

Week 5-操作介面設計及AEDT 函式庫簡介



範例解說：使用QT Designer 設計複數計算機

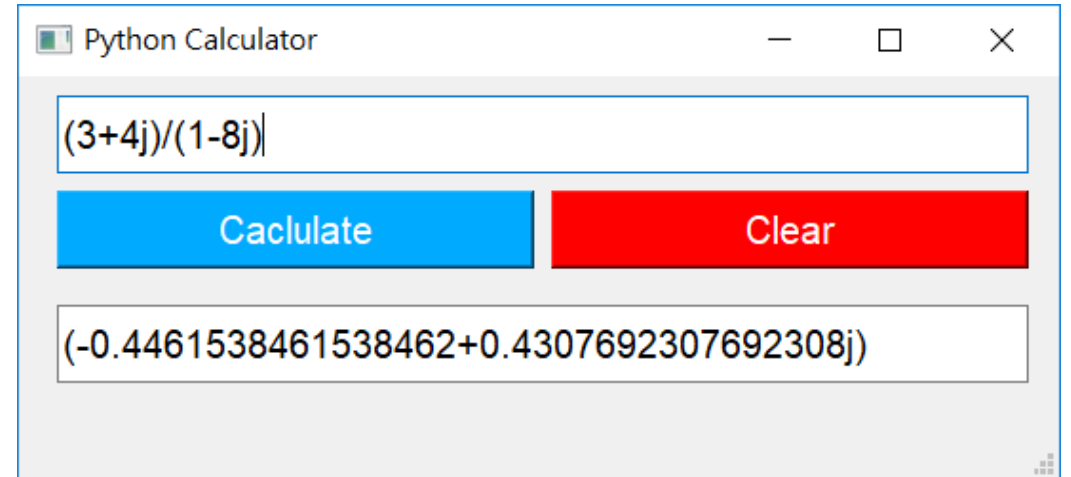
定義需求

- 需求
 - 提供**GUI**供使用者執行複數計算
- 輸入
 - 含有複數運算式
- 輸出
 - 正確計算結果

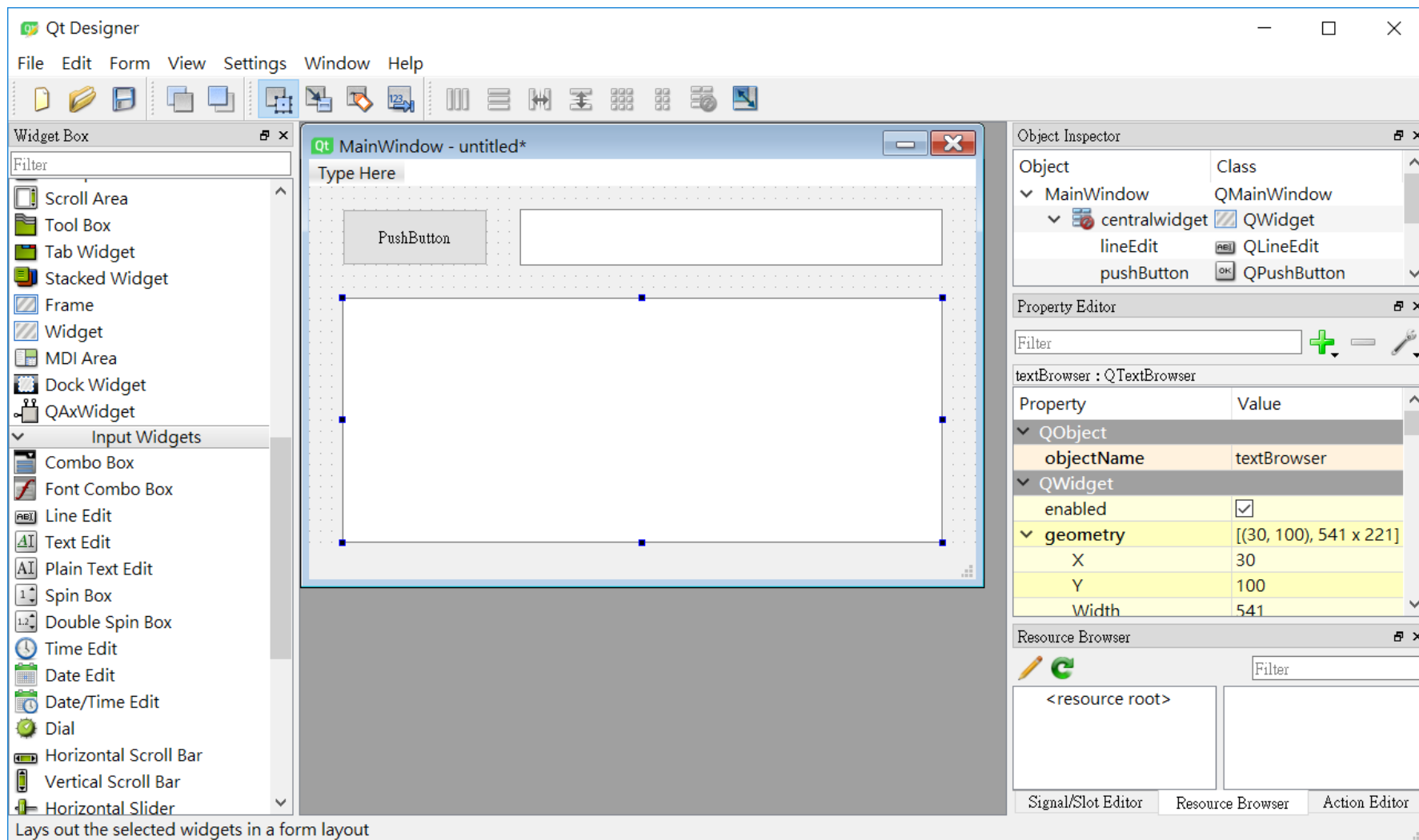


設計流程

- 開啟QT Designer : C:\ProgramData\Anaconda3\Library\bin\designer.exe
- 拖拉控件至主視窗
- 修改控件屬性(顏色及字形)
- 建立交互
- 儲存UI檔
- 開啟Python樣板，並修改UI檔路徑
- 建立事件連結
- 建立函式
- 測試工具
- 建立批次檔



QT Designer UI設計工具



設計好的GUI存成.UI檔(為XML格式)

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3    <class>MainWindow</class>
4    <widget class="QMainWindow" name="MainWindow">
5      <property name="geometry">
6        <rect>
7          <x>0</x>
8          <y>0</y>
9          <width>547</width>
10         <height>211</height>
11        </rect>
12      </property>
13      <property name="windowTitle">
14        <string>Python Calculator</string>
15      </property>
16      <widget class="QWidget" name="centralwidget">
17        <widget class="QLineEdit" name="formula_le">
18          <property name="geometry">
19            <rect>
20              <x>20</x>
21              <y>10</y>
22              <width>511</width>
23              <height>41</height>
24            </rect>
25          </property>
26
27          <property name="styleSheet">
28            <string notr="true">font: 75 12pt "Arial";</string>
29          </property>
30        </widget>
31        <widget class="QPushButton" name="calculate_bt">
32          <property name="geometry">
```

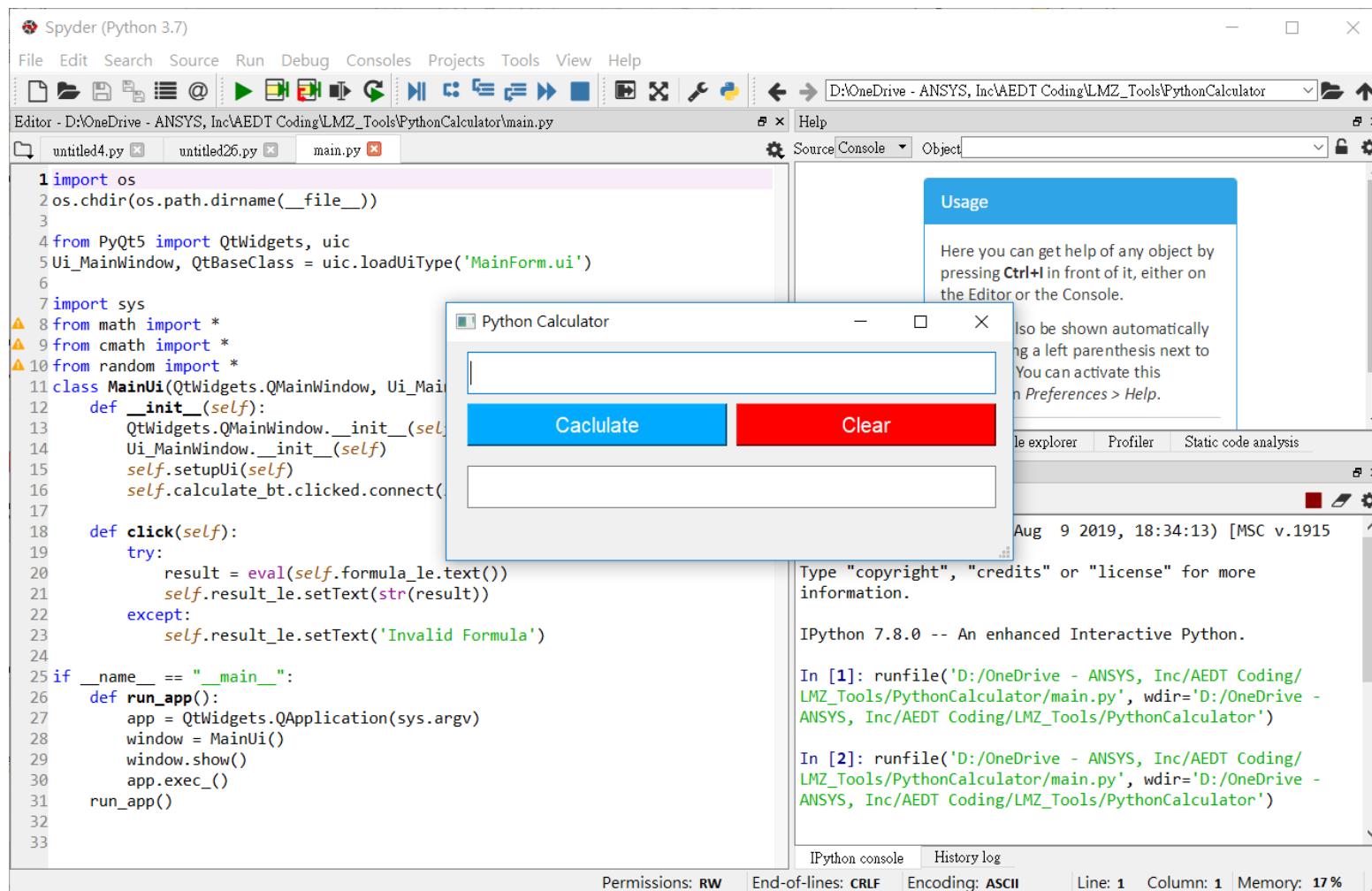
/ 在UI_Template.py連結.ui檔



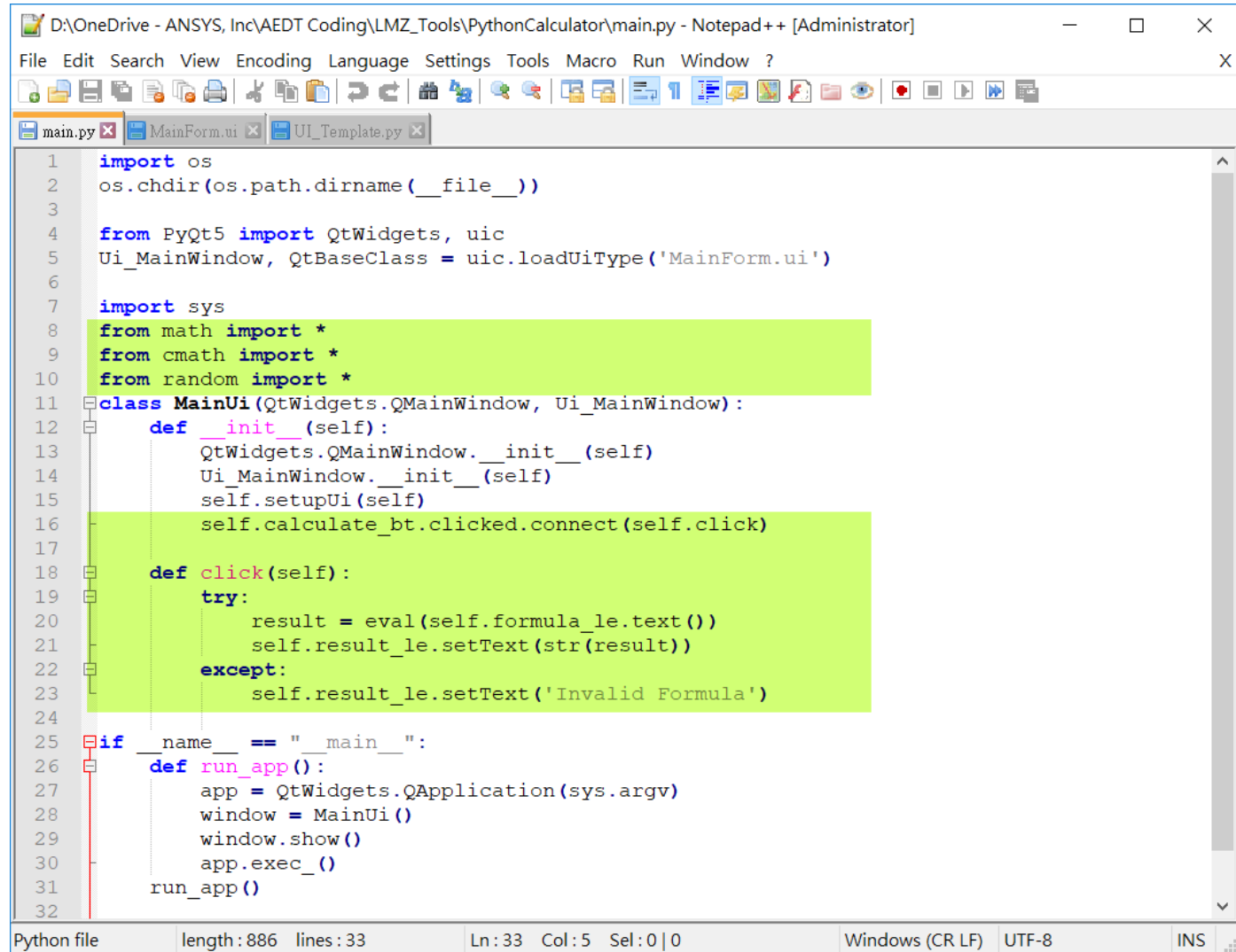
```
1  import os
2  os.chdir(os.path.dirname(__file__))
3
4  from PyQt5 import QtWidgets, uic
5  Ui_MainWindow, QtBaseClass = uic.loadUiType('MainForm.ui')
6
7  import sys
8
9  class MainUi(QtWidgets.QMainWindow, Ui_MainWindow):
10     def __init__(self):
11         QtWidgets.QMainWindow.__init__(self)
12         Ui_MainWindow.__init__(self)
13         self.setupUi(self)
14
15
16  if __name__ == "__main__":
17     def run_app():
18         app = QtWidgets.QApplication(sys.argv)
19         window = MainUi()
20         window.show()
21         app.exec_()
22     run_app()
23
```

修改UI名稱

執行.py檔測試GUI連結是否正常開啟



加入功能代碼

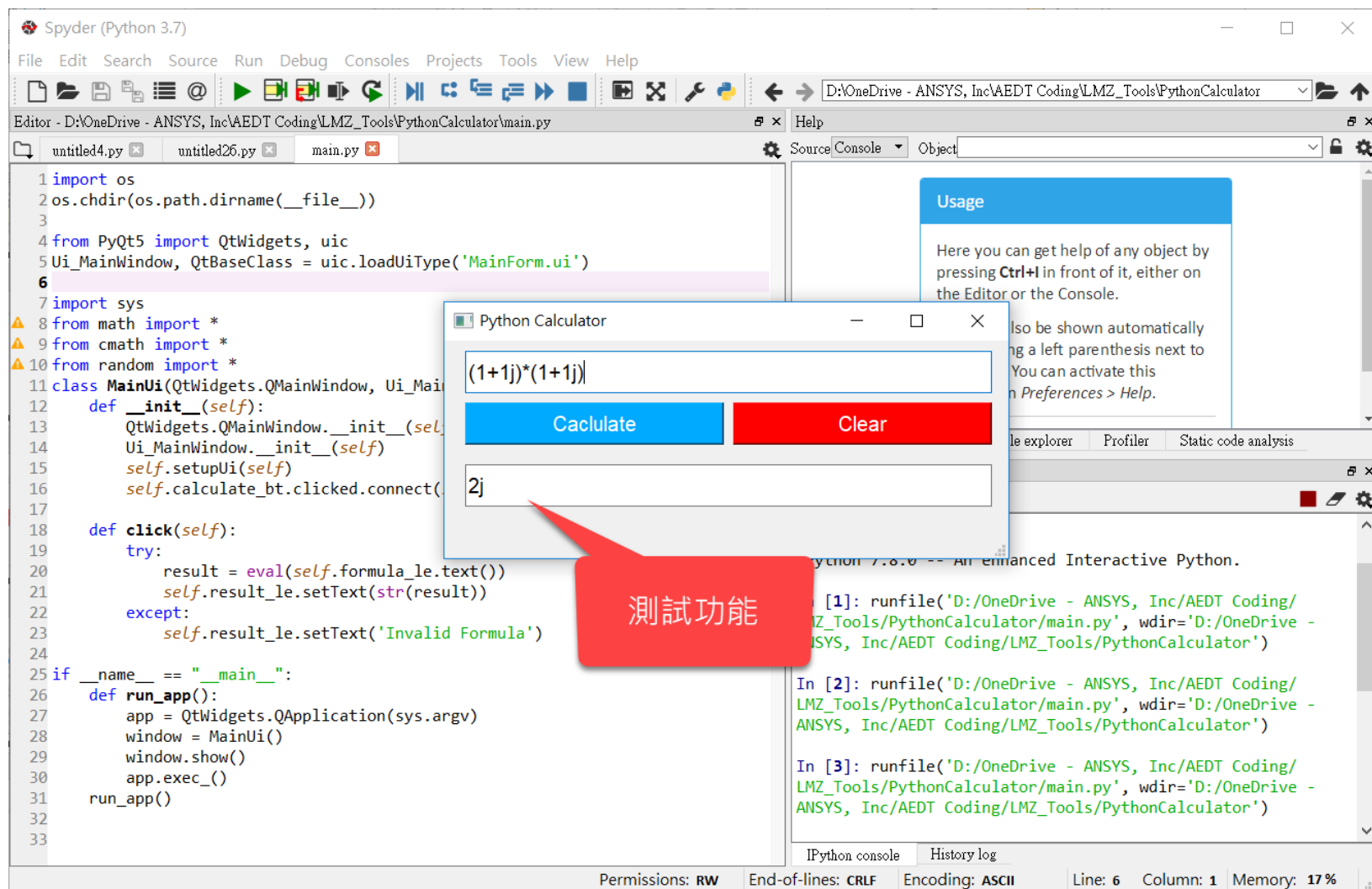


```
D:\OneDrive - ANSYS, Inc\AEDT Coding\LMZ_Tools\PythonCalculator\main.py - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Window ?

main.py MainForm.ui UI_Template.py
1 import os
2 os.chdir(os.path.dirname(__file__))
3
4 from PyQt5 import QtWidgets, uic
5 Ui_MainWindow, QtBaseClass = uic.loadUiType('MainForm.ui')
6
7 import sys
8 from math import *
9 from cmath import *
10 from random import *
11 class MainUi(QtWidgets.QMainWindow, Ui_MainWindow):
12     def __init__(self):
13         QtWidgets.QMainWindow.__init__(self)
14         Ui_MainWindow.__init__(self)
15         self.setupUi(self)
16         self.calculate_bt.clicked.connect(self.click)
17
18     def click(self):
19         try:
20             result = eval(self.formula_le.text())
21             self.result_le.setText(str(result))
22         except:
23             self.result_le.setText('Invalid Formula')
24
25 if __name__ == "__main__":
26     def run_app():
27         app = QtWidgets.QApplication(sys.argv)
28         window = MainUi()
29         window.show()
30         app.exec_()
31     run_app()
32
```

Python file length: 886 lines: 33 Ln: 33 Col: 5 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

測試功能代碼



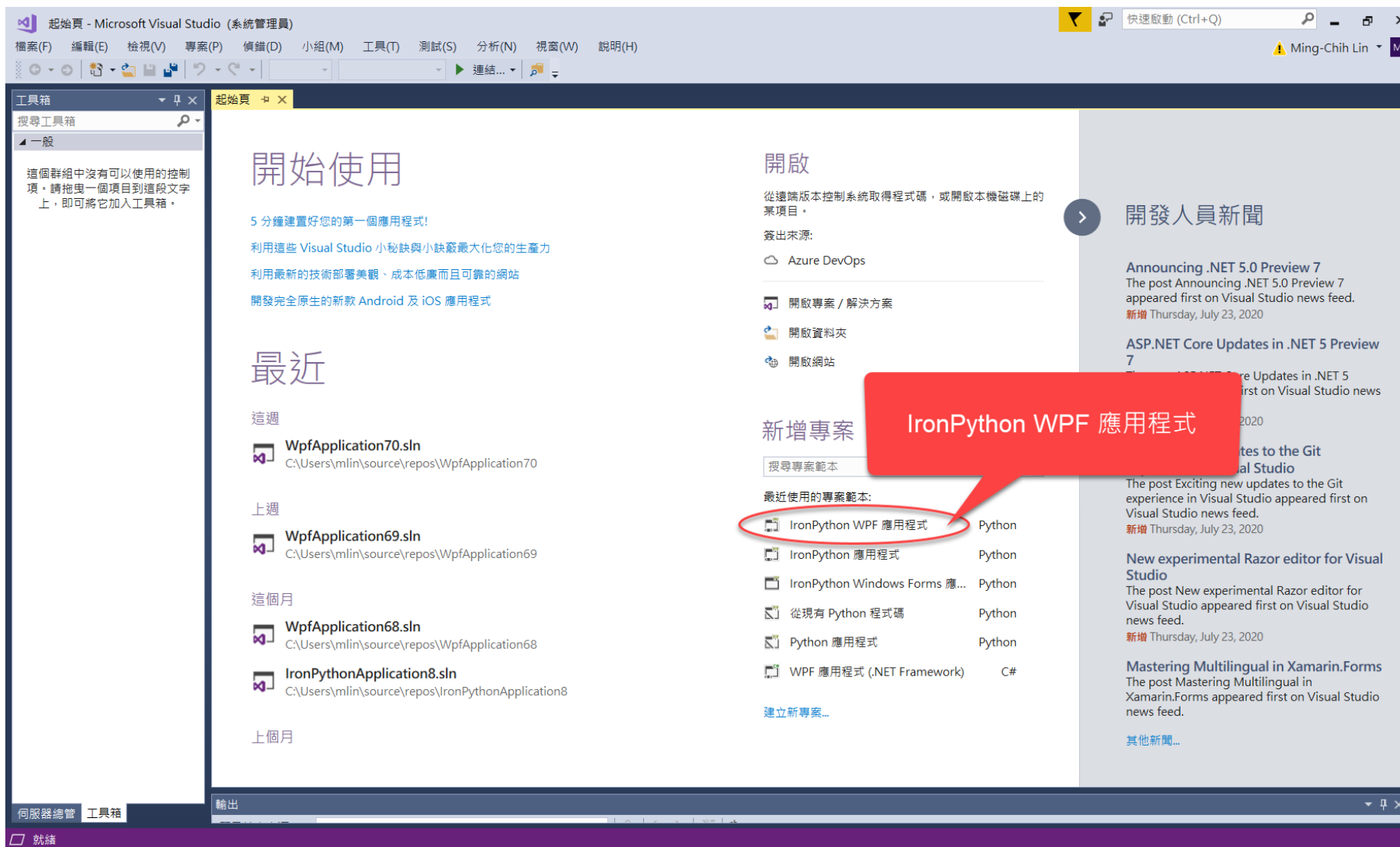
連接到批次檔

透過批次檔執行GUI

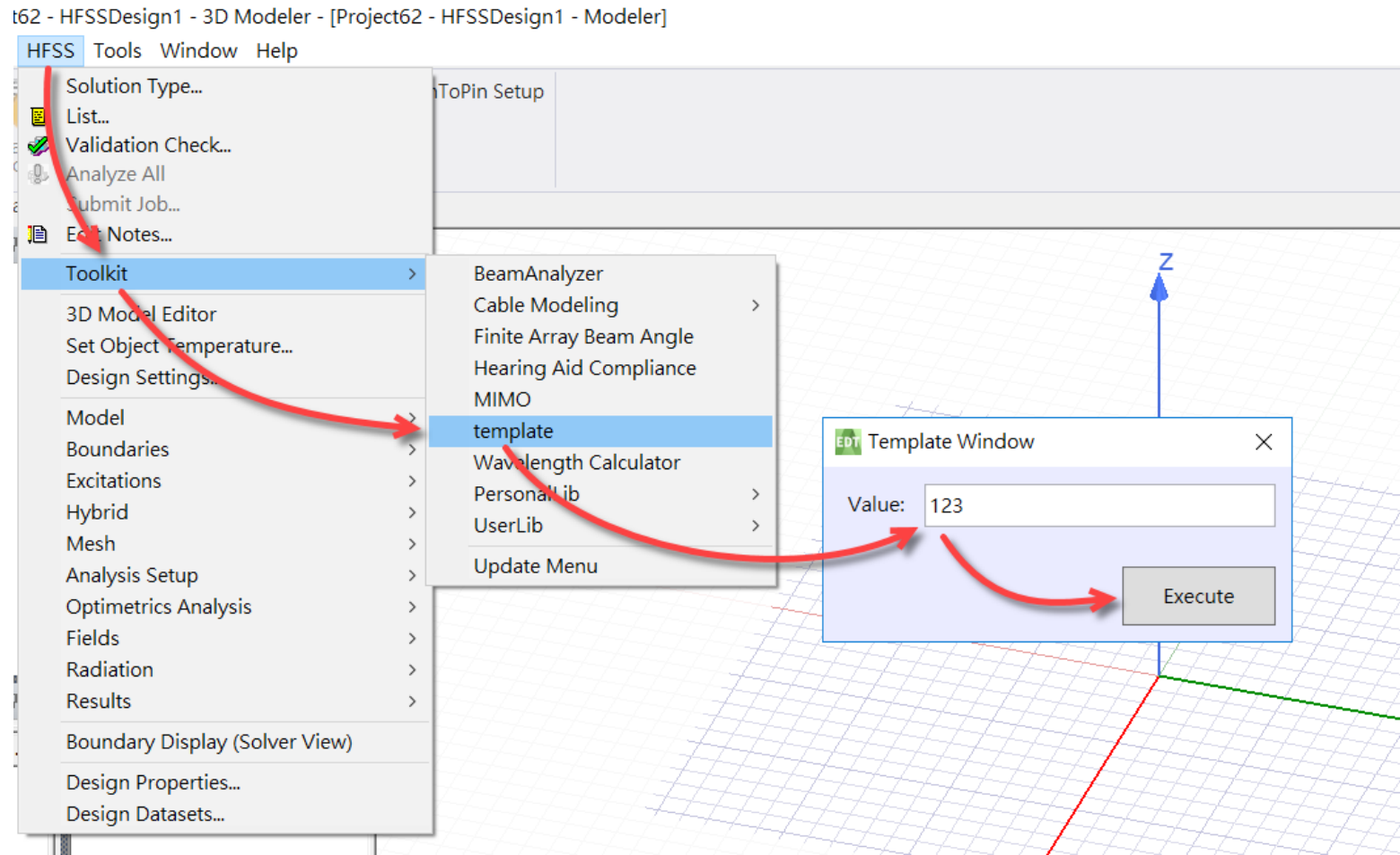
```
call C:\ProgramData\Anaconda3\Scripts\activate.bat  
C:\ProgramData\Anaconda3\python.exe ".\main.py"  
exit
```

範例解說：使用Visual Studio 產生IronPython GUI

Visual Studio IDE 介面



/ template.py + template.xaml



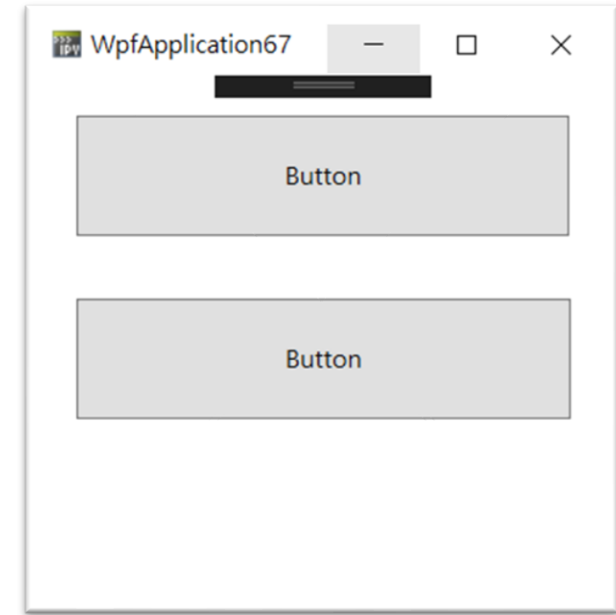
/ WPF的操作介面框架

傳統的介面設計是由程式碼搭出來的。也就是透過函式呼叫來宣告視窗，按鈕，選單等控鍵，也包括了大小，位置，顏色等屬性。除了介面設計，觸發函式的定義及響應也需要編寫函式碼。因此就算是一個簡單的操作介面也動輒超過數百行代碼，不只可讀性差，也很難除錯及擴充。

新的操作介面框架則是將外觀設計與觸發函數定義在xaml檔當中，這讓設計的語法大幅的精簡。一般的介面設計師可以在不需要編程的狀況底下完成介面的外觀設計，著重美觀及流暢的使用者體驗。至於函數響應的部分則仍然由程式工程師來完成，像是查詢或是運算等等。拆分讓整個程式碼大幅的簡化，不會混在一起導致難以閱讀。

XAML代碼及對應之操作介面

- 對於自動化程式設計來說，一般介面比較單純，程式設計師可以用現成“即視即所得”的工具拖放來產生xaml碼，並同時生成觸發函數的宣告及響應的框架。在測試無誤後再行編寫響應函數的實質功能。操作介面設計設計的工作也因而大幅簡化。



```
1 <Window
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     Title="WpfApplication67" Height="300" Width="300">
5     <Grid>
6         <Button Content="Button" Height="60" Margin="20,20,20,0" VerticalAlignment="Top"
7             RenderTransformOrigin="0.571,0.499" Click="Button_Click"/>
8         <Button Content="Button" Height="60" Margin="20,111,19.6,0" VerticalAlignment="Top"
9             RenderTransformOrigin="0.571,0.499" Click="Button_Click1"/>
10    </Grid>
11</Window>
```


何謂觸發及響應

當按下按鈕或是在輸入欄位填入一個字元，都可以觸發對應的函數。也就是函式會去執行對應的工作，比方開啟對話框或是輸出建議字串。在操作介面上的任何動作都可以觸發一個事先定義好的函式。一個介面可以定義的觸發事件可能高達上百上千個。我們只要選擇需要反應的去編寫響應函數即可。簡單介面對介面的響應甚至可以在xaml當中就可以宣告，不需另外編寫函式。

```
1 <Window
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     Title="WpfApplication67" Height="300" Width="300">
5     <Grid>
6         <Button Content="Button" Height="60" Margin="20,20,20,0" VerticalAlignment="Top"
7             RenderTransformOrigin="0.571,0.499" Click="Button_Click"/>
8         <Button Content="Button" Height="60" Margin="20,111,19.6,0" VerticalAlignment="Top"
9             RenderTransformOrigin="0.571,0.499" Click="Button_Click1"/>
10    </Grid>
11</Window>
```

```
import wpf

from System.Windows import Application, Window

class MyWindow(Window):
    def __init__(self):
        wpf.LoadComponent(self, 'WpfApplication67.xaml')

    def Button_Click(self, sender, e):
        pass

    def Button_Click1(self, sender, e):
        pass

if __name__ == '__main__':
    Application().Run(MyWindow())
```

GUI種類介紹

操作介面設計要求

好的操作介面讓使用者可以更直觀的完成設定，許多工程軟體在演算法的準確性及速度並無軒輊，最後卻是操作介面的設計好壞決定了銷售的成敗。對於自動化程式來說，操作介面設計決定了之後有多少人會使用，因此對於自動化程式開發者而言，操作介面的設計不可不慎。一個好的介面具備必須下列的特點：

- 簡潔的畫面
- 清楚的說明
- 正確的互動
- 當輸入值超出範圍時必須提醒使用者
- 當執行時間較長，須提供進度指示
- 本次設定為下次啟動的初始值

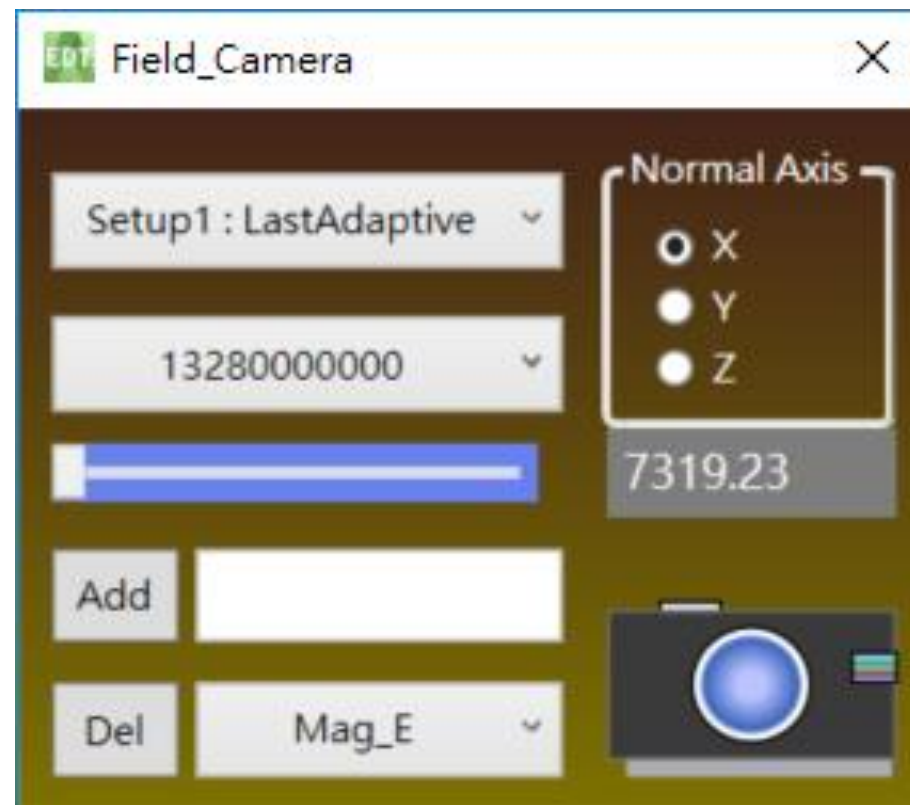
GUI操作介面技術

以下介紹幾種在AEDT當中常用的操作介面技術：

- **WPF**
- 屬性設定視窗
- **EXCEL & CSV**
- **ACT**

用wpf的技術來開發操作視窗簡便又快速。只要拖拉控鍵到視窗之上並輸入屬性就可以輕輕鬆鬆打造出一個相當有質感的視窗。視窗定義代碼記錄在xaml當中，事件函式則是在python當中。

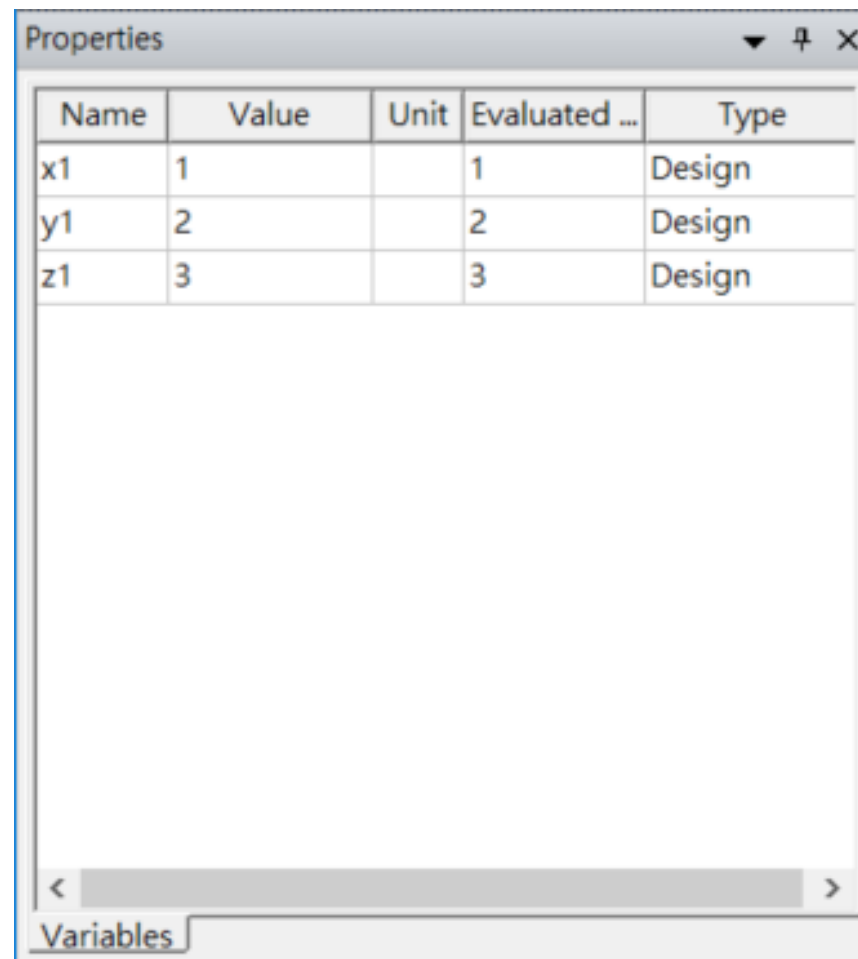
介面與業務代碼分離讓可讀性及擴充性大幅的提高。可惜的是現階段只能使用在windows環境當中。linux無法支援。



屬性設定視窗

這個方法讓使用者可以在屬性視窗設定變量，程式再去屬性視窗讀取數值執行運算。這個方法適用於windows和linux。好處是不需要另外設計操作介面代碼，缺點是只有輸入欄位，缺乏下拉選單或是滑桿等比較進階的控鍵可以使用。

在第一次執行時程式必須加入屬性，提示使用者屬性已被加入並結束程式。只有在屬性已經存在的條件底下程式才會執行工作。屬性的命名必須具備獨特性，避免與使用者自定義變量衝突，造成程式不正常運作。建議可以加入前後底線來做區分。



Name	Value	Unit	Evaluated ...	Type
x1	1		1	Design
y1	2		2	Design
z1	3		3	Design

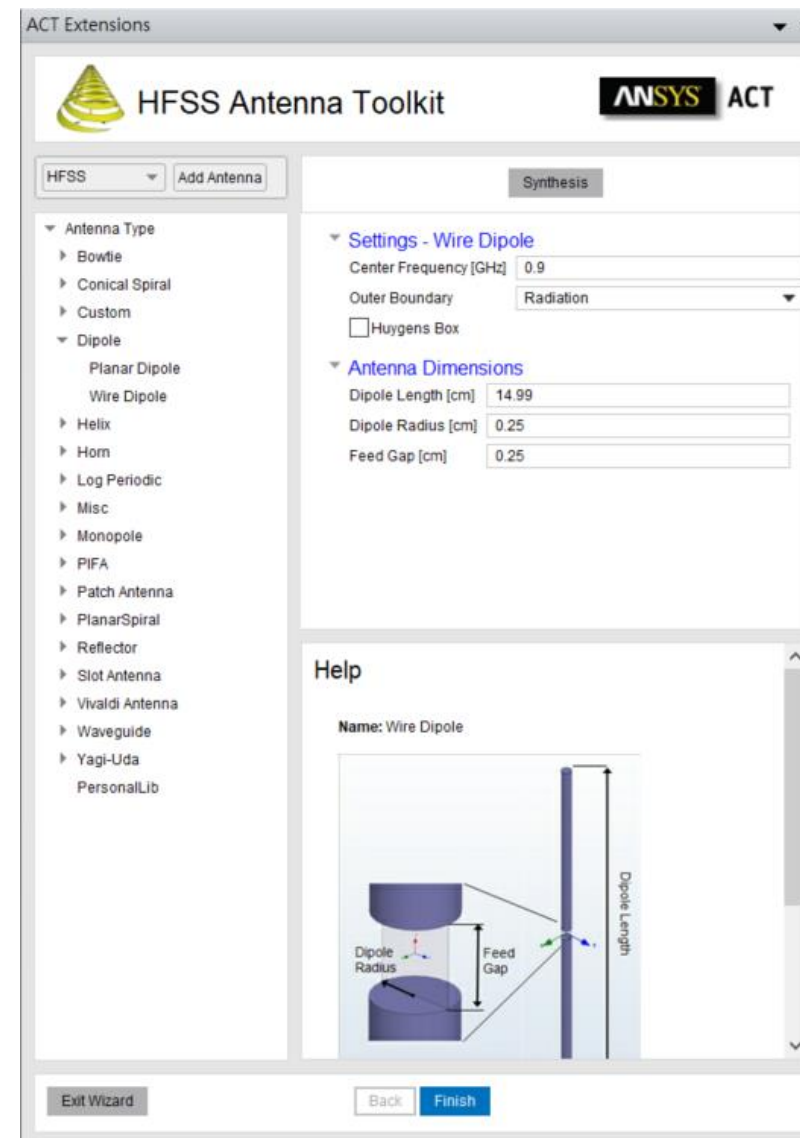
EXCEL & CSV

EXCEL作為介面，程式執行時讀取特定欄位的數值作為輸入執行運算。EXCEL可以輕鬆的做出複雜的介面，還可以加入參數檢查機制，也可以做到一定程度控鍵互動。AEDT內建函式庫提供函數直接讀取EXCEL,不需另外安裝庫。可惜的是這種方式只適用於windows環境，不適用Linux。

如果不需要華麗的介面及繁複的控鍵，最簡單的方法便是讓使用者將參數變量以txt或csv格式填寫。python讀入之後解析出參數值並執行運算即可。適用於windows和linux。可以透過OpenFileDialog()來選擇檔案

	A	B	C	D
2	General Information			
3	Project Name	test		
4	Working Directory	d:/demo2		
5	Package File (.slp)	D:/Customer2020/2020_3_19_SPIL_Automation/RRR03121136.slp		
6	Date		User	
7				
8	BGA Information			
9	Name	Ball Height (um)	Ball Radius (um)	Merge Sink
10	BGA	300	100	TRUE
11				
12	Die Information			
13	Name	Bump Height (um)	Bump Radius (um)	
14	B60874A	50	50	
15				
16				
17				
18				
19	Layers Information			
20	Name	Thickness (um)	Dk	Df
21	UNNAMED_000	50	1	0.2
22	L1	15	4	0.2
23	DRILL	80	4	0.2
24	L2	15	4	0.2
25	UNNAMED_009	50	1	0.2
26				
27				
28				
29				
30				
31				
32				

ACT可以支援幾種簡單的控鍵，如輸入欄位，下拉式選單或是標籤等等。在安裝之後並啟動之後，ACT視窗會顯示在主畫面右側，並持續保持開啟的狀態，ACT操作上相對簡便也同時支援windos和Linux。缺點是ACT屬於獨有的工具，沒有太多範例可以參考。除官方文件，網路上也找不到太多的相關資料，需仰賴使用者自行摸索。



技術總結

要設計出一個好的操作介面，需要從使用者角度來思考，推測使用者輸入時可能發生的錯誤，並針對錯誤提供正確的反饋，讓使用者清楚該如何修正。建議在介面完成初步開發時便可以邀請將來可能的使用者試用，根據使用者意見適當的修改可減少未來使用上的抱怨。

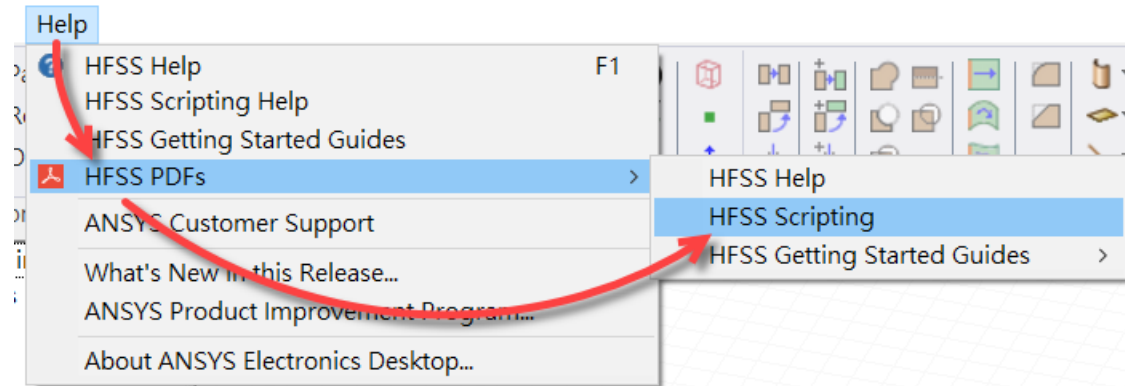
從以上的介紹各位可以了解沒有哪一種技術可以涵蓋所有的好處，開發人員必須視狀況決定採用哪一種技術。

AEDT函式庫架構

/ AEDT函式庫的完整文件

關於函式庫的Help檔可以在C:\Program Files\AnsysEM\AnsysEM20.1\Win64\Help底下找到ScriptingGuide.pdf當中詳細說明了函式的功能及使用方法。或是在Help底下也可以找到。

Modeler - [Project18 - HFSSDesign1 - Modeler]



Q3D Extractor S



ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
<https://www.ansys.com>
(T) 724-746-3304
(F) 724-514-9494



HFSS Scripting Guide

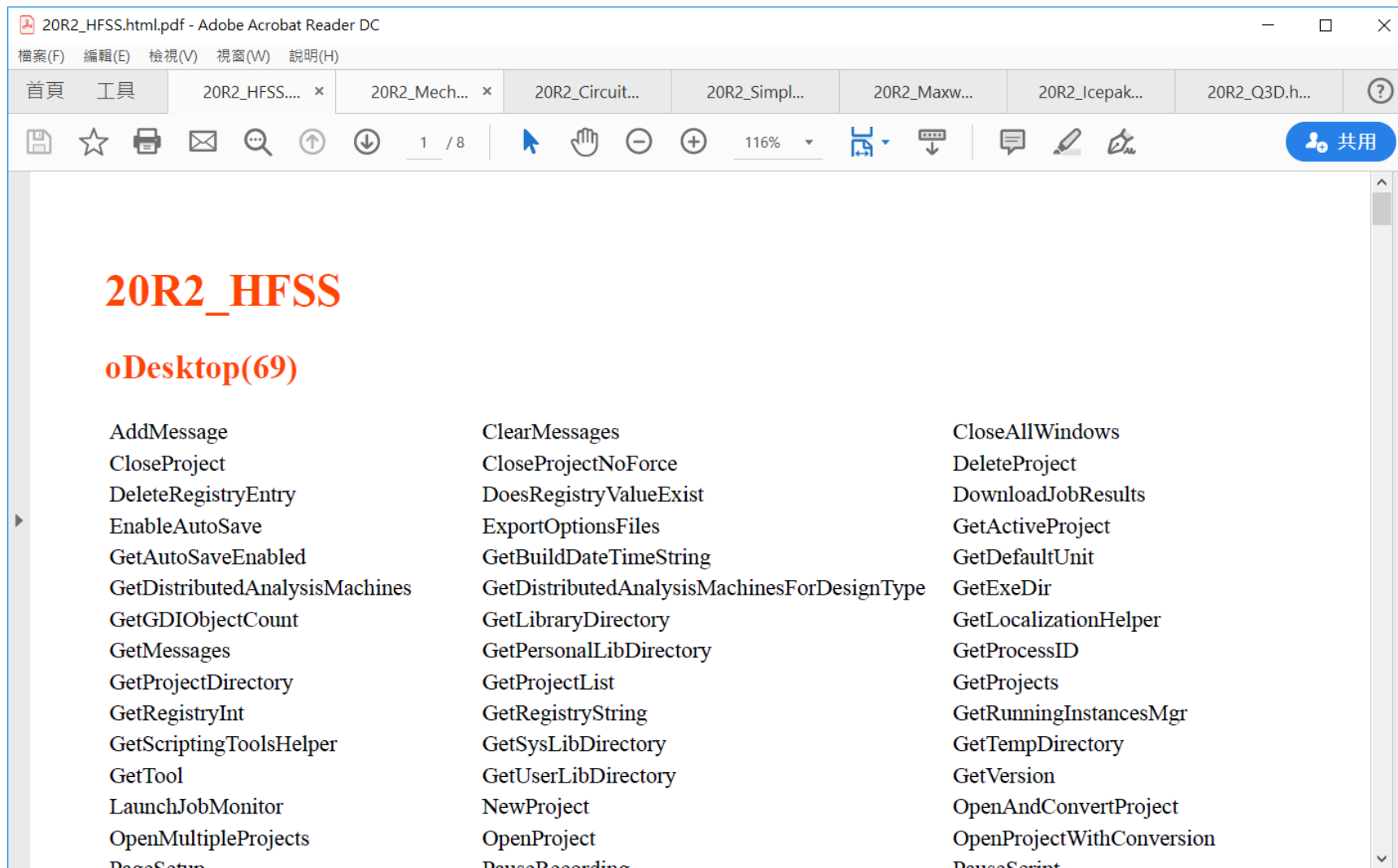


ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
<https://www.ansys.com>
(T) 724-746-3304
(F) 724-514-9494

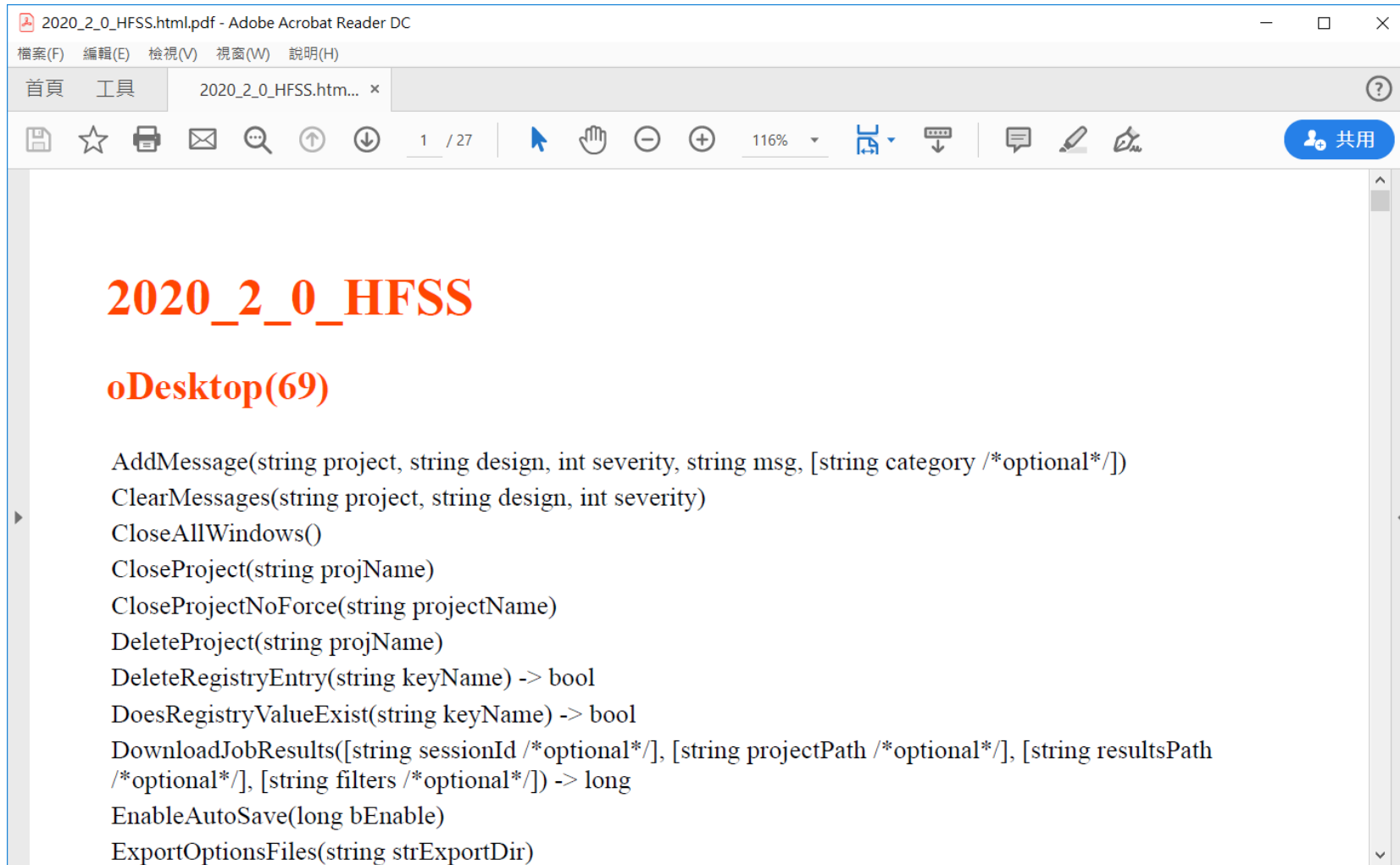
Release 2020 R1
January 2020

ANSYS, Inc. and
ANSYS Europe,
Ltd. are UL
registered ISO
9001:2015 com-
panies.

AEDT函式列表



/ AEDT函式列表+函數參數與返回



輸入dir()顯示最上層的函式與物件oDesktop



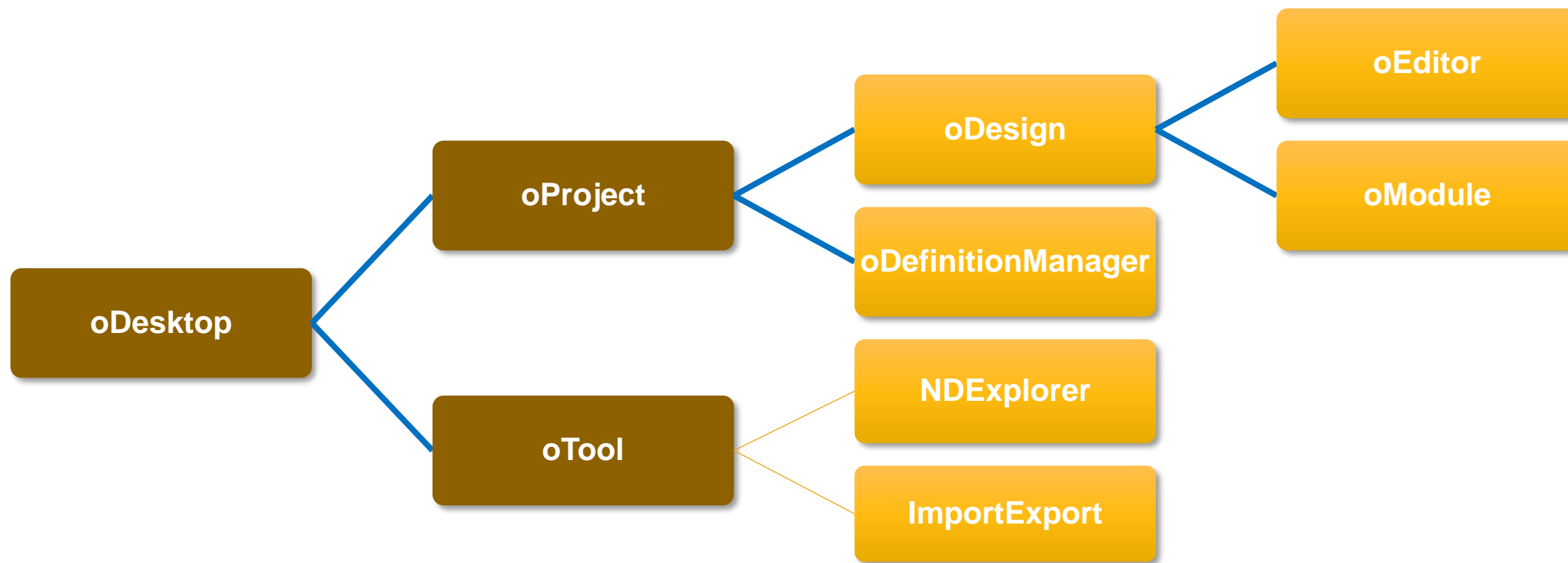
```
=====
ElectronicsDesktop 2020.1.0
IronPython 2.7.0.40 on .NET 4.0.30319.42000
=====
- With Tab completion
- dir()      - lists all available methods and objects
- dir(obj)   - lists all available attributes
- help(obj)  - provides available help on a module
- tutorial() - provides more help on using the software
=====
try executing "dir(oDesktop)" or dir_sig(oDesktop)
=====
>>> dir()
['AddErrorMessage', 'AddFatalMessage', 'AddInfoMessage', 'AddWarningMessage',
'CreateObject', 'LogDebug', 'LogError', 'RunScriptCommand', 'RunScriptFile',
'SetScriptingLanguageToJavascript', 'SetScriptingLanguageToVBScript', '__builtins__',
'__doc__', '__name__', '__scopeID__', 'dir_sig', 'dir_sig_doc', 'oAnsoftApplication',
'oDesktop', 'onlinehelp', 'tutorial']
>>>
```

訊息顯示函式


oDesktop物件

物件的階層

AEDT函式庫分成Desktop, Project, Design, Editor, Module等大類。當中Desktop, Project及Tool是共用的。之後的分類不同的產品會略有不同。



輸入dir(oDesktop)顯示oDesktop的方法和屬性



```
IronPython Command Window
>>> dir(oDesktop)
['AddMessage', 'ClearMessages', 'CloseAllWindows', 'CloseProject',
'CloseProjectNoForce', 'DeleteProject', 'DoesRegistryValueExist',
'DownloadJobResults', 'EnableAutoSave', 'Equals', 'ExportOptionsFiles',
'GetActiveProject', 'GetAutoSaveEnabled', 'GetBuildDateTimeString',
'GetDefaultUnit', 'GetDistributedAnalysisMachines',
'GetDistributedAnalysisMachinesForDesignType', 'GetExeDir', 'GetGDIObjectCount',
'GetHashCode', 'GetLibraryDirectory', 'GetLocalizationHelper', 'GetMessages',
'GetPersonalLibDirectory', 'GetProcessID', 'GetProjectDirectory', 'GetProjectList',
'GetProjects', 'GetRegistryInt', 'GetRegistryString', 'GetScriptingToolsHelper',
'GetSysLibDirectory', 'GetTempDirectory', 'GetTool', 'GetType',
'GetUserLibDirectory', 'GetVersion', 'LaunchJobMonitor', 'MemberwiseClone',
'NewProject', 'OpenAndConvertProject', 'OpenMultipleProjects', 'OpenProject',
'OpenProjectWithConversion', 'PageSetup', 'PauseRecording', 'PauseScript', 'Print',
'QuitApplication', 'ReferenceEquals', 'RefreshJobMonitor', 'ResetLogging',
'RestoreProjectArchive', 'RestoreWindow', 'ResumeRecording', 'RunACTWizardScript',
'RunProgram', 'RunScript', 'RunScriptWithArguments', 'SelectScheduler',
'SetActiveProject', 'SetActiveProjectByPath', 'SetLibraryDirectory',
'SetProjectDirectory', 'SetRegistryFromFile', 'SetRegistryInt',
'SetRegistryString', 'SetTempDirectory', 'ShowDockingWindow', 'Sleep', 'SubmitJob',
'TileWindows', 'ToString', '__class__', '__delattr__', '__doc__', '__format__',
'__getattr__', '__hash__', '__init__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__']
```


/ oDesktop -> oProject -> oDesign

- AEDT的函式庫是以物件導向，階層式的方式建構而成。最上層為oDesktop。底下衍生出來的不同模塊有各自的函數可以操作。比方說oDesktop不存在函式讓我們可以加入新的design。而在oProject底下可以找到InsertDesign()的指令可以插入新的設計。所以正確的順序是先從oDesktop建立新的project並指定到變數oProject，接著再使用oProject.InsertDesign()加入設計並指定到oDesign變數當中。這個順序跟我們手動設定方法是一致的。

```
oProject = oDesktop.NewProject()
```

```
oDesign = oProject.InsertDesign("HFSS", "HFSSDesign1", "DrivenTerminal", "")
```

/ oDesign -> oEditor

以此類推，假設我們要在設計當中加入圓柱，需要用到oEditor.CreateCylinder()指令，而oEditor需要從oDesign當中取得，這時候程式碼為：

```
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateCylinder(
[
  "NAME:CylinderParameters",
  "XCenter:=" , "-0.4mm",
  "YCenter:=" , "-0.4mm",
  "ZCenter:=" , "0mm",
  "Radius:=" , "0.447213595499958mm",
  "Height:=" , "1.2mm",
  "WhichAxis:=" , "Z",
  "NumSides:=" , "0"
],
[
  "NAME:Attributes",
  "Name:=" , "Cylinder1",
  "Flags:=" , "",
  "Color:=" , "(143 175 143)",
  "Transparency:=" , 0,
  "PartCoordinateSystem:=" , "Global",
  "UDMId:=" , "",
  "MaterialValue:=" , "\"vacuum\"",
  "SurfaceMaterialValue:=" , "\"\"",
  "SolveInside:=" , True,
  "IsMaterialEditable:=" , True,
  "UseMaterialAppearance:=" , False,
  "IsLightweight:=" , False
])
```

物件方法多以動詞+名詞來描述

- 建立(Create)
 - Add, New, Open,...
- 更新(Update)
 - Set, Show, Reset, Edit,...
- 讀取(Read)
 - Get, Is,...
- 刪除>Delete)
 - Close, Quit, Delete, Remove, ...

oDesktop

AddMessage
CloseProject
DoesRegistryValueExist
ExportOptionsFiles
GetBuildDateTimeString
GetDistributedAnalysisMachinesForDesignType
GetLibraryDirectory
GetPersonalLibDirectory
GetProjectList
GetRegistryString
GetTempDirectory
GetVersion
OpenAndConvertProject
OpenProjectWithConversion
PauseScript
RefreshJobMonitor
RestoreWindow
RunProgram
SelectScheduler
SetLibraryDirectory
SetRegistryInt
ShowDockingWindow
TileWindows

ClearMessages
CloseProjectNoForce
DownloadJobResults
GetActiveProject
GetDefaultUnit
GetExeDir
GetLocalizationHelper
GetProcessID
GetProjects
GetScriptingToolsHelper
GetTool
LaunchJobMonitor
OpenMultipleProjects
PageSetup
Print
ResetLogging
ResumeRecording
RunScript
SetActiveProject
SetProjectDirectory
SetRegistryString
Sleep

CloseAllWindows
DeleteProject
EnableAutoSave
GetAutoSaveEnabled
GetDistributedAnalysisMachines
GetGDIObjectCount
GetMessages
GetProjectDirectory
GetRegistryInt
GetSysLibDirectory
GetUserLibDirectory
NewProject
OpenProject
PauseRecording
QuitApplication
RestoreProjectArchive
RunACTWizardScript
RunScriptWithArguments
SetActiveProjectByPath
SetRegistryFromFile
SetTempDirectory
SubmitJob

讀懂函式說明

函式名稱

OpenProject

功能解說

Opens a specified project.

輸入參數

UI Access	Click File>Open		
Parameters	Name	Type	Description
	<FileName>	<string>	Full path of the project to open
Return Value	An object reference to the newly opened project.		

返回值

呼叫範例

Python Syntax	OpenProject(<Filename>)
Python Example	<code>oDesktop.OpenProject("C:/Projects/MyProject.aedt")</code>

解析Python模組(Module)和套件(Package)的概念

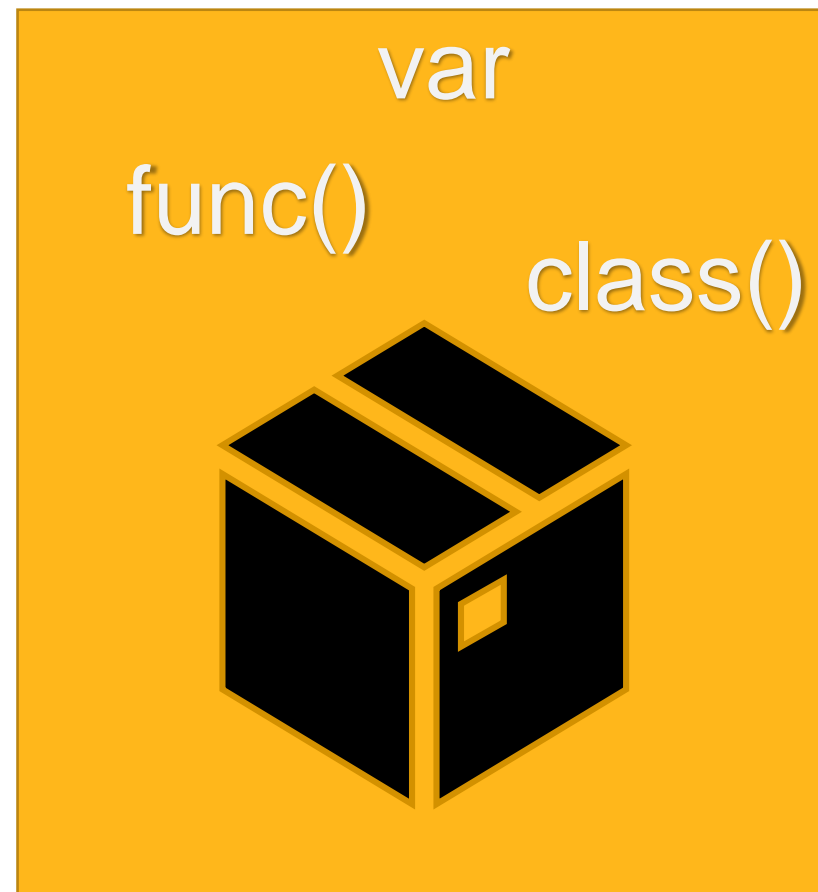
- 當我們在開發大型應用程式時，如果沒有適當的組織程式碼，除了會降低開發的效率外，也不易於維護，所以模組(Module)化就顯得相當的重要，讓程式碼能夠透過引用的方式來重複使用，提升重用性(Reusable)。
- 但是隨著專案模組(Module)的增加，將難以管理及問題的追蹤，這時候就能將模組(Module)打包成套件(Package)，利用其階層式的結構來彈性規劃模組(Module)。

自定義模組(Module)

只要你建立了一個原始碼檔案 `modu.py`，你就建立了一個模組 `modu`，原始碼主檔名就是模組名稱。

`import modu` 陳述句會在相同目錄下尋找 `modu.py`，如果沒找到，則會試著尋找在 `sys.path` 中遞迴地尋找 `modu.py`，如果還是沒有，則會引發 `ImportError` 例外。

模組提供了名稱空間。模組中的變數、函式與類別，基本上需透過模組的名稱空間來取得。在 Python 中，`import`、`import as` 與 `from import` 是陳述句，可以出現在程式中陳述句可出現的任何位置。



將模組編譯成.dll檔(僅限Windows)

PATH=%PATH%;C:\Program Files\AnsysEM\AnsysEM20.2\Win64\common\IronPython

ipy64 "C:/Program Files/AnsysEM/AnsysEM20.2/Win64/common/IronPython/Tools/Scripts/pyc.py" "D:/demo/mymodule.py"

pause

```
D:\demo>ipy64 "C:/Program Files/AnsysEM/AnsysEM20.2/Win64/common/IronPython/Tools/Scripts/pyc.py" "D:/demo/mymodule.py"
Input Files:
    D:/demo/mymodule.py
Output:
    mymodule
Target:
    Dll
Platform:
    ILOnly
Machine:
    I386

Compiling...
Saved to mymodule

D:\demo>pause
請按任意鍵繼續 . . . . .
```

連結.dll檔，匯入模組並測試模組函式

```
import os  
os.chdir(os.path.dirname(__file__))
```

```
import clr  
clr.AddReference("mymodule.dll")
```

```
import mymodule
```

```
x = mymodule.myadd(1,3)  
AddWarningMessage(str(x))
```


如何理解AEDT的方法參數

要理解函式，必須先從輸出入著手。一個函式可以有多個輸入參數，也可以沒有輸入參數。必須以正確的資料型態傳入參數，函式才能正常地完成運算或執行動作。假設參數x是以字串表示的數值，舉例來說“23”，那麼以整數型別傳入23是不被接受的。這是初學者經常犯的錯誤。同樣的，返回資料的型態可能是單純的整數或字串，也可能是物件。清楚返回值的資料格式才能正確的加以處理。在閱讀AEDT的函式說明時，先觀察輸入參數的型別，必要時可以編寫程式碼片段來測試。

如果要判斷返回資料的格式，最簡單的方式便是利用`type()`來得到資料型態，透過`dir()`來得到返回資料可以使用的方法。

 **Ansys**

