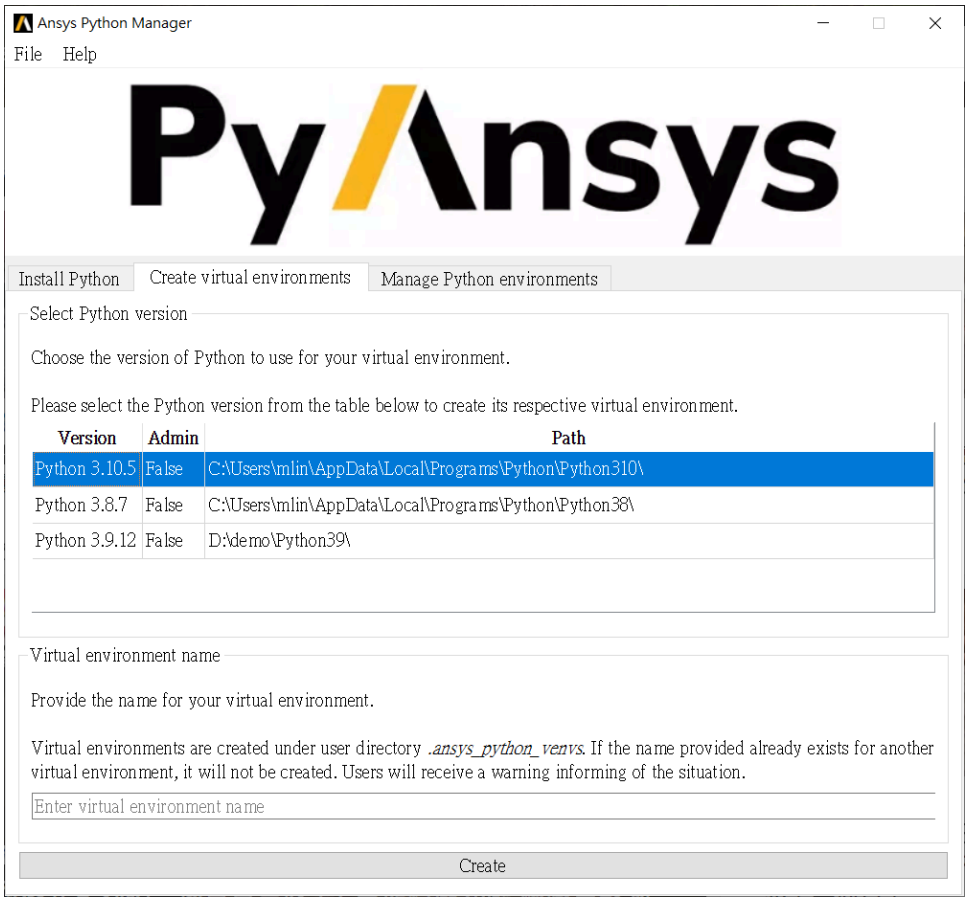
第10章 附錄

10.1 Ansys Python Manager

Ansys Python Manager是一個非常有用的工具，尤其對於那些需要在Ansys軟件中集成和管理Python環境的開發者來說。下面為您提供一個關於如何安裝和使用Ansys Python Manager的建議流程：



ANSYS Python Manager介面

安裝Ansys Python Manager

1. **下載安裝程序** ：

- 訪問Ansys Python Manager的官方頁面。
- 在頁面的Assets部分下，找到並下載安裝程序（檔案名通常為“Ansys-Python-Manager-Setup-v*.exe”）。

2. **執行安裝程序** ：

- 下載完成後，運行安裝程序。
- 按照安裝向導指示進行操作，將Ansys Python Manager安裝到您的電腦上。

3. **啟動Ansys Python Manager** ：

- 安裝完成後，從開始菜單或您選擇的快捷方式啟動Ansys Python Manager。

使用Ansys Python Manager

1. **選擇Python版本** ：

- 在Ansys Python Manager的用戶界面中，選擇所需的Python版本。您可以選擇標準Python或Condaforge Python。

2. **安裝Python** ：

- 選擇所需版本後，點擊“Install”按鈕。程序將自動下載並安裝選擇的Python版本。

3. **創建虛擬環境** ：

- 在“Create Virtual Environments”標籤頁中，您可以通過選擇Python版本並為環境命名來創建新的虛擬環境。
- 點擊創建按鈕，程序將開始創建虛擬環境。

4. **管理虛擬環境** ：

- 在“Manage Virtual Environments”標籤頁中，您可以查看和管理您創建的虛擬環境。
- 這裡可以進行包括刪除環境、啟動環境、管理Python包和PyAnsys包等操作。

5. **訪問文檔和支援** ：

- Ansys Python Manager還提供了方便訪問PyAnsys文檔的方式。您可以通過幫助菜單訪問相關文檔和資源。

透過這些步驟，您可以有效地安裝和使用Ansys Python Manager，從而更好地在Ansys環境中管理Python的安裝和虛擬環境，同時也能更方便地存取PyAnsys包及其相關文檔。這對於需要在Ansys軟件中進行高效Python編程的開發者來說，是一個非常有價值的工具。

10.2 Spyder整合開發環境

Spyder IDE（Scientific Python Development Environment）是一款專為科學計算和數據分析設計的開源集成開發環境。這款IDE特別適合於Python語言中使用科學計算庫（如NumPy、SciPy、Matplotlib等）的使用者。

Spyder的主要特點包括：

1. **互動式Python控制台** ：允許用戶執行Python代碼片段，並立即查看結果。
2. **變數探索器** ：能夠查看存儲在內存中的變數及其內容。
3. **代碼編輯器** ：具有語法高亮、代碼提示和自動縮進等功能。
4. **調試工具** ：支持設置斷點、單步執行、變數檢視和評估等調試功能。

Spyder還集成了其他多種工具和功能，例如檔案瀏覽器、專案管理器和IPython Notebook支持，使其成為一款功能豐富且適合數據分析和科學計算領域的IDE。由於其直觀的用戶界面和豐富的功能集，它對於Python初學者和專業開發者都是一款很好的選擇。

除錯技巧

在使用Spyder進行Python程式碼除錯時，了解不同類型的錯誤和相應的除錯技巧非常重要。程式錯誤大致可分為以下三類：

1. **語法錯誤（Syntax Error）** ：這是由於程式碼的語法不正確導致的錯誤。在Spyder中，這類錯誤通常會在代碼行旁顯示一個紅色的「x」標記，將滑鼠移至該標記上可以看到錯誤的詳細信息。一旦語法問題被修正，該標記會消失。
2. **執行時期錯誤（Runtime Error）** ：這類錯誤在程式運行時發生，例如，由於不兼容的數據類型操作。例如，將字串和整數相加會引發錯誤。Spyder會顯示錯誤信息，指出問題所在的行號和原因。

示例代碼：

```
x = '2'
print(x + 1)  # 這行會引發錯誤，因為字串和整數不能直接相加
```

3. **邏輯錯誤（Logic Error）**：程式可以正常運行並完成，但結果與預期不符。這類錯誤糾正難度較高，可透過設置斷點和逐步執行來找出錯誤所在。

為了有效地進行除錯，您可以運用以下幾種Spyder的功能：

- **設定斷點（Breakpoints）**：在希望程式暫停執行的代碼行上右擊，選擇「設定斷點」。程式運行到該處時將暫停，讓您能檢查當前變量的值。
- **使用偵錯控制台（Debug console）**：當程式在斷點處暫停時，您可以在偵錯控制台中輸入變量名稱查看其值，或執行其他Python命令。
- **變量探查器（Variable Explorer）**：在偵錯模式下，變量探查器可以顯示所有當前可用變量及其值。對於列表或數據框等複雜數據類型，您甚至可以直接在探查器中查看其內容。
- **使用Step Over, Step In, Step Out命令**：當遇到函數或方法調用時，您可以選擇Step In進入該函數或方法進行逐行執行，或選擇Step Over跳過該調用，或在已進入的函數或方法中使用Step Out退出。

這些技巧將有助於您更好地理解程式的運行過程並找出潛在的問題。如需更多關於Spyder除錯的詳細信息，建議查閱Spyder的官方文檔。

10.3 線上學習資源

PyANSYS

PyAnsys開發版本文檔 網站提供了關於PyAnsys項目的詳細信息。該項目最初是作為單一的 `pyansys` 包而起源，現在已成為許多用於通過Python使用Ansys產品的Python包的集合。這裡包括了對各種Ansys服務和工具的Python接口，如PyAEDT、PyMAPDL、PyFluent等，以及其他多種工具和功能的介紹。這些資源對於希望利用Python與Ansys產品進行交互的用戶而言，非常有用。

ANSYS Developer Portal

Ansys開發者門戶 是一個為使用Ansys產品的開發者提供資源的平台。該網站提供了關於Ansys工具和服務的廣泛文檔、知識庫、討論論壇和GitHub存儲庫的鏈接。此外，您還可以找到關於如何利用Ansys開發工具來創建新解決方案、自動化和擴展模擬工作流程的信息，以及各種開發指南和實用示例。這個門戶是Ansys開發者社區的重要資源，有助於分享知識、尋求專業指導並討論項目。

10.4 Git與GitHub

Git 和 GitHub 雖然名稱相似，但它們在軟體開發過程中扮演著不同的角色。以下解釋這兩者的區別：

1. **Git** ：

- **定義** ：Git 是一種分散式版本控制系統，用於追蹤和管理代碼變更。它是由 Linus Torvalds 創建的，旨在為軟體開發者提供高效、靈活的代碼管理工具。
- **功能** ：Git 讓開發者能夠在本地計算機上工作，追蹤所有代碼變更歷史，並允許多個開發者在各自的分支上工作，之後再將這些分支合併到主代碼庫中。
- **本地化** ：Git 是本地化的，這意味著每個開發者在他們的計算機上有整個代碼庫的副本，包括所有歷史記錄。

2. **GitHub** ：

- **定義** ：GitHub 是一個基於網絡的平台，用於代碼托管和協作。它利用 Git 的版本控制功能，為開發者提供了一個中央位置來存儲、分享和管理他們的代碼。
- **功能** ：GitHub 提供了許多協作功能，如問題追蹤、代碼審查、項目管理和社交網絡功能。它也支持公開和私有存儲庫。
- **雲端服務** ：GitHub 是一個雲端服務，允許用戶從任何地方訪問和管理他們的代碼庫。它還促進了開源項目的共享和協作。

總結來說，Git 是一個工具，專注於版本控制和代碼管理，而 GitHub 是一個平台，提供代碼托管服務和協作工具，並建立在 Git 的基礎之上。GitHub 讓開發者可以更容易地共享代碼，並與全球其他開發者協作。

GitHub Desktop

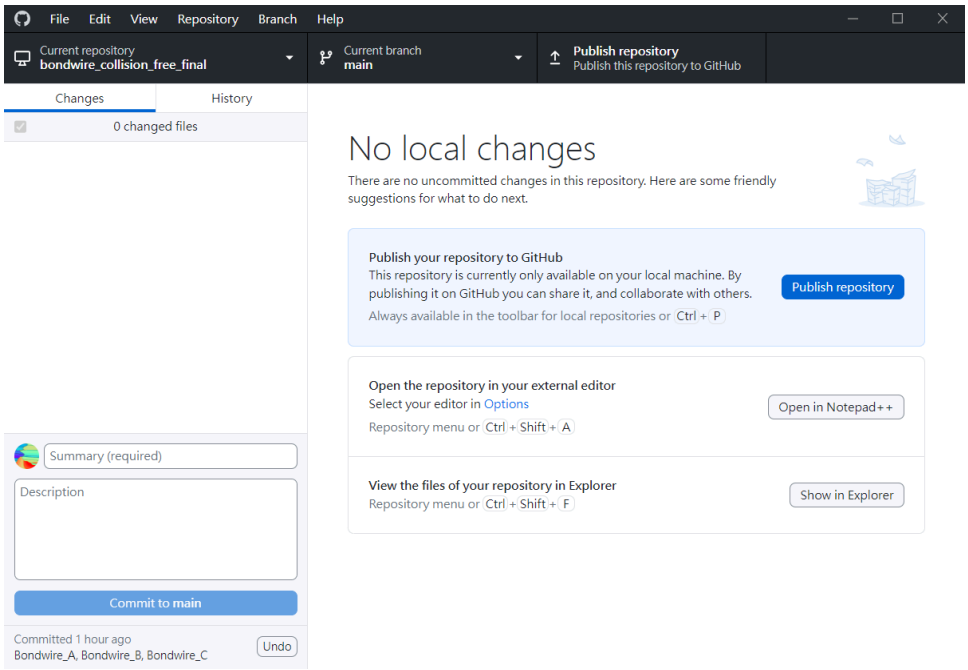
GitHub Desktop 是一款由 GitHub 開發的免費的 Git 客戶端應用程序。它旨在簡化 Git 的版本控制過程，並讓使用者可以更加直觀和方便地使用 GitHub。GitHub Desktop 特別適合那些希望避免使用命令行的用戶，它提供了一個用戶友好的圖形界面來執行日常的 Git 和 GitHub 操作。

GitHub Desktop 的主要特點包括：

1. **易於使用的界面** ：它提供了一個清晰的用戶界面，用戶可以通過點擊按鈕來完成 Git 操作，如提交、拉取、推送等。
2. **簡化的工作流程** ：GitHub Desktop 使版本控制工作流程變得更加直觀，包括分支管理、合併衝突解決等。
3. **跨平台支持** ：GitHub Desktop 可用於 Windows 和 macOS。
4. **與 GitHub 緊密集成** ：用戶可以直接從應用程序內創建新的存儲庫、克隆現有的存儲庫、提交更新和管理分支。
5. **支持本地存儲庫** ：除了與 GitHub 存儲庫的集成外，GitHub Desktop 也支持本地 Git 存儲庫的管理。
6. **提交歷史和更改查看** ：用戶可以輕鬆查看提交歷史，並查看文件之間的差異。

GitHub Desktop 是一個適合初學者和希望提高生產力的專業開發者的工具。它降低了 Git 的學習曲線，並提供了一個更加人性化的方式來管理代碼。

你可以從 GitHub Desktop 的官方網站下載應用程序：GitHub Desktop 官方網站 。在那裡，你也可以找到關於如何使用這個工具的進一步信息和教程。



GitHub Desktop

10.5 Linux環境執行PyAEDT

在Linux環境中使用PyAEDT進行ANSYS Electronics Desktop的腳本編寫和執行，需要先正確設定環境變量。這是因為PyAEDT需要特定的路徑和庫文件才能與ANSYS軟件互動。下面為您簡單解釋設定這些環境變量的步驟：

1. **環境變量** ：

- `ANSYSEM_ROOT222`指向您的ANSYS Electronics Desktop安裝路徑。假設您的ANSYS安裝在`/home/user/ansys`，那麼這個變量應該設為`/home/user/ansys/AnsysEM/v222/Linux64`。
- 在bash shell中，您可以使用以下命令設定：

```
export ANSYSEM_ROOT222=/home/user/ansys/AnsysEM/v222/Linux64
```

2. **環境變量** ：

- `LD_LIBRARY_PATH`用於指定系統在執行程序時查找共享庫的路徑。您需要將ANSYS的共享庫路徑添加到此變量中。
- 使用":"來分隔不同的路徑，按設定的順序從前到後依次搜索。
- 在bash shell中，您可以這樣設定：

```
export LD_LIBRARY_PATH=$ANSYSEM_ROOT222/common/mono/Linux64/lib64:$ANSYSEM_ROOT
```

完成這些設定後，您就可以在Linux環境下使用Spyder或其他編輯器進行PyAEDT的編程和執行了。這些步驟確保了系統能夠正確地找到和加載所需的ANSYS軟件組件，從而順利執行PyAEDT腳本。

10.6 離線安裝PyAEDT

許多公司出於資安考慮，將用於設計模擬的電腦與外部網路隔離，導致無法直接透過pip從PyPI下載並安裝pyaedt。解決這一問題的方法之一是在一台可以連接外部網路的電腦上預先安裝好開發環境（包括pyaedt和spyder），然後下載所有必要的套件，最後將這些套件移至離線電腦上進行安裝。具體操作步驟如下：

1. **在連網電腦上安裝虛擬環境** ，並在該環境中安裝PyAEDT及Spyder。
2. **導出套件列表到requirements.txt** ：

- 在虛擬環境中執行以下指令，將已安裝套件的列表輸出至`requirements.txt`：

```
pip freeze > c:\requirements.txt
```

3. **建立wheelhouse目錄** ：

- 在虛擬環境中，使用以下指令來創建一個名為`wheelhouse`的目錄：

```
mkdir c:\wheelhouse
```

4. **下載套件至wheelhouse目錄** ：

- 根據`requirements.txt`中的內容，將所需套件下載至`wheelhouse`目錄：

```
pip download -r c:\requirements.txt -d c:\wheelhouse
```

5. **目錄複製到離線電腦** ：

- 把這兩個文件和目錄拷貝到離線電腦的C槽（或其他指定位置）。

6. **在離線電腦上建立並啟動虛擬環境** 。
7. **安裝套件** ：

- 在離線電腦的虛擬環境中執行以下指令來安裝套件：

```
pip install -r c:\requirements.txt --no-index --find-links c:\wheelhouse
```

8. **驗證安裝** ：

- 在離線電腦上開啟spyder，並在console中輸入`import pyaedt`測試。若無錯誤訊息出現，則表示安裝成功。

透過以上步驟，您可以在不連網的環境下成功安裝PyAEDT，滿足公司的資安需求，同時也不影響開發工作的進行。

10.7 Logging

在計算機科學中，「logging」（日誌記錄）是一種追蹤和記錄軟件運行時所發生事件的方法。這對於除錯（debugging）、監控軟件行為、以及分析和預防問題非常重要。日誌記錄可以提供寶貴的資訊，幫助開發者了解軟件運行的狀況，並在出現問題時快速找到原因。

日誌級別是日誌記錄中一個重要概念，常見的日誌級別包括：

1. **錯誤（Error）**：嚴重問題，導致軟件無法正常運行。
2. **警告（Warning）**：可能的問題，應予以關注，但不會影響整體運行。
3. **信息（Info）**：正常操作的訊息，用於狀態更新或操作確認。
4. **調試（Debug）**：用於除錯的詳細資訊，通常只在開發過程中使用。
5. **追蹤（Trace）**：最詳細的日誌記錄，用於追蹤問題。

不同的程式語言和框架提供了各自的日誌記錄工具，例如 Python 中的 `logging` 模塊，Java 中的 `Log4j`，等等。這些工具通常允許開發者設定日誌級別、自定義日誌格式，以及選擇日誌輸出的目的地（如控制台、文件、網絡服務器等）。有效的日誌記錄策略可以大大提高維護和除錯的效率。

在 Python 中，`logging` 模塊提供了靈活的日誌記錄系統。這個模塊允許你記錄訊息，這些訊息可以根據嚴重性（日誌級別）被分類，並且可以被導向到不同的目的地，比如控制台或文件。這在開發和維護大型應用程式時尤其有用。

基本用法

1. **導入模塊**：首先，需要導入 `logging` 模塊。

```
import logging
```

2. **設定日誌級別**：選擇要記錄的訊息級別。最常用的級別有 DEBUG, INFO, WARNING, ERROR, 和 CRITICAL。

```
logging.basicConfig(level=logging.DEBUG)
```

3. **記錄訊息** ：使用不同的方法來記錄不同級別的訊息。

```
logging.debug("這是一條 debug 訊息")
logging.info("這是一條 info 訊息")
logging.warning("這是一條 warning 訊息")
logging.error("這是一條 error 訊息")
logging.critical("這是一條 critical 訊息")
```

進階用法

- **自定義日誌格式** ：可以設定日誌的輸出格式。

```
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)
```

- **將日誌輸出到文件** ：可以指定一個文件來儲存日誌。

```
logging.basicConfig(filename='example.log', level=logging.DEBUG)
```

- **使用 Logger 對象** ：為了更細致地控制日誌，可以創建一個 `Logger` 對象。

```
logger = logging.getLogger('my_logger')
logger.setLevel(logging.DEBUG)
```

記住，`logging` 模塊的配置只能在程序的開始時設置一次。若需要動態改變配置，則需要使用更進階的設置方法，如配置文件或者程式碼中配置 Logger。