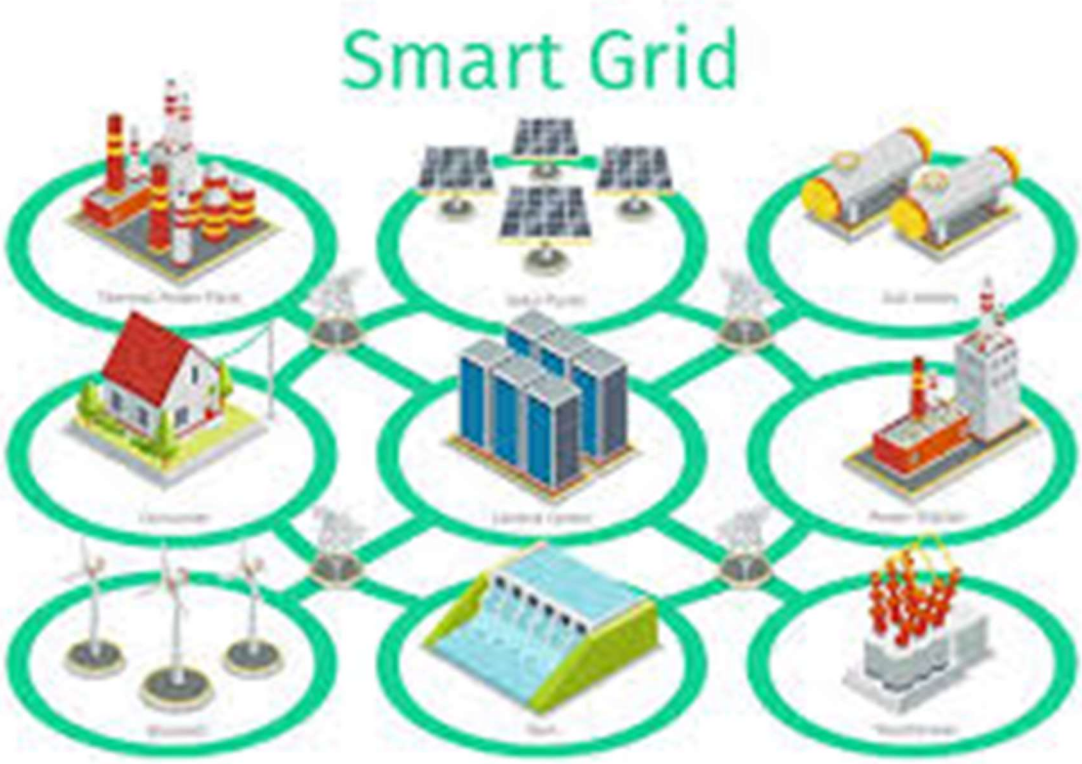
113 學年度第二學期「python 程式設計」課程個人期中報告

模擬智慧微電網初步呈現報告



姓名：陳昱安
學號：B1141065
系級：電機二乙

2024 年 4 月 9 日

Content

（圖一）微電網內部關係圖

一、智慧微電網之介紹

微電網系統（Microgrid）是一個由分散式發電設備和負載所組成的區域電力系統，其與主電網連接但同時也可以獨立運行。微電網為區域內的用戶提供電力，並且可以進行自主的能源管理和控制。

那它主要有以下幾種優勢和功能：

首先是穩定區域電網供電，當主電網發生故障或供電不穩定時，微電網可以提供區域內穩定的電力供應，保證用戶的用電需求。

再來則是調節再生能源的不穩定性，會對電網的穩定性造成一定的影響。微電網可以通過與再生能源的配合運行，調節再生能源的發電波動，降低對主電網的影響。

最後是提高能源利用效率，通過在區域內實現電力、熱能和冷能的綜合利用，進而提高能源的利用效率，以及減少能源的浪費。

智慧微電網為運用資訊、通訊以及自動化科技等技術，讓電網的供電與用電數位化、可視化，並對電力資訊加以整合分析，以達到電力資源的最佳配置；若要確保電網供電安全可靠，必須要更智慧化、即時調節與更精準預測並掌握負載，因此發展智慧電網已成為國際趨勢。

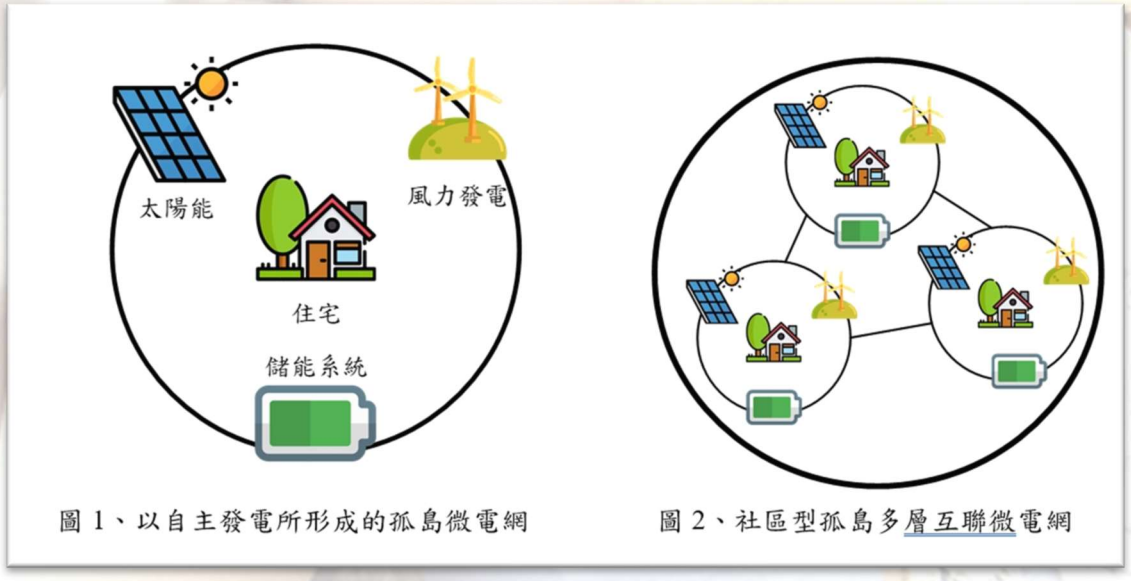


圖 1、以自主發電所形成的孤島微電網　　圖 2、社區型孤島多層互聯微電網

（圖二）不同類型的微電網之呈現

二、預期設計和規劃

（1）預期的功能和設計

對於模擬智慧微電網的製作，我們預期會需要它應有以下幾個功能及相關的規格：

1. 此微電網應有三個發電站系統所組成，分別為 NTUST 的核能、NCHU 的風力能、NPUST 的太陽能。通過三個機構電力的適時供應及傳輸，進而得以維持小區域之緊急電力供給和正常機能運作。
2. 其中每個發電站都可進行內部電量偵測以及其電能在每一時刻的數字變動。此外其偵測的過程也較為人性化。
3. 再有一個系統控制中樞，可對於三個發電站中的目前電量進行彼此調度或是偵測，亦可遠端地進行個別發電站的運作狀況之調整（ex：因為近期有地震，使用者就會藉由此中樞來關閉核能發電站。），進而提升微電網的完整和便利性。

（2）進程規劃

<期中>

一、設計和開發三個發電站系統

1. NTUST 的核能發電系統，包括反應爐模擬、能量產生和數據偵測等功能。
2. NCHU 的風力能發電系統，包括風力發電機模擬、風速偵測和能量產生等功能。
3. NPUST 的太陽能發電系統，包括太陽能電池板模擬、日照偵測和能量產生等功能。

二、內部偵測功能設計

為每個發電站系統添加內部偵測功能，可以監測發電量的變化和系統運行狀況，並將數據傳送至系統控制中樞。

三、電能數值設計

設計能夠實時顯示每個發電站系統的電能數值，包括目前發電量、累計發電量等數據，並能夠以圖表等形式直

觀展示。

〈期末〉

一、系統控制中樞設計和開發

設計並開發微電網的系統控制中樞，可以對三個發電站系統的運行狀況進行遠程監控和調度。

中樞系統應具備自動切換功能，能夠根據系統需求自動調整各發電站的運行模式，並能夠根據外部事件（如地震等）做出相應調整。

二、系統整合和測試

將三個發電站系統和系統控制中樞進行整合，確保各個模塊能夠正常協同工作。

進行系統的測試和驗證，包括功能測試、性能測試和安全測試等，確保系統的穩定性和可靠性。

三、相關系統介紹(期中)

接下來就對於我們微電網中的相關系統和所應用的程式進行說明，那目前我完成了三個發電站系統以及內部電量偵測和電能的單向調度（有點類似使用者直接對單一發電站之電量抽取）。以下是三個發電系統之介紹：

（1）NTUST 核能發電系統

1. 主要程式

```
        def get_hourly_energy_production(self):
                return self.hourly_energy_production

        def extract_energy(self, amount):
                max_extraction = self.total_energy_produced * 0.6
                if amount <= max_extraction and amount <= self.total_energy_produced:
                        self.total_energy_produced -= amount
                        self.available_energy += amount
                        self.total_energy_extracted += amount
                        print(f"成功提取 {amount} kw 能量。")
                        extraction_index = len(self.hourly_energy_production) - 1
                        self.hourly_energy_production[extraction_index] -= amount
                elif amount > max_extraction:
                        print("電量提取失敗：提取的電量超過總發電量的60%。")
                else:
                        print("能量提取失敗：提取的能量超過總發電量。")
                print(f"剩餘電量: {self.total_energy_produced} kw")

        def get_total_energy_extracted(self):
                return self.total_energy_extracted

def main():
        default_reactor_count = 4
        max_reactor_output = 2000
        operation_hours = 24

        efficiency = round((max_reactor_output * default_reactor_count) / (operation_hours),

        print("歡迎使用NTUST核能發電系統模擬器！")
        print(f"目前有 {default_reactor_count} 個反應爐，每個效率為 {efficiency} kw。")
```

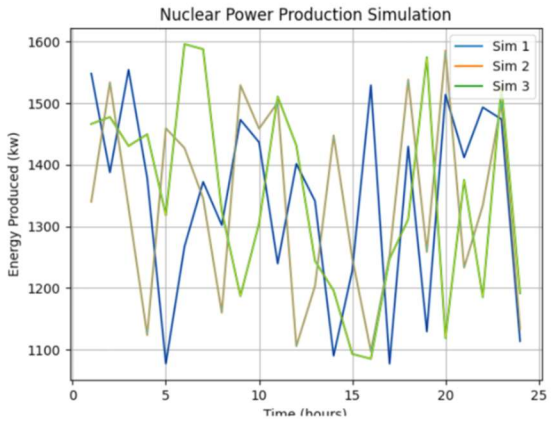
```
import random
import matplotlib.pyplot as plt
import numpy as np

class NuclearPowerPlant:
        def __init__(self, reactor_count, efficiency):
                self.reactor_count = reactor_count
                self.efficiency = efficiency
                self.total_energy_produced = 0
                self.available_energy = 0
                self.total_energy_extracted = 0
                self.hourly_energy_production = []

        def generate_energy(self, hours_of_operation):
                self.hourly_energy_production = []
                for hour in range(hours_of_operation):
                        random_factor = random.uniform(0.8, 1.2)
                        energy_per_reactor = self.efficiency * random_factor
                        total_hourly_energy = energy_per_reactor * self.reactor_count
                        self.total_energy_produced += total_hourly_energy
                        self.hourly_energy_production.append(total_hourly_energy)

        def get_total_energy_produced(self):
                return self.total_energy_produced
```

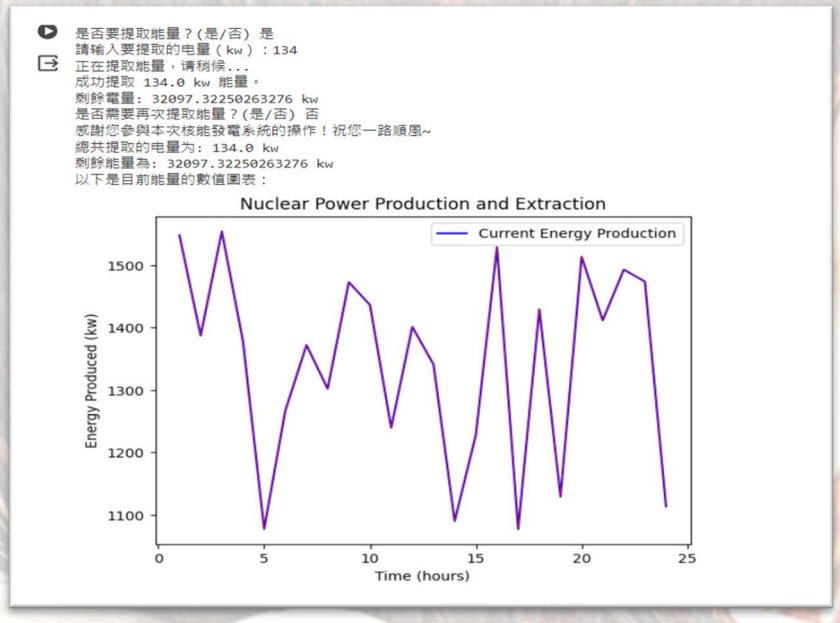
```
        simulation_count = int(input("請輸入要模擬的次數："))
        total_energy_produced_all_simulations = 0
        hourly_energy_production_all_simulations = [[] for _ in range(simulation_count)]

        for i in range(simulation_count):
            print(f"核能發電模擬第{i+1}次開始...")
            nuclear_plant = NuclearPowerPlant(default_reactor_count, efficiency)
            nuclear_plant.generate_energy(operation_hours)
            hourly_energy_production = nuclear_plant.get_hourly_energy_production()
            total_energy_produced_all_simulations += sum(hourly_energy_production)
            hourly_energy_production_all_simulations[i] = hourly_energy_production
            print(f"核能發電模擬第{i+1}次結束。")

        for i in range(simulation_count):
            plt.plot(range(1, operation_hours + 1), hourly_energy_production_all_simulations[i], label=f'Sim {i+1}')

        for i in range(simulation_count):
            plt.plot(range(1, operation_hours + 1), hourly_energy_production_all_simulations[i], linestyle='-',
                     color=plt.cm.jet(i / simulation_count), linewidth=0.5)

        plt.xlabel('Time (hours)')
        plt.ylabel('Energy Produced (kw)')
        plt.title('Nuclear Power Production Simulation')
        plt.legend()
        plt.grid(True)
        print("以下是目前能量的相關圖表：")
        plt.show()
```

```
        while True:
            fig, ax = plt.subplots()
            ax.plot(range(1, operation_hours + 1), hourly_energy_production_all_simulations[0], label='Current Energy Production', color='blue')
            ax.plot(range(1, operation_hours + 1), hourly_energy_production_all_simulations[0], linestyle='-', color='red', linewidth=0.5)

            ax.set_xlabel('Time (hours)')
            ax.set_ylabel('Energy Produced (kw)')
            ax.set_title('Nuclear Power Production and Extraction')
            ax.legend(loc='upper right')

            extract_energy_choice = input("是否要提取能量？(是/否) ").strip().lower()
            if extract_energy_choice != "是":
                break
            amount_to_extract = float(input("請輸入要提取的电量（kw）："))
            print("正在提取能量，请稍候...")
            nuclear_plant.extract_energy(amount_to_extract)
            if input("是否需要再次提取能量？(是/否) ").strip().lower() != "是":
                break
        print("感謝您參與本次核能發電系統的操作！祝您一路順風~")
        print(f"總共提取的电量为: {nuclear_plant.get_total_energy_extracted()} kw")
        print(f"剩餘能量為: {nuclear_plant.get_total_energy_produced()} kw")
        print("以下是目前能量的數值圖表：")

if __name__ == "__main__":
    main()
```

2. 功能呈現

```
歡迎使用NTUST核能發電系統模擬器！
目前有 4 個反應爐，每個效率為 333.333 kw。
請輸入要模擬的次數：3
核能發電模擬第1次開始...
核能發電模擬第1次結束。
核能發電模擬第2次開始...
核能發電模擬第2次結束。
核能發電模擬第3次開始...
核能發電模擬第3次結束。
以下是目前能量的相關圖表：
```





〈程式解釋〉

1. 首先導入三個 Python 標準庫：random、matplotlib.pyplot 和 numpy。random 用於生成隨機數，matplotlib.pyplot 用於繪製圖表，numpy 用於數值運算。

2. 接下來定義一個類別 NuclearPowerPlant，用來表示核能發電廠。在這個類別中，我們定義了一些屬性和方法，包括初始化方法 __init__、生成能量方法 generate_energy、提取能量方法 extract_energy，以及一些用於獲取屬性值的方法。

3. 最後是 main 函式，是整個程式的入口點。在這個函式中，初始化了一些參數，例如反應爐數量、最大反應爐輸出、模擬的運作時間等，並根據這些參數初始化了核能發電廠對象。然後，進行了多次模擬，並繪製了相應的能量產生圖表。最後，提供了用戶選擇是否提取能量的功能，並根據用戶的輸入進行能量的提取操作。

（2）NCHU 風力發電系統

1. 主要程式

```python
import random
import matplotlib.pyplot as plt
import numpy as np

class WindPowerPlant:
    def __init__(self, turbine_count, efficiency):
        self.turbine_count = turbine_count
        self.efficiency = efficiency
        self.total_energy_produced = 0
        self.available_energy = 0
        self.total_energy_extracted = 0
        self.hourly_energy_production = []

    def generate_energy(self, hours_of_wind):
        self.hourly_energy_production = []
        for hour in range(hours_of_wind):
            # 使用弦波函數模擬能量產出的波動
            sin_factor = 0.5 * np.sin(hour * (2 * np.pi / 24)) + 1
            energy_per_turbine = self.efficiency * sin_factor
            total_hourly_energy = energy_per_turbine * self.turbine_count
            self.total_energy_produced += total_hourly_energy
            self.hourly_energy_production.append(total_hourly_energy)

    # 獲取總發電量
    def get_total_energy_produced(self):
        return self.total_energy_produced

    def get_hourly_energy_production(self):
        return self.hourly_energy_production

    def extract_energy(self, amount):
        max_extraction = self.total_energy_produced * 0.6
        if amount <= max_extraction and amount <= self.total_energy_produced:
            self.total_energy_produced -= amount
            self.available_energy += amount
```

```python
            self.total_energy_extracted += amount
            print(f"成功提取 {amount} kw能量。")
            # 更新提取能量後的能量產生列表
            extraction_index = len(self.hourly_energy_production) - 1
            self.hourly_energy_production[extraction_index] -= amount
        elif amount > max_extraction:
            print("電量提取失敗：提取的電量超過總發電量的60%。")
        else:
            print("能量提取失敗：提取的能量超過總發電量。")
        print(f"剩餘電量: {self.total_energy_produced} kw")

    def get_total_energy_extracted(self):
        return self.total_energy_extracted

def main():
    default_turbine_count = 5
    max_turbine_power = 2000
    turbine_blade_length = 15    # 假設風力發電機的葉片長度為15米
    speed_Max = 61.2
    speed_Min = 0

    month_wind_speeds = {
        7: (8, 12),      # 夏季風速快
        8: (8, 12),      # 夏季風速快
        9: (8, 12),      # 夏季風速快
        10: (8, 12),     # 夏季風速快
        11: (5, 8),      # 冬季風速中等
        12: (5, 8),      # 冬季風速中等
        1: (5, 8),       # 冬季風速中等
        2: (5, 8),       # 冬季風速中等
        3: (5, 8),       # 冬季風速中等
        4: (5, 8),       # 冬季風速中等
        5: (3, 5),       # 春季風速較小
        6: (3, 5)        # 春季風速較小
    }
```

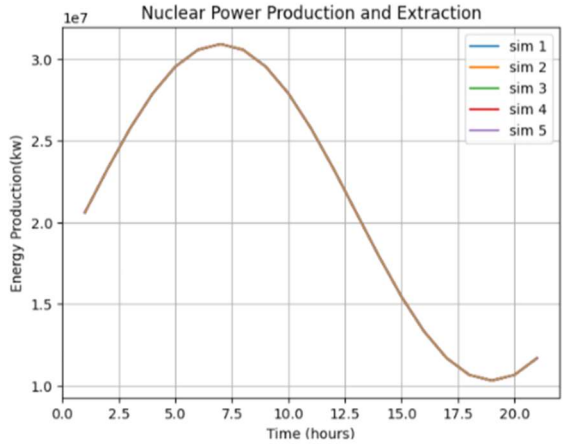
```
        print("歡迎使用NCHU風力發電站模擬系統！")
        know_wind_hours = input("請問是否知道今天的風速持續時間？(是/否) ").strip().lower()

        if know_wind_hours == "是":
            turbine_speed_min = float(input("請輸入風速範圍下限 (m/s)，勿低於0m/s: "))
            while turbine_speed_min < speed_Min:
                turbine_speed_min = float(input(f"風速範圍下限超出範圍，請重新輸入 (不超過{ speed_Min } m/s)："))
            turbine_speed_max = float(input("請輸入風速範圍上限 (m/s)，勿超過61.2m/s: "))

            while turbine_speed_max <= turbine_speed_min or turbine_speed_max > speed_Max:
                if turbine_speed_max <= turbine_speed_min:
                    turbine_speed_max = float(input(f"風速範圍上限必須大於下限 {turbine_speed_min} m/s，請重新輸入："))
                else:
                    turbine_speed_max = float(input(f"風速範圍上限不能超過 { speed_Max } m/s，請重新輸入："))

            hours_of_wind = int(input("請輸入風速持續時間 (小時)："))

        elif know_wind_hours == "否":
            month = int(input("請輸入現在的月份 (1-12)："))
            if month in month_wind_speeds:
                wind_speed_range = month_wind_speeds[month]
            else:
                print("無法識別的月份，將使用預設風速範圍。")
                wind_speed_range = (5, 10)  # 使用預設風速範圍

            print(f"預設風速範圍為：{wind_speed_range[0]} m/s 到 {wind_speed_range[1]} m/s")

            # 隨機選擇下限
            turbine_speed_min = random.randint(wind_speed_range[0], wind_speed_range[1])
            print(f"目前的風速範圍下限設定為：{turbine_speed_min} m/s")

            # 隨機選擇上限，要確保上限大於下限
            turbine_speed_max = random.randint(turbine_speed_min, wind_speed_range[1])
            while turbine_speed_max <= turbine_speed_min:  # 如果 max 小於等於 min，重新選擇 max
                turbine_speed_max = random.randint(turbine_speed_min, wind_speed_range[1])
```

```
efficiency = round((0.5 * max_turbine_power * ((turbine_speed_max + turbine_speed_min) / 2) ** 3 * np.pi * turbine_blade_length ** 2) / 1000, 3)

print(f"目前有 {default_turbine_count} 台風力發電機，每台效率為 {efficiency} kw。")

simulation_count = int(input("請輸入要模擬的次數："))
total_energy_produced_all_simulations = 0
hourly_energy_production_all_simulations = [[] for _ in range(simulation_count)]

for i in range(simulation_count):
    print(f"風力發電模擬第{i+1}次開始...")
    wind_plant = WindPowerPlant(default_turbine_count, efficiency)
    wind_plant.generate_energy(hours_of_wind)
    hourly_energy_production = wind_plant.get_hourly_energy_production()
    total_energy_produced_all_simulations += sum(hourly_energy_production)
    hourly_energy_production_all_simulations[i] = hourly_energy_production
    print(f"風力發電模擬第{i+1}次結束。")

for i in range(simulation_count):
    plt.plot(range(1, hours_of_wind + 1), hourly_energy_production_all_simulations[i], label=f'sim {i+1}')

for i in range(simulation_count):
    plt.plot(range(1, hours_of_wind + 1), hourly_energy_production_all_simulations[i], linestyle='-',
             color=plt.cm.jet(i / simulation_count), linewidth=0.5)

plt.xlabel('Time (hours)')
plt.ylabel('Energy Production(kw)')
plt.title('Nuclear Power Production and Extraction')
plt.legend()
plt.grid(True)
print("以下是目前能量的相關圖表：")
plt.show()
```

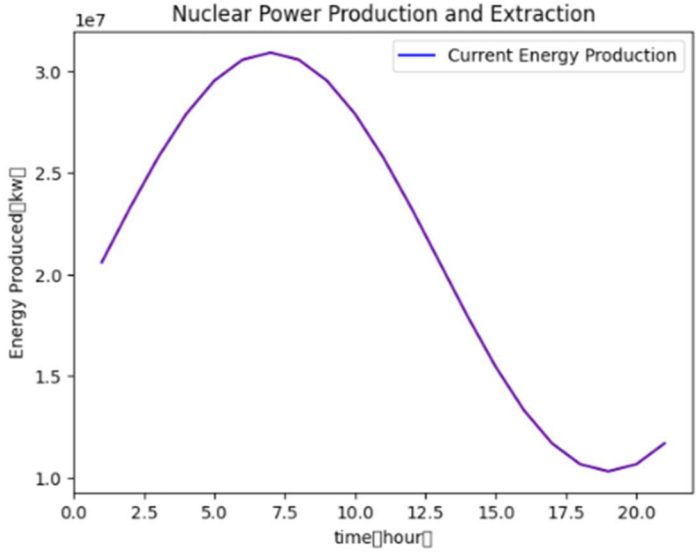
```
        plt.ylabel('Energy Production(kw)')
        plt.title('Nuclear Power Production and Extraction')
        plt.legend()
        plt.grid(True)
        print("以下是目前能量的相關圖表：")
        plt.show()

        # 提取能量過程，同時繪製戴波器圖表
        while True:
            fig, ax = plt.subplots()
            ax.plot(range(1, hours_of_wind + 1), hourly_energy_production_all_simulations[0], label='Current Energy Production', color='blue')
            ax.plot(range(1, hours_of_wind + 1), hourly_energy_production_all_simulations[0], linestyle='-', color='red', linewidth=0.5)

            ax.set_xlabel('time (hour)')
            ax.set_ylabel('Energy Produced (kw)')
            ax.set_title('Nuclear Power Production and Extraction')
            ax.legend(loc='upper right')

            extract_energy_choice = input("是否要提取能量？(是/否) ").strip().lower()
            if extract_energy_choice != "是":
                break
            amount_to_extract = float(input("請輸入要提取的電量 (kw)："))
            print("正在提取能量，請稍候...")
            wind_plant.extract_energy(amount_to_extract)
            if input("是否需要再次提取能量？(是/否) ").strip().lower() != "是":
                break
        print("感謝您參與本次風力發電站的操作！祝您一路順風~")
        print(f"總共提取的電量為: {wind_plant.get_total_energy_extracted()} 千瓦")
        print(f"剩餘的能量為: {wind_plant.get_total_energy_produced()} 千瓦")
        print("以下是目前能量的數值圖：")

if __name__ == "__main__":
    main()
```

2. 功能呈現

```
歡迎使用NCHU風力發電站模擬系統！
請問是否知道今天的風速持續時間？(是/否) 是
請輸入風速範圍下限（m/s），勿低於0m/s：13
請輸入風速範圍上限（m/s），勿超過61.2m/s：23
請輸入風速持續時間（小時）：21
目前有 5 台風力發電機，每台效率為 4122397.88 kw。
請輸入要模擬的次數：5
風力發電模擬第1次開始...
風力發電模擬第1次結束。
風力發電模擬第2次開始...
風力發電模擬第2次結束。
風力發電模擬第3次開始...
風力發電模擬第3次結束。
風力發電模擬第4次開始...
風力發電模擬第4次結束。
風力發電模擬第5次開始...
風力發電模擬第5次結束。
以下是目前能量的相關圖表：
```




```
是否要提取能量？(是/否) 是
請輸入要提取的電量（kw）：1334
正在提取能量，請稍候...
成功提取 1334.0 kw能量。
剩餘電量: 447958267.1963282 kw
是否需要再次提取能量？(是/否) 否
感謝您參與本次風力發電站的操作！祝您一路順風~
總共提取的電量為: 1334.0 千瓦
剩餘的能量為: 447958267.1963282 千瓦
以下是目前能量的數值圖：
```



<程式解釋>

1.首先導入三個 Python 標準庫：random 用於生成隨機數，matplotlib.pyplot 用於繪製圖表，numpy 用於數值運算。

2.接著定義一個類別 WindPowerPlant，表示風力發電站。這個類別包含了風力發電站的屬性和方法，包括初始化方法 __init__、生成能量方法 generate_energy、提取能量方法 extract_energy，以及一些用於獲取屬性值的方法。

3.main 函式是整個程式的入口點。在這個函式中，我們首先設置了一些參數，例如風力發電機數量、最大風力發電機輸出和模擬的時間等。然後，我們提示用戶輸入要模擬的次數，並開始進行多次模擬。每次模擬都會生成能量數據，然後我們繪製了相應的能量產生圖表。

接下來，我們提示用戶是否要提取能量。如果用戶選擇提取能量，我們會要求用戶輸入要提取的能量，然後模擬提取過程，同時更新能量產生列表。然後，我們繪製了提取能量的過程圖表。

最後，我們輸出了總共提取的能量量以及剩餘的能量量，並顯示了最終的能量數值圖表。

（3）NPUST 太陽能發電系統

1. 主要程式

```
import random
import matplotlib.pyplot as plt
import numpy as np

class SolarPowerPlant:
    def __init__(self, solar_panel_count, efficiency):
        self.solar_panel_count = solar_panel_count
        self.efficiency = efficiency
        self.total_energy_produced = 0
        self.available_energy = 0
        self.total_energy_extracted = 0
        self.hourly_energy_production = []

    def generate_energy(self, hours_of_sunshine):
        self.hourly_energy_production = []
        for hour in range(hours_of_sunshine):
            # 添加随机因素模擬能量产出的波動
            random_factor = random.uniform(0.8, 1.2)
            # 使用sin函数模擬能量产出的波動
            sin_factor = 0.5 * np.sin(hour * (2 * np.pi / 24)) + 1
            energy_per_panel = self.efficiency * random_factor * sin_factor
            total_hourly_energy = energy_per_panel * self.solar_panel_count
            self.total_energy_produced += total_hourly_energy
            self.hourly_energy_production.append(total_hourly_energy)

    # 獲取總發電量
    def get_total_energy_produced(self):
        return self.total_energy_produced

    def get_hourly_energy_production(self):
        return self.hourly_energy_production
```

```
    def extract_energy(self, amount):
        max_extraction = self.total_energy_produced * 0.6
        if amount <= max_extraction and amount <= self.total_energy_produced:
            self.total_energy_produced -= amount
            self.available_energy += amount
            self.total_energy_extracted += amount
            print(f"成功提取 {amount} kw 能量。")
            # 更新提取能量后的能量产生列表
            extraction_index = len(self.hourly_energy_production) - 1
            self.hourly_energy_production[extraction_index] -= amount
        elif amount > max_extraction:
            print("电量提取失敗：提取的電量超过總發電量的60%。")
        else:
            print("能量提取失敗：提取的能量超过總發電量。")
        print(f"剩餘電量: {self.total_energy_produced} kw")

    def get_total_energy_extracted(self):
        return self.total_energy_extracted

def main():
    default_solar_panel_count = 20
    max_module_power = 5000
    module_area = 1.6
    solar_irradiance = 1000

    efficiency = round((max_module_power / module_area) / solar_irradiance * 100, 3)

    print("歡迎使用NPUST太陽能發電站模擬系統！")
    print(f"目前有 {default_solar_panel_count} 台太陽能電池板，每塊效率為 {efficiency} kw。")
```

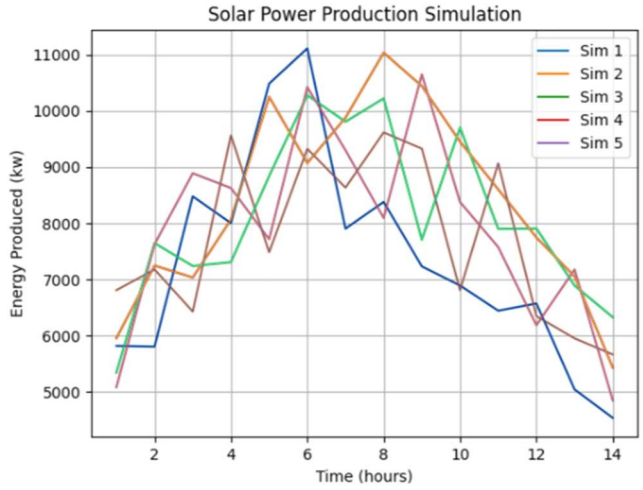
```
        while True:
            know_sunshine_hours = input("是否知道今天的太陽照射時間？(是/否) ").strip().lower()
            if know_sunshine_hours == "是":
                hours_of_sunshine = int(input("請輸入太阳照射时间（小時）："))
                break
            elif know_sunshine_hours == "否":
                weather_condition = input("今天是晴天、陰天还是雨天？ ").strip().lower()
                weather_mapping = {"晴天": 10, "陰天": 5, "雨天": 2}
                hours_of_sunshine = weather_mapping.get(weather_condition, 0)
                if hours_of_sunshine:
                    break
                else:
                    print("無法識別的天气狀況，請重新輸入。")

        simulation_count = int(input("請輸入要模擬的次数："))
        total_energy_produced_all_simulations = 0
        hourly_energy_production_all_simulations = [[] for _ in range(simulation_count)]

        for i in range(simulation_count):
            print(f"太陽能發電模擬第{i+1}次開始...")
            solar_plant = SolarPowerPlant(default_solar_panel_count, efficiency)
            solar_plant.generate_energy(hours_of_sunshine)
            hourly_energy_production = solar_plant.get_hourly_energy_production()
            total_energy_produced_all_simulations += sum(hourly_energy_production)
            hourly_energy_production_all_simulations[i] = hourly_energy_production
            print(f"太陽能發電模擬第{i+1}次结束。")

        for i in range(simulation_count):
            plt.plot(range(1, hours_of_sunshine + 1), hourly_energy_production_all_simulations[i], label=f'Sim {i+1}')
```

```
            plt.plot(range(1, hours_of_sunshine + 1), hourly_energy_production_all_simulations[i], linestyle='-',
                     color=plt.cm.jet(i / simulation_count), linewidth=0.5)

    plt.xlabel('Time (hours)')
    plt.ylabel('Energy Produced (kw)')
    plt.title('Solar Power Production Simulation')
    plt.legend()
    plt.grid(True)
    print("以下是目前能量的相關圖表：")
    plt.show()
```

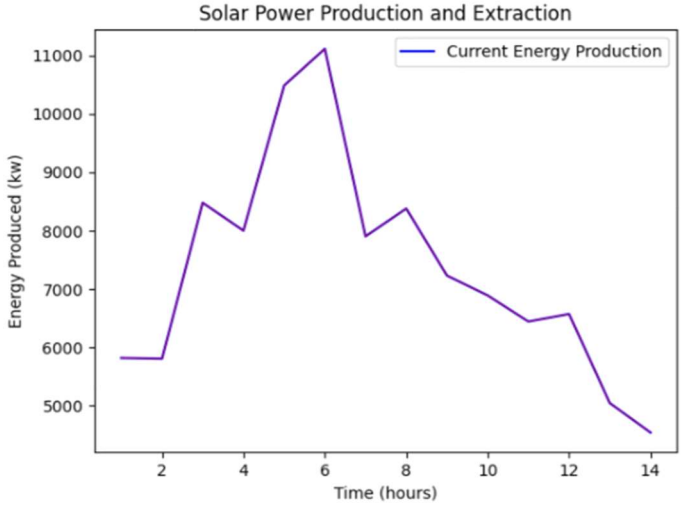
```
        # 提取能量過程，同时繪製截波器圖表
        while True:
            fig, ax = plt.subplots()
            ax.plot(range(1, hours_of_sunshine + 1), hourly_energy_production_all_simulations[0], label='Current Energy Production', color='blue')
            ax.plot(range(1, hours_of_sunshine + 1), hourly_energy_production_all_simulations[0], linestyle='-', color='red', linewidth=0.5)

            ax.set_xlabel('Time (hours)')
            ax.set_ylabel('Energy Produced (kw)')
            ax.set_title('Solar Power Production and Extraction')
            ax.legend(loc='upper right')

            extract_energy_choice = input("是否要提取能量？(是/否) ").strip().lower()
            if extract_energy_choice != "是":
                break
            amount_to_extract = float(input("請輸入要提取的电量（kw）："))
            print("正在提取能量，请稍候...")
            solar_plant.extract_energy(amount_to_extract)
            if input("是否需要再次提取能量？(是/否) ").strip().lower() != "是":
                break
        print("感謝您參與本次太陽能發電站的操作！祝您一路順風~")
        print(f"總共提取的电量为: {solar_plant.get_total_energy_extracted()} kw")
        print(f"剩餘能量為: {solar_plant.get_total_energy_produced()} kw")
        print("以下是目前能量的數值圖表：")

if __name__ == "__main__":
    main()
```

2. 呈現結果

```
歡迎使用NPUST太陽能發電站模擬系統！
目前有 20 台太陽能電池板，每塊效率為 312.5 kw。
是否知道今天的太陽照射時間？(是/否) 是
請輸入太阳照射时间（小時）：14
請輸入要模擬的次數：5
太陽能發電模擬第1次開始...
太陽能發電模擬第1次結束。
太陽能發電模擬第2次開始...
太陽能發電模擬第2次結束。
太陽能發電模擬第3次開始...
太陽能發電模擬第3次結束。
太陽能發電模擬第4次開始...
太陽能發電模擬第4次結束。
太陽能發電模擬第5次開始...
太陽能發電模擬第5次結束。
以下是目前能量的相關圖表：
```




```
是否要提取能量？(是/否) 是
請输入要提取的电量（kw）：1323
正在提取能量，请稍候...
成功提取 1323.0 kw 能量。
剩餘電量: 109200.79403221671 kw
是否需要再次提取能量？(是/否) 否
感謝您參與本次太陽能發電站的操作！祝您一路順風~
總共提取的电量为: 1323.0 kw
剩餘能量為: 109200.79403221671 kw
以下是目前能量的數值圖表：
```



<程式解釋>

1.首先導入三個 Python 標準庫：random 用於生成隨機數，matplotlib.pyplot 用於繪製圖表，numpy 用於數值運算。

2. 接著我們定義了一個類別 SolarPowerPlant，表示太陽能發電站。這個類別包含了太陽能發電站的屬性和方法，包括初始化方法 __init__、生成能量方法 generate_energy、提取能量方法 extract_energy，以及一些用於獲取屬性值的方法。

3.main 函式是整個程式的入口點。在這個函式中，我們首先設置了一些參數，例如太陽能電池板數量、最大模塊輸出功率和太陽照射強度等。然後，我們提示用戶輸入今天的太陽照射時間，並根據用戶的回答設定模擬的時間。接著，我們提示用戶輸入要模擬的次數，並開始進行多次模擬。每次模擬都會生成能量數據，然後我們繪製了相應的能量產生圖表。

4.接下來，提示用戶是否要提取能量。如果用戶選擇提取能量，會要求用戶輸入要提取的能量，然後模擬提取過程，同時更新能量產生列表。然後，我們繪製了提取能量的過程圖表。

5.最後，輸出總共提取的能量以及剩餘的能量，並顯示了最終的能量數值圖表。

四、心得和收穫

通過撰寫有關模擬智慧微電網系統，我不僅深入了解了智慧微電網的概念和優勢，還學到了如何利用編程技術來模擬和探索這一系統的運行。

在撰寫程式的過程中，我對智慧微電網系統的工作原理和運行流程有了更加具體的理解。通過模擬電力發電過程、能量提取和系統監控等功能，我更清晰地了解了智慧微電網如何利用多種能源資源進行能源生產和管理，以及如何實現能源的有效利用和供應的穩定性。

另外，自己從中也對於 Python 編程語言的應用和技巧有了更深入的理解和掌握。在程式設計過程中，我學會了如何使用類和方法來組織程式碼，如何利用循環和條件語句來實現程式的流程控制，以及如何使用第三方庫來實現複雜的數據處理和可視化功能。

總之通過撰寫程式，我不僅加深了對這一系統的理解，還提升了自己的編程能力和技術水平。這一過程不僅讓我更加熟悉了智慧微電網系統的概念和運行方式，還為我未來在能源管理和編程領域的學習和發展打下了堅實的基礎。

五、參考文獻

1.Remotek(2016)。微電網圖。檢自：https://www.remotek.com.tw/zh-hant/%E5%BE%AE%E9%9B%BB%E7%B6%B2/
2.能源教育資源總中心（2021）。互聯微電網展示場域。檢自：https://learnenergy.tw/index.php?inter=knowledge&caid=1&id=832
3.Evcs Production(2024)。Top Challenges and Solutions in Smart Grid Technology.檢自：https://evchargingsummit.com/blog/top-challenges-and-solutions-in-smart-grid-technology/
4.JaneClare（2024）。致需要平靜心靈的你。
檢自：https://www.janeclare.com/page/healing-the-mind