

# Week 1-模擬自動化介紹



## 講師介紹

講師：林鳴志

擔任**EDA**應用工程師**17**年，主要負責模擬軟體推廣，教育訓練及技術支援工作，領域為高頻電磁場，高速訊號，通訊系統模擬等。目前主要負責**5G**毫米波天線模組模擬及分析，並協助客戶建立模擬自動化流程。目前經營Youtube頻道[HowToSim](#)，介紹高頻模擬應用技巧。已累積超過**270**支影片，訂閱人數超過**2000**。

助教：

## 戴偉修(Wayne)，Maxwell專精

## 第一周：模擬自動化介紹

- **AEDT**模擬自動化導論
- 教學資源及討論社群說明
- **Python**基本語法講解(1/4)

## 第二周：AEDT操作錄製與修改

- 將複雜的自動化問題拆解可執行的代碼
- 改編AEDT錄製腳本及除錯
- **Python**基本語法講解(2/4)

## 第三周：檔案處理與格式化輸出

- 檔案讀取與資料分析
- 格式化輸出
- **Python**基本語法講解(3/4)

## 第四周：透過批次指令達到模擬自動化

- **AEDT**批次命令
- **PCB/PKG**前處理自動化
- **Python**基本語法講解(4/4)

## 第五周：GUI設計及AEDT函式庫簡介

- **QT**函式庫與**WPF**介紹
- **EXCEL**及屬性視窗輸入
- **AEDT**函式庫

## 第六周：正規表示式與SIwave自動化

- 正規表示式概念
- **SIwave**自動化簡介
- **SIwave**函式庫

## 第七周：圖表生成及HTML報告輸出

- **AEDT**資料輸出
- **Matplotlib**函式庫介紹
- 整合表格及圖表及html報告輸出

## 第八周：模擬自動化進階

- **Python**程式優化
- **Optislang**
- 分享及結業式(證書與精美禮物)

# 上課進行的方式為何？

八周上課的內容由淺到深。每一週三個小時的內容會涵蓋：

- 自動化的概念，我們會介紹**AEDT**各式的自動化技巧。
- **Python**編程及專題討論。除了介紹基本的**Python**語法，還會討論到自動化常用的函式庫。

前四周為基本課程，參加學員必須構思一個專題題目，並且學習如何分解題目及相關解決手法。後面四周為進階課程，參加學員開始程式開發，並於最後一周作功能展示。專案需用到參加學員個人時間來完成。

# 上課教材及資源：AEDT\_Automation\_Camp (GitHub)

linmingchih / AEDT-Automation-Camp

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security 0

Insights

Settings

AEDT模擬自動化專班的上課教材及網路資源

Edit

Manage topics

126 commits

1 branch

0 packages

0 releases

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

linmingchih Add files via upload

Latest commit 29ebf4d 2 minutes ago

AEDT函式庫列表	Delete test.txt	10 days ago
AEDT檔案	Update 檔案格式說明.md	11 days ago
上課教材	Add files via upload	2 minutes ago
成果發表/2020	Create readme.md	10 days ago
程式碼範例	Create openSaveFileDialog.py	4 hours ago
Python學習資源.md	Update Python學習資源.md	7 days ago
Python常用函式列表.md	Update Python常用函式列表.md	17 hours ago
Python常用類別及方法列表.md	Rename Python常用方法列表.md to Python常用類別及方法列表.md	17 hours ago
課程大綱.md	Update 課程大綱.md	7 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

# 問題發問及討論：AEDT模擬自動化 (Facebook)



# 自動化技術短文：AEDT模擬自動化 (Blog)

## AEDT模擬自動化

2020年6月26日 星期五

### 如何利用Python呼叫nexxim.exe執行電路模擬

在AEDT的電路模擬當中，我們習慣用原理圖（schematic）來建構電路（圖一）。原理圖的好處是容易閱讀及編輯。但是在執行模擬時，原理圖會先被轉換成網表（netlist，圖二），再透過電路模擬器nexxim.exe執行運算(DC, AC, Transient)之後輸出模擬結果。

其實我們也可以在命令列視窗當中或利用Python程式(圖三)讓nexxim.exe直接運算網表來得到模擬結果。因為少了原理圖轉換網表的工作，模擬時間可以大幅縮短。此外，不需要打開AEDT視窗也可以執行。模擬完成之後輸出的資料會以二進制格式儲存在sdf檔案當中。接下來便可以透過sdf2csv.exe將資料轉換成csv文字檔(圖四)。最後就可以在程式當中讀取csv作資料的分析了。

範例代碼如下：

```
import os
os.environ["PATH"] = 'C:/Program Files/AnsysEM/AnsysEM20.1/Win64'

import subprocess
subprocess.run(["nexxim.exe", "d:/demo/rc.cir"])
subprocess.run(["sdf2csv.exe", "d:/demo/rc.cir.sdf"])
```

首頁

AEDT自動化臉書社團

GitHub範例程式碼

網誌存檔

▼ 2020 (4)

▼ 06 (4)

如何利用Python呼叫  
nexxim.exe執行電路模擬

如何讀取HFSS模擬資料

\_\_getattr\_\_()方法動態定義屬  
性

如何避免執行  
ValidateDesign()之後跳出  
錯誤視窗？

搜尋此網誌

搜尋



# 建立模擬自動化基本觀念

# 模擬自動化對公司的好處

一個複雜模擬需要上百個操作步驟才能完成。對於固定的應用(比方說天線設計)而言，某些操作步驟是固定的(比方說設Port, 設Radiation boundary，輸出天線參數)。如果這些固定操作步驟可以寫進到腳本之中，一鍵完成執行。一方面縮短操作時間，又避免了設定出錯的機會，一舉兩得。對於公司來說可以提高模擬工具的產出。

除此之外，自動化可以避免人員異動造成知識斷鏈。當資深工程師異動時，承接模擬工作的新進人員需要一段時間才能充分掌握模擬工作的技巧。平時就將資深工程師摸索出來的模擬流程自動化之後，之後接手的人也可以最少的代價，正確無誤的完成複雜模擬的操作。

# 寫自動化程式對模擬工程師個人的好處

模擬軟體的精通最多不過半年，之後是一個個案子不斷的的操作，其價值是很低的。工程師的價值是持續的攻克工程難題，這必須不斷的學習及思考，只是熟悉工具的操作不可能進階為領域的專家。

技術的拓展不受限於單一工具本身的功能，有能力串聯不同技術形成獨有的解決方案。也不會對龐大的數據量感到手足無措，反而能在梳理數據過程找到樂趣。隨時在思考如何更有效率的完成模擬工作，而不是煩惱這麼多的工作要做到何時。在提升生產力的同時，也更有餘力去嘗試新的技術領域。



# 為什麼ANSYS要開自動化的課程？

模擬是工程裡面相對複雜的技術，需要理論與實務兼備，這樣的人才本來就少，要能寫程式的更是鳳毛麟角。因為市場小，自然也就缺少足夠的師資及教材。許多實務面所遇到的問題必須由工程師自行摸索尋找答案，單打獨鬥不但辛苦且容易產生挫折感。**ANSYS**看到了業界的需求也了解了困難所在，**ANSYS**希望能將過往所累積的模擬自動化經驗分享給業界，並打造一個讓大家可以相互討論切磋成長的社群。

在課程結束之後，各位仍然可以利用臉書的社團來提問及討論。我們也希望各位可以不吝分享自己的經驗指導後進。我也會持續在臉書社團跟各位互動。只有透過不斷的學習及分享才能積累長期的技術資本進而壯大台灣的工程模擬社群，讓台灣工業在面臨世界大廠的挑戰能更有底氣。

## 程式委外可行？

很多人可能會想，我自己不會編程，是不是可以將編程外包給專門寫程式的工作室完成。現在外面可以寫程式的工程師比比皆是，要找到物美價廉的程式設計師應該不難。理論上是這樣，實際上有其困難。

多數程式設計師缺少工程數學與電子工程相關的訓練，要讓程式設計師理解模擬軟體的操作並熟悉當中的原理就需要不少時間。尤其是工程領域的範圍很狹窄，多數程式設計師不會有意願投資許多的時間去學習模擬工程的相關知識。

對程式設計師來說，互聯網商務，網路金融或是APP等才是更大更賺錢的應用。因此比較可行的方法還是由模擬工程師學習編程的技巧來開發所需的自動化程式。

# Why Python ?



Rank	Language	Type	Score
1	Python	🌐 🗨 ⚙	100.0
2	Java	🌐 📱 🗨	96.3
3	C	📱 🗨 ⚙	94.4
4	C++	📱 🗨 ⚙	87.5
5	R	🗨	81.5
6	JavaScript	🌐	79.4
7	C#	🌐 📱 🗨 ⚙	74.5
8	Matlab	🗨	70.6
9	Swift	📱 🗨	69.1
10	Go	🌐 🗨	68.0



NumPy

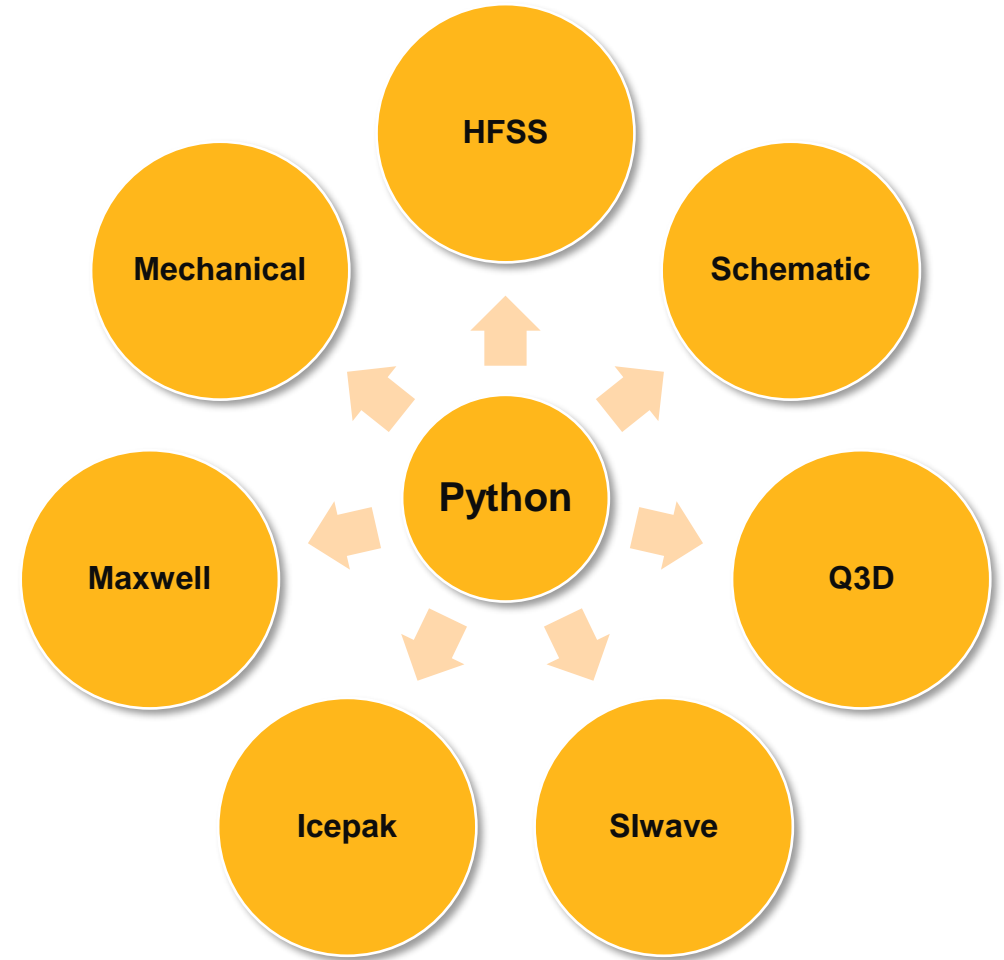


SciPy

pandas



matplotlib



# Python的優缺點

## 優點：

- 語法非常優雅，沒有像其他語言的大括號，分號等特殊符號。入手快，學習曲線低。
- 免費開源且支援多樣平台，如**Windows**、**MacOS**、**Linux**、**Andorid**、**iOS**等等。
- 不須編譯，直譯器把原始碼轉換成稱為位元組碼的中間形式，然後再把它翻譯成計算機使用的機器語言並執行。
- 支援程序導向設計及物件導向設計，這樣程式設計就更加靈活。
- 本身具有有豐富而且強大的庫。更有許多第三方庫用在**web**開發、爬蟲、科學計算等等。

## 缺點：

- 由於**Python**是解釋型語言，所以速度會比**C**、**C++**慢一些。不過對於使用者而言，機器上執行速度是可以忽略的。多數時候速度的差異不影響使用。

# 是不是學會Python就可以開發自動化程式了

並不是。除了基本語法之外，你還要熟悉以下這些函式庫，你才能透過Python控制函式庫來與AEDT互動

- 基本函式庫：**math, os, sys,...**
- 進階函式庫：**re, tkinter, matplotlib, os, sys, ...**
- AEDT函式庫：**Project, Design, Edit, Module**

熟悉的意思並不是要背下所有的函式以及語法，而是要知道函式庫提供了何種功能，可以解決何種問題。只要知道如何發問問題，便可以透過Google找到答案。



# / 可以從哪些步驟切入模擬自動化？

初學者可以考慮下面幾個方向：

- **前處理**：比方說是建模，設port，合成電路原理圖等等。
- **後處理**：比方說是匯出模擬資料並生成報告，輸出模擬資源等。
- **簡化操作**：將多個步驟連接成單一步驟，並連接到熱鍵。



## / 可以介紹"前處理"自動化嗎？

在執行模擬之前，我們必須先完成設計或3D建模，材料庫的設定，Port的設置等等。對於某些設計，上述所提到的工作相當的繁複。例如大型天線陣列。手動設置往往曠日廢時又容易出錯。此時便可以透過前處理自動化來加速完成。

前處理自動化主要是透過呼叫AEDT的API(Application Programming Interface)函式來完成建模或加入新的材料等等。一般開發前處理自動化程式的方式是先行錄製一段操作，再加以修改擴充。比方說是加入迴圈來重複其操作，以節省人為操作所耗費的時間。錄製的Python程式可以用一般文字編輯器加以修改，個人偏好Notepad++。

## / 可以簡單描述"前處理"自動化的流程嗎？

首先是錄製腳本，如果操作步驟繁複，可以分段錄製到不同的腳本當中。接下來將不同功能的程式碼包裝到不同函式當中，只留下需要的參數即可。每當包裝完一組程式碼，隨即編寫測試程式，如有必要可以用AddWarningMessage()將函式當中的變數輸出到AEDT的訊息視窗當中來協助除錯。最後串接所有程式之後進行完整功能測試。一切無誤，再視需要編寫使用者介面，並連結功能碼及使用者介面碼，完成開發工作。最後便是編寫使用說明並發布。



## / 可以說明"後處理"自動化嗎？

後處理指的是模擬後的資料處理。模擬的大量資料產出提供了設計的重要訊息，**AEDT**當中的各式報表可以支持大部分的需求，比方說是**XY**圖，表格，密度分布圖等等。但是一旦要分析的資料較為複雜，**Report**無法勝任時，我們便需要將模擬完成的原始資料匯出到第三方軟體(如**Matlab**, **Python**)來分析。

這部分牽涉了資料匯出、運算、視覺化、排版等等。需要自動化來簡化這一連串的瑣碎流程。舉例來說，一組複雜的**QFN**封裝完成模擬之後，生成了上百組的**RLC**等效電路模型，便可以透過腳本快速找出**RLC**值超出標準的腳位，而不需要人工逐條檢視紀錄。

# / Anaconda

Anaconda是一個免費開源的Python和R語言的發行版本，用於計算科學（資料科學、機器學習、巨量資料處理和預測分析），Anaconda致力於簡化包管理和部署。Anaconda的包使用軟體套件管理系統Conda進行管理。超過1200萬人使用Anaconda發行版本，並且Anaconda擁有超過1400個適用於Windows、Linux和MacOS的資料科學軟體包[Wiki]。



# / 完成一個AEDT自動化程式要多久時間？最大的困難是什麼？

簡單的程式數十行代碼即可完成，開發時間不過數十分鐘。複雜的幾千行，開發時間需要數天到數周。這跟工程師對程式設計的掌握度及AEDT的熟悉度有絕對的關係。

開發自動化程式需要使用到AEDT的函式庫，找到所需的函式並熟悉函式的輸出入是最花時間的地方。單單HFSS就提供了上千個函式，如果要透過人工從無到有開發自動化程式其工作量不敢想像。所幸AEDT可以讓使用者錄製操作步驟並輸出腳本，大幅縮短前處理自動化的開發時間。





/ “Progress is made by lazy men looking for easier ways to do things.” --Robert A. Heinlein



# 了解AEDT模擬自動化程 式的框架



## AEDT使用的是Iron Python，這和一般的Python有何差別？

一般的Python又稱為Cpython。兩者的語法是類似的，內建基本函式庫也相同。但是目前多數科學用函式庫如Matplotlib，Scipy及Numpy僅支援Cpython。Iron Python主要建構在微軟的.NET框架，好處是可以使用到WPF技術，在AEDT當中建立使用者介面相對容易。

比方說。兩者都可以讀寫EXCEL或PPT，但是各自使用不同的函式庫。此外，目前Cpython已來到3.8版，而Iron Python仍停留在2.7版。部分語法已有差異。總的來說。已經學會CPython的很容易就可以習慣IronPython。

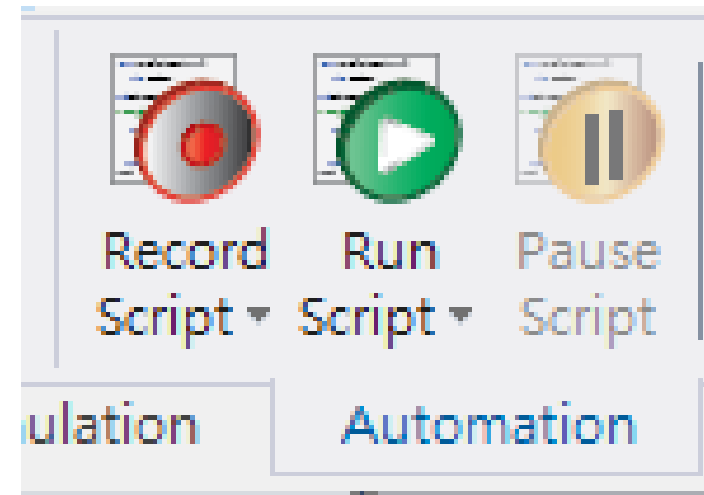
# 在AEDT執行自動化程式的方式有哪些？

主要方法有下列幾種：

- 從Automation執行script
- 從Toolkit執行script
- 熱鍵執行script
- ACT啟動GUI以執行script
- UDO(User Defined Outout) , 建立客製化方程式
- UDP(User Defined Primitive) , 建立客製化3D模型
- 將資料輸出到外部用Cpython處理(Anaconda)
- 透過批次檔控制AEDT完成前處理及後處理

## / 透過Run Script來執行腳本

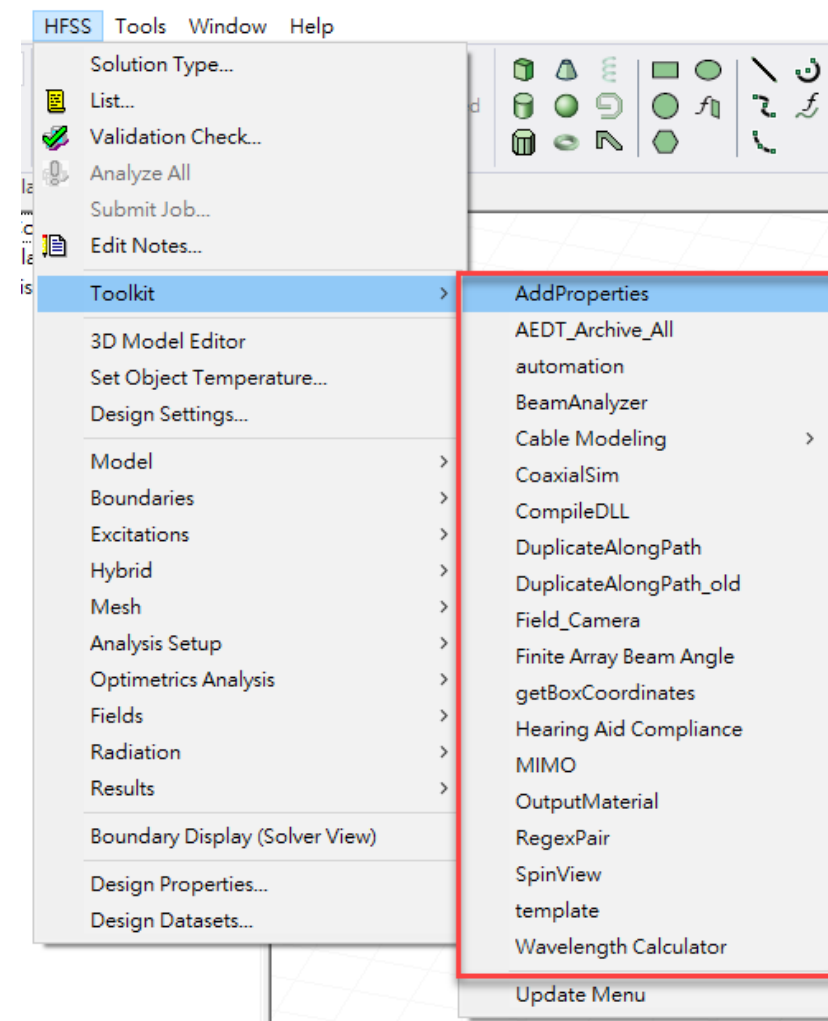
這是最常用來執行腳本的方法，也常用在腳本開發及除錯過程。在Automation類別找到並按下Run Script並選擇要執行Python的腳本即可。



# 通過toolkit來執行腳本

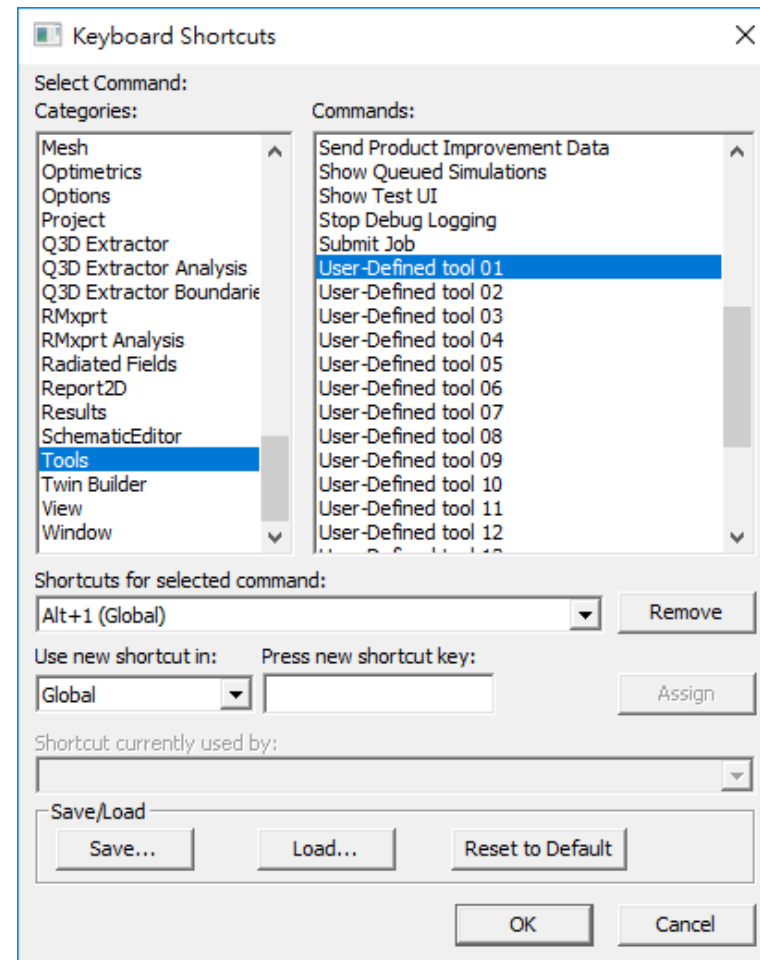
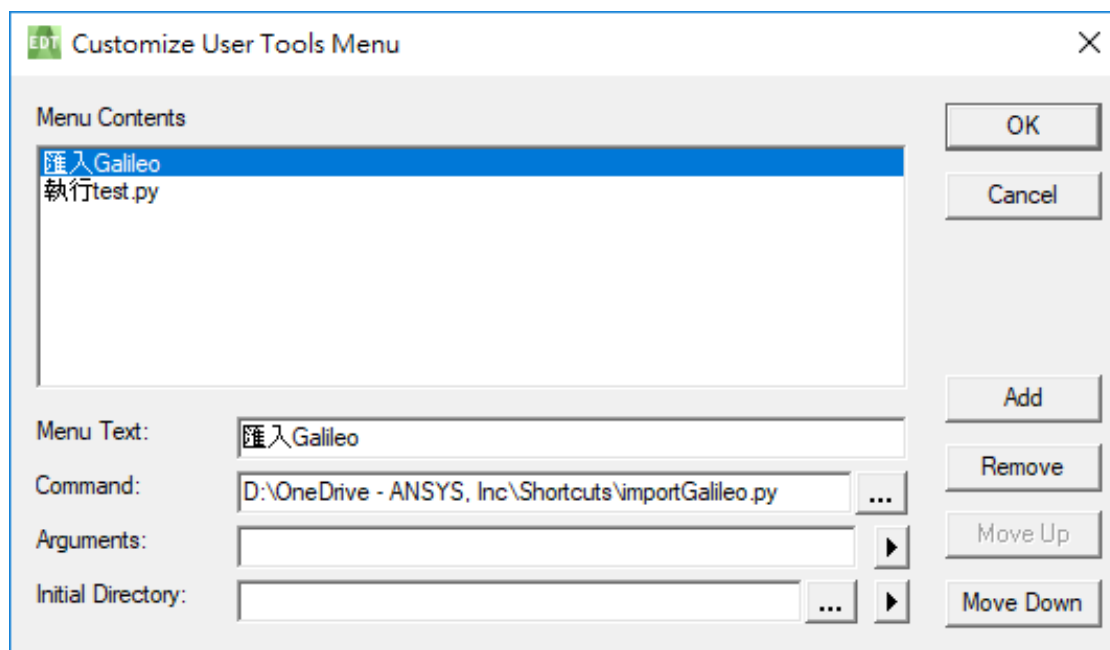
一旦完成腳本開發、便可以將腳本放置在特定目錄之下，之後就可以在Toolkit當中選取並執行。

Toolkit按照產品類別分類，只有符合該產品的腳本才會被顯示在Toolkit選單當中。



# 熱鍵執行腳本

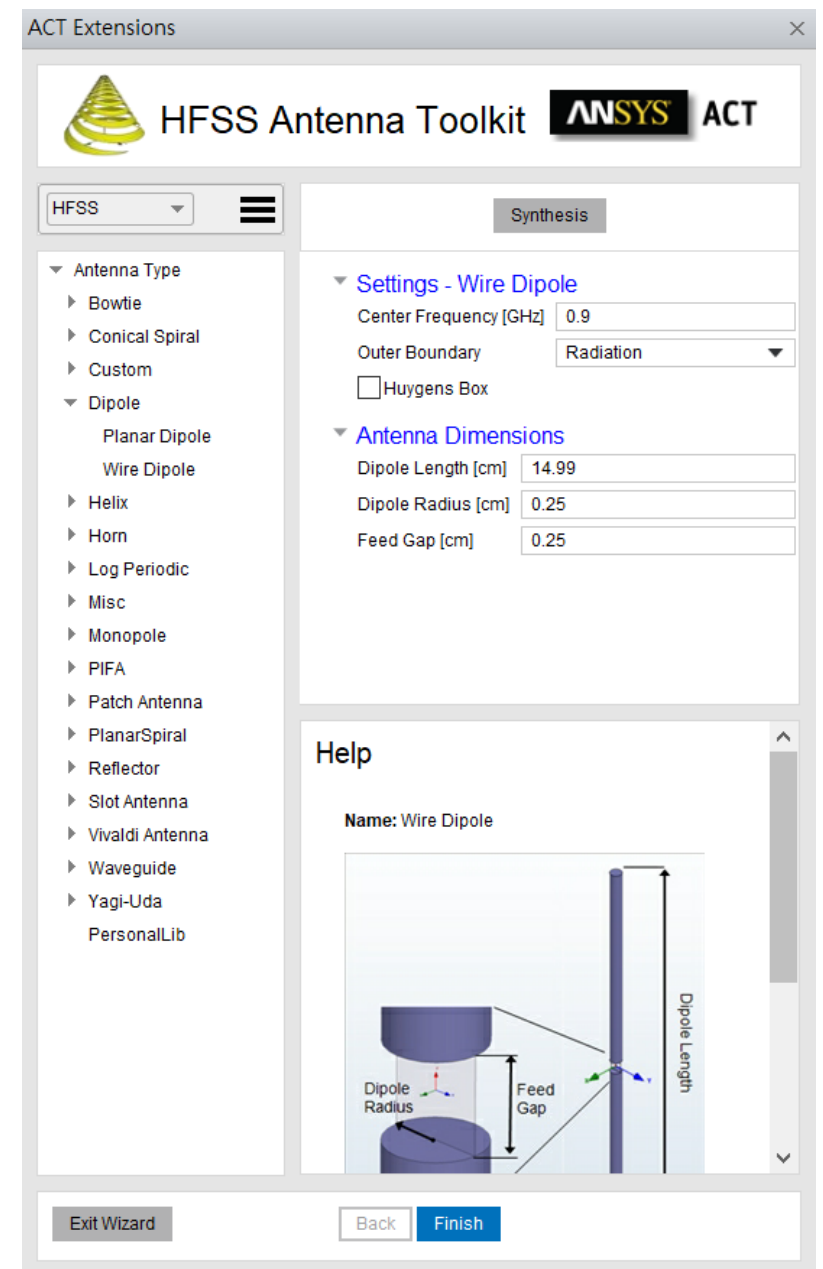
- 在External tool設定腳本及對應的名稱，名稱便會顯示在Tool選單下層的選項。
- 可以在Keyboard Shortcuts當中連結到熱鍵，以方便執行。



# ACT(ANSYS Customization Toolkit)

ACT是一種外掛套件的技術，在ANSYS的軟件上可以外掛ACT套件來提供額外的功能。由於軟體工具不可能滿足所有的需求，工程師便可以根據自己的需求編寫程式碼在ACT框架當中，來達到特定功能的需求。ACT的開發工具已經整合在AEDT環境當中，不需要額外license，也不需要另外安裝。

舉例來說，HFSS當中提供了Antenna ACT，使用者只要輸入頻段並選擇天線結構，ACT便自動生成天線3D模型，省去使用者建立模型的功夫。本質也是自動化的一種。與Script不同的是：可提供加密保護，支持簡單的GUI介面，可上架銷售。

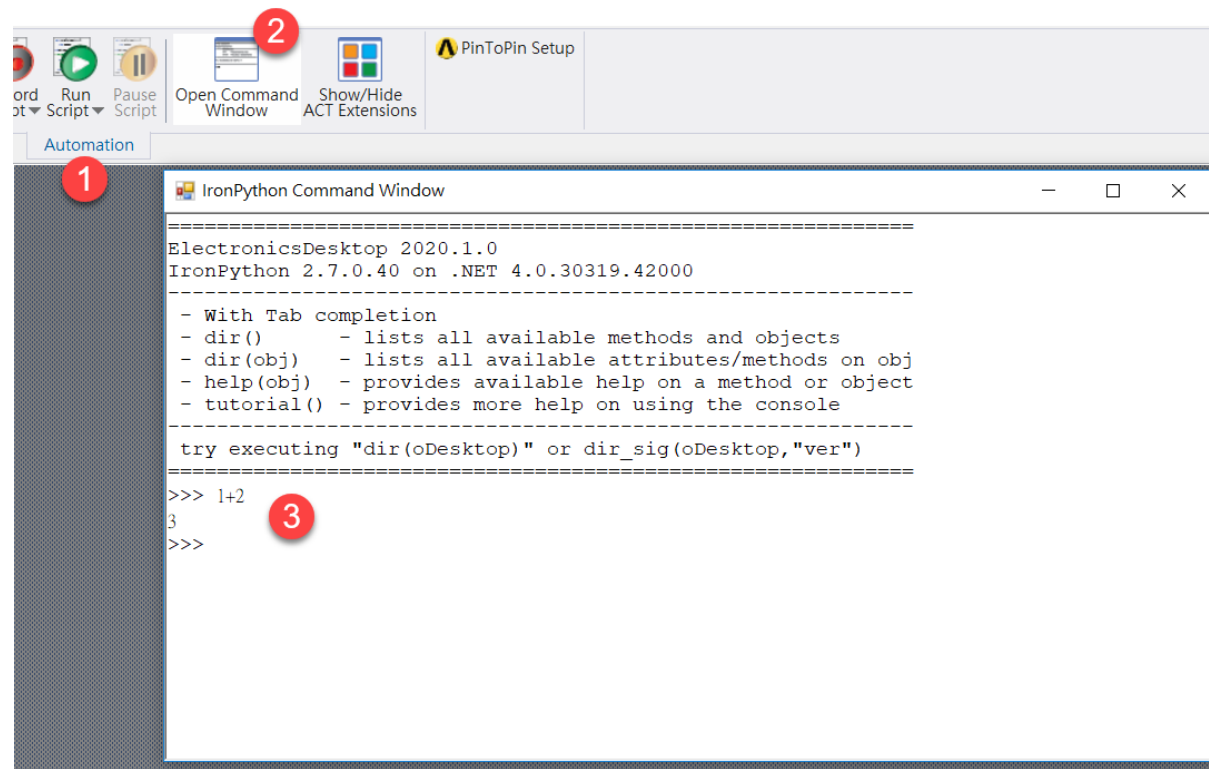


# AEDT編程環境

# / IronPython命令列視窗

使用者可以在命令列視窗輸入指令，執行之後即時返回結果。比方說輸入1+2，按下Enter鍵，即可得到答案3。

設定過的變數名會儲存起來，直到關閉視窗。通常我們並不會在命令列視窗編程，命令列視窗僅適合做一些函式功能測試。



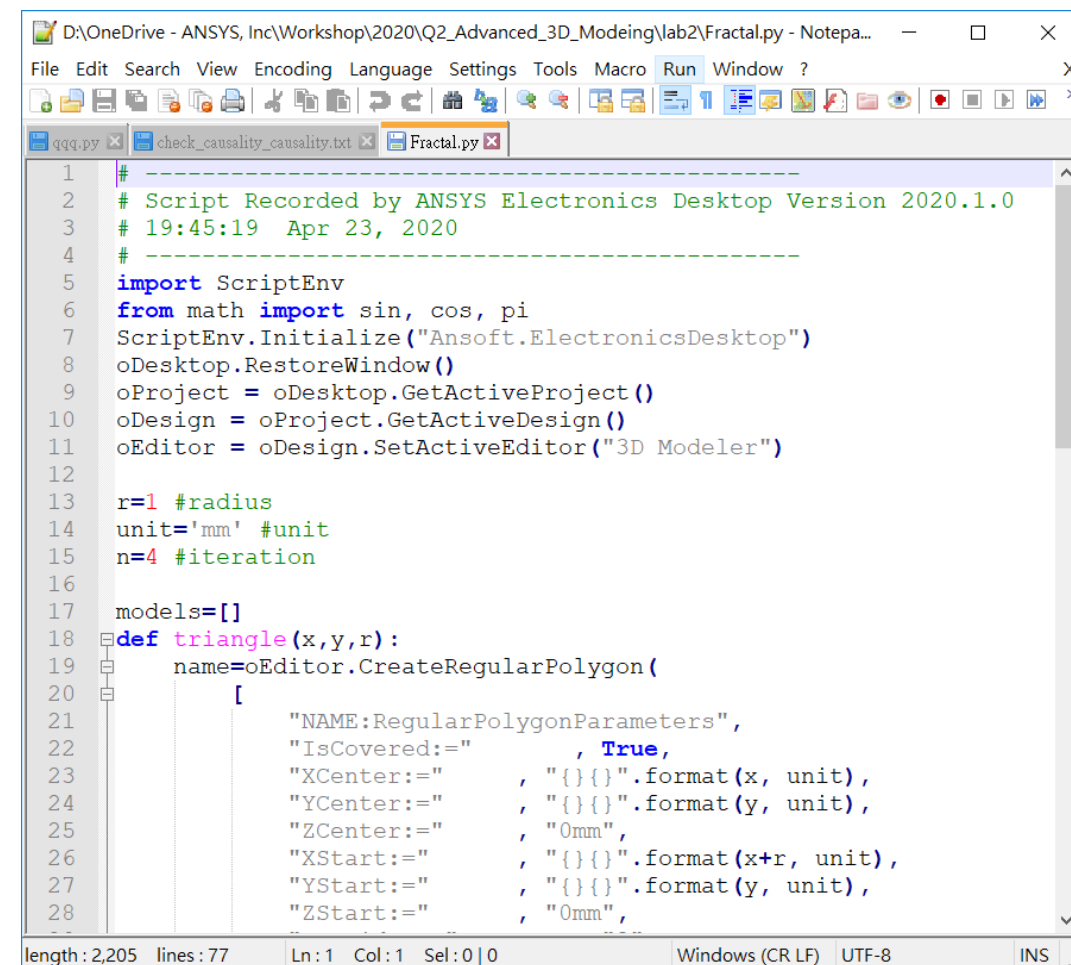


# 使用Notepad++來編輯程式

Notepad++是一套免費，輕量且功能強大的純文本編輯器，開啟或關閉都很快速。可支援上百種不同檔案格式關鍵字的顏色標示。由於編程過程中不只會編輯python，也有機會讀取像是CSV,xml, html等等，Notepad++都可以很好的處理。搜尋及取代功能都做得相當不錯。是在做前處理腳本編輯的首選。

下載連結：

- <https://notepad-plus-plus.org/downloads/>

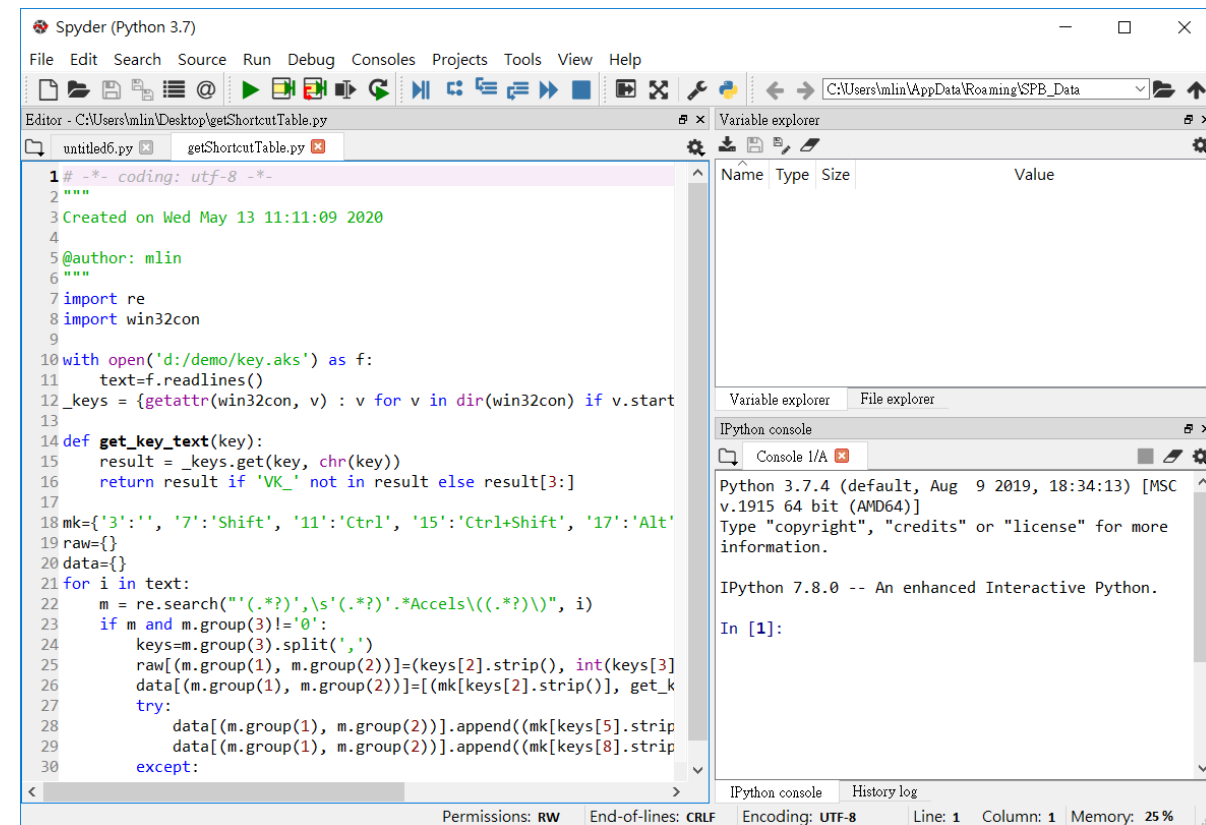


The screenshot shows the Notepad++ application window with a file named 'Fractal.py' open. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Window, and Help. The toolbar contains various icons for file operations and editing. The code is a Python script that initializes the ANSYS ScriptEnv and creates a regular polygon. The code is syntax-highlighted: comments are green, keywords are blue, and strings are red. The script includes imports for math functions (sin, cos, pi) and the ScriptEnv module. It initializes the ScriptEnv, gets the active project and design, and sets the active editor to '3D Modeler'. It then defines a function 'triangle' that creates a regular polygon with specified parameters. The status bar at the bottom shows 'length: 2,205 lines: 77', 'Ln: 1 Col: 1 Sel: 0 | 0', 'Windows (CR LF)', 'UTF-8', and 'INS'.

```
1 # -----
2 # Script Recorded by ANSYS Electronics Desktop Version 2020.1.0
3 # 19:45:19 Apr 23, 2020
4 # -----
5 import ScriptEnv
6 from math import sin, cos, pi
7 ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
8 oDesktop.RestoreWindow()
9 oProject = oDesktop.GetActiveProject()
10 oDesign = oProject.GetActiveDesign()
11 oEditor = oDesign.SetActiveEditor("3D Modeler")
12
13 r=1 #radius
14 unit='mm' #unit
15 n=4 #iteration
16
17 models=[]
18 def triangle(x,y,r):
19     name=oEditor.CreateRegularPolygon(
20         [
21             "NAME:RegularPolygonParameters",
22             "IsCovered:=", True,
23             "XCenter:=", "{}{}".format(x, unit),
24             "YCenter:=", "{}{}".format(y, unit),
25             "ZCenter:=", "0mm",
26             "XStart:=", "{}{}".format(x+r, unit),
27             "YStart:=", "{}{}".format(y, unit),
28             "ZStart:=", "0mm",
```

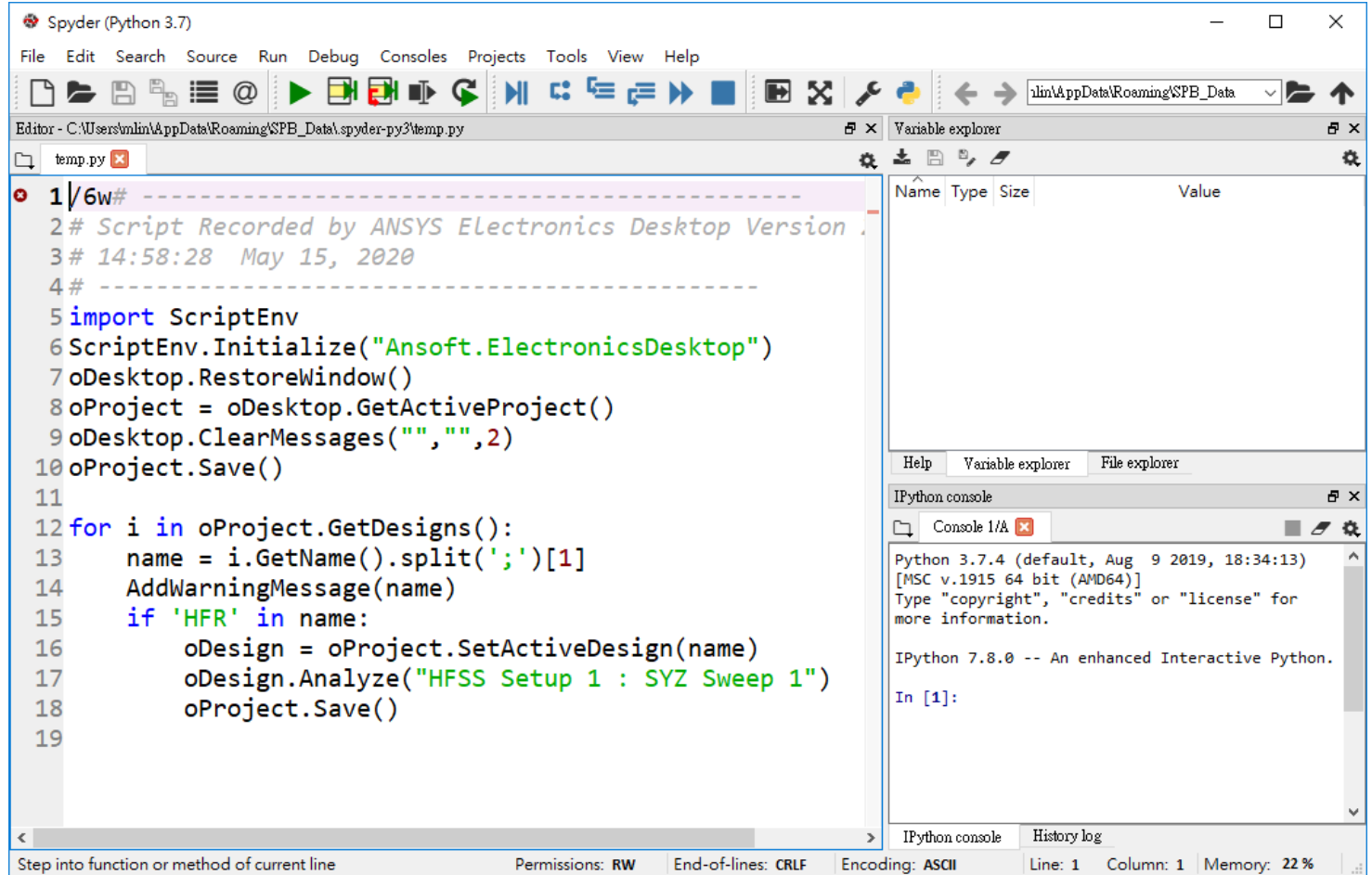
# Anaconda Spyder (Data Post Processing)

這個仿Matlab的編程環境不只提供了程式碼編輯器，還包括了變量視窗，命令列視窗及完整的除錯功能。可以讓使用者快速的匯入csv檔，檢視變數，產生圖片。是一套專門用於科學運算的程式碼編輯器。提供了功能可以簡單檢視物件所支援的函數。相當適合支援後處理程式的設計工作。



# / Anaconda Spyder

- Numpy and Scipy
- Matplotlib
- Spyder
  - Cell Definition
  - Dynamic Syntax Checker
  - Variable Explorer
  - Intelligence
  - Debugger
  - Data Import
  - ...



# Spyder的優點

市面上Python的整合發展環境(IDE)有數十種，各有各的優缺點。但是如果說到最適合科學工程應用的IDE絕對是Anaconda的Spyder。首先，spyder編輯器支援IntelliSense。IntelliSense包含一些功能的程式碼完成輔助工具：列出成員、參數資訊、快速諮詢和自動完成文字。免去工程師查找的時間。變數顯示視窗將所有變量的型別，大小及數值用不同色塊區分開來，一目了然。支援斷點，及步階執行方便除錯。支援Cell，不須重複執行耗時的計算工作。除此之外，其他像是自動標示語法錯誤，支援資料檔匯入等等也都很好的支援。

由於Spyder的介面類似Matlab，又內建了Numpy，Scipy，Matplotlib等科學工程常用的函式庫，在資料分析領域已經成為多數工程師的首選。

# Jupyter Notebook

Yammer - 5G mmWave x | 頻道資訊主頁 - YouTube x | 存取面板應用程式 x | 常用代碼片段 - Jupyter x | (2) AEDT 模擬自動化 | F x

localhost:8888/notebooks/常用代碼片段.ipynb

應用程式 私人 Ansys 共享 編程 函式 Slwave 教材 資源 提交 工作 20R1 20R2 參考 Lumerical

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run

### Pickle檔案讀取

In [16]:

```
1 import pickle
2
3 data = {'A': [1, 2, 3], 'B': [4, 5, 6]}
4
5 with open('d:/demo/test.pickle', 'wb') as f:
6     pickle.dump(data, f)
7
8 with open('d:/demo/test.pickle', 'rb') as f:
9     data = pickle.load(f)
10
11 print(data)
```

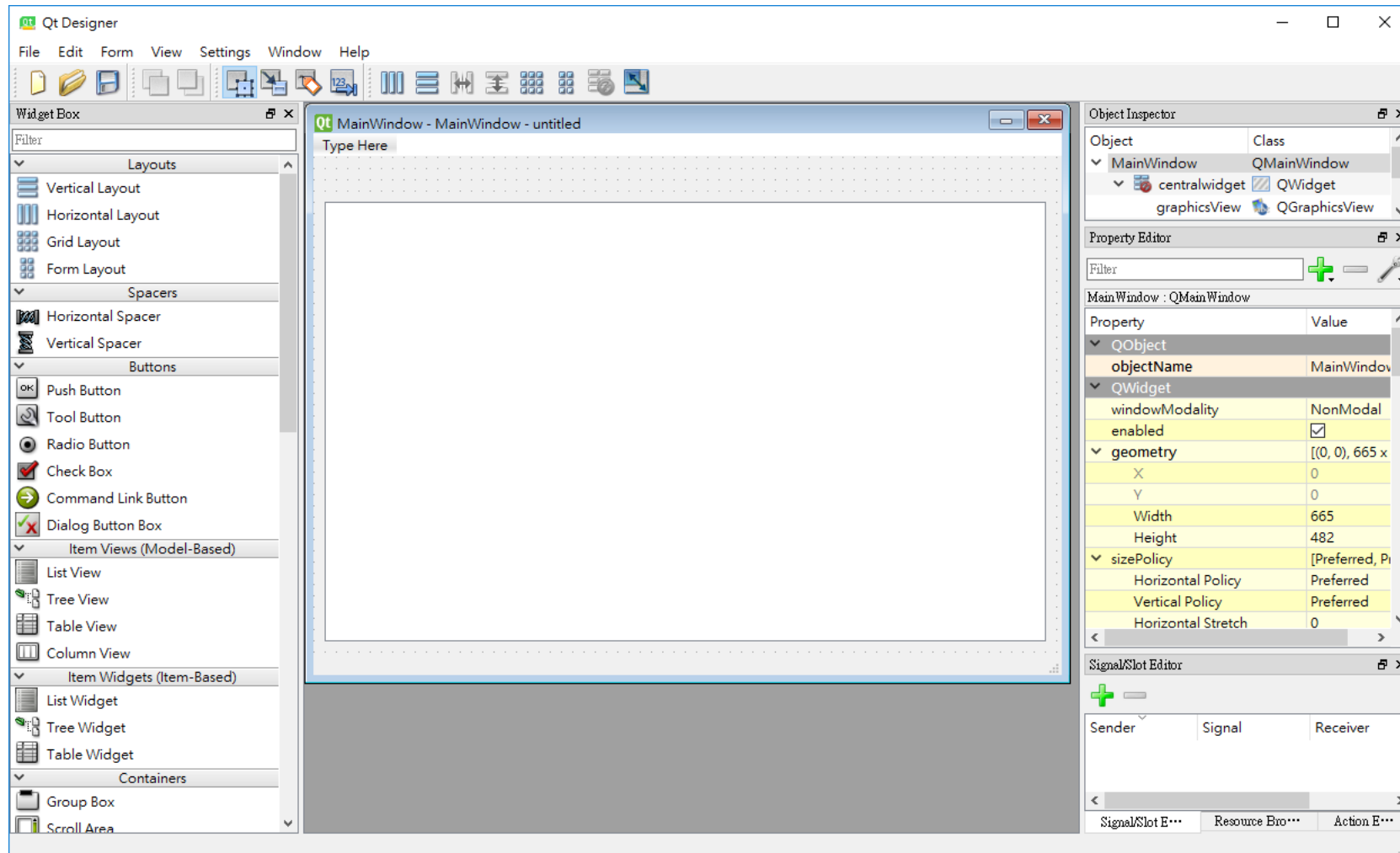
```
{'A': [1, 2, 3], 'B': [4, 5, 6]}
```

### Json檔案讀取

In [17]:

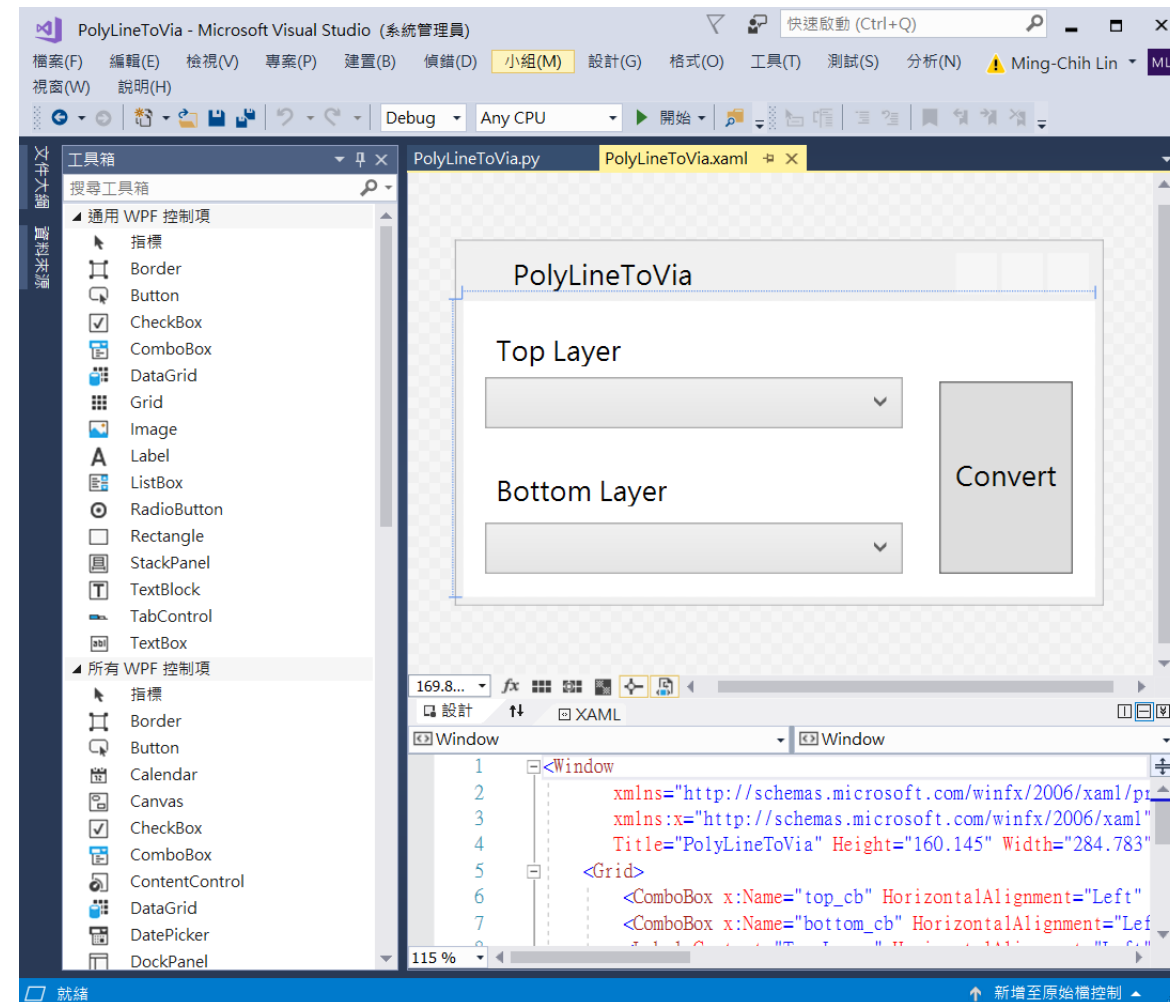
```
1 import json
2
3 data = {'A': [1, 2, 3], 'B': [4, 5, 6]}
4
5 with open('d:/demo/test.json', 'w') as f:
```

# QT Designer



# Visual Studio (GUI Design Environment)

號稱是地表最強的文字編輯器。然而其功能過於強大，對初學者來說難以駕馭。主要用來建立windows環境的操作介面，其支援了完整的視窗控鍵，像是按鈕，滑桿，下拉式選單，表列等等。只要使用拖放便可以輕鬆的建立圖形化操作介面及連結事件。顏色，位置，大小都可以輕鬆調整。



# Python基本語法簡介(1/4)



 <https://www.tutorialspoint.com/python/index.htm>

 Python - Basic Syntax

 Python - Variable Types

# / 輸出訊息指令：print() & AddWarningMessage()

print()

- 輸出到Command Window

AddWarningMessage()

- 輸出到Message Manager

oDesktop.ClearMessages("", "", 2)

- 清空Message Manager



The screenshot shows the IronPython Command Window interface. The main window displays the IronPython version (2.7.0.40) and .NET version (4.0.30319.42000). Below this, a list of commands is shown: dir(), dir(obj), help(obj), and tutorial(). The command prompt shows the execution of print("Hello World!") and AddWarningMessage("Hello World!"). The output "Hello World!" is displayed. Below the command window, the Message Manager is visible, showing a warning message "Hello World!" with a yellow warning icon.

```
IronPython Command Window

=====
ElectronicsDesktop 2020.1.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
=====

- With Tab completion
- dir()          - lists all available methods and objects
- dir(obj)       - lists all available attributes/methods on obj
- help(obj)      - provides available help on a method or object
- tutorial()     - provides more help on using the console
=====

try executing "dir(oDesktop)" or dir_sig(oDesktop,"ver")
=====

>>> print("Hello World!")
Hello World!
>>> AddWarningMessage("Hello World!")
>>> |

Message Manager
*Global - Messages
[Warning] Hello World!
```

# / Python Keyword

- Keyword(保留字)不能用於變數或函數命名。

種類	關鍵字
常數	<code>False</code> <code>None</code> <code>True</code>
運算子	<code>and</code> <code>del</code> <code>in</code> <code>is</code> <code>lambda</code> <code>not</code> <code>or</code>
簡單陳述	<code>as</code> <code>assert</code> <code>break</code> <code>continue</code> <code>from</code> <code>global</code> <code>import</code> <code>nonlocal</code> <code>pass</code> <code>raise</code> <code>return</code> <code>yield</code>
複合陳述	<code>else</code> <code>elif</code> <code>except</code> <code>finally</code> <code>for</code> <code>if</code> <code>try</code> <code>while</code> <code>with</code>
定義	<code>class</code> <code>def</code>

# 縮排

Python透過縮排來組織程式碼並提高可讀性。在控制結構的複合陳述會需要用到：

- **if ...: else:**
- **for ... in ...:**
- **def foo():**
- **class foo():**
- **with open():**

縮排規則如下：

- 「:」之後的下一行必須縮排，除非程式碼緊接在「:」之後的同一行且只有一行
- 慣例縮排的空白數為4，同一個縮排等級的程式碼必須使用相同的空白數進行縮排
- 違反縮排規則會出現**IndentationError**

# 註解

- 在電腦語言中，註解是電腦語言的一個重要組成部分，用於在原始碼中解釋代碼的功用，可以增強程式的可讀性，可維護性。
- 如果沒辦法使用少量的單字為變數命名，或是用適當的動詞作為函式的命名，就得用註解完整描述程式碼。換句話說，如果可以透過命名解釋意圖，則不用註解釋釋程式碼的意圖。
- 請使用英文註解。中文註解在AEDT執行時會產生編碼錯誤使得程式無法執行。

- 單行

```
#This is comment
```

- 多行：

```
""" comment1  
comment2  
comment3"""
```

# 變數型別與宣告

## Python基本資料型別

- Numbers:  $x = 3$ ,  $x = 1.34$ ,  $x = 3+4j$
- String:  $x = \text{"Hello World"}$ ,  $x = \text{'Hello World'}$ ,  $x = \text{"'Hello", he said"}$ ,  $x = \text{'3.14'}$
- List:  $x = [1,2,3]$ ,  $x = [\text{'A'}, \text{'B'}, \text{'C'}]$
- Tuple:  $x = (\text{'John'}, 37, \text{"Male"})$
- Dictionary:  $x = \{\text{'John'}: (37, \text{'Male'}), \text{'Mary'}: (18, \text{'Female'}) \}$
- Boolean: True, False
  - 判斷為False
    - None
    - 數字0
    - 空字串及空資料，如：`"`、`[]`、`()`、`{}`
  - 其他判斷為True

# Python運算子類型

可分成下面幾類:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators(自動化不常用)
- Membership operators
- Bitwise operators(自動化不常用)



# 算術運算子(Arithmetic operators)

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$



# 指派運算子(Assignment operators)

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

# 比較運算子(Comparison operators)

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

# 成員運算子(Membership operators)

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

# 邏輯運算子(Logical Operator)

Operator	Description	Example
and	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
or	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>

# 專題預備

# 為何課程要完成專題？如果遇到困難該怎麼辦？

只有學以致用，才能真正的落實學習到的知識。透過專題的演練，可以認知到知識不足的部分，並拓展思維的廣度。這是專題實作的意義。此外，講師會在課程期間協助個人專案的開發。透過討論及腦力激盪來完完成專案。



# / 如何制定自動專題題目(1/2)

制定自動化專題可以考慮下列幾點

- 作業系統：**Windows**或**Linux**或兩者
- 環境：**HFSS**、**Q3D**、**Schematic**、**Maxwell...**
- 工作：前處理或後處理
- 輸入及操作介面
- 輸出結果



## 如何制定自動化專題題目(2/2)

建議可以從平常操作所遇到的問題著手，找出工作流程的瓶頸。除此之外。要衡量自己的編程能力：

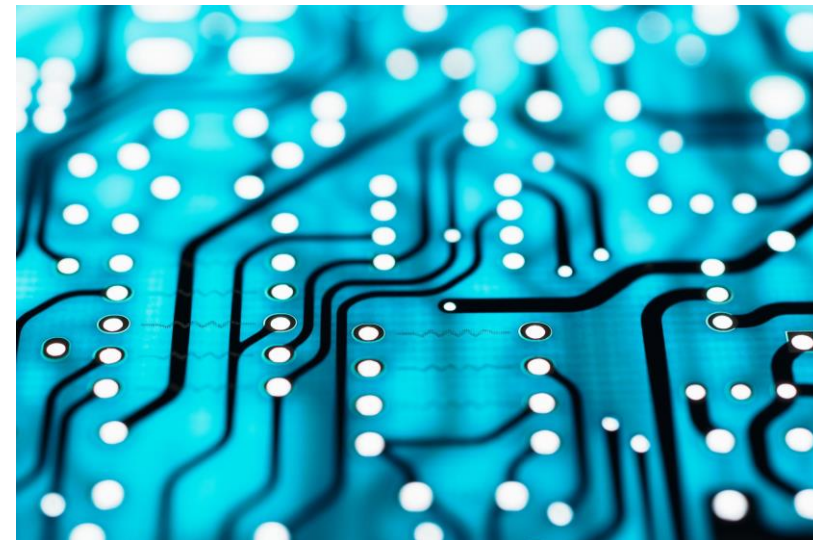
- 平常沒有做編程工作的學員建議選擇前處理作為專題題目。透過修改錄製好的腳本來逐漸改進編程技巧並建立自信心。
- 平時已有編程經驗的可以試著挑戰後處理，學習如何處理上百萬筆的資料並輸出報表。後處理技術也可以用在其他領域。
- 前處理後處理都熟稔的學員可以挑戰GUI編程或是串連多個環境，像是將HFSS/Q3D結果輸出到原理圖當中並接上電路元件做模擬並輸出報告到ppt檔。
- 如果上面都難不倒你，不妨來挑戰ACT。ACT整合了介面設計，也可以串聯多平台。寫得好還可以上架ANSYS Store銷售。



# ／ 自動化專題舉例

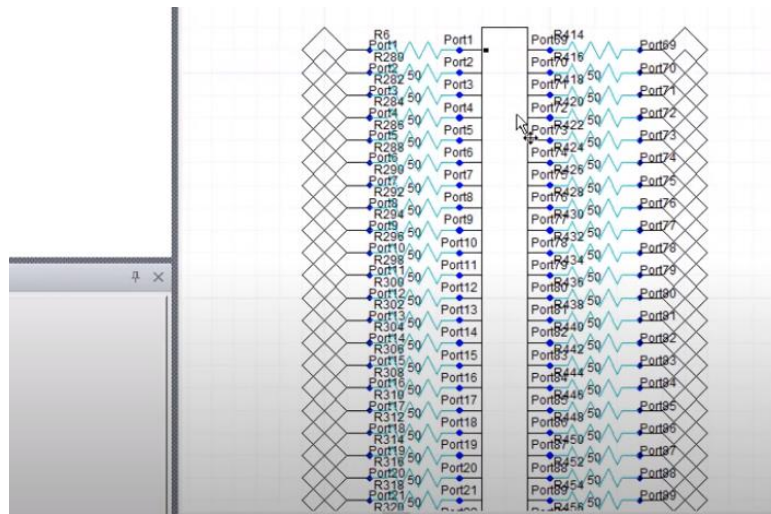
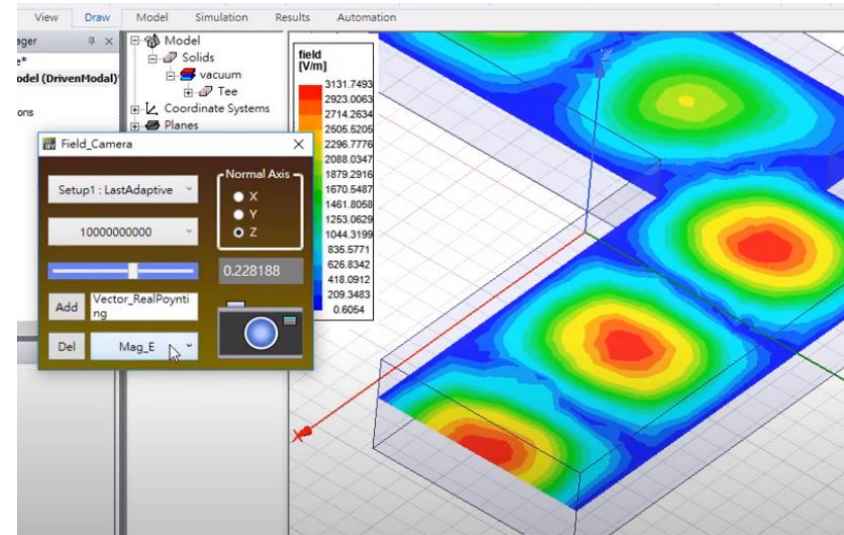
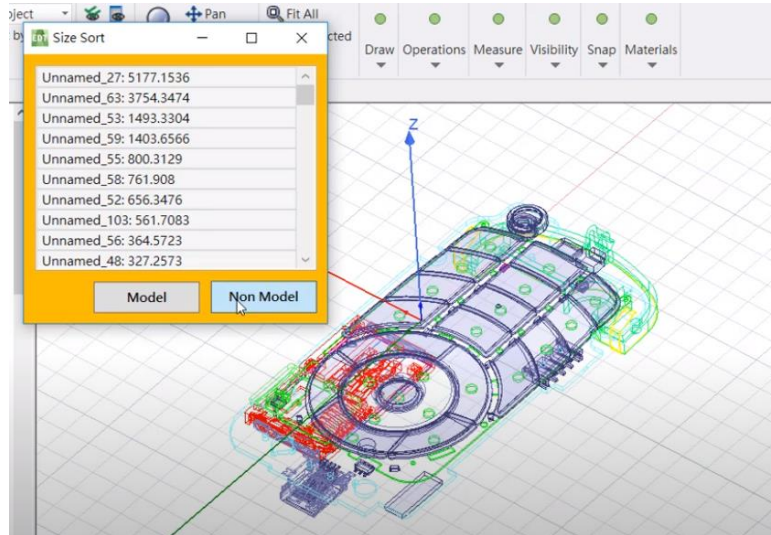
回想自己在軟體操作時耗費時間最多的步驟，比方說：

- 沿著弧形傳輸線兩側佈建via
- 每次建立新的設計時都要加入同樣的材料與模擬設定
- 在所有的pad上設置port
- 輸出每一組走線的頻寬曲線圖
- 將數十個port的S參數模型接上電路
- 計算每一組訊號眼高及眼寬並建立統計分布曲線
- 其它



總而言之，任何你覺得煩瑣的操作，都是自動化腳本可以發揮的空間。

# 模擬自動化案例介紹



# Homework

- 熟悉AEDT模擬自動化專班的上課教材及網路資源
- 加入臉書討論群AEDT模擬自動化
- 瀏覽推薦Python學習網站
- 找出一個自動化專題題目並定義程式的輸入與輸出。之後三周上課時間每位學員要輪流上台分享自動化專題題目。
  - 作業系統：Windows或Linux或兩者
  - 環境：HFSS、Q3D、Schematic、Maxwell...
  - 工作：前處理或後處理
  - 輸入及操作介面
  - 輸出結果