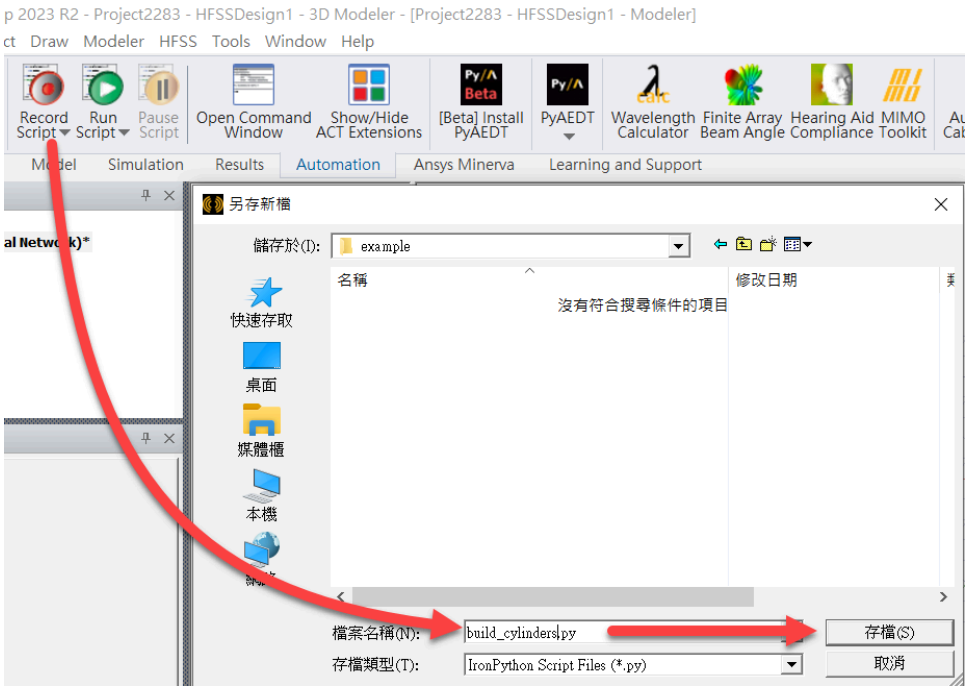
第6章 工具開發範例

6.1 讀取圓柱座標及尺寸並生成模型

本節將深入介紹一個開發實例。在此例中，使用者將通過選單來啟動一項工具。隨後，一個檔案選擇視窗會出現，讓使用者選取csv格式的檔案。程式接著會從該檔案中讀取座標、半徑以及高度的數據，並利用這些數據在HFSS軟體中生成圓柱形結構。

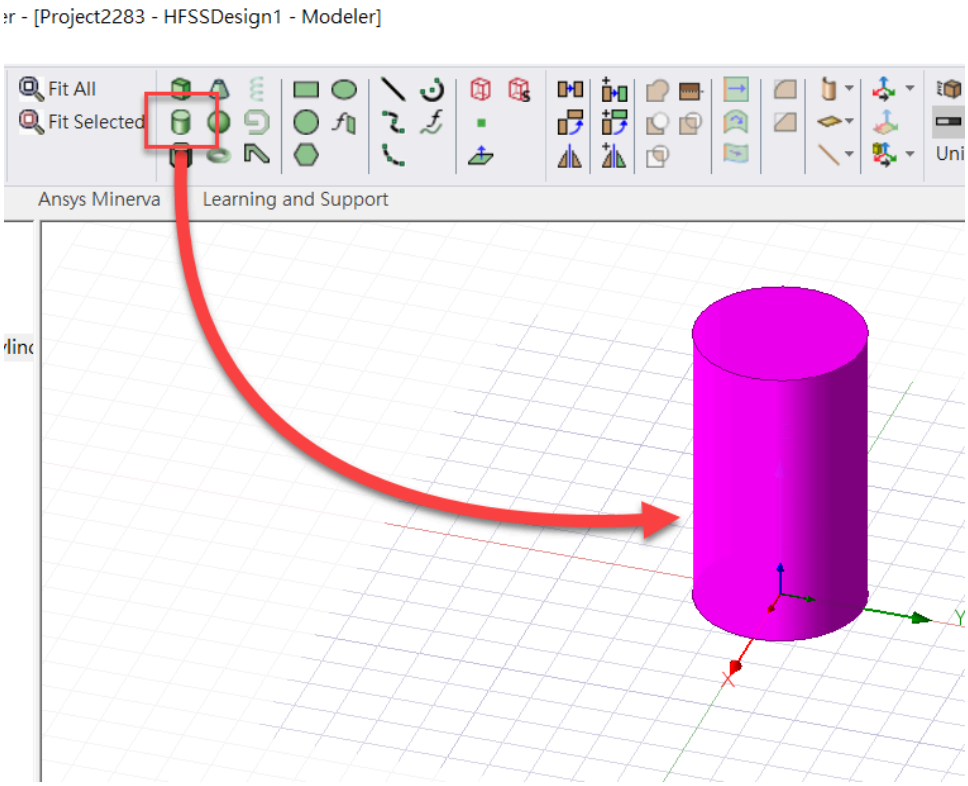
錄製腳本

1. **開始錄製**：首先，我們開始錄製操作過程，並命名我們的腳本為「build_cylinders.py」



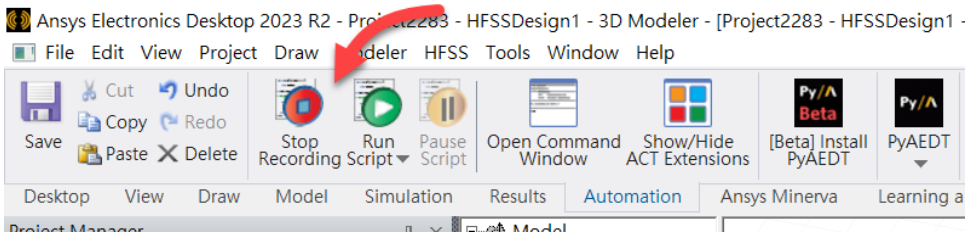
這裡是圖片的說明

2. **建立圓柱**：完成命名後，我們回到HFSS軟體中，進入建模環境並使用圓柱工具創建一個圓柱形結構。圓柱的尺寸並不重要，可以隨意設定。



這裡是圖片的說明

3. **停止錄製**：一旦圓柱創建完畢，停止錄製。



這裡是圖片的說明

4. **編輯腳本**：在此過程中接下來，使用文本編輯器打開「build_cylinders.py」腳本，您將會看到生成的程式碼。

```
# ----------------------------------------------
# Script Recorded by Ansys Electronics Desktop Version 2023.2.0
# 19:59:16  十二月 27, 2023
# ----------------------------------------------
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("Project2283")
oDesign = oProject.SetActiveDesign("HFSSDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oEditor.CreateCylinder(
        [
                "NAME:CylinderParameters",
                "XCenter:="             , "0mm",
                "YCenter:="             , "0mm",
                "ZCenter:="             , "0mm",
                "Radius:="              , "0.4mm",
                "Height:="              , "1.4mm",
                "WhichAxis:="           , "Z",
                "NumSides:="            , "0"
        ],
        [
                "NAME:Attributes",
                "Name:="                , "Cylinder1",
                "Flags:="               , "",
                "Color:="               , "(143 175 143)",
                "Transparency:="        , 0,
                "PartCoordinateSystem:=", "Global",
                "UDMId:="               , "",
                "MaterialValue:="       , "\"vacuum\"",
                "SurfaceMaterialValue:=", "\"\"",
                "SolveInside:="         , True,
                "ShellElement:="        , False,
                "ShellElementThickness:=", "0mm",
                "ReferenceTemperature:=", "20cel",
                "IsMaterialEditable:="  , True,
                "UseMaterialAppearance:", False,
```

```
                "IsLightweight:="       , False
        ])
```

觀察上述我們錄製的程式碼，我們可以發現，oEditor.CreateCylinder()這個函數被用於建立圓柱結構。然而，這個函數所需的參數結構相當繁複，包含了多個細節設定，這對於初學者來說可能顯得有些混亂且難以理解。接下來我們要針對程式碼做一番梳理。

程式碼整理

首先，我們需要將程式碼中的註解移除，並刪除以下幾行程式碼：

```
import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
```

並將SetActiveProject，SetActiveDesign改成GetActiveProject，GetActiveDesign。

接著，我們的目標是將 `oEditor.CreateCylinder()` 函數封裝成一個新的函數 `create_cylinder()`。為了達成這一點，我們將 `"Name:"`, `"XCenter:"`, `"YCenter:"`, `"ZCenter:"`, `"Radius:"` 等參數後的對應數值替換為變數，並將這些變數定義為新函數的參數。通過這樣的處理，新的函數將變得更加簡潔明瞭。修改之後的程式碼如下：

```
# -*- coding: utf-8 -*-
oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

def create_cylinder(name, x_center, y_center, z_center, radius, height):
    oEditor.CreateCylinder(
        [
            "NAME:CylinderParameters",
            "XCenter:=", x_center,
            "YCenter:=", y_center,
            "ZCenter:=", z_center,
            "Radius:=", radius,
            "Height:=", height,
            "WhichAxis:=", "Z",
            "NumSides:=", "0"
        ],
        [
            "NAME:Attributes",
            "Name:=", name,
            "Flags:=", "",
            "Color:=", "(143 175 143)",
            "Transparency:=", 0,
            "PartCoordinateSystem:=", "Global",
            "UDMId:=", "",
            "MaterialValue:=", "\"vacuum\"",
            "SurfaceMaterialValue:=", "\"\"",
            "SolveInside:=", True,
            "ShellElement:=", False,
            "ShellElementThickness:=", "0mm",
            "ReferenceTemperature:=", "20cel",
            "IsMaterialEditable:=", True,
            "UseMaterialAppearance:=", False,
            "IsLightweight:=", False
        ]
    )
```

在這段程式碼中，create_cylinder 函數接收圓柱的名稱、中心座標、半徑和高度作為參數，並用這些參數來調用 oEditor.CreateCylinder() 函數。這樣的封裝使得代碼更加整潔，並且更容易重複使用和維護。

```
# 例子: 使用 create_cylinder 函數來建立一個圓柱
create_cylinder("Cylinder1", "0mm", "0mm", "0mm", "0.4mm", "1.4mm")
```

> 在Python程式碼中，*# -- coding: utf-8 --* 是一個特殊的註解，用於指定文件的編碼格式。這行代碼通常位於Python檔案的頂部，用來告訴Python解釋器該文件使用UTF-8編碼。這對於包含非ASCII字符（例如中文、日文或其他非英文字符）的程式碼來說尤為重要。

定義腳本專屬CSV檔格式

我們的目標是利用.csv檔案來記錄圓柱結構所需的各項資訊，如座標、半徑和高度。藉由這種方式，程式能夠讀取.csv檔案，並根據其中的資料來自動生成圓柱結構。我們可以設計這樣的.csv檔案格式，其中每行代表一個圓柱的參數，按照「X座標, Y座標, Z座標, 半徑, 高度」的順序排列。例如：

```
# mm
6.4,7.11,7.47,0.25,2.16
11.86,15.77,16.56,1.65,1.0
20.65,23.05,25.88,1.9,4.21
30.53,28.93,32.64,1.0,1.91
40.02,38.59,42.47,0.33,3.9
47.31,48.28,48.11,0.15,1.22
56.05,54.47,55.1,1.34,3.5
65.71,60.18,62.02,1.63,3.79
71.85,66.12,67.91,1.88,1.13
81.1,72.78,77.66,0.72,1.66
88.32,80.94,83.39,0.9,4.11
95.86,88.28,90.9,0.2,2.86
101.03,93.76,98.84,1.87,0.97
109.16,101.3,108.69,0.14,2.84
119.15,106.93,117.75,1.91,1.42
124.48,112.81,127.41,1.44,3.98
134.11,117.86,134.34,1.9,2.38
139.62,127.26,140.17,0.23,2.64
148.19,134.13,149.54,1.35,2.83
154.13,143.82,157.09,0.8,2.51
```

在這個格式中，第一行#號空格之後為單位，之後的每一行都包含了創建一個圓柱所需的完整資訊，使程式可以依此生成多個圓柱結構。接下來我們要用讀檔來將這20筆圓柱的資料讀到變數當中。將上述內容複製到c:/locations.csv當中。

CSV檔案讀取

為了讀取該.csv檔案並將數值與單位結合，我們可以撰寫一個Python函數來實現此功能。這個函數將從.csv檔案中讀取數據，將每一行的數值與第一行指定的單位合併，並將這些數據存儲在一個列表中返回。請注意，這個函數假設.csv檔案格式如上一節所提供的範例，且第一行包含單位資訊。

以下是函數的實現：

```
def read_csv_with_unit(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()

    unit = lines[0].strip().split(' ')[-1]  # 取得單位
    data_list = []

    for line in lines[1:]:  # 跳過第一行的單位
        values = line.strip().split(',')
        data_with_unit = [value.strip() + unit for value in values]
        data_list.append(data_with_unit)

    return data_list

# 使用函數
data = read_csv_with_unit('c:/locations.csv')
print(data)
```

這段程式碼的目的是從一個CSV檔案中讀取數據，並在每個數據後面加上單位。以下是逐行解釋：

1. `def read_csv_with_unit(filename):`
   定義了一個函式 `read_csv_with_unit`，這個函式接受一個參數 `filename`，它是一個字符串，表示要讀取的CSV檔案的檔案名。
2. `with open(filename, 'r') as file:`
   使用 `with` 語句打開檔案。這種方式可以在檔案操作完成後自動關閉檔案。`'r'` 表示以讀取模式開啟檔案。
3. `lines = file.readlines()`
   讀取檔案中的所有行，並將它們儲存到 `lines` 這個列表中。每一行都是列表中的一個元素。
4. `unit = lines[0].strip().split(' ')[-1]`
   從檔案的第一行提取單位。首先去除開頭和結尾的空格（`strip()`），然後以空格分割字符串（`split(' ')`），最後取分割後的最後一個元素（`[-1]`），這就是數據的單位。

5. `data_list = []`
   創建一個空列表 `data_list`，用於儲存處理後的數據。
6. `for line in lines[1:]:`
   遍歷從第二行開始的每一行（因為第一行是單位，已經被處理過了）。`lines[1:]` 創建了一個從第二行開始的新列表。
7. `values = line.strip().split(',')`
   對於每一行，先去除開頭和結尾的空格，然後以逗號分割，獲得一個值的列表。
8. `data_with_unit = [value.strip() + unit for value in values]`
   對於每一個值，去除開頭和結尾的空格，然後將單位加到值的後面，使用列表推導式創建一個新的列表 `data_with_unit`。
9. `data_list.append(data_with_unit)`
   將加上單位的數據列表 `data_with_unit` 加入到 `data_list` 中。
10. `return data_list`
    返回最終的 `data_list`，它包含了所有處理過的數據行，每行數據都加上了單位。

這段程式碼是對CSV檔案進行讀取並添加單位的一個很好的例子，展示了文件處理和字符串處理的基本方法。

strip()與split()字串處理

在 Python 中，`strip()` 和 `split()` 是處理字符串的兩個常用方法。它們各自有不同的用途：

1. **strip()方法** ：

- 功能：用於移除字符串開頭和結尾的空白字符（包括空格、換行符 `\n`、製表符 `\t` 等）。
- 語法：`str.strip([chars])`。
- `chars`：可選參數，指定要從字符串中移除的字符集。如果未指定，則默認移除空白字符。
- 例子：`' hello '.strip()` 會返回 `'hello'`。

2. **split()方法** ：

- 功能：用於將字符串根據指定的分隔符拆分成多個子字符串，並將這些子字符串保存在列表中。
- 語法：`str.split(sep=None, maxsplit=-1)`。
- `sep`：分隔符。預設為任何空白字符，如空格、換行符、製表符等。
- `maxsplit`：可選參數，指定最大拆分次數。默認為 `-1`，表示不限制拆分次數。
- 例子：`'a,b,c'.split(',')` 會返回 `['a', 'b', 'c']`。

這兩個方法常常在數據處理和文本處理中使用，幫助處理和組織字符串數據。

加入迴圈

這段代碼的目的是從CSV檔案中自動讀取一系列圓柱的參數，並使用這些參數在HFSS中創建對應的圓柱結構。每個圓柱根據其在CSV檔案中的順序，被賦予一個獨特的名稱，並根據檔案中的數據設定其幾何特性。這樣的自動化過程可以大大節省時間，特別是在需要創建大量類似物體時。

```
csv_path = 'c:/locations.csv'
data = read_csv_with_unit(csv_path)

for n, (x, y, z, radius, height) in enumerate(data):
    create_cylinder('cylinder{}'.format(n), x, y, z, radius, height)
```

這一段代碼主要進行的是從CSV檔案中讀取數據並根據這些數據在HFSS中建立一系列圓柱形結構的過程。以下進一步解釋每個部分的功能和運作方式。

首先，csv_path 變量被設定為字符串 'c:/locations.csv'，這是您存儲圓柱數據的CSV檔案的路徑。這個檔案應該包含了一系列用於建立圓柱的參數，如X、Y、Z座標、半徑和高度。

接下來，read_csv_with_unit 函數被調用，並將 csv_path 作為參數傳入。這個函數的作用是打開指定路徑的CSV檔案，讀取裡面的數據，並將這些數據轉換成適合後續處理的格式。具體來說，它會讀取每一行的數值，並將這些數值連同它們的單位（假設第一行指定了單位）存儲在一個列表中。

然後，程式碼進入一個循環，使用 enumerate 函數遍歷 read_csv_with_unit 函數返回的數據列表。enumerate 函數會在每次迭代中返回兩個值：一個是正在處理的元素

的索引（在這裡是 n），另一個是元素本身（在這裡是一組圓柱參數 (x, y, z, radius, height)）。

在循環的每次迭代中，都會調用 create_cylinder 函數。這個函數負責在HFSS中創建一個圓柱，它需要圓柱的名稱和各種參數作為輸入。圓柱的名稱是通過格式化字符串 'cylinder{}'.format(n) 生成的，這樣每個圓柱都會有一個唯一的名稱（如 'cylinder0', 'cylinder1' 等）。其餘的參數則是從CSV檔案中讀取的數據。

enumerate函數

`enumerate()` 是一個在 Python 中非常實用的函數，它用於在遍歷（迭代）序列（如列表、元組或字符串）時同時獲取元素的索引和值。

基本語法如下：

```
enumerate(iterable, start=0)
```

- `iterable`：要迭代的序列。
- `start`：索引開始的數字，預設為0。

使用 `enumerate()` 時，它會返回一個 enumerate 物件，這個物件生成一系列包含索引和對應值的元組。

例如，假設我們有一個列表 `['a', 'b', 'c']`，並想打印出每個元素及其索引：

```
for index, value in enumerate(['a', 'b', 'c']):
    print(index, value)
```

這段代碼會輸出：

```
0 a
1 b
2 c
```

這樣，你就可以很容易地同時獲取到元素的索引和值。

加入檔案開啟視窗

至此，我們的程式功能基本完善。然而，當前的程式碼中，CSV檔案的路徑是固定寫入的，這限制了其靈活性。為了讓程式更加靈活，我們可以加入一段代碼來實現檔案選擇對話框的功能。以下是這段新代碼：

```
import sys
import clr
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import DialogResult, OpenFileDialog
dialog = OpenFileDialog()
dialog.Multiselect = False  # 確保只能選擇一個檔案
dialog.Title = "選擇圓柱資料檔案"  # 設置對話框的標題
dialog.Filter = "CSV檔案 (*.csv)|*.csv"  # 設置檔案篩選器，只顯示CSV檔案

if dialog.ShowDialog() == DialogResult.OK:
    csv_path = dialog.FileName  # 獲取選擇的檔案路徑
    AddWarningMessage(csv_path)  # 顯示檔案路徑
else:
    pass  # 如果未選擇檔案，則不執行任何操作
```

這段代碼使用了Windows表單來創建一個檔案選擇對話框，允許用戶選擇CSV檔案。如果用戶選擇了一個檔案，程式將獲取該檔案的路徑並顯示出來。這樣，我們就不必在程式碼中硬編碼CSV檔案的路徑，從而提高了程式的可用性和靈活性。

這種改進不僅使程式更加用戶友好，而且也方便了在不同情境下重複使用該程式，因為用戶可以根據需要選擇不同的CSV檔案。

完整版本

以下是對完整程式碼的簡要概述：

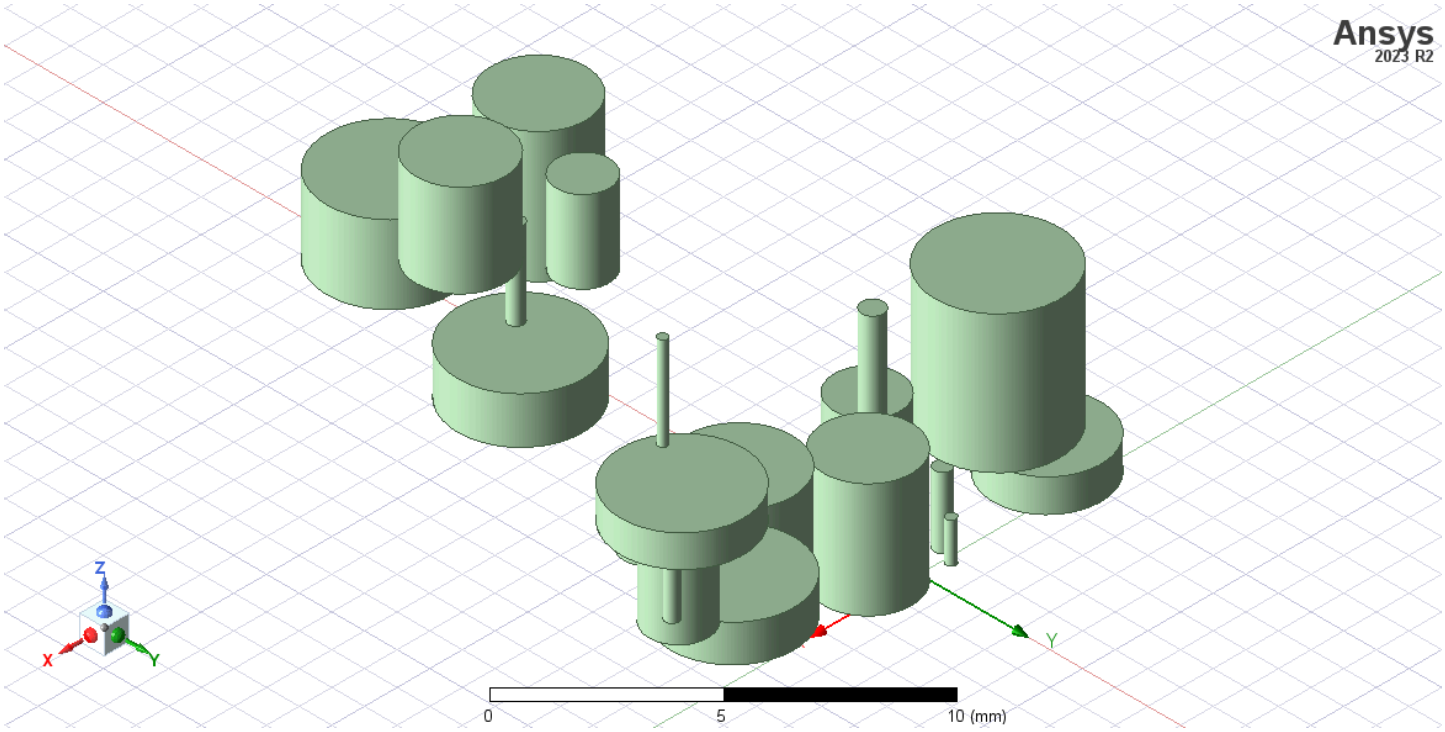
1. **初始化HFSS環境：** 代碼首先設定HFSS的專案、設計和編輯器環境。
2. **定義 create_cylinder函數：** 此函數負責在HFSS中創建圓柱。它接受圓柱的名稱和幾何參數作為輸入，然後調用HFSS的API來實現圓柱的創建。

3. **定義 read_csv_with_unit函數：** 這個函數用於讀取CSV檔案，提取其中的圓柱參數及其單位，並將它們轉換成適合後續處理的格式。
4. **檔案選擇對話框：** 代碼中加入了一段使用Windows表單的代碼，允許用戶通過對話框選擇CSV檔案，而不是在代碼中硬編碼檔案路徑。這增加了代碼的靈活性和用戶友好性。
5. **讀取並處理CSV檔案：** 代碼讀取用戶選擇的CSV檔案，並使用 `read_csv_with_unit` 函數解析檔案內容。
6. **生成圓柱：** 代碼遍歷CSV檔案中的每一行數據，並對每一組數據調用 `create_cylinder` 函數，根據這些數據在HFSS中生成相應的圓柱形結構。

```
# -*- coding: utf-8 -*-
oProject = oDesktop.GetActiveProject()
oDesign = oProject.GetActiveDesign()
oEditor = oDesign.SetActiveEditor("3D Modeler")

def create_cylinder(name, x_center, y_center, z_center, radius, height):
    oEditor.CreateCylinder(
        [
            "NAME:CylinderParameters",
            "XCenter:=", x_center,
            "YCenter:=", y_center,
            "ZCenter:=", z_center,
            "Radius:=", radius,
            "Height:=", height,
            "WhichAxis:=", "Z",
            "NumSides:=", "0"
        ],
        [
            "NAME:Attributes",
            "Name:=", name,
            "Flags:=", "",
            "Color:=", "(143 175 143)",
            "Transparency:=", 0,
            "PartCoordinateSystem:=", "Global",
            "UDMId:=", "",
            "MaterialValue:=", "\"vacuum\"",
            "SurfaceMaterialValue:=", "\"\"",
            "SolveInside:=", True,
            "ShellElement:=", False,
            "ShellElementThickness:=", "0mm",
            "ReferenceTemperature:=", "20cel",
            "IsMaterialEditable:=", True,
            "UseMaterialAppearance:=", False,
            "IsLightweight:=", False
        ]
    )

def read_csv_with_unit(filename):
```

```
    with open(filename, 'r') as file:
        lines = file.readlines()

    unit = lines[0].strip().split(' ')[-1]  # 取得單位
    data_list = []

    for line in lines[1:]:  # 跳過第一行的單位
        values = line.strip().split(',')
        data_with_unit = [value.strip() + unit for value in values]
        data_list.append(data_with_unit)

    return data_list


import sys
import clr
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import DialogResult, OpenFileDialog
dialog = OpenFileDialog()
dialog.Multiselect = False  # 確保只能選擇一個檔案
dialog.Title = "選擇圓柱資料檔案"  # 設置對話框的標題
dialog.Filter = "CSV檔案 (*.csv)|*.csv"  # 設置檔案篩選器，只顯示CSV檔案

if dialog.ShowDialog() == DialogResult.OK:
    csv_path = dialog.FileName  # 獲取選擇的檔案路徑
    AddWarningMessage(csv_path)  # 顯示檔案路徑
else:
    pass  # 如果未選擇檔案，則不執行任何操作


#csv_path = 'c:/locations.csv'
data = read_csv_with_unit(csv_path)
for n, (x, y, z, radius, height) in enumerate(data):
    create_cylinder('cylinder{}'.format(n), x, y, z, radius, height)
```

執行程式碼，會見到以下結果：



總結

以上的開發流程展示了從開始到結束的一個完整的腳本開發過程。這個過程始於錄製操作的步驟，即在HFSS中進行的用戶互動，這些互動被轉換成可重用的腳本。隨後，這些腳本中的核心操作被封裝成獨立的函數，如 create_cylinder，使其更加模組化且易於管理。

接著，開發過程中引入了進一步的功能增強，比如讀取CSV檔案的函數 read_csv_with_unit。這個步驟使得腳本能夠處理外部數據，增加了靈活性和可擴展性。為了進一步提升用戶體驗，加入了圖形化的檔案選擇對話框，使用者可以通過可視化的界面選擇檔案，而不是在代碼中硬編碼檔案路徑。這一點尤其重要，因為它使得程式更加用戶友好，並允許在不同的情境下重複使用程式。

最終，整個開發流程以整合這些部分並完成最終腳本結束。這個腳本不僅包括圓柱體的自動生成，還包括從用戶選擇的檔案中讀取數據的功能。整個過程是一個典型的從原型到成品的開發旅程，展示了如何逐步構建和完善一個軟件應用的各個方面。