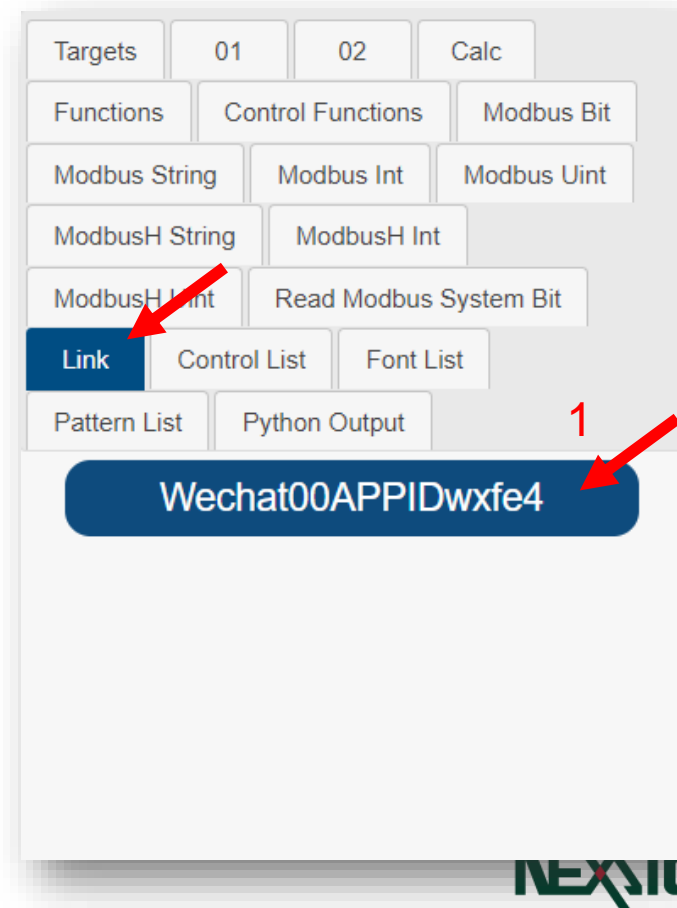


- 函式

- **SEND.WECHAT_P**：發送 WeChat 通知給所有關注此公眾號的一個用戶

1. id：要使用的 WeChat 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
2. templateID：要傳送的模板的ID
3. openID：要接收通知的用戶微信號
4. keyword1：要傳送的模板內的{{keyword1.DATA}}
5. keyword2：要傳送的模板內的{{keyword2.DATA}}
6. keyword3：要傳送的模板內的{{keyword3.DATA}}
7. sec：執行間隔時間

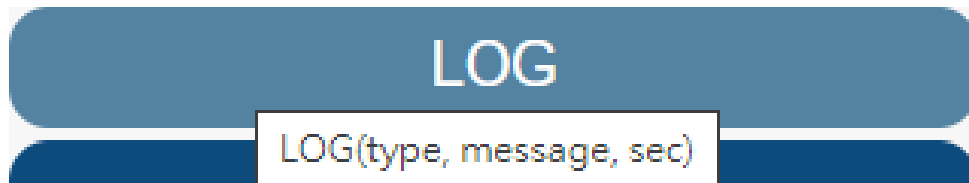
```
SEND.WECHAT_P  
SEND.WECHAT_P(id, templateID, openID, keyword1, keyword2, keyword3, sec)  
SEND.WECHAT_P( , , , , , , )
```



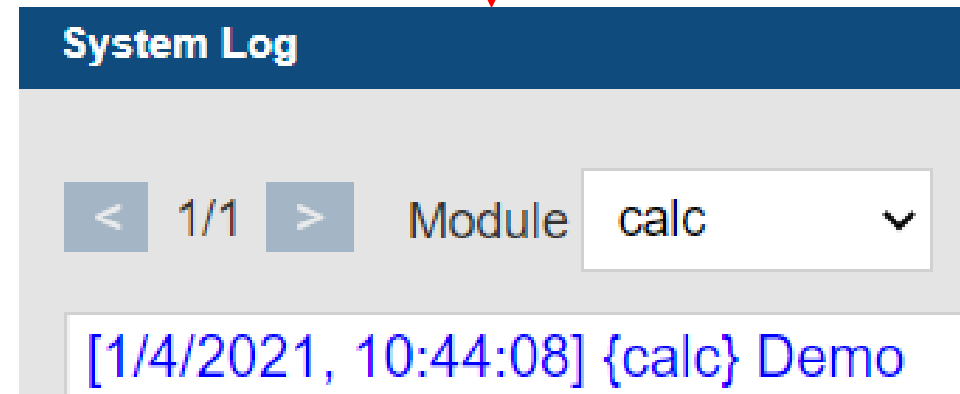
- 函式

- **LOG**：記錄訊息在系統日誌中
 1. type：記錄在系統日誌中的訊息文字顏色，0 = 黑色，1 = 紅色，2 = 藍色
 2. message：要記錄的訊息內容
 3. sec：執行間隔時間

LOG(2, 'Demo', 10)



LOG(, ,)



- 函式
 - **RT.PAGE_NO** : 當前辨識頁面的編號

RT.PAGE_NO

RT.PAGE_NO

RT.PAGE_NO

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    print(RT.PAGE_NO)
```

```
    pass
```

Pattern List

Python Output

Apply successfully.

01

- 函式
 - **RT.RESULT** : 當前辨識結果
 - 0 = Pass
 - 1 = NG
 - 2 = No Matching



RT.RESULT

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    print(RT.RESULT)
```

```
    pass
```

Apply successfully.

0

- 函式

- **RT.CURRENT_TIME**：執行函式的當前時間，可再細分為

RT.CURRENT_TIME.YEAR：執行函式的當前時間的年份

RT.CURRENT_TIME.MONTH：執行函式的當前時間的月份

RT.CURRENT_TIME.DAY：執行函式的當前時間的日期

RT.CURRENT_TIME.HOUR：執行函式的當前時間的時

RT.CURRENT_TIME.MINUTE：執行函式的當前時間的分

RT.CURRENT_TIME.SECOND：執行函式的當前時間的秒



RT.CURRENT_TIME
RT.CURRENT_TIME.YEAR
RT.CURRENT_TIME.MONTH
RT.CURRENT_TIME.DAY
RT.CURRENT_TIME.HOUR
RT.CURRENT_TIME.MINUTE
RT.CURRENT_TIME.SECOND

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    print(RT.CURRENT_TIME)  
    print(RT.CURRENT_TIME.YEAR)  
    print(RT.CURRENT_TIME.MONTH)  
    print(RT.CURRENT_TIME.DAY)  
    print(RT.CURRENT_TIME.HOUR)  
    print(RT.CURRENT_TIME.MINUTE)  
    print(RT.CURRENT_TIME.SECOND)
```

```
    pass
```

Font List	Pattern List	Python Output
Apply successfully.		
2021-01-04 10:47:55		
2021		
1		
4		
10		
47		
55		

- 函式

- **PLAY.PROJECT**：設定專案啟動/停止

1. status：專案執行狀態

0 = 啟動專案

1 = 停止專案

PLAY.PROJECT

PLAY.PROJECT(status)

PLAY.PROJECT()

- 函式

- **WRITE.DATABASE**：設定辨識資料是否寫入資料庫

1. status：資料庫寫入狀態

0 = 不寫入資料庫

1 = 寫入資料庫

WRITE.DATABASE

WRITE.DATABASE(status)

WRITE.DATABASE()

- 函式

- **CHECK.COLOR**：偵測顏色是否閃爍

- 1. page_color：要被偵測的顏色

- 2. calc：存放偵測結果的運算值

若在偵測時間內顏色比對結果有改變則輸出為 1，反之則為 0

- 3. sec：偵測時間

CHECK.COLOR

CHECK.COLOR(page_color, calc, sec)

CHECK.COLOR(, ,)

- 函式

- **WRITE_TEXT**：寫入文字到本地文字檔

1. filepath：儲存絕對路徑或檔案名稱

輸入儲存絕對路徑：文字檔會被儲存在該路徑

輸入檔案名稱：文字檔會被儲存在 VIC7000 資料夾內的 Export

2. content：要寫入文字檔的內容

3. sec：執行間隔時間

WRITE_TEXT

WRITE_TEXT(filepath, content, sec)

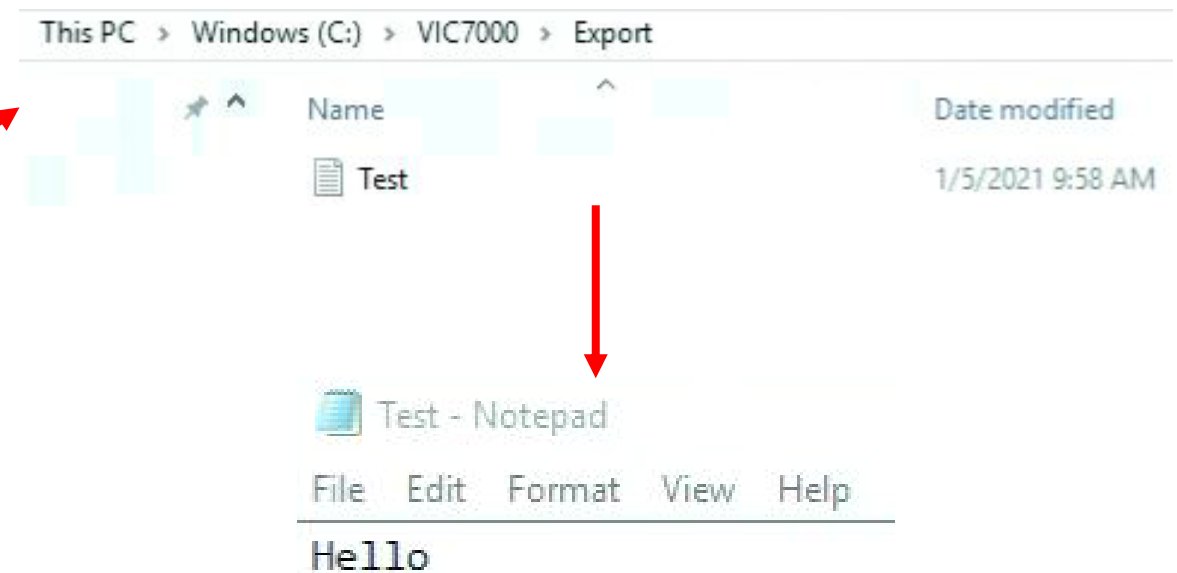
WRITE_TEXT(, ,)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    WRITE_TEXT('Test','Hello', 1)
```

```
    pass
```



- 函式

- **WRITE_TEXT_LN**：寫入文字到本地文字檔，寫入後換行

1. filepath：儲存絕對路徑或檔案名稱

輸入儲存絕對路徑：文字檔會被儲存在該路徑

輸入檔案名稱：文字檔會被儲存在 VIC7000 資料夾內的 Export

2. content：要寫入文字檔的內容

3. sec：執行間隔時間

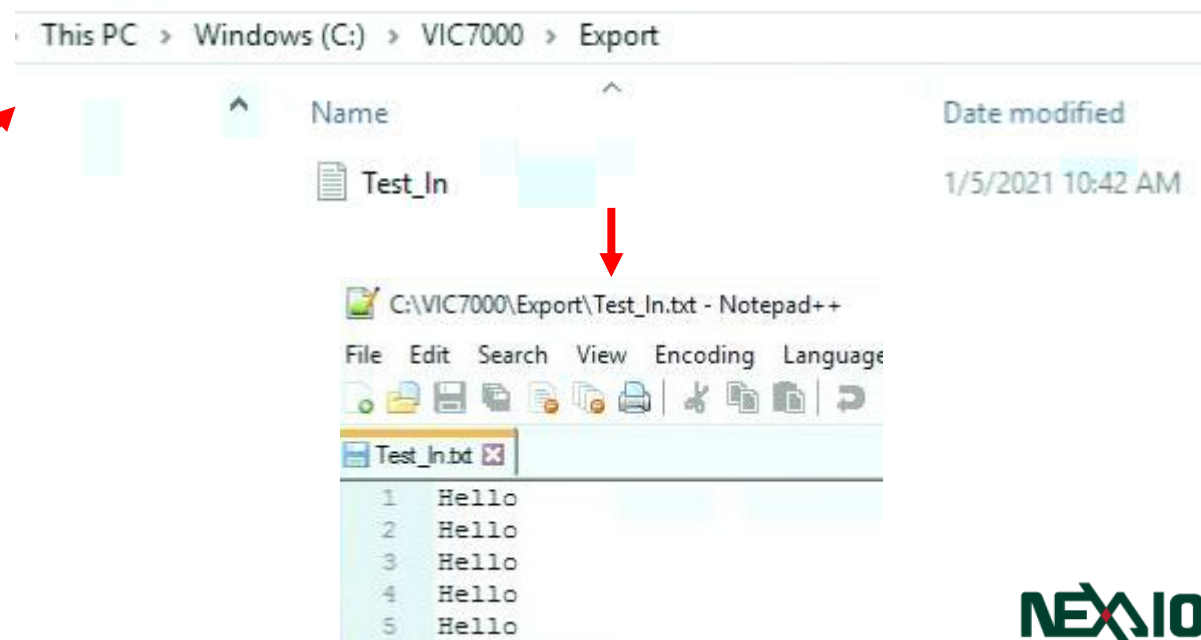
WRITE_TEXT_LN
WRITE_TEXT_LN(filepath, content, sec)
WRITE_TEXT_LN(, ,)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    WRITE_TEXT_LN('Test_In','Hello', 1)
```

```
    pass
```



- 函式

- **SAVE_IMAGE**：儲存當前辨識的圖片到本地

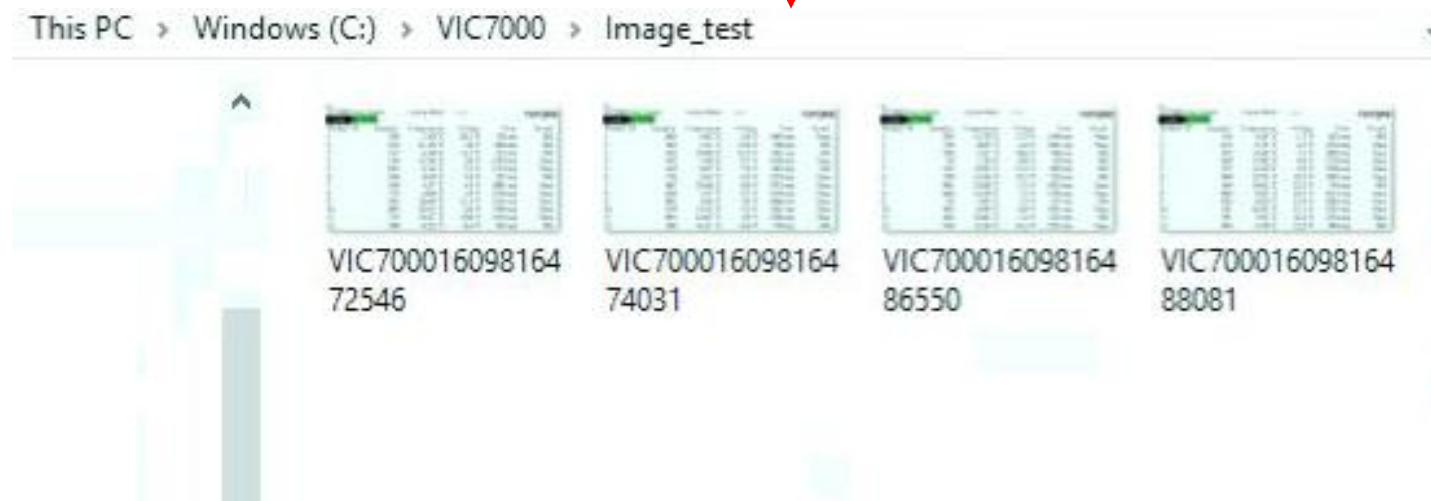
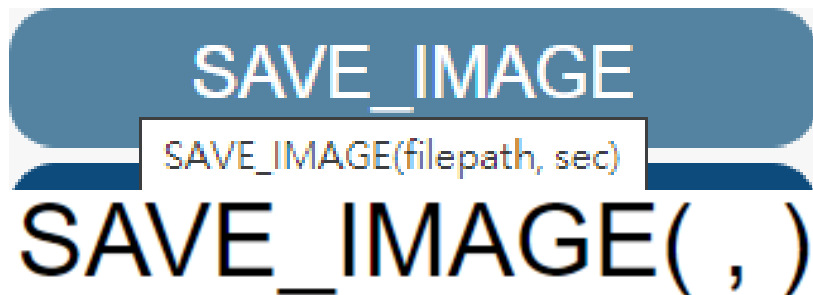
1. filepath：儲存圖片的資料夾路徑
2. sec：執行間隔時間

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    SAVE_IMAGE('C:\\\\VIC7000\\Image_test',1 )
```

```
    pass
```

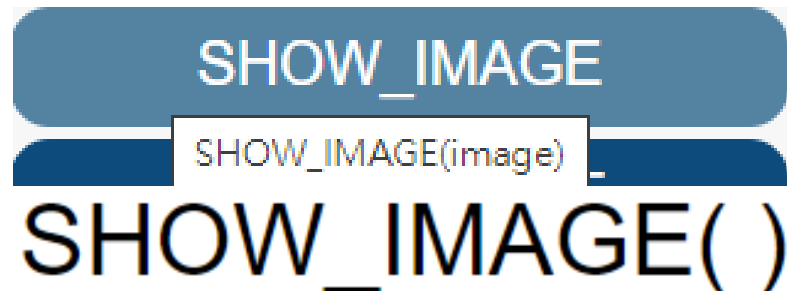


- 函式

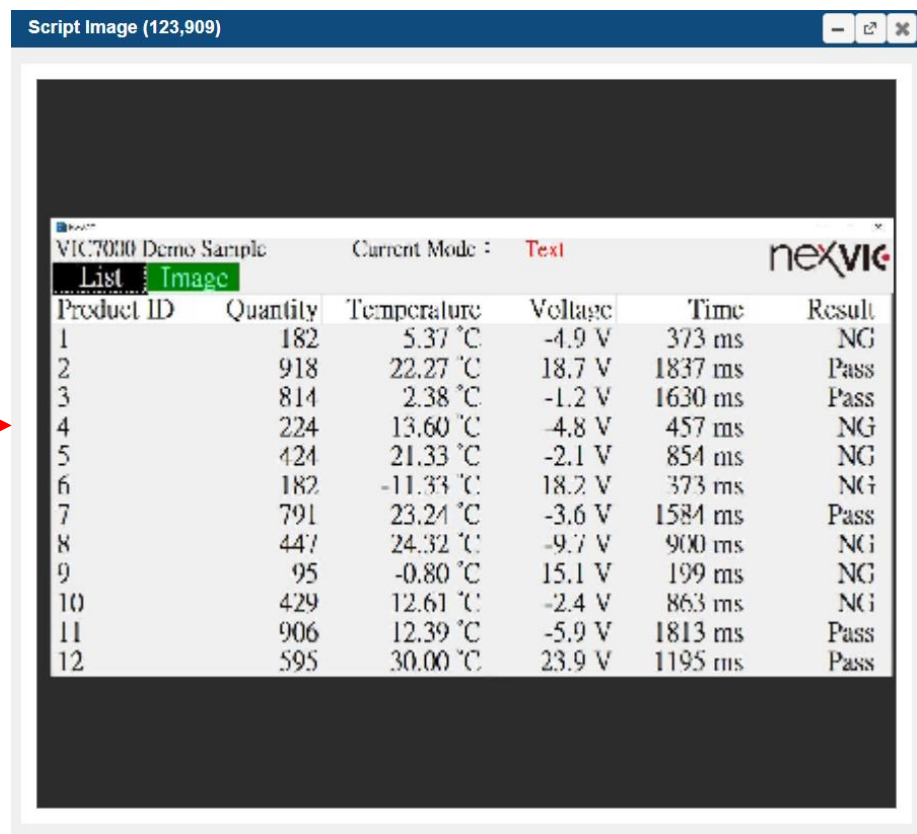
- **SHOW_IMAGE**：在腳本圖像中顯示圖片

1. image：要顯示的圖片

mainLoop 內的 arr 代表專案啟動時辨識的圖片



```
1 from vic import *
2
3
4 def mainLoop(info, arr):
5
6     SHOW_IMAGE(arr)
7
8     pass
```



腳本

- 函式
 - **CURRENT_IMAGE** : 當前辨識的圖片

CURRENT_IMAGE
CURRENT_IMAGE()
CURRENT_IMAGE()

```
1 from vic import *
2
3 def mainLoop(info, arr):
4     SHOW_IMAGE(CURRENT_IMAGE())
5     pass
6
```

Script Image (123,909)

VIC7000 Demo Sample						Current Mode : Text	nexVIG
Product ID	Quantity	Temperature	Voltage	Time	Result		
1	182	5.37 °C	-4.9 V	373 ms	NG		
2	918	22.27 °C	18.7 V	1837 ms	Pass		
3	814	2.38 °C	-1.2 V	1630 ms	Pass		
4	224	13.60 °C	-4.8 V	457 ms	NG		
5	424	21.33 °C	-2.1 V	854 ms	NG		
6	182	-11.33 °C	18.2 V	373 ms	NG		
7	791	23.24 °C	-3.6 V	1584 ms	Pass		
8	447	24.32 °C	-9.7 V	900 ms	NG		
9	95	-0.80 °C	15.1 V	199 ms	NG		
10	429	12.61 °C	-2.4 V	863 ms	NG		
11	906	12.39 °C	-5.9 V	1813 ms	Pass		
12	595	30.00 °C	23.9 V	1195 ms	Pass		

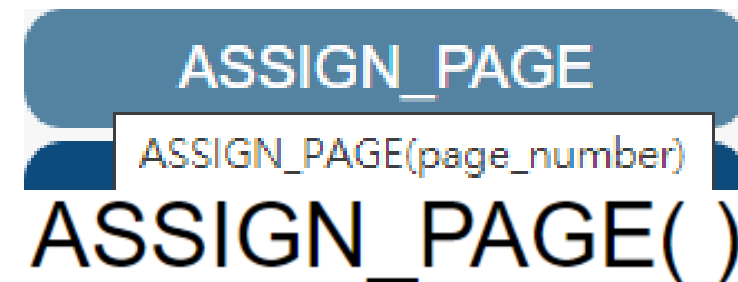
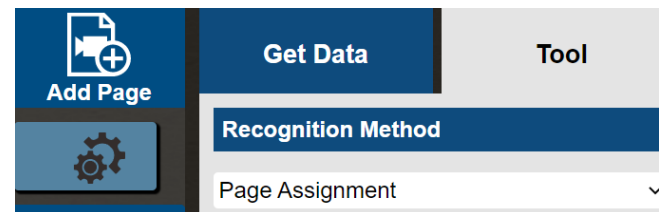
腳本

- 函式

- **ASSIGN_PAGE**：指定當前辨識的頁面，須將辨識方法改為頁面指定

若有管理員登入，此功能無效

1. page_number：要指定的辨識頁面編號

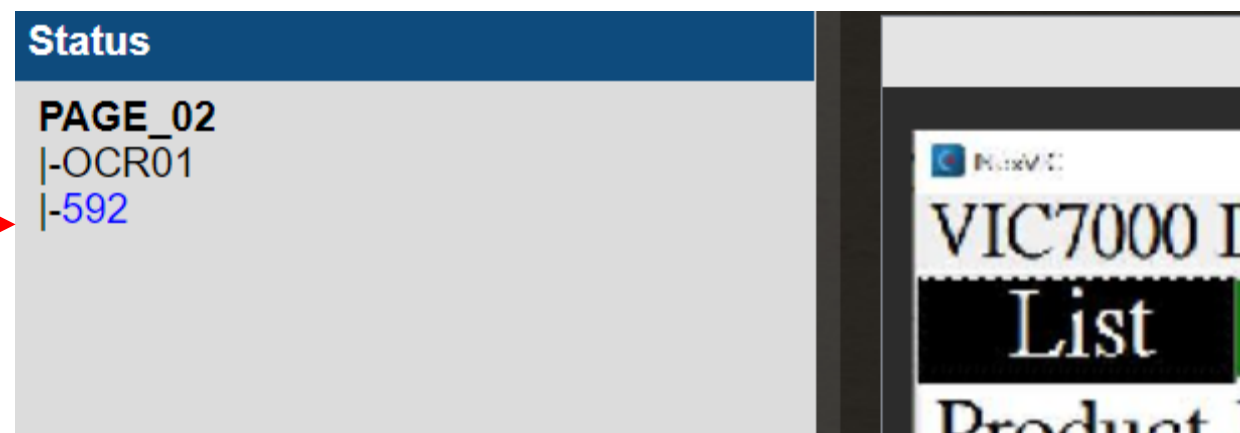


```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    ASSIGN_PAGE(2)
```

```
    pass
```



腳本

- 函式

- TOOL.OCR**：在當前辨識的圖像上進行文字辨識

- select：選擇字型
- fontName：要使用的辨識字型檔名稱，
可從字型檔列表選擇
- roi_x：辨識 ROI 的 x 座標
- roi_y：辨識 ROI 的 y 座標
- roi_width：辨識 ROI 的寬度
- roi_height：辨識 ROI 的高度

OCR Font	
Segmentation Mo	Single Line
1 Select Font	English
Load Font	Font File
Font File	English
Learn Font	Submit

0: 使用字型檔
1: 使用預設字型庫

Font List	Pattern List	Python Output
OCR_Actual		
OCR_Number		
OCR_Power		
OCR_Setpoint		

TOOL.OCR
TOOL.OCR(select, fontName, roi_x, roi_y, roi_width, roi_height)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    CALC[0]=TOOL.OCR( 0,"OCR_Number" , 360,172,73,43 )
```

```
    pass
```

Calculation	Quality
-CALC00	231
-231	439
	625
	260

1,3 frames

- 函式

- TOOL.OCR：進階應用

1. select：選擇字型

2. fontName：要使用的辨識字型檔名稱，可從字型檔列表選擇

3. roi_x：辨識 ROI 的 x 座標

4. roi_y：辨識 ROI 的 y 座標

5. roi_width：辨識 ROI 的寬度

6. roi_height：辨識 ROI 的高度

7. img：要辨識的圖片

8. params：前處理參數

```
CALC[0]=TOOL.OCR(0,"OCR_Number", 360,172,73,43,arr,  
{'segmentation':7, 'preprocess_resize':4,  
'preprocess_resize_method': 0, 'preprocess_threshold_method':1,  
'preprocess_threshold_algorithm': 1, 'preprocess_thredshold_value': 120})
```

```
params = {'segmentation':a, 'preprocess_resize':b, 'preprocess_resize_method': c,  
'preprocess_threshold_method':d, 'preprocess_threshold_algorithm': e,  
'preprocess_thredshold_value': f}
```

• 函式

- TOOL.OCR : 進階應用

'segmentation':7, 'preprocess_resize':4,

'segmentation':a

a = 0 ~ 13

a	Mode
0	PSM_OSD_ONLY
1	PSM_AUTO_OSD
2	PSM_AUTO_ONLY
3	PSM_AUTO
4	PSM_SINGLE_COLUMN
5	PSM_SINGLE_BLOCK_VERT_TEXT
6	PSM_SINGLE_BLOCK

a	Mode
7	PSM_SINGLE_LINE
8	PSM_SINGLE_WORD
9	PSM_CIRCLE_WORD
10	PSM_SINGLE_CHAR
11	PSM_SPARSE_TEXT
12	PSM_SPARSE_TEXT_OSD
13	PSM_RAW_LINE

'preprocess_resize':b

b = 0 ~ 9

b	Resize
0	x1
1	x2
2	x3
3	x4
4	x5
5	x6
6	x7
7	x8
8	X9
9	x10

- 函式

- TOOL.OCR : 進階應用

'preprocess_resize_method': 0, 'preprocess_threshold_method': 1,

'preprocess_resize_method': c

c = 0 ~ 4

c	Method
0	INTER_NEAREST
1	INTER_LINEAR
2	INTER_CUBIC
3	INTER_AREA
4	INTER_LANCZOS4

'preprocess_threshold_method': d

d = 0 ~ 2

d	Method
0	None
1	Binary
2	Binary Inverse

• 函式

```
'preprocess_threshold_algorithm': 1, 'preprocess_thredshold_value': 120})
```

-
- TOOL.OCR**
- : 進階應用

'preprocess_threshold_algorithm':e

e = 0 ~ 2

'preprocess_threshold_method':d

f = 0 ~ 255

Binary Threshold Value

e	Method
0	None
1	THRESH_OTSU
2	THRESH_TRIANGLE

- 函式

- **TOOL.COLOR**：在當前辨識的圖像上進行顏色比對

1. red：要比對的 RGB 色域紅色值
2. green：要比對的 RGB 色域綠色值
3. blue：要比對的 RGB 色域藍色值
4. roi_x：辨識 ROI 的 x 座標
5. roi_y：辨識 ROI 的 y 座標
6. roi_width：辨識 ROI 的寬度
7. roi_height：辨識 ROI 的高度
8. tolerance：比對容錯度

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    CALC[0]=TOOL.COLOR(0,128,0,161,89,13,13,5)
```

```
    pass
```

TOOL.COLOR
TOOL.COLOR(red, green, blue, roi_x, roi_y, roi_width, roi_height, tolerance)
TOOL.COLOR(, , , , , , ,)

Calculation

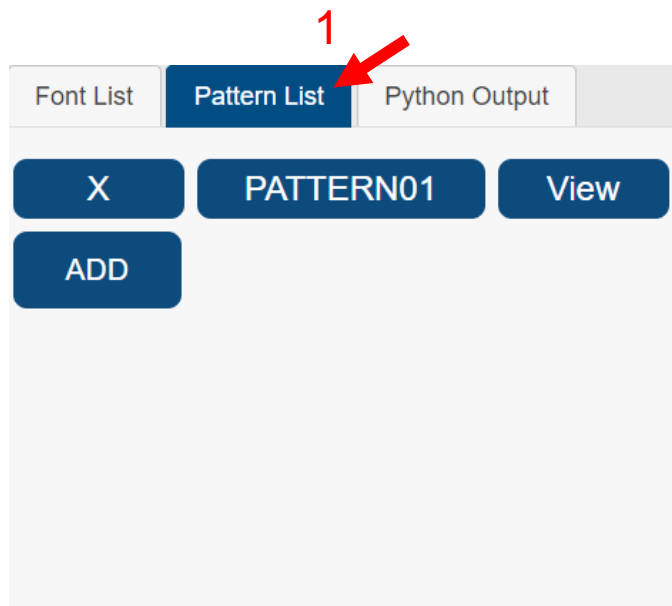
```
-CALC00  
-true
```



- 函式

- **TOOL.PATTERN**：在當前辨識的圖像上進行樣式比對

1. pattern_id：要比對的樣式，從樣式列表進行選擇和設定
2. roi_x：辨識 ROI 的 x 座標
3. roi_y：辨識 ROI 的 y 座標
4. roi_width：辨識 ROI 的寬度
5. roi_height：辨識 ROI 的高度



TOOL.PATTERN
TOOL.PATTERN(pattern_id, roi_x, roi_y, roi_width, roi_height)
TOOL.PATTERN(, , , ,)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    CALC[0]=TOOL.PATTERN("PATTERN01",1142,35,205,63)
```

```
    pass
```



- 函式

- **TOOL.PATTERN**：進階應用

1. pattern_id：要比對的樣式，從樣式列表進行選擇和設定

2. roi_x：辨識 ROI 的 x 座標

3. roi_y：辨識 ROI 的 y 座標

4. roi_width：辨識 ROI 的寬度

5. roi_height：辨識 ROI 的高度

6. img：要辨識的圖片

7. params：樣式比對參數

params = {'gray_match':True/False,'min_score':0~1.0}

gray_match：是否使用灰階比對

Min_score：比對最小分數

```
CALC[0]=TOOL.PATTERN("PATTERN01",1142,35,205,63,  
arr,{'gray_match':True,'min_score':0.95})
```


- 函式
 - **SET_PREFERENCE**：設定系統變數的值
 1. key：變數名稱
 2. value：設定值

SET_PREFERENCE

SET_PREFERENCE(key, value)

SET_PREFERENCE(,)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    SET_PREFERENCE('abc','123')
```

```
    pass
```

- 函式
 - **GET_PREFERENCE** : 取得系統變數的值
 1. key : 變數名稱
 2. default : 若未找到該變數的預設值

GET_PREFERENCE

GET_PREFERENCE(key, default_value)

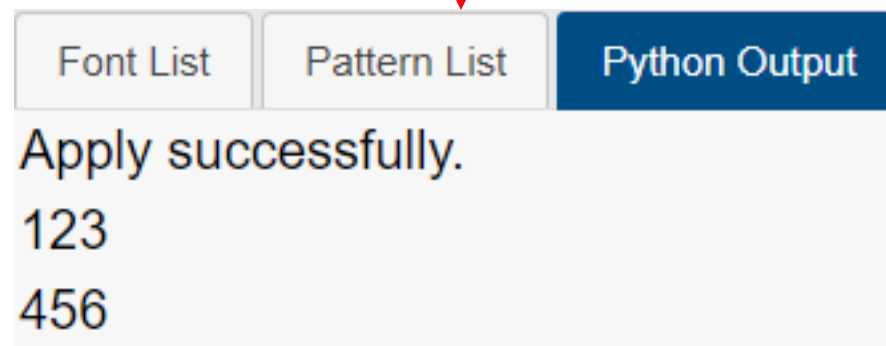
GET_PREFERENCE(,)

```
from vic import *
```

```
def mainLoop(info, arr):
```

```
    print(GET_PREFERENCE('abc','456'))  
    print(GET_PREFERENCE('def','456'))
```

```
    pass
```



- 函式

- **SHOW_DIALOG**：彈出視窗

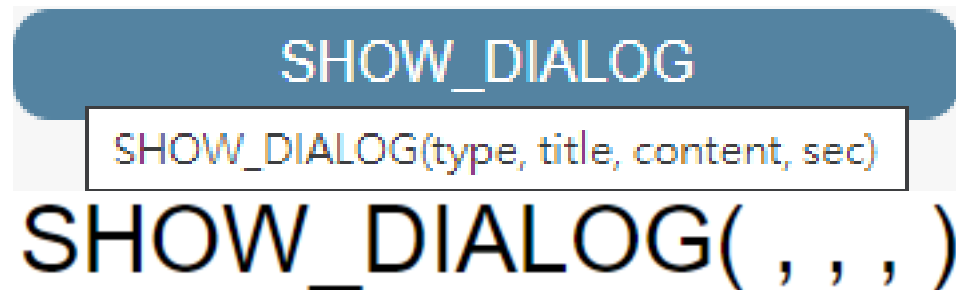
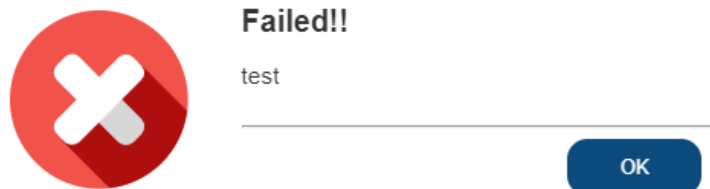
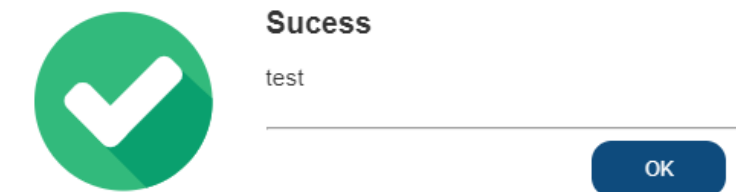
- 1. type：視窗類型

- 0 = 成功，1 = 失敗，2 = 資訊，3 = 警告

- 2. title：視窗標題

- 3. content：訊息內容

- 4. sec：執行間隔時間



- 控制函式
 - **PLAY.CONTROL**：設定控制啟動停止
 1. status：控制執行狀態
 - 0 = 啟動控制
 - 1 = 停止控制

PLAY.CONTROL

PLAY.CONTROL(status)

PLAY.CONTROL()

- 控制函式
 - **MOUSE_MOVE**：控制滑鼠游標移動到特定位置
 1. control_type：要使用的控制方式
0 = USB，1 = 本機
 2. x：目標 x 座標
 3. y：目標 y 座標

MOUSE_MOVE
MOUSE_MOVE(control_type, x, y)
MOUSE_MOVE(, ,)

- 控制函式

- **MOUSE_CLICK**：控制滑鼠游標移動到特定位置，並點擊一次

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. x：目標 x 座標

3. y：目標 y 座標

4. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

MOUSE_CLICK

MOUSE_CLICK(control_type, x, y, click_type)

MOUSE_CLICK(, , ,)

- 控制函式

- **MOUSE_DOUBLE_CLICK**：控制滑鼠游標移動到特定位置，並連點兩次

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. x：目標 x 座標

3. y：目標 y 座標

4. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

MOUSE_DOUBLE_CLICK

`MOUSE_DOUBLE_CLICK(control_type, x, y, click_type)`

MOUSE_DOUBLE_CLICK(, , ,)

- 控制函式

- **MOUSE_DRAG**：控制滑鼠游標從一點移動另一點，同時進行拖曳

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

3. x：拖曳起點 x 座標

4. y：拖曳起點 y 座標

5. x1：拖曳終點 x 座標

6. y1：拖曳終點 y 座標



- 控制函式

- MOUSE_OCR_CLICK**：控制滑鼠在符合比對條件的文字的位置點擊一次

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

3. text：文字比對條件

4. select：選擇字型

5. fontName：要使用的辨識字型檔名稱，可從字型檔列表選擇

6. roi_x：辨識 ROI 的 x 座標

7. roi_y：辨識 ROI 的 y 座標

8. roi_width：辨識 ROI 的寬度

9. roi_height：辨識 ROI 的高度

MOUSE_OCR_CLICK

MOUSE_OCR_CLICK(control_type, click_type, text, select, font_name, roi_x, roi_y, roi_width, roi_height)

MOUSE_OCR_CLICK(, , , , , , , ,)

OCR Font

Segmentation Mo	Single Line	▼
4 Select Font	English	▼
Load Font	Font File	English
Font File		
Learn Font	Submit	

0：使用字型檔
1：使用預設字型庫

5

Font List Pattern List Python Output

OCR_Actual

OCR_Number

OCR_Power

OCR_Setpoint

- 控制函式

- **MOUSE_OCR_CLICK**：進階使用

1. control_type：要使用的控制方式，0 = USB，1 = 本機
2. click_type：點擊按鍵類型，1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵
3. text：文字比對條件
4. select：選擇字型
5. fontName：要使用的辨識字型檔名稱，可從字型檔列表選擇
6. roi_x：辨識 ROI 的 x 座標
7. roi_y：辨識 ROI 的 y 座標
8. roi_width：辨識 ROI 的寬度
9. roi_height：辨識 ROI 的高度
10. img：要辨識的圖像
11. params：前處理參數

```
params = {'segmentation':a, 'preprocess_resize':b,  
'preprocess_resize_method': c,  
'preprocess_threshold_method':d,  
'preprocess_threshold_algorithm': e,  
'preprocess_thredshold_value': f}
```

- 控制函式

'segmentation':7, 'preprocess_resize':4,

- **MOUSE_OCR_CLICK** : 進階使用

'segmentation':a

a = 0 ~ 13

a	Mode
0	PSM_OSD_ONLY
1	PSM_AUTO_OSD
2	PSM_AUTO_ONLY
3	PSM_AUTO
4	PSM_SINGLE_COLUMN
5	PSM_SINGLE_BLOCK_VERT_TEXT
6	PSM_SINGLE_BLOCK

a	Mode
7	PSM_SINGLE_LINE
8	PSM_SINGLE_WORD
9	PSM_CIRCLE_WORD
10	PSM_SINGLE_CHAR
11	PSM_SPARSE_TEXT
12	PSM_SPARSE_TEXT_OSD
13	PSM_RAW_LINE

'preprocess_resize':b

b = 0 ~ 9

b	Resize
0	x1
1	x2
2	x3
3	x4
4	x5
5	x6
6	x7
7	x8
8	X9
9	x10

- 控制函数

- **MOUSE_OCR_CLICK** : 進階使用

'preprocess_resize_method':c

c = 0 ~ 4

c	Method
0	INTER_NEAREST
1	INTER_LINEAR
2	INTER_CUBIC
3	INTER_AREA
4	INTER_LANCZOS4

'preprocess_resize_method': 0, 'preprocess_threshold_method':1,

'preprocess_threshold_method':d

d = 0 ~ 2

d	Method
0	None
1	Binary
2	Binary Inverse

- 控制函数 `'preprocess_threshold_algorithm': 1, 'preprocess_thredshold_value': 120}}`

- **MOUSE_OCR_CLICK** : 進階使用

`'preprocess_threshold_algorithm':e`

$e = 0 \sim 2$

`'preprocess_threshold_method':d`

$f = 0 \sim 255$

Binary Threshold Value

e	Method
0	None
1	THRESH_OTSU
2	THRESH_TRIANGLE

- 控制函式

- **MOUSE_OCR_DOUBLE_CLICK**：控制滑鼠在符合比對條件的文字的位置連點兩次，進階使用同 MOUSE_OCR_CLICK

1. control_type：要使用的控制方式，0 = USB，1 = 本機

2. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

3. text：文字比對條件

4. select：選擇字型

5. fontName：要使用的辨識字型檔名稱，可從字型檔列表選擇

6. roi_x：辨識 ROI 的 x 座標

7. roi_y：辨識 ROI 的 y 座標

8. roi_width：辨識 ROI 的寬度

9. roi_height：辨識 ROI 的高度

MOUSE_OCR_DOUBLE_CLICK
MOUSE_OCR_DOUBLE_CLICK(control_type, click_type, text, select, font_name, roi_x, roi_y, roi_width, roi_height)
MOUSE_OCR_DOUBLE_CLICK(, , , , , , , ,)

OCR Font

Segmentation Mode	Single Line
4 Select Font	English
Load Font	Font File
Font File	English
Learn Font	Submit

0: 使用字型檔
1: 使用預設字型庫

5

Font List Pattern List Python Output

OCR_Actual

OCR_Number

OCR_Power

OCR_Setpoint

- 控制函式

- **MOUSE_PATTERN_CLICK**：控制滑鼠在符合比對條件的樣式的位置點擊一次

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. click_type：點擊按鍵類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

3. pattern_id：要比對的樣式，從樣式列表進行選擇和設定

4. roi_x：辨識 ROI 的 x 座標

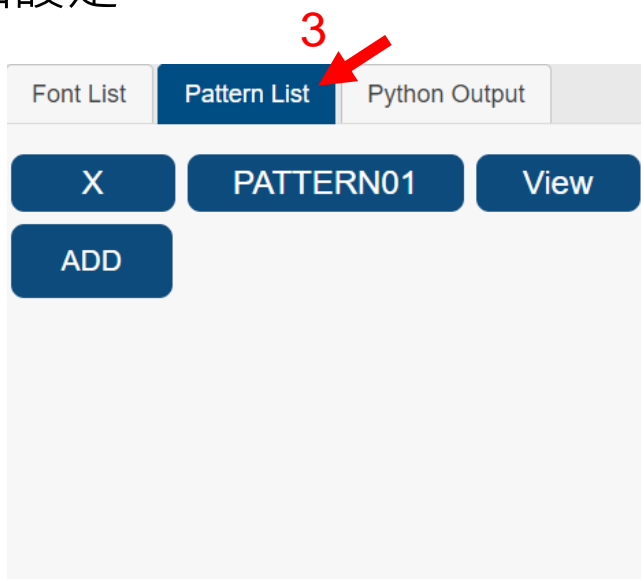
5. roi_y：辨識 ROI 的 y 座標

6. roi_width：辨識 ROI 的寬度

7. roi_height：辨識 ROI 的高度

8. score：最小比對分數

MOUSE_PATTERN_CLICK
MOUSE_PATTERN_CLICK(control_type, click_type, pattern_id, roi_x, roi_y, roi_width, roi_height, score)



- 控制函式

- **MOUSE_PATTERN_CLICK**：進階使用

1. control_type：要使用的控制方式，0 = USB，1 = 本機
2. click_type：點擊按鍵類型
1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵
3. pattern_id：要比對的樣式，從樣式列表進行選擇和設定
4. roi_x：辨識 ROI 的 x 座標
5. roi_y：辨識 ROI 的 y 座標
6. roi_width：辨識 ROI 的寬度
7. roi_height：辨識 ROI 的高度
8. score：最小比對分數
9. img：要辨識的圖片
10. params：樣式比對參數

params =

{'gray_match':True/False,'min_score':0~1.0}

gray_match：是否使用灰階比對

Min_score：比對最小分數

- 控制函式

MOUSE_PATTERN_DOUBLE_CLICK：控制滑鼠在符合比對條件的樣式的位置連點兩次，
進階使用同MOUSE_PATTERN_CLICK

1. control_type：要使用的控制方式

0 = USB，1 = 本機

2. click_type：點擊按鈕類型

1 = 左鍵，2 = 滑鼠中鍵，3 = 右鍵

3. pattern_id：要比對的樣式，從樣式列表進行選擇和設定

4. roi_x：辨識 ROI 的 x 座標

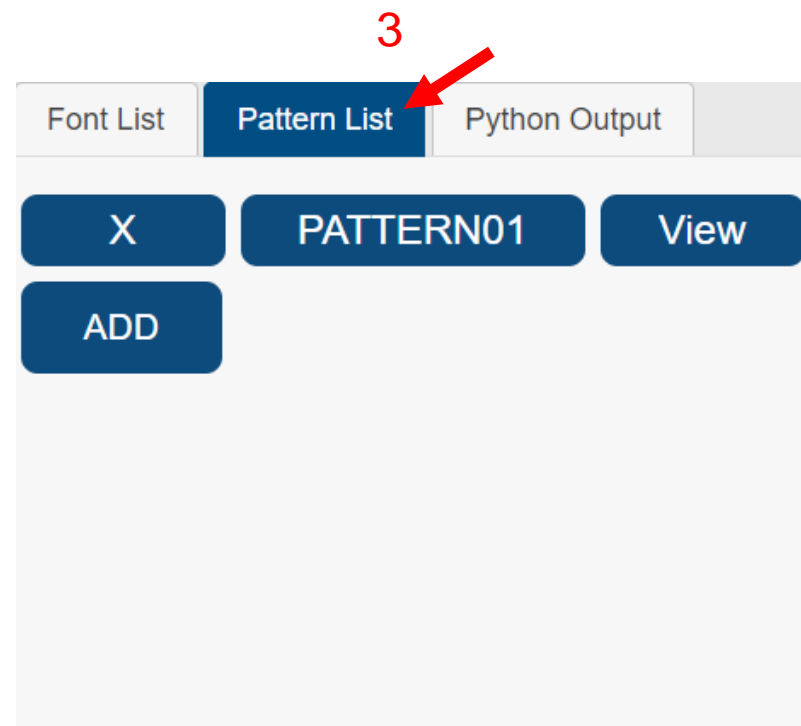
5. roi_y：辨識 ROI 的 y 座標

6. roi_width：辨識 ROI 的寬度

7. roi_height：辨識 ROI 的高度

8. score：最小比對分數

MOUSE_PATTERN_CLICK
MOUSE_PATTERN_CLICK(control type, click type, pattern id, roi x, roi y, roi width, roi height, score)



- 控制函式

- **KEYBOARD_EVENT**：控制鍵盤按下設定的字串內容

1. control_type：要使用的控制方式

0 = USB，1 = 本機，2 = 序列埠(PS/2)，3 = 瀏覽器

2. content：要按下的字串內容

實際輸入內容會以受控裝置的設定為主

KEYBOARD_EVENT

KEYBOARD_EVENT(control_type, content)

KEYBOARD_EVENT(,)

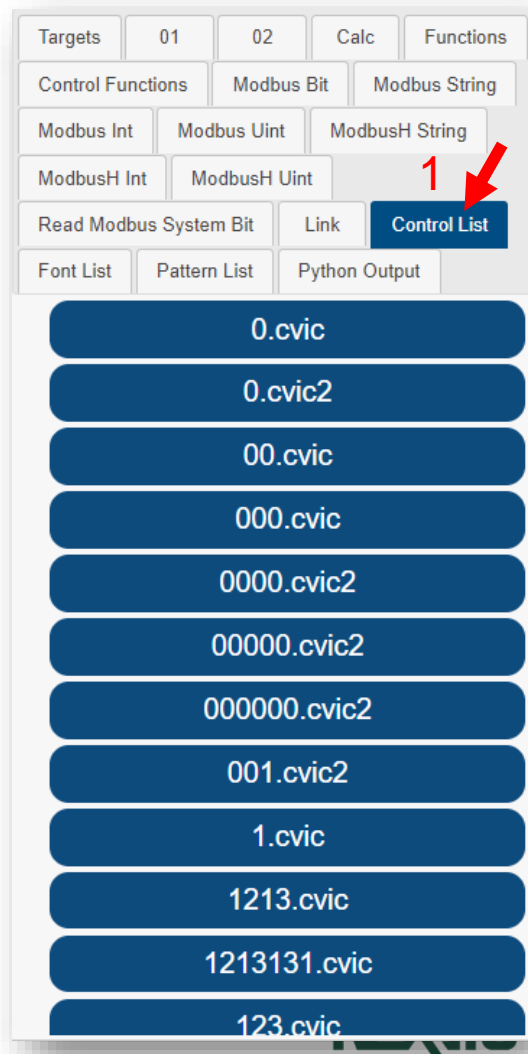
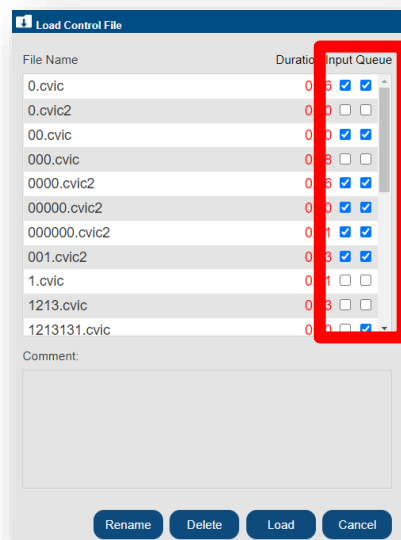
- 控制函式

- PLAY.CONTROL_FILE_IQ**：套用佇列和輸入屬性，執行特定控制檔

須先到載入控制檔視窗內設定，* 代表可不設定，含進階設定

1. control_file_name：設定要執行的控制檔，可從控制檔列表選擇
2. sec：執行間隔時間
3. *offset_x：控制檔執行時的 x 軸偏移量，預設為 0
4. *offset_y：控制檔執行時的 y 軸偏移量，預設為 0
5. *sync：是否同步執行，預設為 False

```
PLAY.CONTROL_FILE_IQ  
PLAY.CONTROL_FILE_IQ(control_file_name, sec, offset_x=0, offset_y=0, bSync=False)
```



- **Modbus函式**

- **MODBUS_MASTER_READ_BIT**：從 Modbus slave/server 讀取位元資料

1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
2. prefix：Modbus 表格代碼，0 = Coil Status，1 = Input Status
3. regs：讀取位址，1-based
4. sec：執行間隔時間

```
MODBUS_MASTER_READ_BIT
MODBUS_MASTER_READ_BIT(id, prefix, regs, sec)
MODBUS_MASTER_READ_BIT( , , , )
```



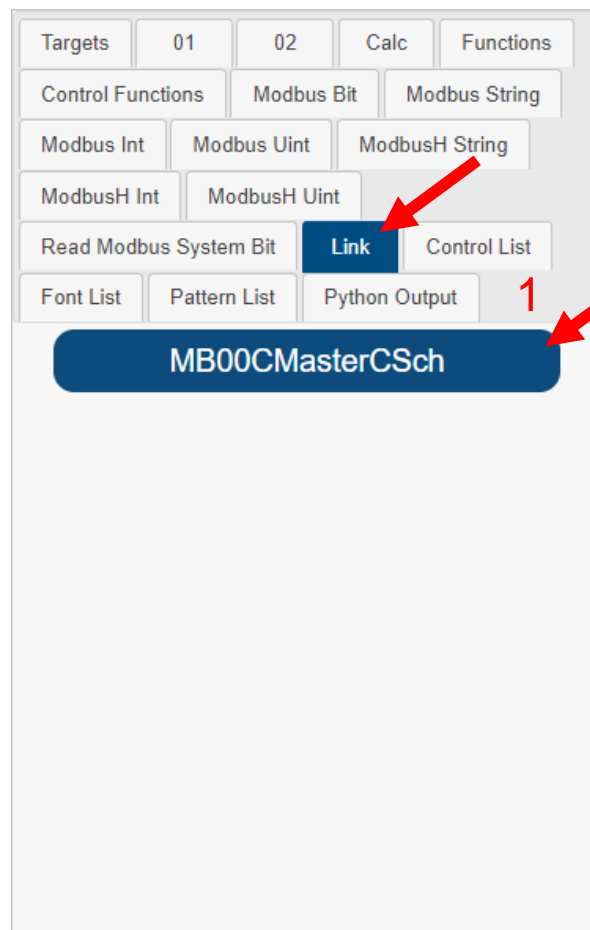
- **Modbus函式**

- **MODBUS_MASTER_WRITE_BIT**：寫入位元資料到 Modbus slave/server
 1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
 2. prefix：Modbus 表格代碼，0 = Coil Status
 3. regs：寫入位址，1-based
 4. value：要寫入的位元資料
 5. sec：執行間隔時間

MODBUS_MASTER_WRITE_BIT

MODBUS_MASTER_WRITE_BIT(id, prefix, regs, value, sec)

MODBUS_MASTER_WRITE_BIT(, , , ,)



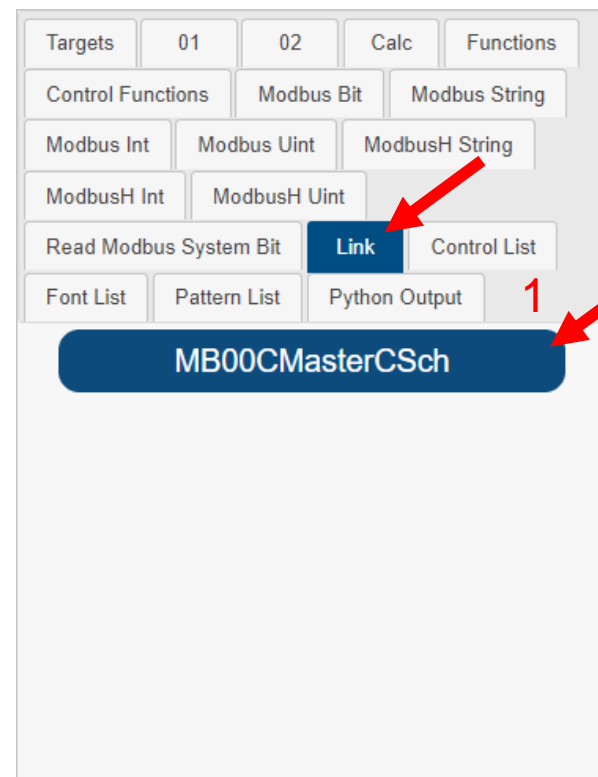
- **Modbus函式**

- **MODBUS_MASTER_READ_STR**：從 Modbus slave/server 讀取字串資料
 1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
 2. prefix：Modbus 表格代碼，3 = Input Register，4 = Holding Register
 3. regs：讀取位址，1-based
 4. count：讀取位址數量
 5. sec：執行間隔時間

MODBUS_MASTER_READ_STR

MODBUS_MASTER_READ_STR(id, prefix, regs, regs_count, sec)

MODBUS_MASTER_READ_STR(, , , ,)

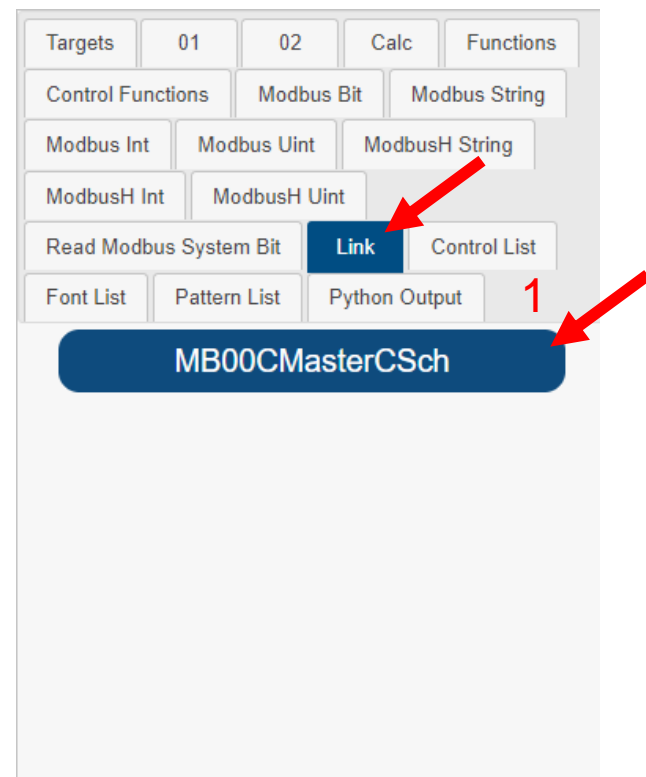


- **Modbus函式**

- **MODBUS_MASTER_WRITE_STR**：寫入字串資料到 Modbus slave/server

1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
2. prefix：Modbus 表格代碼，4 = Holding Register
3. regs：寫入位址，1-based
4. value：要寫入的字串資料
5. sec：執行間隔時間

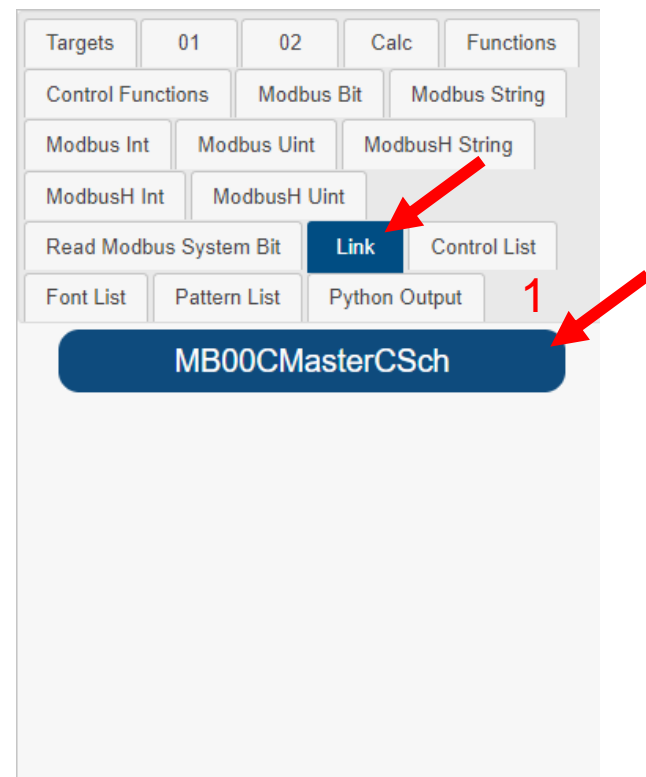
```
MODBUS_MASTER_WRITE_STR
MODBUS_MASTER_WRITE_STR(id, prefix, regs, value, sec)
MODBUS_MASTER_WRITE_STR( , , , , )
```



- **Modbus函式**

- **MODBUS_MASTER_READ_UINT**：從 Modbus slave/server 讀取整數資料
 1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
 2. prefix：Modbus 表格代碼，3 = Input Register，4 = Holding Register
 3. regs：讀取位址，1-based
 4. count：讀取位址數量
 5. sec：執行間隔時間

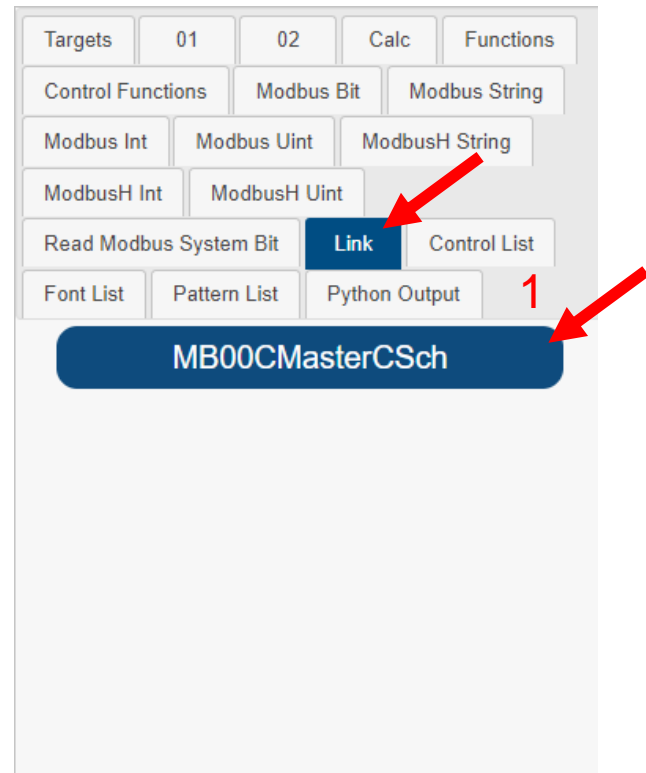
```
MODBUS_MASTER_READ_UINT
MODBUS_MASTER_READ_UINT(id, prefix, regs, regs_count, sec)
MODBUS_MASTER_READ_UINT( , , , , )
```



- **Modbus函式**

- **MODBUS_MASTER_WRITE_UINT**：寫入整數資料到 Modbus slave/server
 1. id：要使用的 Modbus 連線，要先在連結頁面內建立連線，可從連結中選擇要使用的連線
 2. prefix：Modbus 表格代碼，4 = Holding Register
 3. regs：寫入位址，1-based
 4. value：要寫入的整數資料
 5. sec：執行間隔時間

```
MODBUS_MASTER_WRITE_UINT
MODBUS_MASTER_WRITE_UINT(id, prefix, regs, value, sec)
MODBUS_MASTER_WRITE_UINT( , , , , )
```



- 呼叫自定義函式

自定義函式設置

1. func_name : 自定義函式名稱
2. kwargs : 輸入參數引數
3. param1_name : 輸入參數1名稱
4. param2_name : 輸入參數2名稱
- .
- .
5. return1_name : 回傳參數1名稱
6. return2_name : 回傳參數2名稱

.

.

```
def demo_call_python_function(**kwargs):  
    param1=kwargs['param1_name']  
    param2=kwargs['param2_name']  
    .  
    .  
    return1=int(param1)+int(param2)  
    return2=int(param1)*int(param2)  
    .  
    .  
    return{'return1_name':return1,'return2_name':return2}
```

- 呼叫自定義函式

透過 RESTful API 呼叫 VIC 執行自定義函式

1. username : admin or user
2. password : 123456(default)
3. cmd : req_call_python_function
4. value : 1
5. func : func_name
6. param1_name : param1
7. param2_name : param1

The screenshot shows a REST client interface with a PUT request configured for the endpoint `10.12.1.169/restful/put`. The request body is set to `x-www-form-urlencoded`. The parameters are as follows:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	admin	or user
<input checked="" type="checkbox"/> password	123456	default
<input checked="" type="checkbox"/> cmd	req_call_python_function	
<input checked="" type="checkbox"/> value	1	
<input checked="" type="checkbox"/> func	func_name	name of custom function
<input checked="" type="checkbox"/> param1_name	param1	input param1
<input checked="" type="checkbox"/> param2_name	param2	input param2

- 呼叫自定義函式

範例：透過 RESTful API 發出兩個參數，透過 VIC 計算後回傳計算值

PUT call python function

VIC Demo / call python function

PUT 10.12.169/restful/put

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> username	admin
<input checked="" type="checkbox"/> password	123456
<input checked="" type="checkbox"/> cmd	req_call_python_function
<input checked="" type="checkbox"/> value	1
<input checked="" type="checkbox"/> func	demo_call_python_function
<input checked="" type="checkbox"/> param1_name	3
<input checked="" type="checkbox"/> param2_name	2
Key	Value

Body Cookies Headers (2) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "login_success": "1",
3   "return1_name": 5,
4   "return2_name": 6
5 }
```

```
1 from vic import *
2
3 def demo_call_python_function(**kwargs):
4
5     param1=kwargs['param1_name']
6     param2=kwargs['param2_name']
7     print('param1='+param1)
8     print('param2='+param2)
9
10    return1=int(param1)+int(param2)
11    return2=int(param1)*int(param2)
12    print('return1='+str(return1))
13    print('return2='+str(return2))
14
15    return{'return1_name':return1,'return2_name':return2}
```

MODBUS	LINK	Control List	Font List	Pattern List
Python Output				
param1=3				
param2=2				
return1=5				
return2=6				

- 匯入 **py**：匯入自行編寫的 **py** 檔

將自行編寫的 py 檔放置於 VIC 的 py 資料夾中，路徑如下

Windows：C:\VIC7000\bin\py

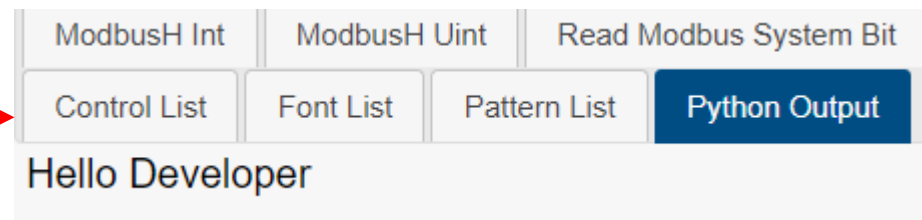
Linux：/opt/VIC7000/bin/py

就可以在 Python 腳本中匯入 py 檔並使用它，如下範例第 3 行

若該 py 檔會被更改，則需要 reload，如下範例第 2、4 行

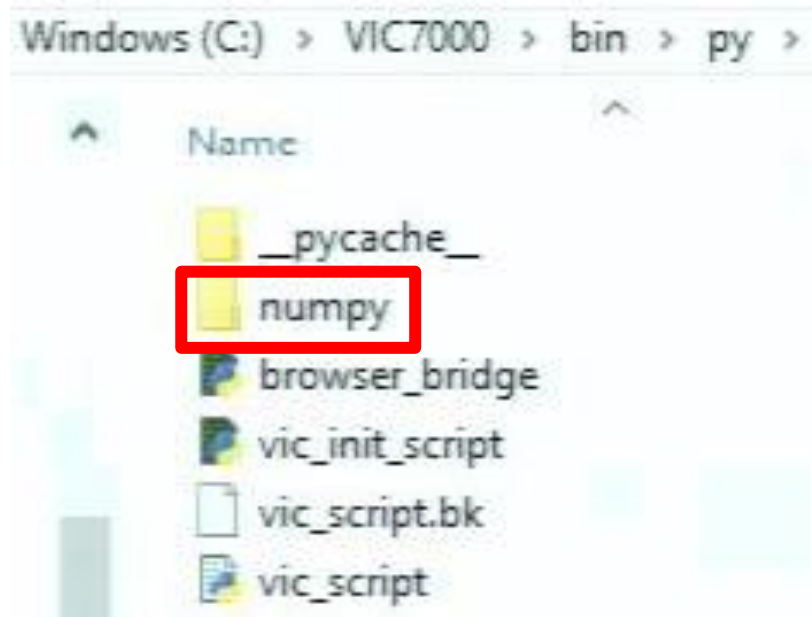
```
def demo_develop():  
    print('Hello Developer')
```

```
1 from vic import *  
2 from imp import reload  
3 import Demo_Develop_Import  
4 reload(Demo_Develop_Import)  
5  
6 def mainLoop(info, arr):  
7     Demo_Develop_Import.demo_develop()  
8
```



- 匯入套件PYPI

- 線上安裝：使用 Windows 的 Command Prompt (Admin) 和 Linux 的 terminal，輸入 `pip install XXXX`，XXXX 為套件名稱，即可安裝套件，使用 Python 腳本時要再次 import 該套件
- 離線安裝：使用另一台可連網並安裝 python 的電腦，使用上述 pip 安裝後，將套件放入 VIC7000 電腦上的 C:\VIC7000\bin\py 內即可，使用 Python 腳本時要再次 import 該套件



```
from vic import *  
import numpy
```

```
def mainLoop(info, arr):
```

```
pass
```

NEXIOT

Your Partner in Smart Manufacturing



Q&A



[NexVIC產品教學影片](#)



NexVIC產品服務群組



FB Page:

NexAloT in Action @