

pyAEDT的天線陣列設計 分析介紹 – Worksho -1

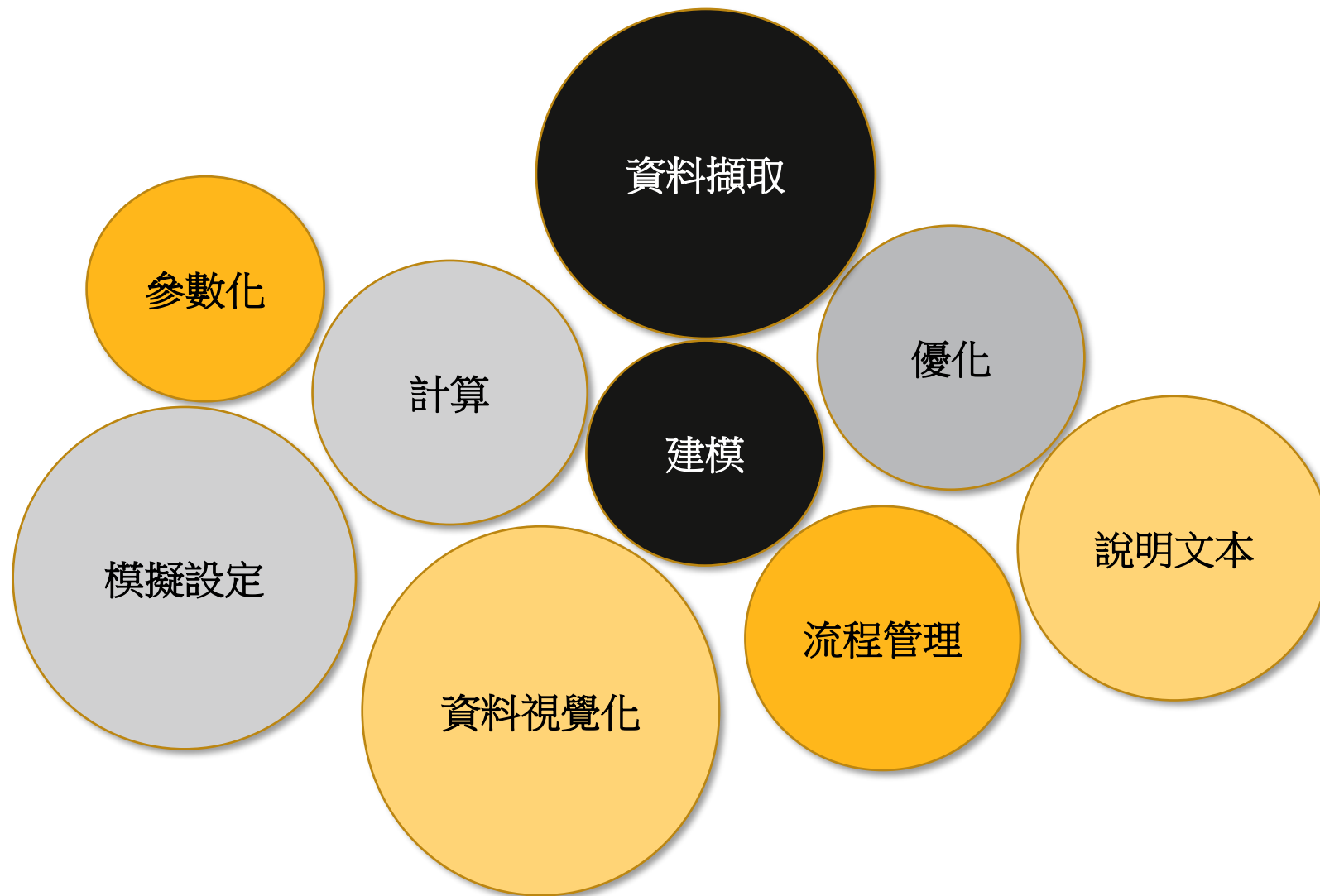
Apr 15, 2022



PyAEDT

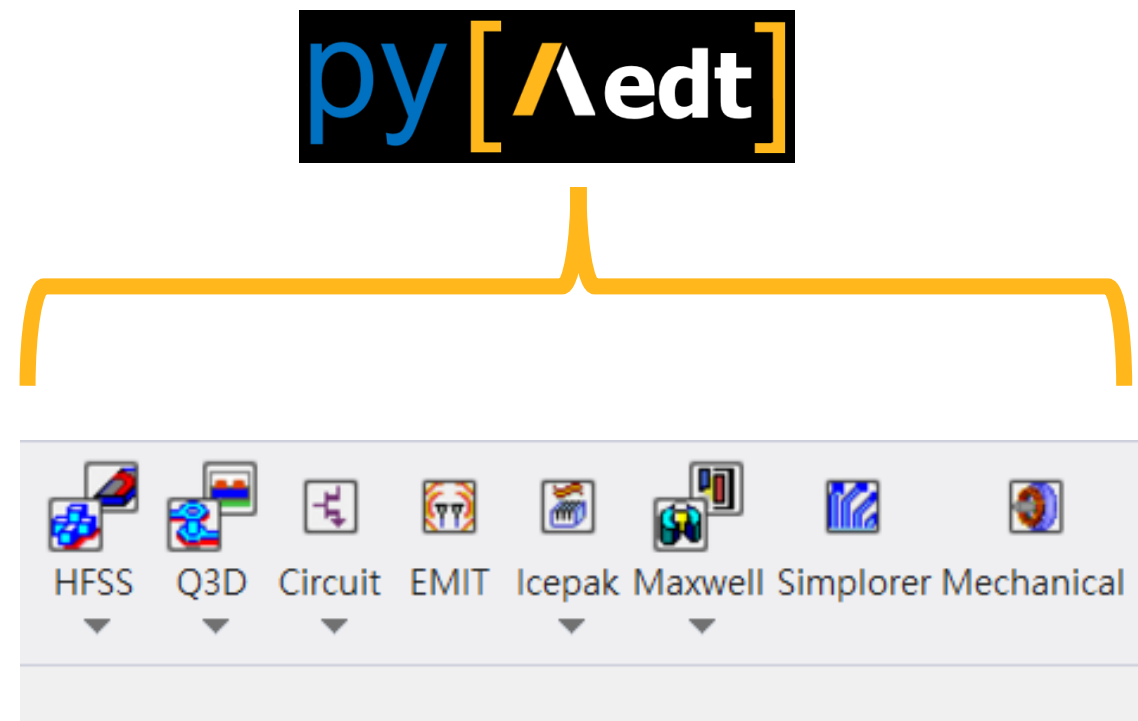


複雜設計由許多工作所組成



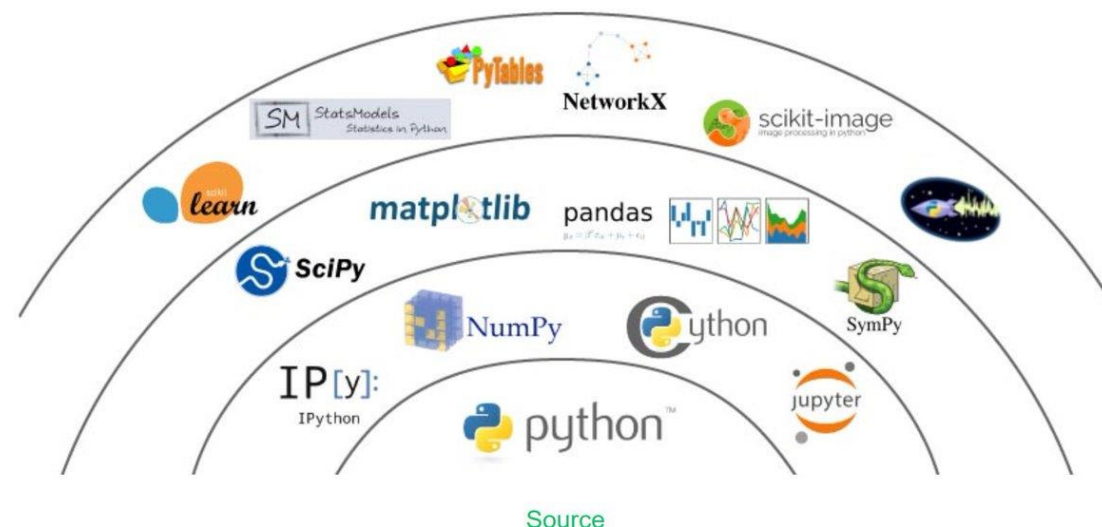
PyAEDT

- PyAEDT 旨在整合和擴展圍繞 Ansys Electronics Desktop (AEDT) 腳本編寫的所有現有功能，以允許重用現有代碼、共享最佳實踐並加強協作。PyAEDT 在 MIT 許可下獲得許可。
- 讓使用者可以透過Python程式在AEDT上面開發自動化程式，開展技術探索，開發教程等。

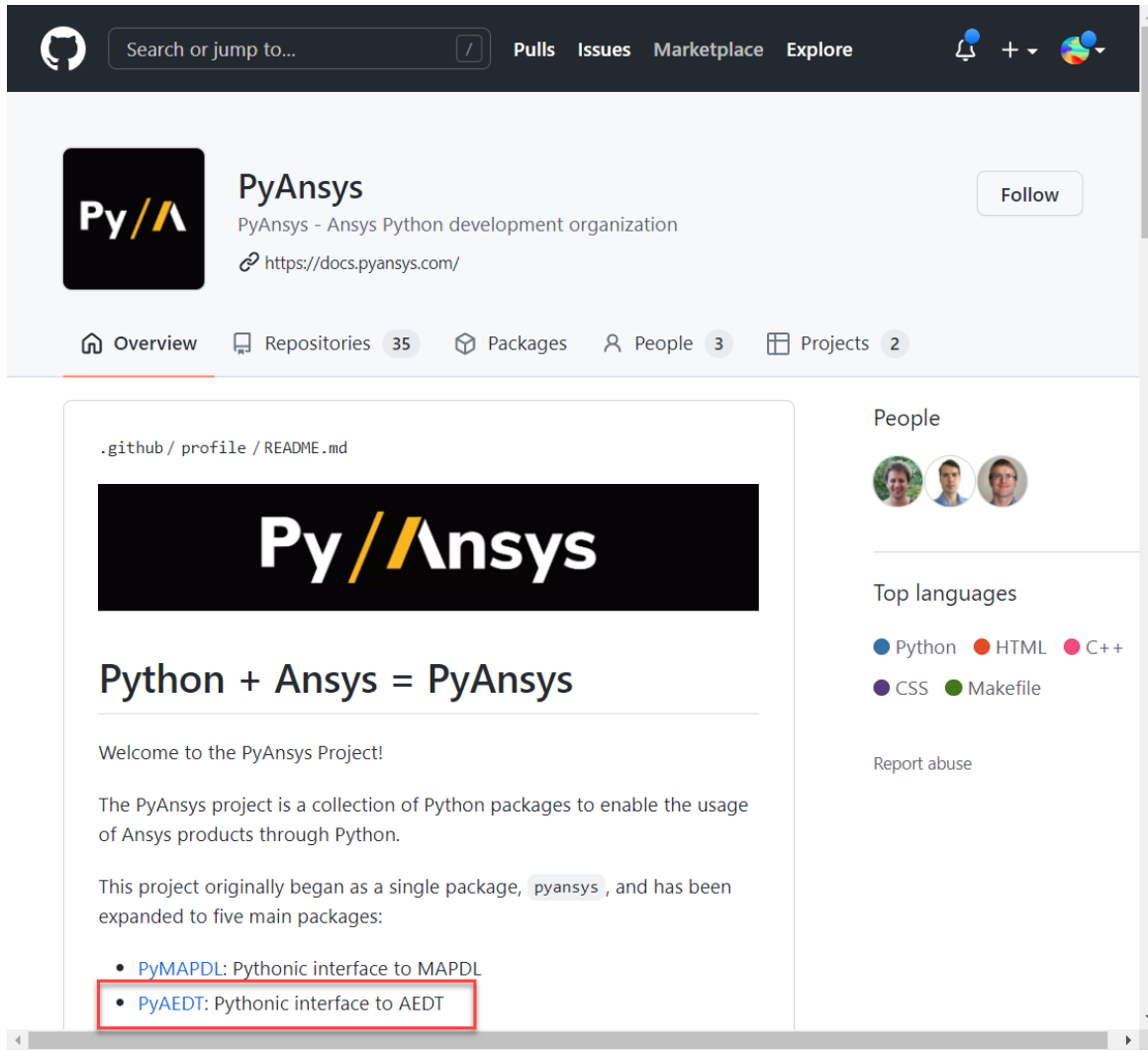


PyAEDT API的主要優勢 (與傳統API相比)

- 自動初始化所有 AEDT 對象，例如編輯器、邊界等桌面對象
- 錯誤管理，日誌管理，變量管理
- 與 IronPython 和 CPython 的兼容性
- 使用數據對象簡化複雜的 API 語法，同時保持 PEP8 合規性。
- 跨不同求解器的代碼可重用性
- 清晰的函數和 API 文檔
- 代碼單元測試以提高不同 AEDT 版本的質量
- 可支援NonGUI模式



PyAEDT是PyAnsys的一部份



The screenshot shows the GitHub profile of the PyAnsys organization. The header includes the GitHub logo, a search bar, and navigation links for Pulls, Issues, Marketplace, and Explore. The profile section features the PyAnsys logo, the name "PyAnsys", and the description "PyAnsys - Ansys Python development organization" with a link to the documentation. Below this are tabs for Overview, Repositories (35), Packages, People (3), and Projects (2). The main content area displays the README for the profile, which includes the PyAnsys logo and the text "Python + Ansys = PyAnsys". It welcomes users to the project and describes it as a collection of Python packages. A list of packages is shown, with "PyAEDT: Pythonic interface to AEDT" highlighted by a red box. To the right, there is a "People" section with three profile pictures and a "Top languages" section showing Python, HTML, C++, CSS, and Makefile.

PyAnsys
PyAnsys - Ansys Python development organization
<https://docs.pyansys.com/>

Overview Repositories 35 Packages People 3 Projects 2

.github / profile / README.md

Python + Ansys = PyAnsys

Welcome to the PyAnsys Project!

The PyAnsys project is a collection of Python packages to enable the usage of Ansys products through Python.

This project originally began as a single package, `pyansys`, and has been expanded to five main packages:

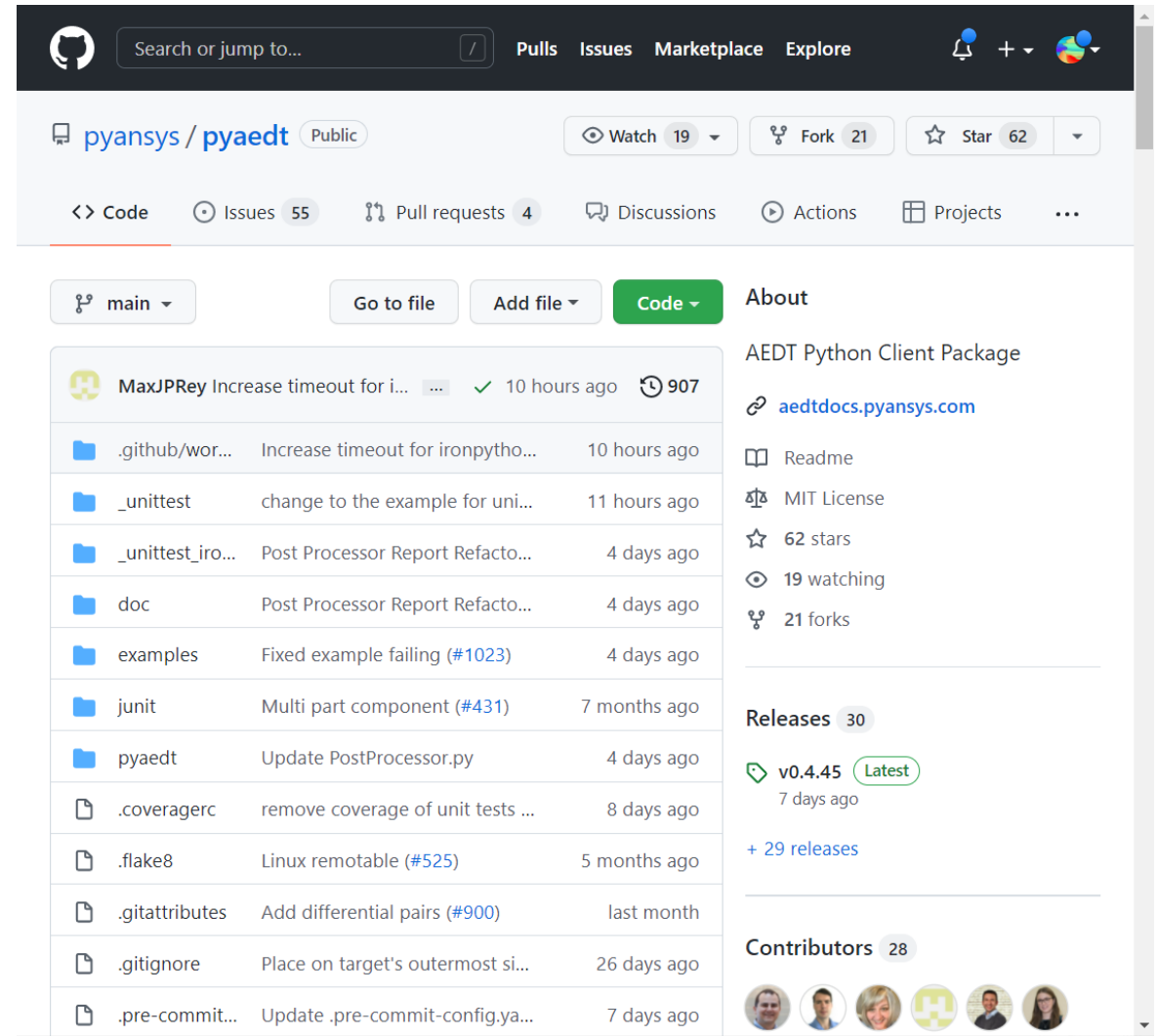
- PyMAPDL: Pythonic interface to MAPDL
- PyAEDT: Pythonic interface to AEDT**

People

Top languages

- Python
- HTML
- C++
- CSS
- Makefile

Report abuse



The screenshot shows the GitHub repository page for pyansys/pyaedt. The header includes the GitHub logo, a search bar, and navigation links for Pulls, Issues, Marketplace, and Explore. The repository section features the repository name "pyansys / pyaedt", the visibility "Public", and statistics for Watch (19), Fork (21), and Star (62). Below this are tabs for Code, Issues (55), Pull requests (4), Discussions, Actions, Projects, and more. The main content area displays a list of files and folders, including .github/workflows, _unittest, _unittest_ironpy, doc, examples, junit, pyaedt, .coveragerc, .flake8, .gitattributes, .gitignore, and .pre-commit-config.yaml. To the right, there is an "About" section describing the repository as the "AEDT Python Client Package" with a link to the documentation. It also shows the MIT License, 62 stars, 19 watchers, and 21 forks. Below this is a "Releases" section showing the latest release "v0.4.45" (Latest) from 7 days ago, with a link to view all 30 releases. At the bottom, there is a "Contributors" section showing 28 contributors.

pyansys / pyaedt Public

Watch 19 Fork 21 Star 62

Code Issues 55 Pull requests 4 Discussions Actions Projects

main Go to file Add file Code

MaxJPRey Increase timeout for i... 10 hours ago 907

File/Folder	Description	Time
.github/workflows	Increase timeout for ironpytho...	10 hours ago
_unittest	change to the example for uni...	11 hours ago
_unittest_ironpy	Post Processor Report Refacto...	4 days ago
doc	Post Processor Report Refacto...	4 days ago
examples	Fixed example failing (#1023)	4 days ago
junit	Multi part component (#431)	7 months ago
pyaedt	Update PostProcessor.py	4 days ago
.coveragerc	remove coverage of unit tests ...	8 days ago
.flake8	Linux remotable (#525)	5 months ago
.gitattributes	Add differential pairs (#900)	last month
.gitignore	Place on target's outermost si...	26 days ago
.pre-commit-config.yaml	Update .pre-commit-config.ya...	7 days ago

About

AEDT Python Client Package

aedtdocs.pyansys.com

Readme

MIT License

62 stars

19 watching

21 forks


Releases 30

v0.4.45 Latest 7 days ago

+ 29 releases

Contributors 28

文件官方網站：https://aedtdocs.pyansys.com/



PyAnsys » PyAEDT Documentation 0.4.50

 Search the do

PyAEDT Documentation 0.4.50

 **pypi** **v0.4.50** **PyPI downloads** **3.3k/month**  **CI** **passing**



 **codecov** **81%** **License** **MIT** **code style** **black**

Introduction


PyAEDT is part of the larger [PyAnsys](#) effort to facilitate the use of Ansys technologies directly from Python.

PyAEDT is intended to consolidate and extend all existing functionalities around scripting for Ansys Electronics Desktop (AEDT) to allow reuse of existing code, sharing of best practices, and increased collaboration. PyAEDT is licensed under the [MIT License](#).

PyAEDT includes functionality for interacting with the following AEDT tools and Ansys products:



PyAnsys » PyAEDT » API Documentation

 Search the do

API Documentation

Welcome to PyAEDT API documentation. Use the search feature or click the following links to view API documentation.

- **Initial Setup and Launching AEDT**
 - [Initial Setup and Launching AEDT Locally](#)
 - [Initial Setup and Launching AEDT Remotely](#)
 - [CPython on Linux with Client-Server](#)
- **AEDT Applications**
 - [pyaedt.Desktop](#)
 - [pyaedt.Hfss](#)
 - [pyaedt.Q3d](#)
 - [pyaedt.Q2d](#)
 - [pyaedt.Edb](#)
 - [pyaedt.Maxwell2d](#)
 - [pyaedt.Maxwell3d](#)
 - [pyaedt.Icepak](#)
 - [pyaedt.Hfss3dLayout](#)
 - [pyaedt.Mechanical](#)

Initial Setup and Launching AEDT

AEDT Applications

LogHandler

AEDT Modules

EDB Modules

Modeler and Primitives

Mesh Classes

Material and Stackup

Setup

Postprocessing

Setup Templates

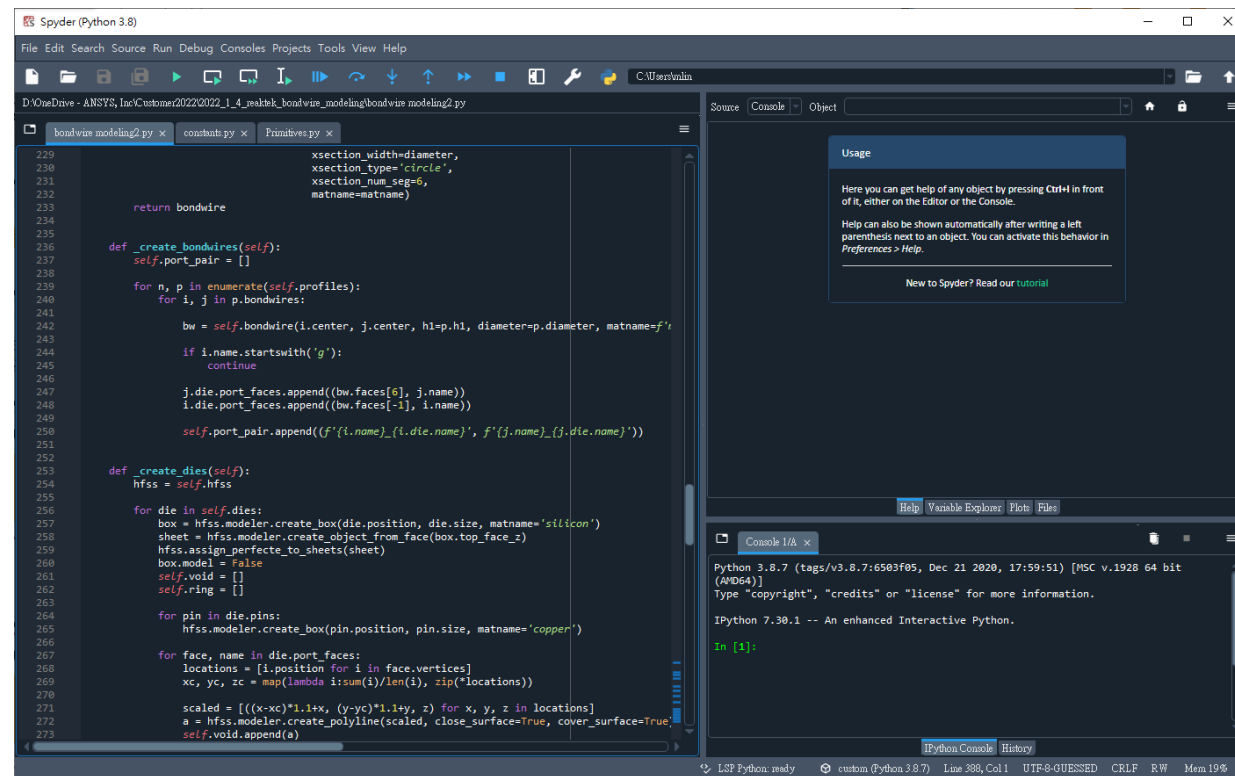
Boundaries Objects

Multi-Part Components

Constants

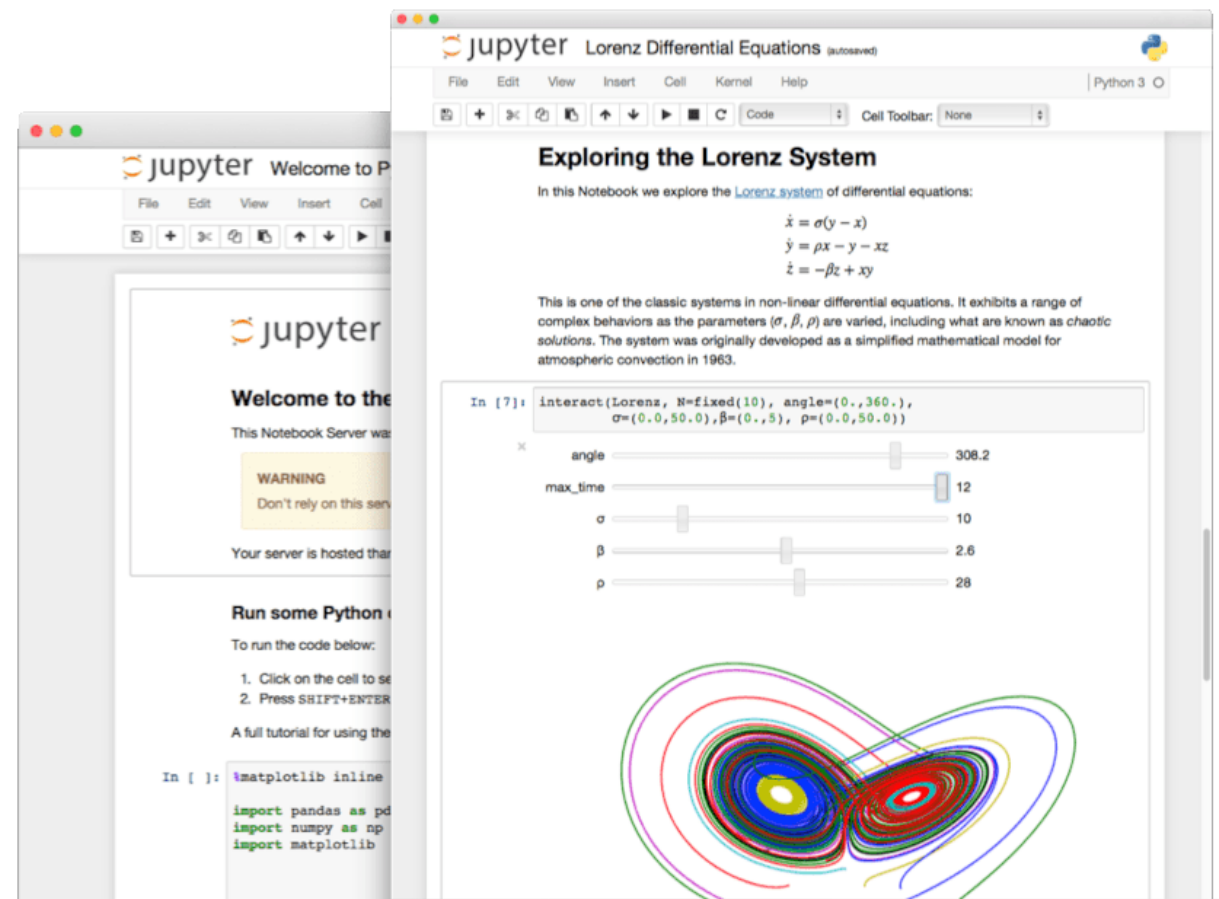
整合開發工具：Spyder

- 編輯器：
 - 具有函式和類檢視器
 - 代碼分析特性
 - 代碼補全
 - 直接跳入定義
- 互動視窗：
 - Python或IPython埠都在工作區可以調整和使用。支援對編輯器里的代碼直接除錯。。
- 文件瀏覽器：
- 在編輯器或埠中顯示任意類或函式呼叫的文件。
- 變數瀏覽視窗
- Matplotlib的圖表顯示視窗
- 歷史記錄



操作使用介面：Jupyter Notebook

- Jupyter Notebook是一個基於Web的互動式計算環境，用於創建Jupyter Notebook檔案。
- Jupyter Notebook檔案是一個JSON檔案，包含一個有序的輸入/輸出單元格列表，這些單元格可以包含程式碼、文字（使用Markdown語言）、數學、圖表和富媒體 (Rich media)，通常以「.ipynb」結尾附檔名。
- 可做為複雜流程管理及視覺化介面。



傳統設計流程 v.s. PyAEDT設計流程

- 傳統流程
 - 不容易參數化複雜結構
 - 需手動傳遞參數並個別執行模擬
 - 需另外計算延遲補償並回填
 - 手動複製模型並調整位置
 - 需手動繪製傳輸線
 - 需開啟GUI操作
 - 需要輸出模擬資料與圖表以生成報告
- 設計者需連結不同設計，設計方法無法完整表示。
- PyAEDT設計流程
 - 相對容易參數化複雜結構
 - 在不同設計當中自動傳遞參數
 - 結合延遲計算
 - 根據算法自動調整位置
 - 自動繪製傳輸線
 - 可不開啟GUI執行
 - 可結合報告輸出與圖表
- 設計方法以程式碼方式完整保存設計參數，具備可復現性。

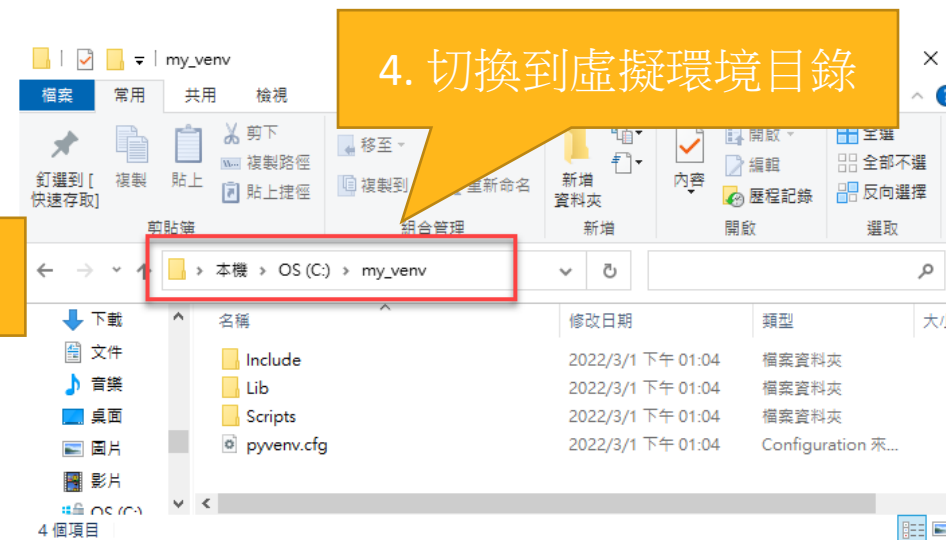
PyAEDT與Spider安裝



Python 虛擬環境

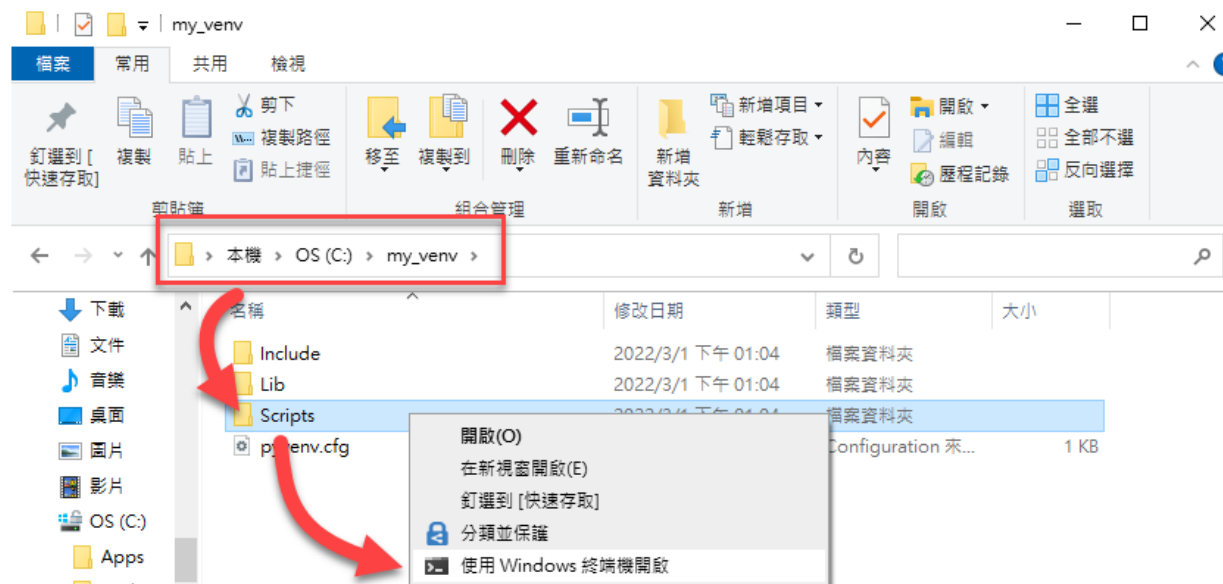
- Python 應用程式通常會用到不在標準函式庫的套件和模組。應用程式有時候會需要某個特定版本的函式庫，因為這個應用程式可能需要某個特殊的臭蟲修正，或是這個應用程式是根據該函式庫特定版本的介面所撰寫。
- 這意味著不太可能安裝一套 Python 就可以滿足所有應用程式的要求。如果應用程式 A 需要一個特定的模組的 1.0 版，但另外一個應用程式 B 需要 2.0 版，那麼這整個需求不管安裝 1.0 或是 2.0 都會衝突，以致於應用程式無法使用。
- 解決方案是創建一個虛擬環境 (virtual environment)，這是一個獨立的資料夾，並且裡面裝好了特定版本的 Python，以及一系列相關的套件。
- 不同的應用程式可以使用不同的虛擬環境。以前述中需要被解決的例子中，應用程式 A 能夠擁有它自己的虛擬環境，並且是裝好 1.0 版，然而應用程式 B 則可以用另外一個有 2.0 版的虛擬環境。要是應用程式 B 需要某個函式庫被升級到 3.0 版，這並不會影響到應用程式 A 的環境。

建立Python虛擬環境



透過網路連線安裝

- 選擇Scripts目錄並開啟終端機
- 啟動虛擬環境
 - `.\activate`
- 安裝pyaedt
 - `pip install pyaedt`
- 安裝Spyder IDE
 - `pip install spyder`



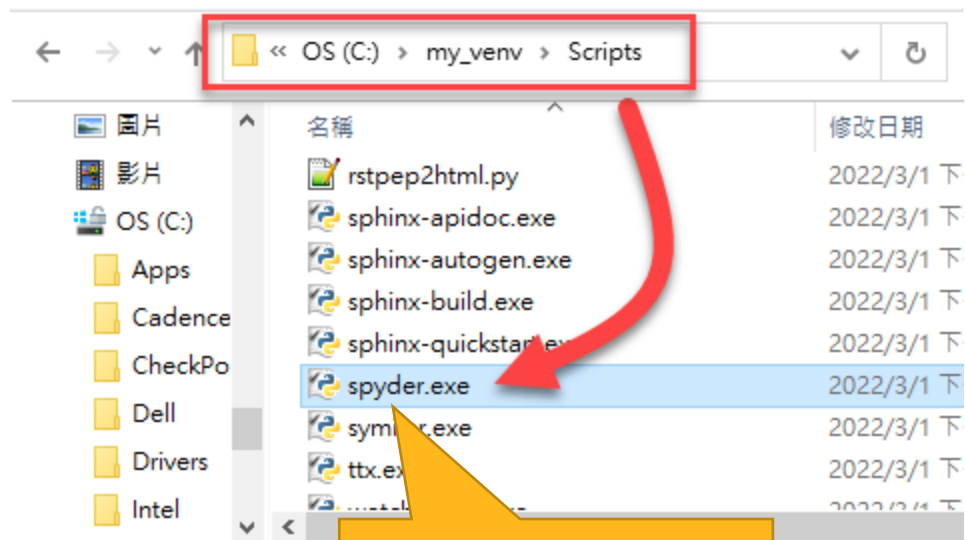
```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore6

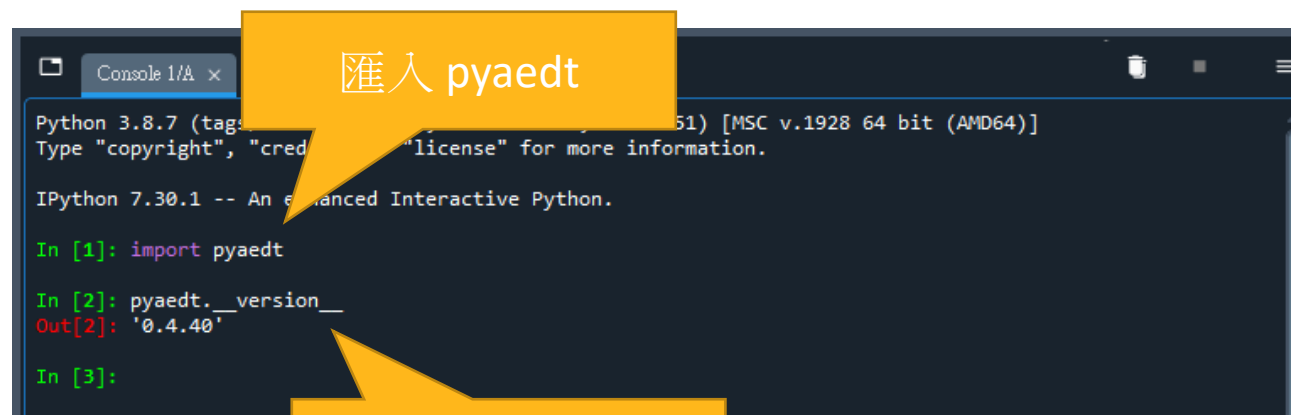
PS C:\my_venv\Scripts> .\activate
(my_venv) PS C:\my_venv\Scripts> pip install pyaedt
Collecting pyaedt
```

```
33 pycparser-2.21 pygments-2.11.2 pyparsing-3.0.7 python-date
5.0.1 scooby-0.5.11 six-1.16.0 traitlets-5.1.1 typing-extensi
WARNING: You are using pip version 20.1.1; however, version 2
You should consider upgrading via the 'c:\my_venv\scripts\pyt
(my_venv) PS C:\my_venv\Scripts> pip install spyder
Collecting spyder
Using cached spyder-5.2.2-py3-none-any.whl (14.8 MB)
```

安裝完成之後，開啟Spyder，測試pyaedt模組安裝



1. 開啟Spyder

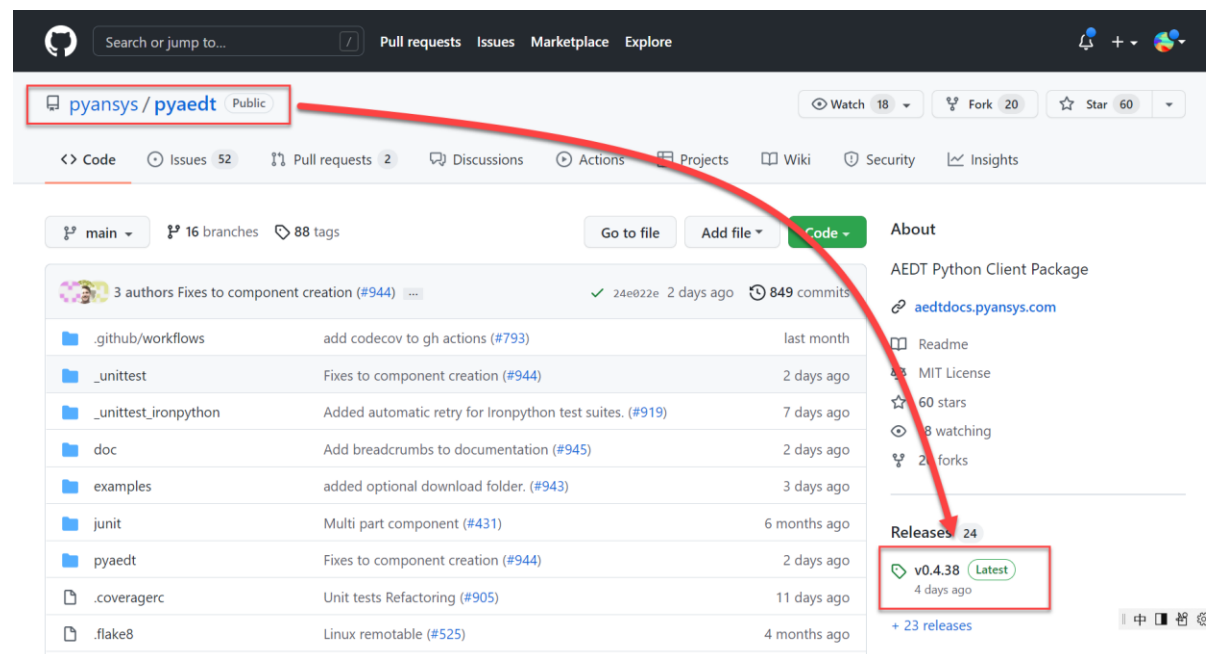


匯入 pyaedt

檢查版本

昇級pyaedt版本

- 最新pyaedt版本可至Github pyaedt repository(儲存庫)查詢
- 升級pyaedt版本
 - 到虛擬環境並activate環境
 - 輸入 `pip install pyaedt --upgrade`

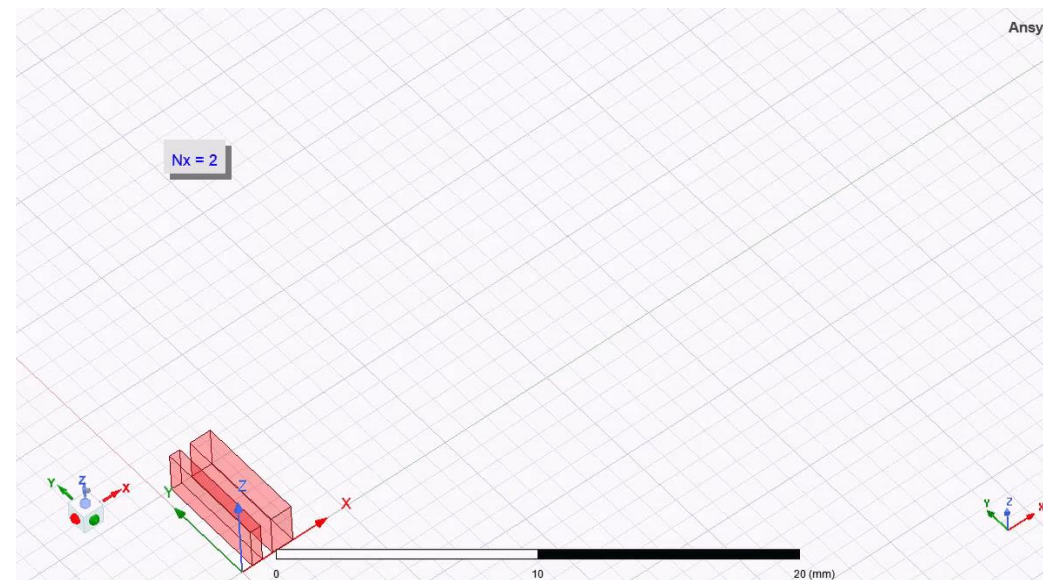


```
系統管理員: Windows PowerShell
(my_venv3) PS D:\my_venv3\Scripts> pip install pyaedt --upgrade
WARNING: Ignoring invalid distribution -yaedt (d:\my_venv3\lib\site-packages)
WARNING: Ignoring invalid distribution -yaedt (d:\my_venv3\lib\site-packages)
Requirement already satisfied: pyaedt in d:\my_venv3\lib\site-packages (0.4.35)
Collecting pyaedt
  Downloading pyaedt-0.4.40.tar.gz (1.8 MB)
    1.8/1.8 MB 3.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: rpyc==5.0.1 in d:\my_venv3\lib\site-packages (from pyaedt) (5.0.1)
Requirement already satisfied: pyvista>=0.32.0 in d:\my_venv3\lib\site-packages (from pyaedt) (0.32.1)
```


複雜結構參數化

參數化

- 3D物件可以透過參數改變其位置，形狀，數量，角度等，稱之為參數化元件
- 參數化元件可以搭配Sweep, Optimization等找出最佳系統表現，或檢視系統變異量。
- 對強健性設計來說，參數化是系統分析的前提。
- 如何建立參數化結構是一門綜合數學各領域的工程問題：幾何+邏輯+編程



HFSS基本參數化方法

內部參數化

- Sweep/Optimetrics
- 可分散式處理
- 可建立3D Component
- 資料儲存在同一個設計當中
- 無法建立複雜模型
- 難度：★

UDP (User Define Primitive)

- Sweep/Optimetrics
- 可分散式處理
- 可建立3D Component
- 資料儲存在同一個設計當中
- 可建立複雜模型
- 難度：★★★★★

外部參數化

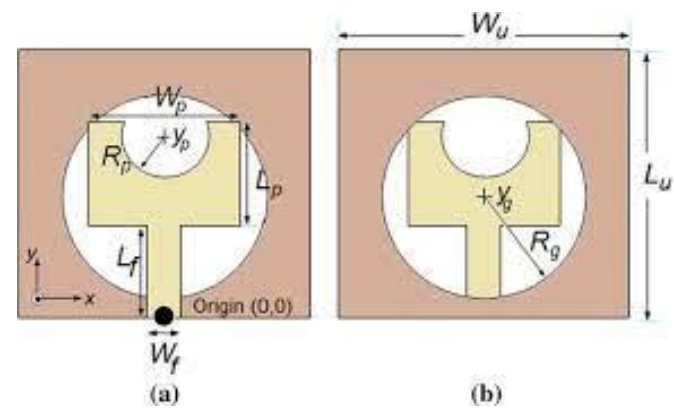
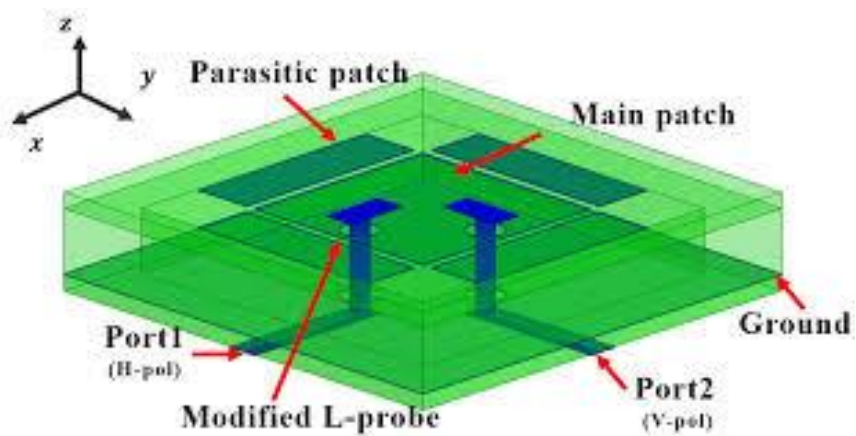
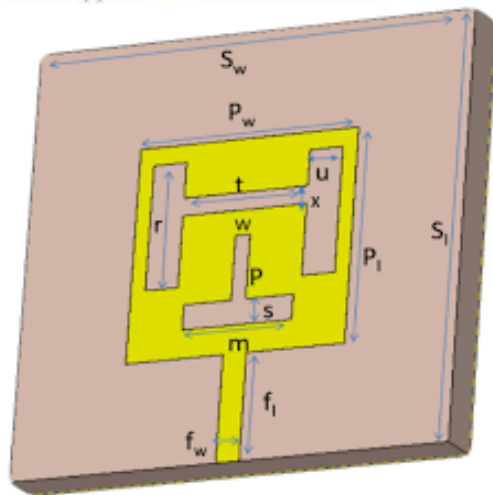
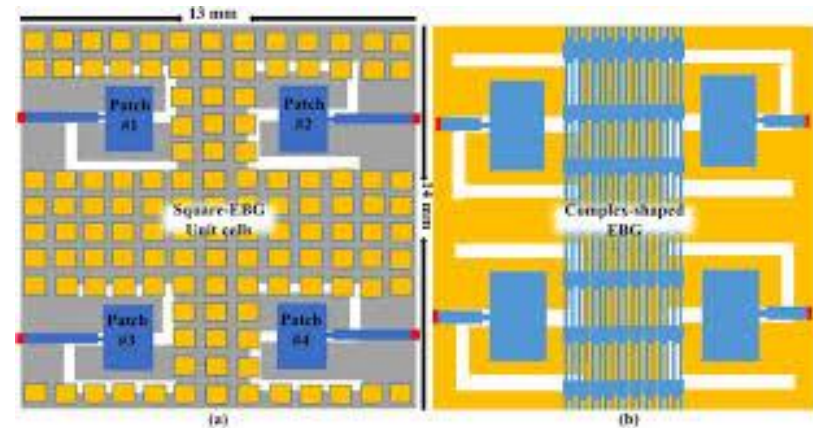
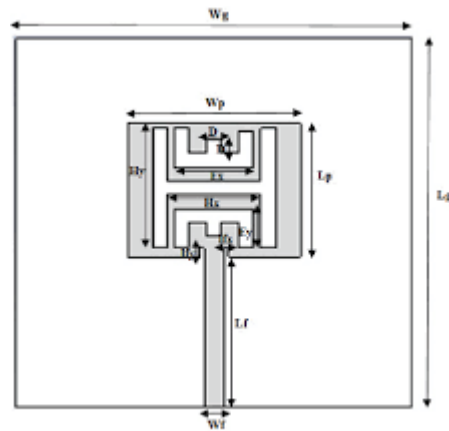
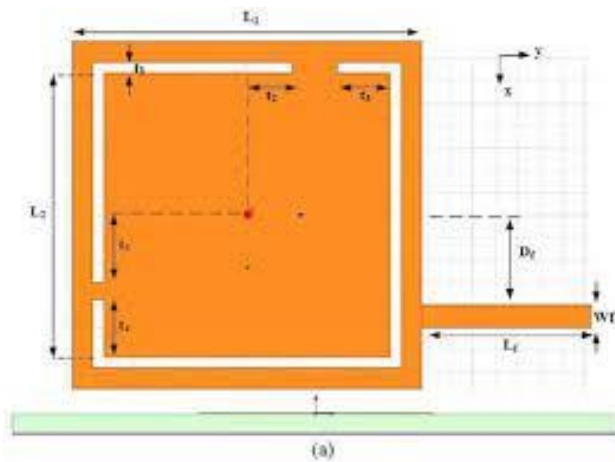
- For-Loop
Sweep/Scipy.Minimize()
- 資料儲存分散在多個設計
- 可建立複雜模型
- 難度：★★

PyAEDT+HFSS基本參數化方法

PyAEDT+內部參數化

- Sweep/Optimetrics
 - 可分散式處理
 - 可建立3D Component
 - 資料儲存在同一個設計當中
 - 可以建立複雜模型
 - 難度：★★
- 設定參數
 - 決定參考點
 - 判斷是否存在對稱
 - 利用迴圈及判斷式設定座標點
 - 使用`hfss.modeler.create_polyline()`建立多邊形
 - 使用`hfss.modeler.duplicate_and_mirror()`建立對稱
 - 使用`hfss.modeler.duplicate_along_line()`建立陣列

各式各樣的平面天線設計



漸變尺寸參數化陣列

```
from pyaedt import Hfss

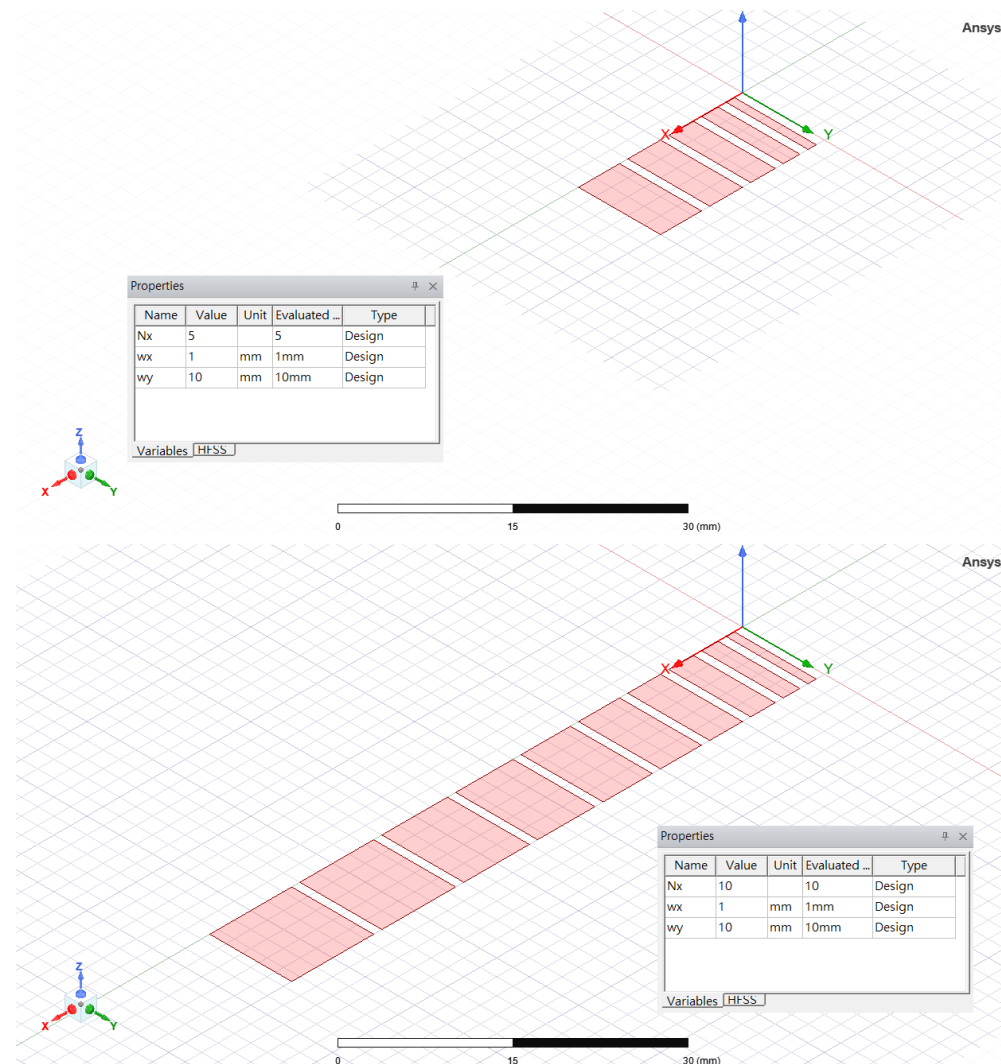
hfss = Hfss(specified_version='2021.2')

hfss['Nx'] = '4'
hfss['wx'] = '1mm'
hfss['wy'] = '10mm'
u=1

models = []
for i in range(100):
    x = hfss.modeler.create_rectangle(2,
                                      [f'if(Nx>{i}, {u}*wx, wx)', '0mm', '0mm'],
                                      [f'if(Nx>{i}, {i+1}*wx, wx)', 'wy'],
                                      name='rect')

    x.color = (255,0,0)
    u+=i+2
    models.append(x)

hfss.modeler.unite(models)
```



週期環形結構

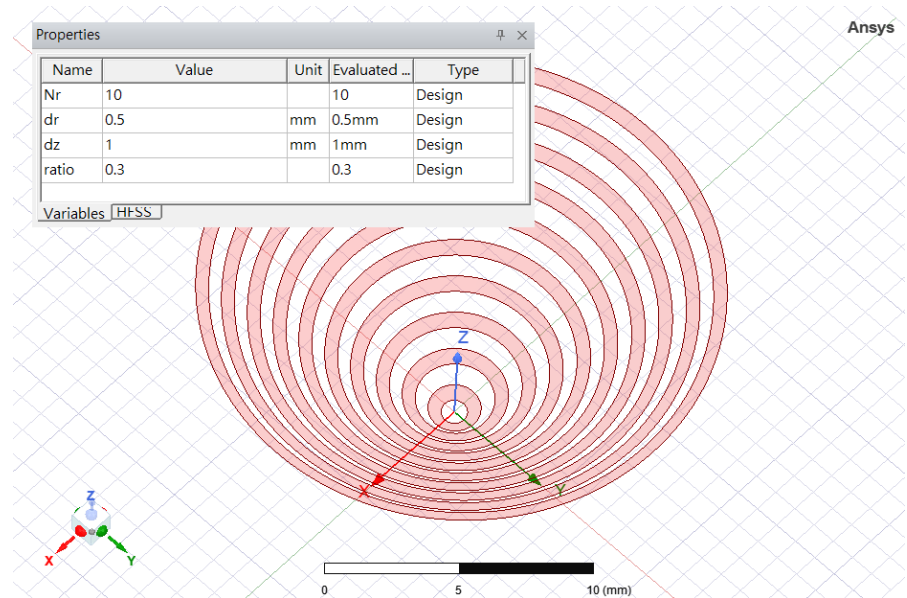
```
from pyaedt import Hfss

hfss = Hfss(specified_version='2022.1')

hfss['Nr'] = 4
hfss['dr'] = '1mm'
hfss['dz'] = '1mm'
hfss['ratio'] = 0.7

rings = []
for i in range(0, 20):
    c1 = hfss.modeler.create_circle(2, (0, 0, f'if({i}<Nr, {i}*dz, 0)'), f'if({i}<Nr, {2*i+1}*dr, 1*dr)', is_covered=False, name=f'c_{2*i}')
    c2 = hfss.modeler.create_circle(2, (0, 0, f'if({i}<Nr, ({i}+ratio)*dz, ratio*dz)'), f'if({i}<Nr, {2*i+2}*dr, 2*dr)', is_covered=False, name=f'c_{2*i+1}')
    ring = hfss.modeler.connect([c1, c2])

sheets = hfss.modeler.get_objects_in_group('Sheets')
hfss.modeler.unite(sheets)
```



矩陣方塊

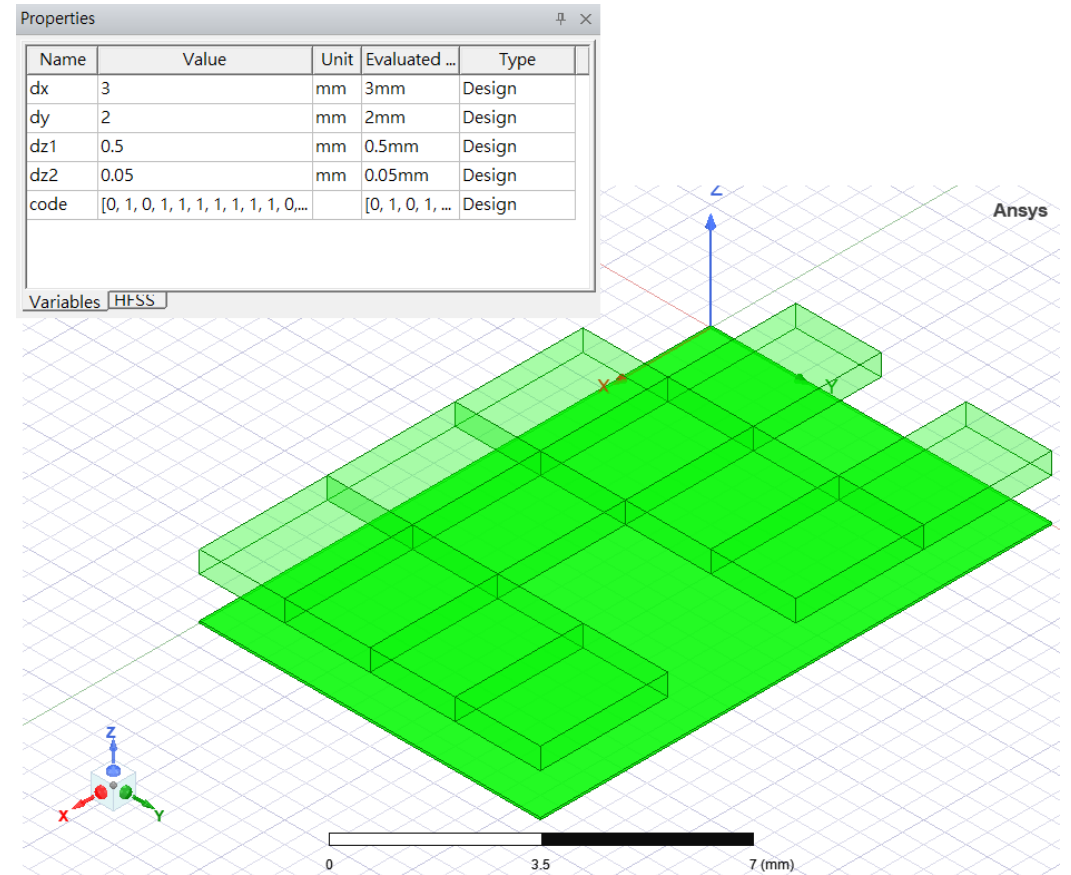
```
from pyaedt import Hfss

hfss = Hfss(specified_version='2022.1')
N=4

hfss['dx'] = '4mm'
hfss['dy'] = '3mm'
hfss['dz1'] = '0.2mm'
hfss['dz2'] = '0.1mm'
hfss['code'] = [1]*(N**2)

create_rectangle = hfss.modeler.create_rectangle
keys = [(i, j) for i in range(N) for j in range(N)]

x = create_rectangle(2, ('0mm', '0mm', '0mm'), (f'{N}*dx', f'{N}*dy'), name='m0')
hfss.modeler.thicken_sheet(x, 'dz2')
for n, (nx, ny) in enumerate(keys):
    x = create_rectangle(2,
        (f'if(code[{n}]==1, {nx}*dx, 0)',
         f'if(code[{n}]==1, {ny}*dy, 0)', f'if(code[{n}]==1, 1mm, 0mm)'),
        (f'if(code[{n}]==1, dx, {N}*dx)',
         f'if(code[{n}]==1, dy, {N}*dy)'), name=f'mp{nx}_{ny}_0')
    hfss.modeler.thicken_sheet(x, f'if(code[{n}]==1, dz1, dz2)')
```



多邊螺旋

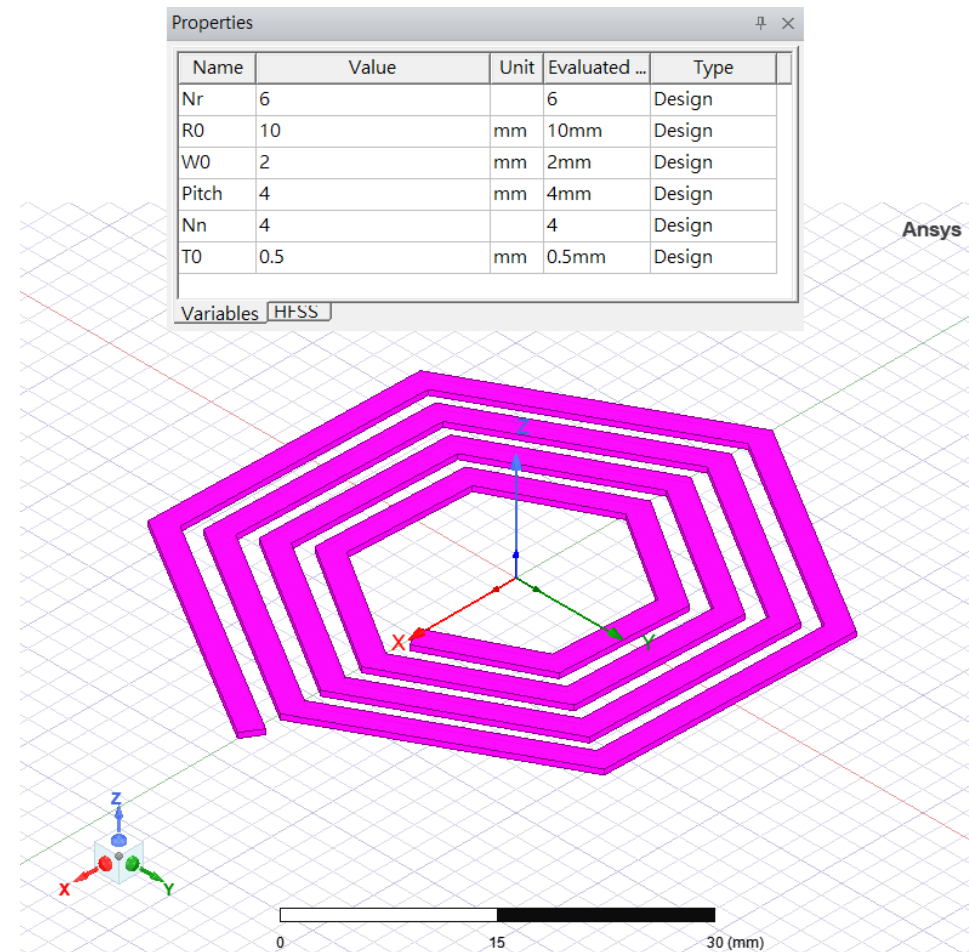
```
from pyaedt import Hfss

hfss = Hfss(specified_version='2022.1', designname='case3')

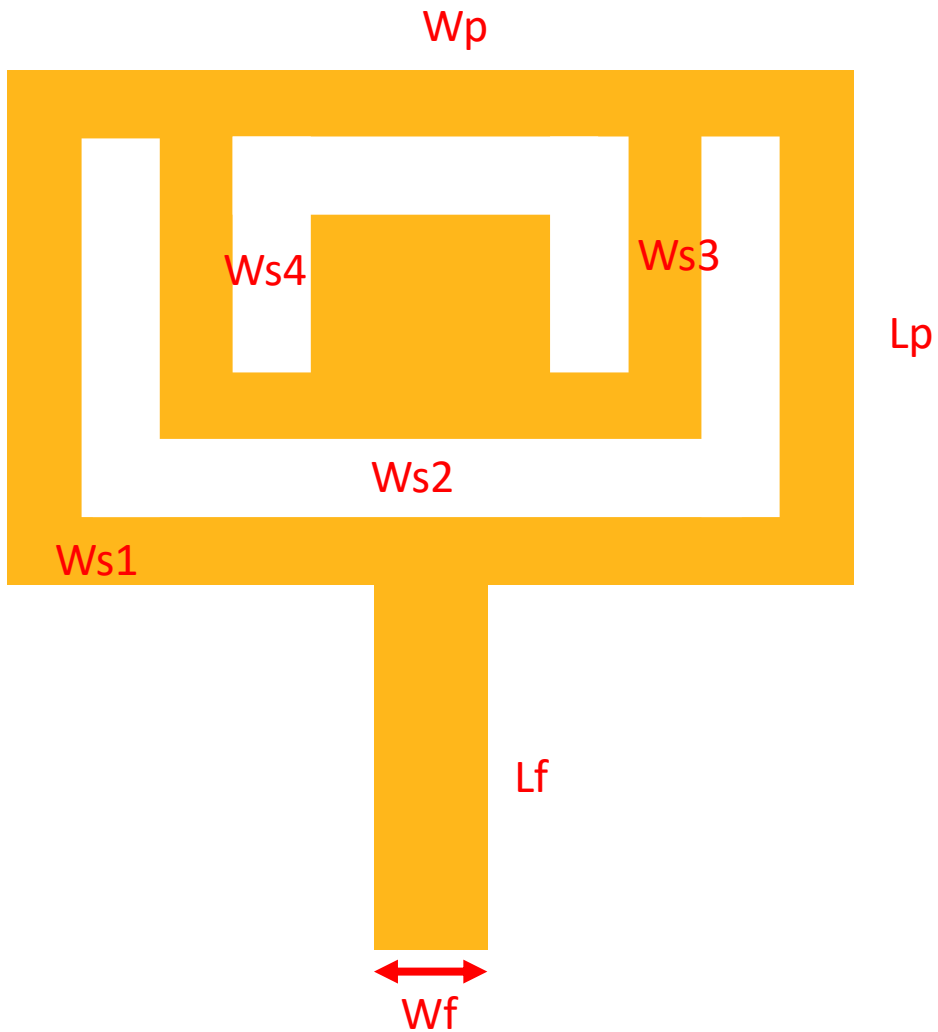
hfss['Nr'] = 6
hfss['R0'] = '10mm'
hfss['W0'] = '2mm'
hfss['Pitch'] = '4mm'
hfss['Nn'] = 4
hfss['T0'] = '0.5mm'

locations = []
for i in range(200):
    locations.append((f'if({i}<Nr*Nn, (R0+Pitch*{i}/6)*cos(2*pi*{i}/Nr),\n(R0+Pitch*Nr*Nn/6)*cos(2*pi*Nn))',\n                    f'if({i}<Nr*Nn, (R0+Pitch*{i}/6)*sin(2*pi*{i}/Nr),\n(R0+Pitch*Nr*Nn/6)*sin(2*pi*Nn))',\n                    '0mm'))

hfss.modeler.create_polyline(locations, xsection_type='Line', xsection_width='W0',\nname='spiral')
hfss.modeler.thicken_sheet('spiral', 'T0')
```



定義變數



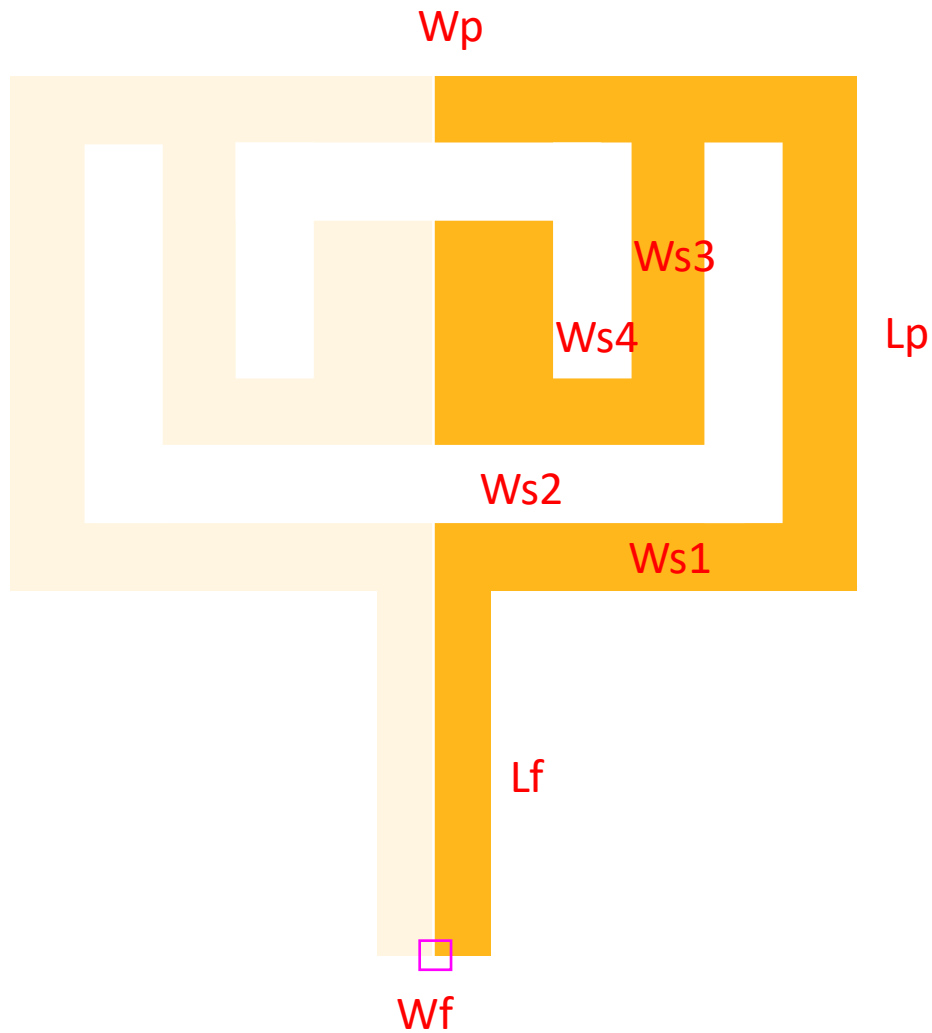
```
#匯入模組
import matplotlib.pyplot as plt
from pyaedt import Hfss
from scipy.optimize import minimize

#開啟HFSS
hfss = Hfss(specified_version='2022.1')

#初始化HFSS參數
hfss['Wp'] = '5.35mm'
hfss['Lp'] = '4.54mm'
hfss['Ws1'] = '0.5mm'
hfss['Ws2'] = '0.5mm'
hfss['Ws3'] = '0.5mm'
hfss['Ws4'] = '0.5mm'
hfss['Wf'] = '1mm'
hfss['Lf'] = '3mm'
hfss['Tm'] = '0.1mm'
hfss['Ts'] = '1mm'

hfss['dx'] = '10mm'
hfss['dy'] = '10mm'
hfss['P'] = '1mm'
```

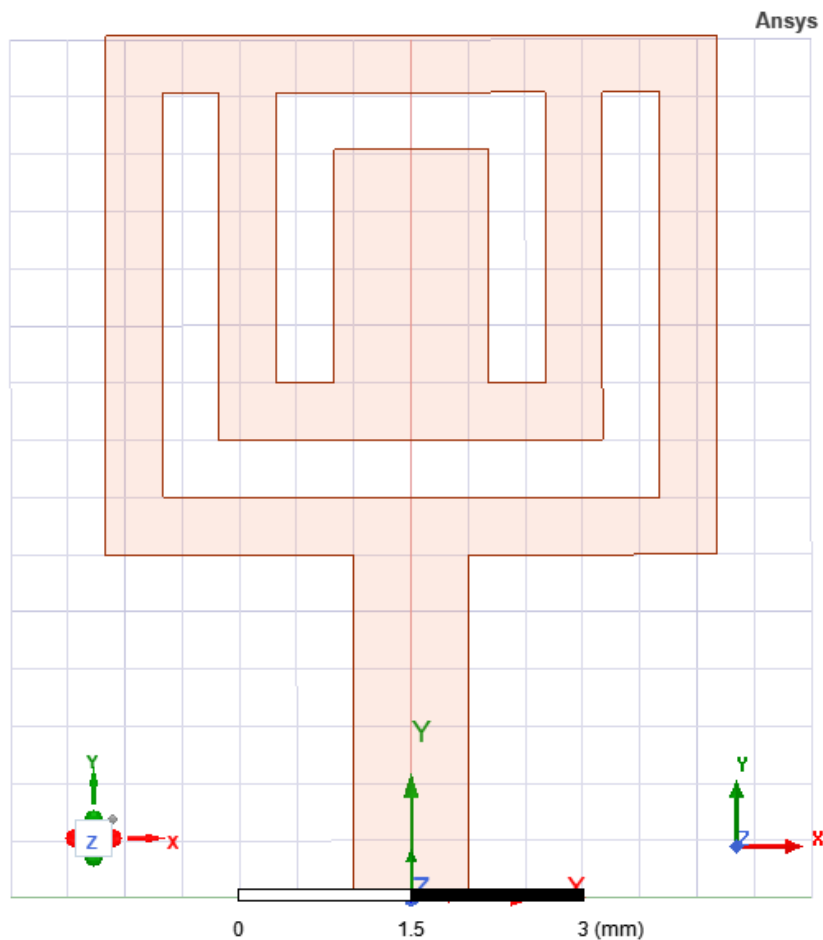
定義參考點與對稱平面



#定義天線右半邊座標點

```
half_patch = [('0mm', '0mm'),  
              ('Wf/2', '0mm'),  
              ('Wf/2', 'Lf'),  
              ('Wp/2', 'Lf'),  
              ('Wp/2', 'Lf+Lp'),  
              ('0mm', 'Lf+Lp'),  
              ('0mm', 'Lf+Lp-Ws1'),  
              ('Wp/2-Ws1-Ws2-Ws3', 'Lf+Lp-Ws1'),  
              ('Wp/2-Ws1-Ws2-Ws3', 'Lf+Ws1+Ws2+Ws3'),  
              ('Wp/2-Ws1-Ws2-Ws3-Ws4',  
               'Lf+Ws1+Ws2+Ws3'),  
              ('Wp/2-Ws1-Ws2-Ws3-Ws4', 'Lf+Lp-Ws1-Ws4'),  
              ('0mm', 'Lf+Lp-Ws1-Ws4'),  
              ('0mm', 'Lf+Ws1+Ws2'),  
              ('Wp/2-Ws1-Ws2', 'Lf+Ws1+Ws2'),  
              ('Wp/2-Ws1-Ws2', 'Lf+Lp-Ws1'),  
              ('Wp/2-Ws1', 'Lf+Lp-Ws1'),  
              ('Wp/2-Ws1', 'Lf+Ws1'),  
              ('0mm', 'Lf+Ws1'),]
```

建立多邊形，鏡像及對稱



```
#加入Z座標點
half_patch = [(i, j, '0mm') for i, j in half_patch]

#產生多邊形
x1 = hfss.modeler.create_polyline(half_patch, cover_surface=True, matname='copper')

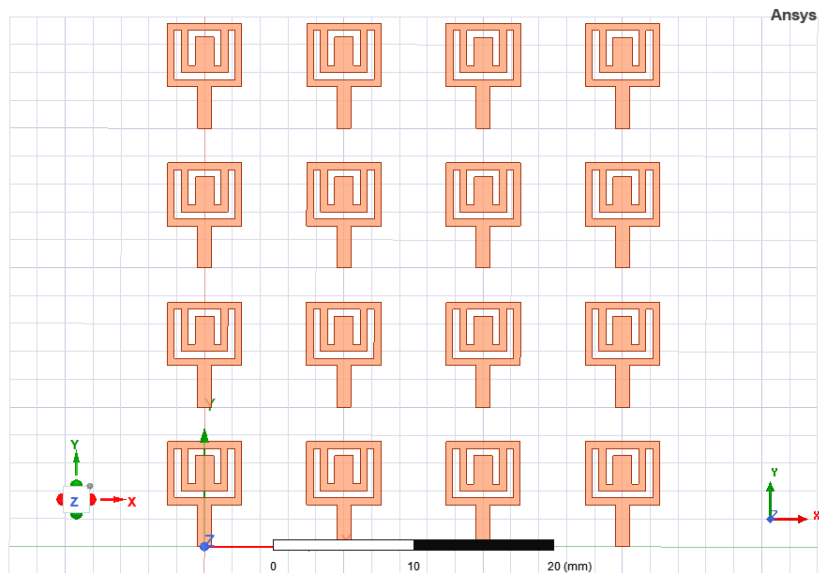
#複製到左半邊
x2 = hfss.modeler.duplicate_and_mirror(x1, (0, 0, 0), (1, 0, 0))

#合併左右半邊
x = hfss.modeler.unite([x1.name, x2[1][0]])

#生成厚度
hfss.modeler.thicken_sheet(x1, 'Tm')

#生成sheet，之後可在其上設port
sheet = hfss.modeler.create_rectangle(1, ('-Wf/2', '0mm', '0mm'), ('-Ts', 'Wf'))
```

建立陣列



#定義陣列大小

Nx = 4

Ny = 4

#複製單元到相對位置

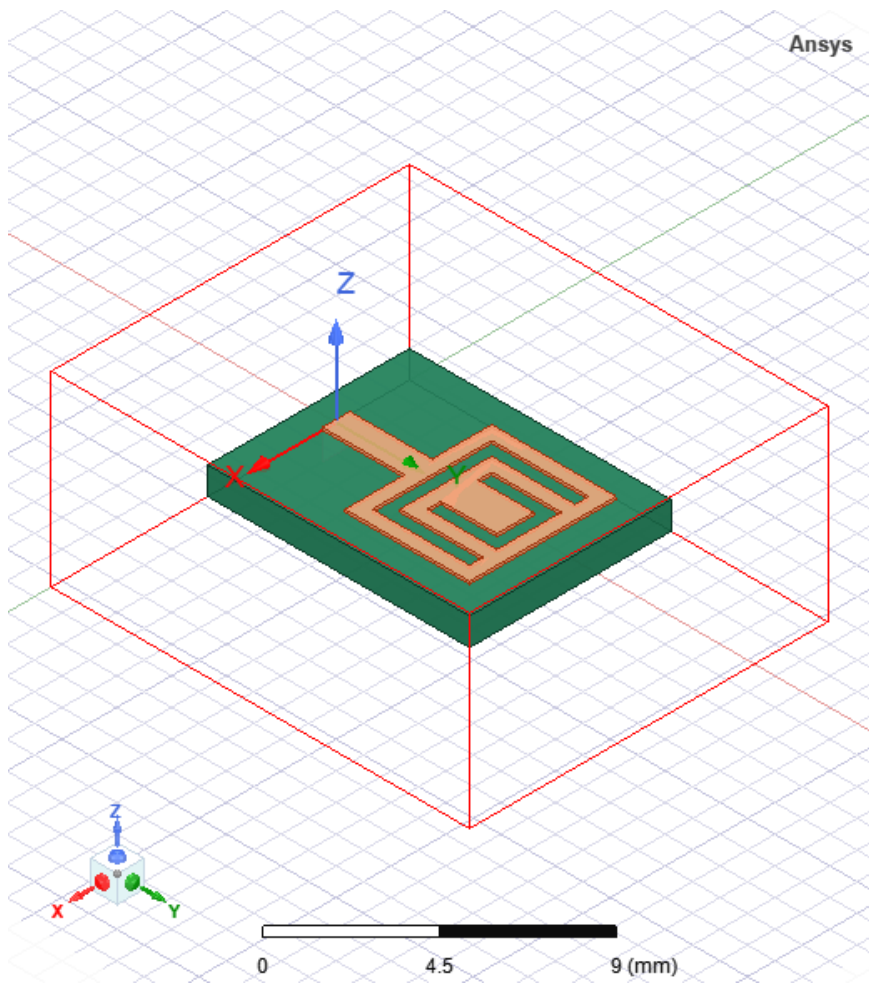
```
for i in range(Nx):
```

```
    for j in range(Ny):
```

```
        if i + j != 0:
```

```
            hfss.modeler.duplicate_along_line([x1.name, sheet.name], (f'dx*{i}', f'dy*{j}', '0mm'))
```

設定基板、邊界條件與Ports



```
#設定基板
p0 = ('-Wp/2-P', '-P', '0mm')
size = (f'({Nx}-1)*dx+Wp+2*P', f'({Ny}-1)*dy+(Lf+Lp)+2*P', '-Ts')
substrate = hfss.modeler.create_box(p0, size, matname='FR4_epoxy')

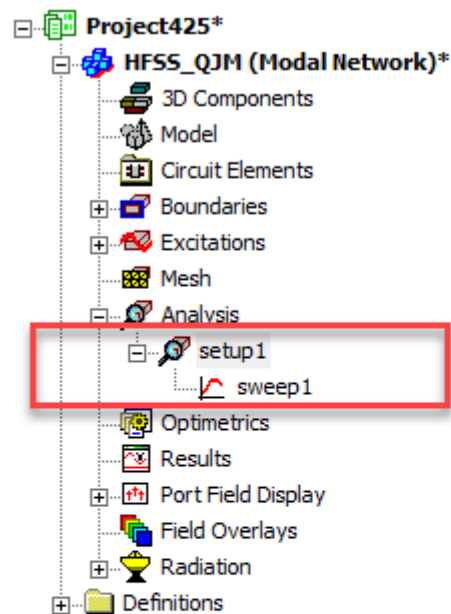
#將基板底部設定為PEC
hfss.assign_perfecte_to_sheets(substrate.bottom_face_z)

#找出所有sheets並在上面設定ports
for n, i in enumerate(hfss.modeler.get_objects_in_group('Sheets'), 1):
    port = hfss.create_lumped_port_to_sheet(i, 2, portname=f'port{n}')

#設定Open Region
hfss.create_open_region(fc)
```

設定、模擬與優化

建立Setup與Sweep



#設定基板

```
p0 = ('-Wp/2-P', '-P', '0mm')
size = (f'({Nx}-1)*dx+Wp+2*P', f'({Ny}-1)*dy+(Lf+Lp)+2*P', '-Ts')
substrate = hfss.modeler.create_box(p0, size, matname='FR4_epoxy')
```

#將基板底部設定為PEC

```
hfss.assign_perfecte_to_sheets(substrate.bottom_face_z)
```

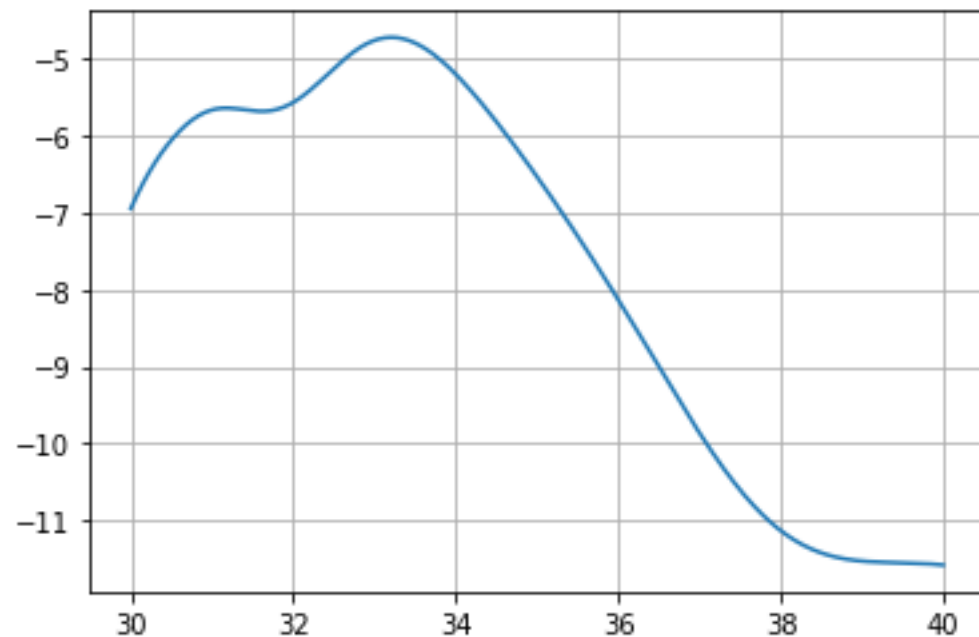
#找出所有sheets並在上面設定ports

```
for n, i in enumerate(hfss.modeler.get_objects_in_group('Sheets'), 1):
    port = hfss.create_lumped_port_to_sheet(i, 2, portname=f'port{n}')
```

#設定Open Region

```
hfss.create_open_region(fc)
```


啟動模擬並抓取資料



```
#啟動模擬，可指定CPU核心數
hfss.analyze_nominal(4)

#抓取資料
RL = hfss.post.get_report_data('dB(S11)')
y = RL.data_real()
x = RL.sweeps['Freq']
plt.plot(x, y)
plt.grid()
plt.show()
```

內部優化演算法

定義目標

```
optim = hfss.optimizations.add(calculation="dB(S(1,1))", ranges={"Freq": fc}, condition='<', goal_value=-20)
optim.props['AnalysisStopOptions']['MaxNumIteration'] = 20
```

#定義數值範圍

```
optim.add_variation('Ws1', 0.35, 0.65)
optim.add_variation('Ws2', 0.35, 0.65)
optim.add_variation('Ws3', 0.35, 0.65)
optim.add_variation('Ws4', 0.35, 0.65)
```

#啟動模擬

```
hfss.analyze_setup(optim.name, num_cores=4, num_tasks=4)
```

外部優化

```
# 定義目標函數
def target(parameters):
    print(parameters)
    Ws1, Ws2, Ws3, Ws4 = parameters
    hfss['Ws1'] = f'{Ws1}mm'
    hfss['Ws2'] = f'{Ws2}mm'
    hfss['Ws3'] = f'{Ws3}mm'
    hfss['Ws4'] = f'{Ws4}mm'

    hfss.analyze_nominal(4)

    RL = hfss.post.get_report_data('dB(S11)')
    y = RL.data_real()
    x = RL.sweeps['Freq']
    plt.title(str(parameters))
    plt.plot(x, y)
    plt.grid()
    plt.show()

    result = hfss.post.get_report_data('dB(S11)')
    print(result.data_real()[0])
    return result.data_real()[0]

# 初始值
x0 = [0.5, 0.5, 0.5, 0.5]

# 數值範圍
bnds = ((0.35, 0.65), (0.35, 0.65), (0.35, 0.65), (0.35, 0.65))

# 啟動優化
sol = minimize(target, x0, bounds=bnds, tol=1e-6, options={'maxiter': 100, 'disp': True})
print(sol)
```

 **Ansys**

