

# ANSYS AEDT程式開發

## 目錄

- Chapter 1. 概要
- Chapter 2. Python語法介紹
- Chapter 3. 模擬資料處理
- Chapter 4. 資料處理範例
- Chapter 5. AEDT腳本開發
- Chapter 6. 開發流程範例
- Chapter 7. PyAEDT介紹
- Chapter 8. 用PyAEDT完成完整模擬
- Chapter 9. Python 進階
- Chapter 10. 附錄

所有權利保留。本書的任何部分未經出版社和作者書面允許，不得以任何形式或手段（電子、機械、影印、錄音或其他方式）複製、傳播或儲存於檢索系統。

# 第1章 概要

## 1.1 模擬工程面臨的挑戰與解決方案

在當今高度競爭的市場中，設計和優化產品已經成為企業贏得市場份額的關鍵。

ANSYS作為一家提供工程模擬工具和服務的公司，工具涵蓋有限元分析、流體力學分析、熱傳分析、電磁場分析、優化分析、多物理場仿真分析、系統仿真分析、材料仿真分析等多個領域。ANSYS的產品和服務在汽車、航空航天、醫療、能源、電子等行業得到廣泛應用。

然而，在競爭激烈且不斷變化的市場中，客戶對軟體的需求和期望變得越來越複雜和多樣化，這對工程模擬軟體的開發帶來了重大挑戰。例如，半導體業者使用ANSYS電磁模擬軟體進行先進封裝設計，天線業者則利用該軟體開發陣列天線，而系統業者則運用該軟體進行EMI/EMC研究。不同領域希望軟體能夠包含各自所需的功能，但這些功能可能僅適用於特定領域，對其他行業而言，可能並不需要。這需要軟體開發人員投入大量時間研究特定領域並開發相應功能，這不僅消耗大量資源，還可能導致軟體變得臃腫，進而影響其穩定性。

從使用者角度看，工程軟體無法及時滿足需求也是一個令人困擾的問題。例如，毫米波陣列天線的研發工程師需要在短時間內對模擬出來的電磁波場型進行大量波束資料計算以便產生提交給通訊協會的報告。這個功能對封裝模擬或EMI/EMC模擬並無幫助，但對天線工程師來說卻是極為重要。如果軟體未提供此功能，天線研發進度將受到嚴重影響，導致巨大損失。

### 讓業界專家加入模擬工程應用開發

對ANSYS與使用ANSYS工程軟體的使用者來說，隨著技術的深入與分化，這樣的難題會不斷的出現。這困境僅單純依靠ANSYS是難以獨立解決的，需要一個具有創新思維的作法來突破這些困境，ANSYS提出了一個大膽的創新作法：引入外部各領域的技術專家參與模擬工程應用軟體的開發。通過業界專家根據特定應用調整最適當的設置並重新打包使用介面，生成新的應用軟體。使用者只需要簡單輸入必要的參數就能得到可信賴且有用的結果，從而大大提高使用者的生產力。這個作法既能滿足不同專業領域的模擬需求，也能加速工程業的設計創新。

以手機為例，手機充當了一個平台的角色，提供了陀螺儀、螢幕、攝像頭、麥克風、擴音器、網路等硬體以及相應的API（應用程式接口，Application Programming Interface）。應用開發者可以在這個平台上開發各種APP，以滿足使用者多樣化的需求，例如地圖導航、短視頻錄製和社交媒體等。同樣地，ANSYS計畫搭建一個工程模擬平台，提供多樣模擬核心引擎，如電磁模擬、流體力學模擬、光學模擬、結構力學模擬等等，以及相應的API。在這個平台上，應用開發者可以根據不同領域的需求，開發出各種定制的應用軟體，滿足使用者的多元化需求。

這種類似於手機平台的策略，讓ANSYS可以專注於為各行業提供強大且靈活的核心功能，而業界專家和開發者則可以在此基礎上創建針對特定需求的應用軟體。如此一來，使用者能夠輕鬆地獲得適合自己需求的解決方案，提高工作效率，同時也滿足各個行業的特殊需求。這種模式有助於推動工程模擬軟體市場的創新與發展，為各行業提供更專業、更高效的解決方案，同時降低開發成本並提高靈活性。

## 開發者社群應運而生

為此，ANSYS提出的方案是建立開發者社群，廣邀業界專家加入應用的開發，希望能利用他們的知識、數據和經驗結合ANSYS的模擬軟體，開發出針對特定問題的定制化解決方案。具體做法是開放ANSYS軟體的接口PyANSYS並成立ANSYS Developer入口網站。

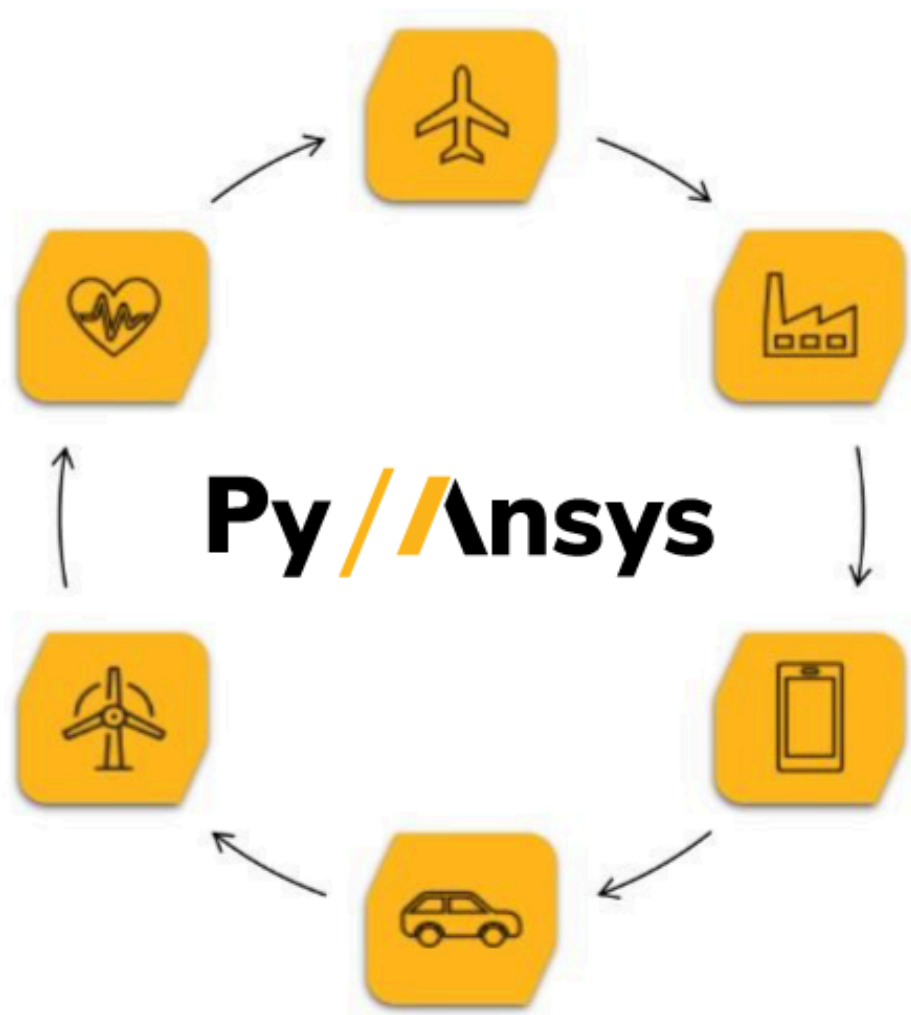
應用開發者可以是業界專家、學術界研究人員或對工程模擬有濃厚興趣的個人。通過軟體開放接口和ANSYS開發者社群提供的豐富資源，開發者可以高效地開發出各種定制化的應用。開發者可以向使用應用的企業收取費用，這成為開發者獲利的一種方式。領域專家可以通過開發解決方案獲得報酬，企業使用這些應用可以提高生產效率和產品質量，ANSYS則可以服務更多使用者，擴大應用範圍。如此一來，企業、開發者和ANSYS三方都可以從中獲利。

另一種使用場景是公司內部資深的模擬工程師能夠將日常的模擬工作包裝成不同的網頁應用，這樣部門內的工程師可以透過瀏覽器上傳設計檔案到遠端伺服器，由PyANSYS程式自動完成設置、執行模擬並輸出報告。這樣的做法不僅可以解決模擬人才短缺的問題，也可以大幅提高部門內的生產力，減少繁瑣的手動操作，並節省大量的時間和成本。

過去，模擬軟體的使用通常需要高度專業的技能 and 知識，這限制了大多數使用者的使用。應用的出現將可以讓更多的不懂模擬技巧的使用者也能夠輕鬆地進行工程仿真。

這種民主化的應用方式，使得工程仿真和設計不再是高門檻的專業領域，而是變得更加容易和普及化。

## PyANSYS生態系統



PyANSYS 生態系統

**PyAnsys**計畫是一系列Python套件的集合，用於透過Python操控Ansys產品。這個計畫最初源自單一的pyansys套件，現在已經發展成許多不同的套件，包括：

1. **PyAdditive**：用於Ansys Additive服務的Python介面。
2. **PyAEDT**：用於AEDT（Ansys電子桌面）的Python介面。
3. **PyAnsys Geometry**：用於Ansys幾何服務的Python介面。
4. **PyAnsys Math**：用於PyAnsys數學庫的Python介面。
5. **PyDPF-Core**：用於DPF（數據處理框架），建立更高級和定制的工作流程的Python介面。
6. **PyDPF-Post**：用於訪問和後處理Ansys求解器結果文件的Python介面。
7. **PyDPF-Composites**：用於後處理分層和短纖維複合材料模型的Python介面。

8. **PyDyna** : 用於構建Ansys DYNA輸入牌組、提交到Ansys LS-DYNA求解器及其結果後處理的Python介面。
9. **PyDynamicReporting** : 用於Ansys動態報告服務及其數據庫和報告的控制的Python介面。
10. **PyEnSight** : 用於EnSight · Ansys模擬後處理器的Python介面。
11. **PyFluent** : 用於Ansys Fluent的Python介面。
12. **PyFluent - Parametric** : 用於Ansys Fluent參數化工作流程的Python介面。
13. **PyFluent - Visualization** : 用於可視化Ansys Fluent模擬的Python介面。
14. **PyMAPDL** : 用於Ansys MAPDL ( 機械APDL ) 的Python介面。
15. **PyMAPDL Reader** : 用於讀取舊版MAPDL結果文件 ( MAPDL 14.5及以後版本 ) 的Python介面。
16. **PyMechanical** : 用於Ansys Mechanical的Python介面。
17. **PyMotorCAD** : 用於Ansys Motor-CAD的Python介面。
18. **PyOptislang** : 用於Ansys Optislang的Python介面。
19. **PyPIM** : 用於與Ansys PIM ( 產品實例管理 ) API通訊的Python介面。
20. **PyPrimeMesh** : 用於Ansys Prime Server , 提供核心Ansys網格技術的Python介面。
21. **PySeascape** : 用於與Ansys RedHawkSC和TotemSC通訊的Python介面。
22. **PySherlock** : 用於與Ansys Sherlock通訊的Python介面。
23. **PySystem Coupling** : 用於與Ansys系統耦合通訊的Python介面。
24. **PyTurbogrid** : 用於Ansys TurboGrid , 一個高品質渦輪機械網格軟件應用的Python介面。
25. **PyTwin** : 用於與Ansys數字雙胞胎消費工作流程通訊的Python介面。
26. **Granta MI BoM Analytics** : 用於Ansys Granta MI BoM分析服務的Python介面。
27. **Granta MI RecordLists** : 用於Ansys Granta MI列表API的Python介面。
28. **Shared Components** : 共享Ansys軟件組件 , 以實現套件互操作性並最小化維護

# 1.2 Python + AEDT模擬

利用 Python 結合 Ansys Electronics Desktop (AEDT) 在工程領域中帶來的諸多優勢。以下是進一步解釋和補充：

## 1. 開發工具包

- **定制化工具**：Python 的豐富函式庫允許開發者創建定制化的工具包，這些工具包可以直接與 AEDT 接口，實現特定的工作流程和功能。
- **增強用戶體驗**：這些工具包可以整合到 AEDT 的用戶界面中，使得操作更加直觀，同時提供更豐富的功能，如自動化數據輸入、複雜參數設定等。

## 2. 自動化模擬工作

- **減少重複工作**：通過 Python 腳本自動化建模、模擬設定和運行過程，可以顯著降低重複性工作，提高整體效率。
- **快速迭代和優化**：自動化可以更快地進行模型迭代和參數調整，從而快速找到最優解。

## 3. 模擬數據處理與分析

- **強大的數據分析能力**：Python 的數據處理庫（如 Pandas）可以用來清洗和轉換模擬數據，而可視化庫（如 Matplotlib 和 Seaborn）則能夠直觀地展示數據，幫助更好地理解 and 解釋模擬結果。
- **交互式分析環境**：結合 JupyterLab，Python 提供了一個交互式數據分析環境，這對於探索性數據分析和數據呈現尤為有用。

## 4. 設計探索

- **優化和參數探索**：利用 Python 的優化庫（如 Scipy）可以進行複雜的設計優化和參數探索，從而找到最佳設計方案。
- **預測和機器學習**：結合機器學習庫（如 scikit-learn），可以開發出能夠預測設計性能的模型，這對於預先評估設計選擇的效果十分重要。

## 綜合優勢

結合 Python 和 AEDT，工程師不僅能夠提高模擬工作的效率，還能進行更深入的數據分析和優化，這有助於做出更科學、更準確的設計決策。這種組合能夠在複雜的工

程項目中節省大量時間和資源，並提高整體設計質量。



## 1.3 工程模擬腳本開發

### 工程物理模擬

工程物理模擬是一種使用計算機模型來模擬工程問題的技術。這種方法允許工程師在建造或製造之前，就能在虛擬環境中測試和優化設計。工程模擬的應用範圍廣泛，包括但不限於結構分析、流體動力學、熱傳遞、電磁學等。

以下是幾個關鍵點：

- 結構分析 ( **Structural Analysis** )：模擬建築物、橋樑、車輛等結構在各種載荷下的行為。這包括應力、變形和振動分析。
- 流體動力學 ( **Computational Fluid Dynamics, CFD** )：模擬氣體和液體的流動特性，應用於航空航天、汽車設計、氣象預報等。
- 熱傳遞模擬：分析和預測熱在物體之間的傳遞過程，重要於能源系統、電子散熱等領域。
- 電磁場模擬：用於分析電磁波的傳播，應用於通信、雷達和微波工程等領域。

工程模擬通常依賴於複雜的數學模型和算法，並需要大量的計算資源。軟體工具如 **ANSYS**、**COMSOL Multiphysics** 等廣泛應用於這些模擬中。這些模擬可以提高設計的效率和安全性，減少物理測試的需要，並幫助工程師更好地理解 and 優化他們的設計。

### 關於ANSYS

**ANSYS** 是一家著名的工程模擬軟體開發公司，其總部位於美國賓夕法尼亞州。自 1970 年代末成立以來，**ANSYS** 一直是工程模擬領域的先驅和領導者。這家公司提供廣泛的軟體工具，用於幫助工程師和科學家在虛擬環境中模擬各種工程設計和分析。

以下是 **ANSYS** 公司的幾個主要特點：

- 廣泛的產品線：**ANSYS** 提供多種模擬工具，包括結構分析 ( 如 **ANSYS Mechanical** )，流體動力學 ( 如 **ANSYS Fluent** )，電磁場分析 ( 如 **ANSYS Maxwell** )，以及多物理場交互分析。
- 跨學科應用：**ANSYS** 的軟體廣泛應用於航空航天、汽車、電子、能源和醫療等多個行業。它們幫助設計更安全的飛機，更高效的汽車，以及更先進的醫療設備。

- 創新和技術領先：**ANSYS**致力於持續創新和改進其模擬技術，以滿足不斷變化的工程需求。它們在模擬精確度、用戶友好性和計算效率方面不斷取得進步。
- 客戶支持和培訓：除了提供軟體，**ANSYS**還為客戶提供廣泛的技術支持、培訓和諮詢服務，以確保用戶能夠最大限度地利用其軟體。

總之，**ANSYS**是工程模擬領域的重要玩家，其軟體和服務對於推動設計創新和提高工程效率具有重要意義。

**ANSYS**的官方網站為 [www.ansys.com](http://www.ansys.com)。在這個網站上，你可以找到關於他們的多物理軟體解決方案和數位任務工程的資訊，這些工具幫助企業以新穎的方式進行創新和驗證。這個網站提供了關於**ANSYS**產品的詳細資訊，包括免費試用和查看所有產品的選項。

## AEDT設計模擬平台

**ANSYS Electronics Desktop (AEDT)** 是一個集成的電磁、電路和系統模擬平台，專為解決從射頻微波、天線到高速數字電子設計的各種挑戰而設計。**AEDT** 提供了多種工具，包括**ANSYS HFSS**、**ANSYS Maxwell**、**ANSYS Q3D Extractor**、**ANSYS SIwave**和**ANSYS Icepak**等。可用於分析電磁場、信號完整性、功率完整性、熱傳遞和振動問題，適用於各種應用，從消費電子到航空航天和防禦系統。

以下是 **AEDT** 的一些主要特點和功能：

- 多物理場仿真：**AEDT** 結合了電磁場模擬、電路模擬和系統級模擬。這樣可以在同一個平台上分析電磁場與電路和系統性能之間的交互作用。
- 電磁場模擬：**AEDT** 包括射頻和微波設計的模擬工具，如 **HFSS** ( 高頻結構模擬器 )，專門用於三維全波電磁場模擬。
- 信號完整性分析：**AEDT** 提供了工具來分析高速電子系統中的信號完整性問題，例如信號衰減、串擾和反射。
- 功率完整性分析：用於分析電子設備中的電源分配網絡 ( **PDN** )，確保電子系統中的穩定電源供應。
- 熱分析：**AEDT** 允許分析電子設備的熱特性，幫助設計更有效的散熱解決方案。
- 用戶友好的界面：**AEDT** 擁有直觀的用戶界面，使工程師可以輕鬆設置和管理複雜的模擬。
- 集成設計流程：**AEDT** 支持與其他 **ANSYS** 工具的集成，如結構分析和流體動力學模擬，從而實現跨學科的設計和分析。

這個平台適合於需要精密電磁場模擬的高科技產品開發，如衛星通訊、高速數據傳輸設備和精密儀器等。使用 **AEDT**，工程師可以更精準地預測產品性能，優化設計，並加速產品上市時間。

## 工程模擬流程

模擬流程通常涉及一系列的步驟，從問題定義到最終結果分析。以下是一個典型的工程模擬流程，這裡以**ANSYS**或類似的工程模擬軟體為例：

### 1. 問題定義：

- 確定要解決的工程問題。
- 明確模擬的目標和預期結果。

### 2. 預處理：

- **幾何建模**：建立或導入模擬的物體或系統的幾何模型。
- **網格生成**：將幾何模型劃分為更小的單元（網格），以便進行數值計算。網格的質量和細度對模擬結果的準確性有重要影響。

### 3. 設置物理參數和邊界條件：

- 定義材料特性（如密度、彈性模量、導熱系數等）。
- 設置邊界條件（如固定點、負載、溫度、壓力等）和初始條件。

### 4. 選擇和設置求解器：

- 選擇適當的數學模型和求解器（如靜態、動態、線性、非線性等）。
- 設定求解過程的參數，如迭代次數、收斂標準等。

### 5. 運行模擬：

- 啟動計算過程，求解器將根據設定的物理模型和數學算法計算結果。
- 監控計算過程，必要時調整設定以確保模擬的準確性和效率。

### 6. 後處理和結果分析：

- 評估和解釋計算結果，如位移、應力、溫度分佈等。
- 使用圖形化工具顯示結果，如等值線圖、矢量圖、動畫等。

- 根據需要進行數據後處理，如結果整合、極值分析等。

## 7. 驗證和調整：

- 對模擬結果進行驗證，可能通過實驗數據或理論分析。
- 根據驗證結果調整模型或模擬設定，以提高準確性。

## 8. 報告和決策支持：

- 編寫模擬報告，包括方法、結果和結論。
- 將模擬結果用於設計決策、性能優化或問題診斷。

在整個模擬流程中，重要的是要保持對問題的清晰理解，並根據模擬結果不斷迭代和完善模型。此外，模擬的準確性極大地依賴於初始模型的準確性、物理參數的正確設定以及求解過程的控制。

## 開發者所需技能

進行工程模擬軟體的二次開發，如ANSYS或其他專業模擬工具，需要一系列專業技能和知識。以下是一些關鍵的技能和知識領域：

### 1. 深厚的專業知識：

- 對相關工程領域（如機械、電子、流體力學等）有深入理解。
- 理解模擬的物理原理和數學模型。

### 2. 編程和腳本語言技能：

- 熟悉一種或多種編程語言，如Python、C++、MATLAB等，這些語言常用於定制算法和工具。

### 4. 軟體開發能力：

- 了解軟體開發的基本原則，包括數據結構、算法、軟體設計模式等。
- 能夠設計和實現用戶友好的介面和工具。

### 6. 問題解決和創新思維：

- 能夠識別和解決在模擬過程中遇到的具體問題。

- 創新思維，不斷尋求改進模擬效率和準確性的方法。

## **7. 項目管理和溝通能力：**

- 能夠有效管理二次開發項目，包括時間管理、資源協調等。
- 良好的溝通能力，能夠與團隊成員、客戶和利益相關者有效溝通。

二次開發不僅要求技術專長，還需要對特定應用領域有深入理解。此外，由於這是一個不斷發展的領域，持續學習和適應新技術和方法也非常重要。

## 1.4 Python基本

Python 是一種高級的、通用的程式語言，由 Guido van Rossum 在 1991 年創建。它以語法簡單、易讀且清晰為特色，並且強調程式碼的可讀性。Python 支援多種編程範式，包括程序化（procedural）、物件導向（object-oriented）和函數式（functional）程式設計。

Python 提供了大量的內建資料型態，包括列表（list）、字典（dictionary）、集合（set）和元組（tuple）。此外，Python 的標準庫包含了許多有用的功能，包括檔案 I/O、網路通訊、日期和時間處理、數據序列化等等。Python 的應用範圍非常廣泛，包括 Web 開發、數據分析、機器學習、AI、科學計算、網路爬蟲、遊戲開發、桌面應用等等。

Python 的生態系統強大且成熟，有許多第三方套件可以用於各種任務，例如 NumPy 和 Pandas 用於數據分析，Django 和 Flask 用於 Web 開發，TensorFlow 和 PyTorch 用於機器學習等等。

Python 的社區非常活躍，有大量的學習資源、線上論壇和使用者群組。Python 的開發和維護由 Python 軟體基金會（Python Software Foundation）負責。在許多大學和線上課程中，Python 也被廣泛地用作教授程式設計的入門語言。

### 優點

- 易於學習和使用**：Python 擁有清晰、直觀且閱讀性強的語法，這使得它成為初學者學習編程的理想選擇。
- 廣泛的應用範圍**：Python 在網絡開發、數據科學、人工智能、機器學習、科學計算等多個領域都有廣泛應用。
- 豐富的庫和框架**：Python 有一個龐大的標準庫，以及大量的第三方庫和框架，如 Django（網絡開發）、Pandas（數據分析）、NumPy（科學計算）等。
- 跨平台兼容性**：Python 可以在多種作業系統上運行，包括 Windows、macOS、Linux 等。
- 高效的代碼開發**：Python 使開發者能夠用更少的代碼完成任務，提高開發效率。
- 社群支持強大**：Python 擁有一個活躍且支持性強的開發者社群。

## 缺點

1. **執行速度相對較慢**：由於 Python 是一種解釋型語言，其執行速度通常比編譯型語言（如 C、C++）慢。
2. **記憶體消耗**：Python 的記憶體消耗相比於一些更低層的語言要高，這可能是在大型應用或系統級開發中的一個考慮因素。
3. **多線程的限制**：由於全局解釋器鎖（GIL），Python 的多線程並不能實現真正的並行執行（尤其是在 CPython 中）。

## Python實現

Python 是一種通用的程式語言，其設計理念是可移植性和易於擴展。這意味著 Python 理論上可以被實作（或“移植”）到不同的環境和平台中。不同的 Python 實現（如 CPython, Jython, IronPython 等）就是這一理念的體現，它們使 Python 能夠在各種系統和架構上運行。

談到 Python 的標準庫和第三方模組：

- **標準庫**：Python 的標準庫提供了一系列基本功能，這些功能在大多數 Python 實現中都是可用的。這些標準庫模組通常被認為是 Python 語言的一部分，並且在不同的實現之間保持相對一致。然而，某些模組可能會因為依賴特定平台的特性（比如操作系統特定的功能）而在某些實現中無法使用或有所差異。
- **第三方模組**：Python 的一個重要特色是其龐大的第三方模組生態系統。這些模組擴展了 Python 的功能，允許它處理各種各樣的任務，從網絡編程到數據科學。然而，並非所有第三方模組都能在所有 Python 實現上運行。有些模組可能只與特定的 Python 實現（如 CPython）兼容，特別是那些包含 C 語言擴展的模組。

由於不同的 Python 實現可能會對標準庫和第三方庫的支持有所不同，因此在開發跨平台應用時，了解這些差異是非常重要的。特別是在涉及到特定系統調用或平台依賴特性時，這些差異可能會成為關鍵因素。Python 的實現有幾種不同的方式，主要體現在不同的解釋器上。每種解釋器都有其獨特的特點和適用場景。以下是一些主要的 Python 實現：

- **CPython**: 這是最常見的 Python 實現，由 Guido van Rossum（Python 的創建者）主導開發。它用 C 語言編寫，提供了 Python 的標準實現和擴展。

- **Jython**: 這個實現是為了在 Java 平台上運行 Python 程式。Jython 把 Python 代碼編譯成 Java 字節碼，允許 Python 程式直接調用 Java 類庫和物件。
- **IronPython**: 主要用於 .NET 環境，允許 Python 腳本與 .NET 框架進行交互。它把 Python 代碼轉換成 .NET 中間語言 ( MSIL ) 。
- **PyPy**: 這是一個以速度為目標的 Python 實現，使用稱為 RPython ( 一種靜態子集的 Python ) 的語言開發。PyPy 通過即時編譯 ( JIT ) 來提高 Python 程序的執行效率。
- **MicroPython**: 這是專為微控制器和嵌入式系統設計的 Python 實現。它對 Python 標準庫進行了精簡，以適應資源有限的環境。

AEDT在介面運行 Python 腳本時使用的是 IronPython 解釋器，而在透過集成開發環境 ( IDE ) 執行與 AEDT 連接的腳本時則使用 CPython。這種情況下，理解這兩種 Python 解釋器之間的差異以及它們如何與 AEDT 互動非常重要。

## Python軟體基金會

Python軟體基金會 ( Python Software Foundation, 縮寫為 PSF ) 是一個非營利的組織，致力於推廣和發展Python程式語言。PSF成立於2001年，其目標是推廣、保護並提升Python程式語言的發展，並支援和鼓勵Python的全球社區。

PSF的主要活動包括：

- 管理和保護Python的智慧財產權：PSF擁有Python的版權，並負責保護Python的商標。
- 贊助Python相關的活動和項目：PSF提供資助給各種Python相關的活動，包括研討會、教育項目和開源項目。
- 組織Python的開發和維護：PSF支持Python的核心開發團隊，並管理Python的開發流程。
- 提供社區支援：PSF鼓勵和協助Python的社區發展，包括用戶組織、在線社區和教育活動。

任何人都可以成為PSF的成員，並參與到PSF的活動中。PSF的運作主要依賴於贊助、捐款以及成員的志願工作。

Python Software Foundation (PSF) 的官方網站是 [Python.org](https://python.org)。PSF 的使命是促進、保護和推進 Python 編程語言的發展，同時支持和促進一個多元化和國際化



## 關於PyPI

PyPI ( Python Package Index ) 是 Python 的官方第三方軟件庫，用戶可以從中下載和安裝各種 Python 軟件包。以下是 PyPI 的主要特點和功能：

1. **豐富的軟件庫**：PyPI 擁有成千上萬的 Python 軟件包，這些包覆蓋了從數據分析、網頁開發到機器學習等各種應用領域。
2. **易於安裝**：使用 Python 的包管理工具（如 pip）可以輕鬆地從 PyPI 安裝軟件包。通常只需一條簡單的命令即可完成安裝。
3. **社區驅動**：PyPI 是一個社區驅動的平台，許多開發者在此分享他們的作品，並對其他人的項目做出貢獻。
4. **版本控制**：在 PyPI 上，軟件包可能有多個版本可用，使用者可以根據自己的需求選擇安裝特定版本的包。
5. **開源**：絕大多數 PyPI 上的軟件包是開源的，這意味著用戶可以自由地使用、修改和分發這些軟件。
6. **依賴管理**：當安裝一個軟件包時，如果該包依賴於其他包，pip 會自動安裝所需的所有依賴包。
7. **安全性**：PyPI 積極管理軟件包的安全性，但用戶在安裝和使用時仍需要注意檢查軟件包的可信度和安全性。

使用 PyPI，Python 開發者能夠輕鬆地訪問、分享和利用各種有用的資源，從而提高開發效率和促進項目進展。

Python Package Index (PyPI) 的官方網站是 [pypi.org](https://pypi.org)。PyPI 是一個為 Python 編程語言提供的軟件庫，用戶可以在這裡查找、安裝和發佈 Python 軟件包。

## 1.5 Python開發環境

下載Python時應考慮以下幾點：

1. **選擇合適的版本**：考慮使用最新穩定版本以獲得最新功能和安全修復。如果有特定需求（如兼容性），可能需要選擇特定版本。
2. **操作系統兼容性**：確保下載與你的操作系統兼容的版本（Windows、macOS或Linux）。
3. **環境配置**：安裝時選擇將Python添加到環境變量，這樣可以在命令行中直接使用。
4. **安全性**：僅從官方網站或可信來源下載。
5. **安裝前的準備**：確認系統是否已經安裝了Python，避免重複安裝或版本衝突。
6. **官方文檔**：查看官方文檔以獲取安裝和配置的詳細指南。

記得在進行安裝前，閱讀相關的發布說明和安裝指南，以確保順利安裝並配置Python環境。

- [Python官方下載頁面](#) 是下載Python的主要來源。在這個頁面上，你可以找到所有可用的Python版本，包括最新的穩定版本和舊版本。這個頁面會根據你的操作系統自動推薦合適的Python版本，同時也提供了其他操作系統版本的下載選項。此外，該頁面還提供了每個版本的詳細發布說明和安裝指南，這對於了解版本間的差異和新功能特別有幫助。建議在下載前閱讀相關的發布說明，以確保選擇適合你需求的Python版本。

## 解釋器與包

在Python中，「解釋器（Interpreter）」和「包（Packages）」是兩個核心概念，它們之間有著密切的關係。

### Python 解釋器：

- 解釋器是運行Python代碼的環境。它讀取你的Python程式碼，並將其轉換成計算機可以執行的指令。

- 常見的 Python 解釋器有 CPython ( 官方提供，用 C 語言編寫 )、Jython ( 運行在 Java 平台 )、IronPython ( 運行在 .NET 和 Mono 平台 ) 等。

## Python 包：

- 包是一種封裝 Python 代碼 ( 例如模組、函數、類等 ) 的方式，用於模組化和重用代碼。它們可以包含一個或多個模組。
- 你可以通過包管理工具 ( 如 pip ) 來安裝、更新或刪除包。這些包可能是第三方提供的 ( 例如 NumPy、Pandas 等 )，也可能是你自己創建的。

## 解釋器與包的關聯：

- 解釋器負責執行安裝在系統上的包中的代碼。當你在 Python 程式碼中導入一個包時 ( 例如 `import numpy as np` )，解釋器會尋找這個包並載入其內容，使你可以在你的代碼中使用這個包提供的功能。
- 每個 Python 解釋器可能會有自己的包安裝位置。這意味著，不同解釋器之間的包是隔離的，安裝在一個解釋器上的包不會影響到另一個解釋器。

## 虛擬環境：

- 為了解決不同項目間的依賴衝突問題，Python 提供了「虛擬環境」的概念。一個虛擬環境是一個獨立的目錄樹，包含一個特定版本的解釋器以及一系列安裝的包。
- 使用虛擬環境，你可以為每個項目創建一個隔離的環境，並且為它們安裝不同版本的包，而不會影響到系統中的其他項目。

總結來說，Python 解釋器是運行 Python 代碼的環境，而包是被解釋器用來擴展功能的代碼集合。通過虛擬環境，可以實現解釋器和包之間的有效隔離，從而使得 Python 項目更加模組化和可管理。

## Python 虛擬環境

Python 虛擬環境是個目錄，開發者可以在一台電腦當中創建多個隔離的 Python 環境。每個虛擬環境都有其自己的 Python 執行器和一套獨立的庫。這種隔離可以避免不同項目之間的庫版本衝突，並使得項目的依賴管理更加清晰。

主要用途包括：

1. **不同項目間的隔離**：不同項目可能需要不同版本的庫，使用虛擬環境可以避免版本衝突。
2. **實驗與測試**：在虛擬環境中測試新庫或更新，不會影響到主系統。
3. **便於部署**：可以將項目所需的所有依賴打包，便於在其他系統或生產環境中部署。

創建和使用虛擬環境通常很簡單，常用的工具包括Python自帶的 `venv` 模組和第三方的 `virtualenv` 包。通過簡單的命令行操作，就可以創建、激活或刪除虛擬環境。

## PIP

- `pip` 是 Python 的一個包管理工具，它用於安裝和管理 Python 軟件包。`pip` 使得從 Python Package Index (PyPI) 或其他軟件包索引下載和安裝軟件包變得簡單快捷。以下是 `pip` 的一些主要功能：
1. **安裝軟件包**：使用 `pip install package_name` 命令安裝指定的 Python 軟件包。
  2. **升級軟件包**：使用 `pip install --upgrade package_name` 命令升級已安裝的軟件包。
  3. **卸載軟件包**：使用 `pip uninstall package_name` 命令卸載不再需要的軟件包。
  4. **列出已安裝的軟件包**：使用 `pip list` 命令查看已安裝的所有軟件包。
  5. **查詢軟件包信息**：使用 `pip show package_name` 命令查看特定軟件包的詳細信息。
  6. **管理軟件包依賴**：`pip` 會自動管理軟件包的依賴關係，確保安裝時依賴包也會一同被安裝。

`pip` 是 Python 開發者常用的工具之一，因為它簡化了軟件包管理過程，使得安裝、升級、卸載和管理 Python 軟件包變得更加高效和方便。

## 集成開發環境 (IDE)

使用文字編輯器（如 Notepad++、Sublime Text 等）進行編程是完全可行的，尤其對於小型項目或簡單腳本。但是，使用集成開發環境 (IDE) 在許多方面更有優勢，特別是對於大型項目或複雜的應用程序。以下是使用 IDE 的幾個主要優點：

1. **代碼自動完成和語法高亮**：IDE 提供代碼自動完成功能，這可以加速編程過程並減少錯誤。語法高亮則可以幫助開發者更快地閱讀和理解代碼。

2. **錯誤檢測和除錯工具**：IDE 通常具有即時錯誤檢測和強大的除錯工具，可以幫助你快速定位並解決代碼中的問題。
3. **版本控制集成**：許多 IDE 提供了與版本控制系統（如 Git）的集成，這使得代碼版本管理和協作更加方便。
4. **內建的測試和構建工具**：IDE 常常包含用於測試和構建應用程序的工具，這有助於保持代碼質量和效能。
5. **專案管理**：IDE 為大型專案提供了更好的組織和管理功能，包括文件管理、專案結構概覽等。
6. **插件和擴展支持**：許多 IDE 支持插件和擴展，使你可以根據自己的需求定制開發環境。
7. **集成開發和測試環境**：IDE 常常提供集成的開發和測試環境，這意味著開發者可以在同一環境中編寫、測試和除錯代碼。

總的來說，雖然文字編輯器在某些情況下足夠使用，但IDE通過其豐富的功能和工具，提供了一個更高效、更強大的開發體驗，特別是對於複雜的專案和專業的開發環境。

## Python IDE

對於選擇 Python 開發環境（IDE），有幾個受歡迎的選項可以考慮：

1. **PyCharm**：PyCharm 是一款功能強大的 IDE，適合專業開發人員。它提供了代碼自動完成、錯誤檢測、集成的測試和版本控制系統等功能。PyCharm 適合從事大型項目或者需要複雜功能的開發者。
2. **Visual Studio Code (VS Code)**：VS Code 是一款輕量級但功能強大的編輯器，支援許多語言，包括 Python。它通過擴展（extensions）提供額外功能，例如代碼提示、版本控制和多種語言支援。
3. **Jupyter Notebook**：Jupyter Notebook 特別適合進行數據分析、數據可視化和機器學習項目。它允許你在瀏覽器中直接編寫代碼，並即時查看代碼的執行結果。非常適合教育和演示目的。
4. **Spyder**：Spyder 是一個專為科學計算和工程開發的 IDE。它提供了內置的變數探查器、集成的 IPython 控制台等功能。特別適合從事科學計算或數據分析的使用者。
5. **Thonny**：Thonny 是一個專為初學者設計的 Python IDE。它具有簡潔的用戶界面和易於理解的錯誤提示，非常適合初學者使用。

選擇哪一個 IDE 取決於你的具體需求和偏好。如果你是初學者，可能會更喜歡 Thonny 或 VS Code。對於進行科學計算或數據分析，Jupyter Notebook 和 Spyder 是很好的選擇。對於專業開發，PyCharm 提供了更多高級功能。

## 開發環境安裝

下面是一個詳細的步驟指南，幫助您下載並安裝 Python 3.10，建立虛擬環境，安裝 PyAEDT，以及安裝 Spyder IDE。

### 1. 下載並安裝 Python 3.10

1. 訪問 **Python 官方網站**：在瀏覽器中打開 [Python 官方網站](#)。
2. 選擇 **Python 3.10**：轉到「下載」區域，選擇 Python 3.10 版本。
3. 下載安裝程式：根據您的操作系統（Windows、macOS、Linux）下載相應的安裝程式。
4. 運行安裝程式：打開下載的文件，並運行安裝程式。Windows 用戶應該選擇「為所有用戶安裝」和「將 Python 添加到 PATH」這兩個選項。
5. 完成安裝：按照提示完成安裝。

### 2. 建立虛擬環境

1. 打開命令行介面：在 Windows 上打開命令提示符，或在 macOS/Linux 上打開終端。
2. 建立一個新目錄：使用 `mkdir your_project_name` 建立一個新目錄作為您的專案資料夾。
3. 進入該目錄：使用 `cd your_project_name` 命令進入該目錄。
4. 創建虛擬環境：執行 `python -m venv venv`（這裡的 `venv` 是虛擬環境的名稱，您可以自行命名）。

### 3. 安裝 PyAEDT

1. 啟動虛擬環境：在該目錄中，Windows 使用 `venv\Scripts\activate`，而 macOS/Linux 使用 `source venv/bin/activate` 啟動虛擬環境。
2. 安裝 **PyAEDT**：在虛擬環境中，使用命令 `pip install pyaedt` 安裝 PyAEDT。

## 4. 安裝 Spyder

1. 在虛擬環境中安裝 **Spyder**：還在虛擬環境中，執行 `pip install spyder` 來安裝 Spyder IDE。
2. 啟動 **Spyder**：安裝完成後，可以通過執行 `spyder` 命令來啟動 Spyder IDE。

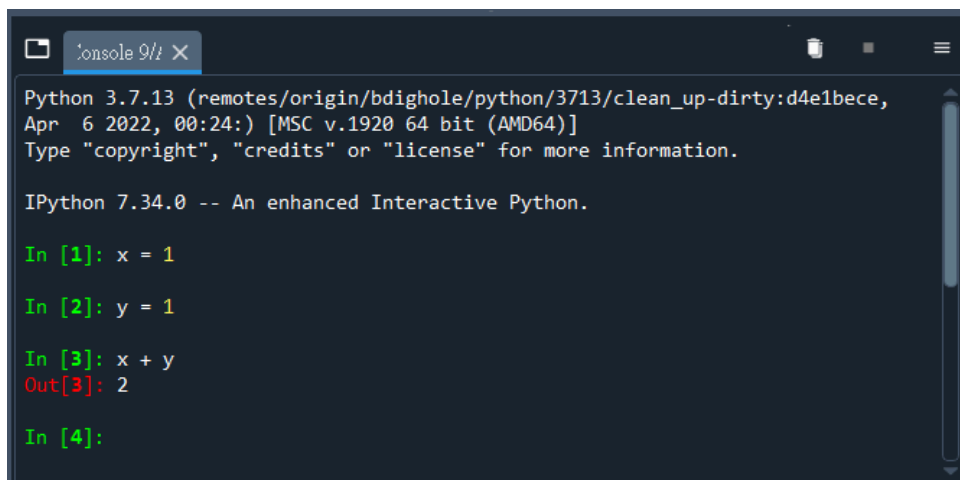
### 注意事項

- 確保在執行這些步驟時處於穩定的網絡環境中。
- 如果您在使用過程中遇到任何問題，例如安裝錯誤或相容性問題，請檢查錯誤信息並尋求相關的幫助文檔或論壇。

## 執行模式

Python的兩種主要執行模式——交互模式和腳本模式——各自具有獨特的特點和用途，適用於不同的開發情境：

### 交互模式 ( Interactive Mode )

A screenshot of a terminal window titled 'console 9/1'. The window shows the output of a Python 3.7.13 interpreter. It displays the version information, the IPython 7.34.0 banner, and a series of interactive commands and their outputs: 'In [1]: x = 1', 'In [2]: y = 1', 'In [3]: x + y', 'Out[3]: 2', and 'In [4]:'. The window has a dark background and a scrollbar on the right side.

```
Python 3.7.13 (remotes/origin/bdighole/python/3713/clean_up-dirty:d4e1bece,
Apr  6 2022, 00:24:) [MSC v.1920 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.34.0 -- An enhanced Interactive Python.

In [1]: x = 1

In [2]: y = 1

In [3]: x + y
Out[3]: 2

In [4]:
```

交互模式

#### 1. 即時反饋：

- 在交互模式中，當您輸入命令並按下Enter鍵後，Python會立即執行該命令並提供輸出結果。
- 這種模式適合於實時測試和調試，因為您可以即時看到每一行代碼的執行結果。

#### 2. REPL ( Read-Eval-Print-Loop )：

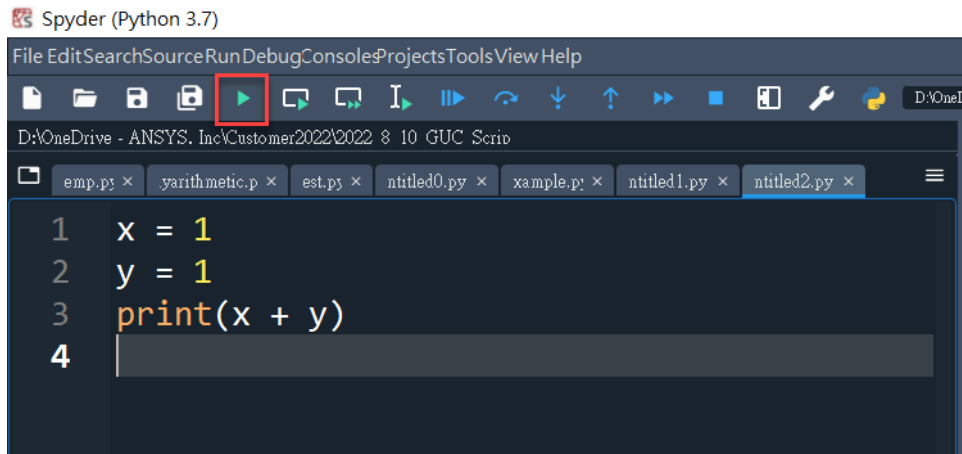
- 這種模式被稱為REPL，即讀取 ( Read )、評估 ( Evaluate )、列印 ( Print ) 和循環 ( Loop )。

- 在REPL中，Python解釋器讀取您的輸入，評估並執行該輸入，然後打印出結果，並重複此過程。

### 3. 適用於短代碼和初學者：

- 對於需要寫一些短小的代碼片段，或者剛開始學習Python的初學者來說，交互模式非常適合，因為它可以幫助理解每一步代碼的執行結果。

## 腳本模式 ( Script Mode )



腳本模式

### 1. 編寫長代碼：

- 在腳本模式下，開發者通常在集成開發環境 ( IDE ) 或文本編輯器中編寫Python程序。
- 這種模式適合編寫較長的代碼，因為它允許整體規劃和組織代碼結構。

### 2. 文件保存與執行：

- 程序被保存在一個以“.py”為副檔名的文件中。
- 您可以直接在IDE中執行此文件，或使用命令行提示符來執行。

### 3. 適用於複雜程序和專案開發：

- 對於需要編寫複雜程序或參與更大型專案開發的情況，腳本模式更加合適，因為它提供了更好的代碼管理和結構化功能。

總結來說，交互模式適合於快速測試和學習，而腳本模式則適用於開發完整的程序和專案。根據您的需求和開發階段選擇合適的模式，可以提高您的開發效率和代碼質



量。

# 1.6 Classical API vs PyAEDT

Classical API 是 AEDT 內建的原生 API，已有十多年的歷史。支持直接錄製用戶操作以便於自動化。這使得使用者可以在 AEDT 的圖形用戶介面內直接執行操作。然而，由於其歷史因素，這套 API 在參數設定方面相對複雜，並不完全符合 **Pythonic**（即遵循 Python 慣用語法和風格的編程方式）。此外，Classical API 運行於 IronPython 環境上，這限制了它與通用的 CPython 生態系統（即標準 Python 發行版和其廣泛的第三方庫）的整合能力。

有鑑於此，ANSYS 在 2021 年推出了 PyAEDT，這是一個重新包裝和優化的 API，不僅更加 **Pythonic**，也允許在 CPython 環境下運行，從而實現與廣泛的 Python 生態系統的無縫協作。這標誌著從傳統 API 向更現代、靈活且易於使用的解決方案的轉變。以下進一步概括和比較 AEDT Classical API 和 PyAEDT 兩種開發框架的特點和適用場景：

## AEDT Classical API

- **適用於特定工作和操作流程**：如果您的目標是為 AEDT 的特定操作或工作流程開發工具套件，尤其是當這些操作需要與 AEDT 的圖形用戶界面 (GUI) 緊密結合時，Classical API 是合適的選擇。
- **錄製和自動化功能**：AEDT 的操作介面提供錄製功能，可將用戶操作轉換為 Python 腳本，這有助於快速開發和自動化重複性操作。
- **豐富但複雜的 API 函數**：Classical API 提供了豐富的功能，但由於其參數結構複雜，開發出來的程式碼不易閱讀且難以維護。

## PyAEDT

- **適用於整個模擬流程和複雜任務**：當工作涵蓋整個模擬流程，或需要整合第三方 Python 模組時，PyAEDT 是更佳的选择。它提供了一種更加直觀、易於使用的接口。
- **高度整合性和靈活性**：PyAEDT 可以在一般的 Python 開發環境中運行，不必直接在 AEDT 的 GUI 中操作。這使得它可以輕鬆地與其他 Python 科學和工程模組（如數據分析、優化、機器學習等）結合。
- **較好的可讀性和易用性**：PyAEDT 的函數參數結構簡潔清晰，相較於 Classical API 更易於學習和使用。

## 選擇依據

- **開發目的**：如果您的開發重點是為了提高特定操作的效率，並與 AEDT 的 GUI 緊密結合，選擇 **Classical API**。如果您的目標是涵蓋更廣泛的模擬流程，並希望有更好的靈活性和擴展性，選擇 **PyAEDT**。
- **開發環境和集成需求**：考慮您的開發環境和是否需要將 AEDT 集成到更廣泛的數據分析和模擬流程中。**PyAEDT** 在這方面提供了更多的可能性。
- **學習曲線和可用資源**：考慮您和您的團隊的技能水平以及可用於學習和掌握這些工具的時間和資源。

綜上所述，選擇適合的開發框架取決於具體的應用需求、開發環境、以及期望達到的靈活性和整合程度。每種框架都有其獨特的優勢，因此選擇應基於您的具體需求和目標。

以下是 AEDT API 和 PyAEDT 兩種腳本開發框架的比較表格：

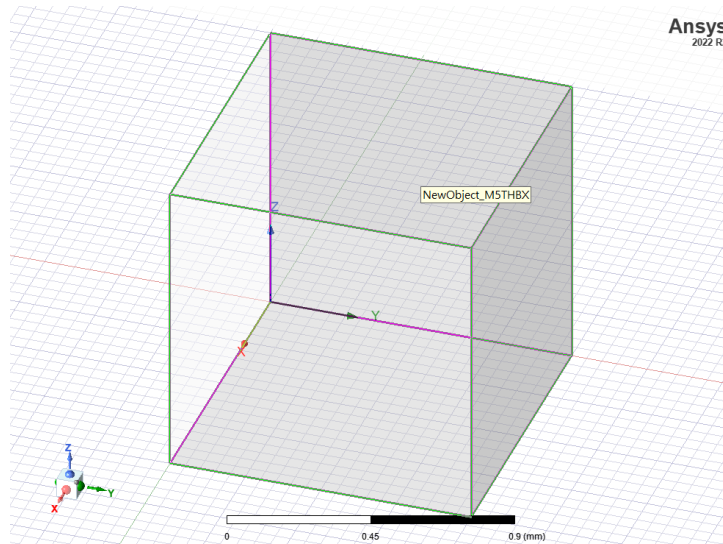
項目	AEDT API	PyAEDT
平台	IronPython	CPython
主要執行方式	在AEDT內執行	作業系統端執行
錄製	可	否
可讀性	低	高
擴充	困難	容易
支援模組	少	多
遠端連線	否	可
應用	單點工具	全流程或後處理
AEDT GUI	需開啟	可不需開啟

從表格中可以看出，AEDT API 和 PyAEDT 在各方面都有明顯的差異。選擇哪個框架取決於您的具體需求，如執行環境、功能覆蓋範圍、可擴展性以及與其他模組的整合程度等。例如，如果您需要在 AEDT 環境內進行密切的交互操作，可能會傾向於使用

AEDT API；相反，如果您需要更廣泛地使用 Python 生態系統中的各種科學計算模組，則 PyAEDT 可能是更好的選擇。

## API與PyAEDT函式風格比較

我們以創建一個立方體為例來比較兩種不同的框架API時，可以發現顯著的風格差異。



程式碼創建立方體

在傳統的API中，創建立方體的代碼如下所示：

```

oEditor.CreateBox(
    [
        "NAME:BoxParameters",
        "XPosition:=", "0mm",
        "YPosition:=", "0mm",
        "ZPosition:=", "0mm",
        "XSize:=", "1mm",
        "YSize:=", "1mm",
        "ZSize:=", "1mm"
    ],
    [
        "NAME:Attributes",
        "Name:=", "Box1",
        "Flags:=", "",
        "Color:=", "(143 175 143)",
        "Transparency:=", 0,
        "PartCoordinateSystem:=", "Global",
        "UDMId:=", "",
        "MaterialValue:=", "\"vacuum\"",
        "SurfaceMaterialValue:=", "\"\"",
        "SolveInside:=", True,
        "ShellElement:=", False,
        "ShellElementThickness:=", "0mm",
        "IsMaterialEditable:=", True,
        "UseMaterialAppearance:=", False,
        "IsLightweight:=", False
    ]
)

```

而在PyAEDT中，相同的操作可以通過更簡潔的代碼完成：

```
hfss.modeler.create_box((0,0,0),(1,1,1))
```

從這兩種方法的比較中，我們可以看出PyAEDT的代碼風格更為簡潔且易於閱讀和維護。這主要是因為PyAEDT對一些常用功能進行了有效的封裝，讓使用者能夠專注於模型的物理參數，而非繁瑣的底層API細節。儘管PyAEDT的框架API更為精簡，但在

某些情況下，傳統API由於其可透過錄製獲得且內嵌於AEDT中，仍然具有一定的優勢。