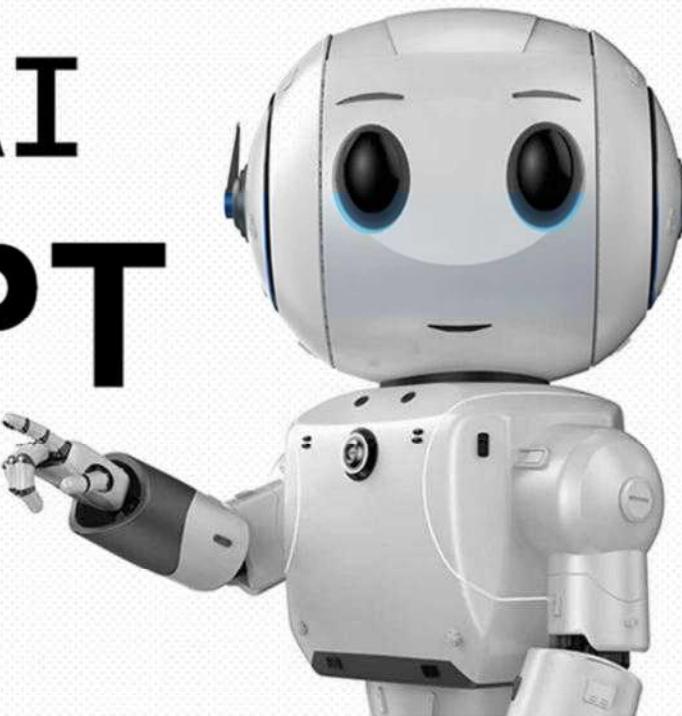


# ChatGPT 簡介與設定

ChatGPT Introduction & Setting



OpenAI  
**ChatGPT**



# AIGC 生成式 AI

---

- ◆ 人工智慧（AI）是一種涵蓋範圍很廣的技術，它包括機器學習、深度學習、自然語言處理等多種子領域。
- AIGC ( Artificial Intelligence Generated Content ) 則是 AI 的一個特定應用領域，它專注於使用 AI 建立新的、原創的內容。



# 生成式 AI 與一般 AI 的差異

---

- ◆ 人工智慧（AI）是一種涵蓋範圍很廣的技術，它包括機器學習（ML）、深度學習（DL）、自然語言處理（NLP）等多種子領域。AI 可以被設計來完成各種任務，從簡單的如排序和分類，到複雜如圖片識別、語音識別等。
- ◆ 人工智慧產生內容（AIGC）則是 AI 的一個特定應用領域，它專注於使用 AI 建立新的、原創的內容，包括文字、圖片、音樂、影片等各種形式的內容。



# AIGC 應用範例

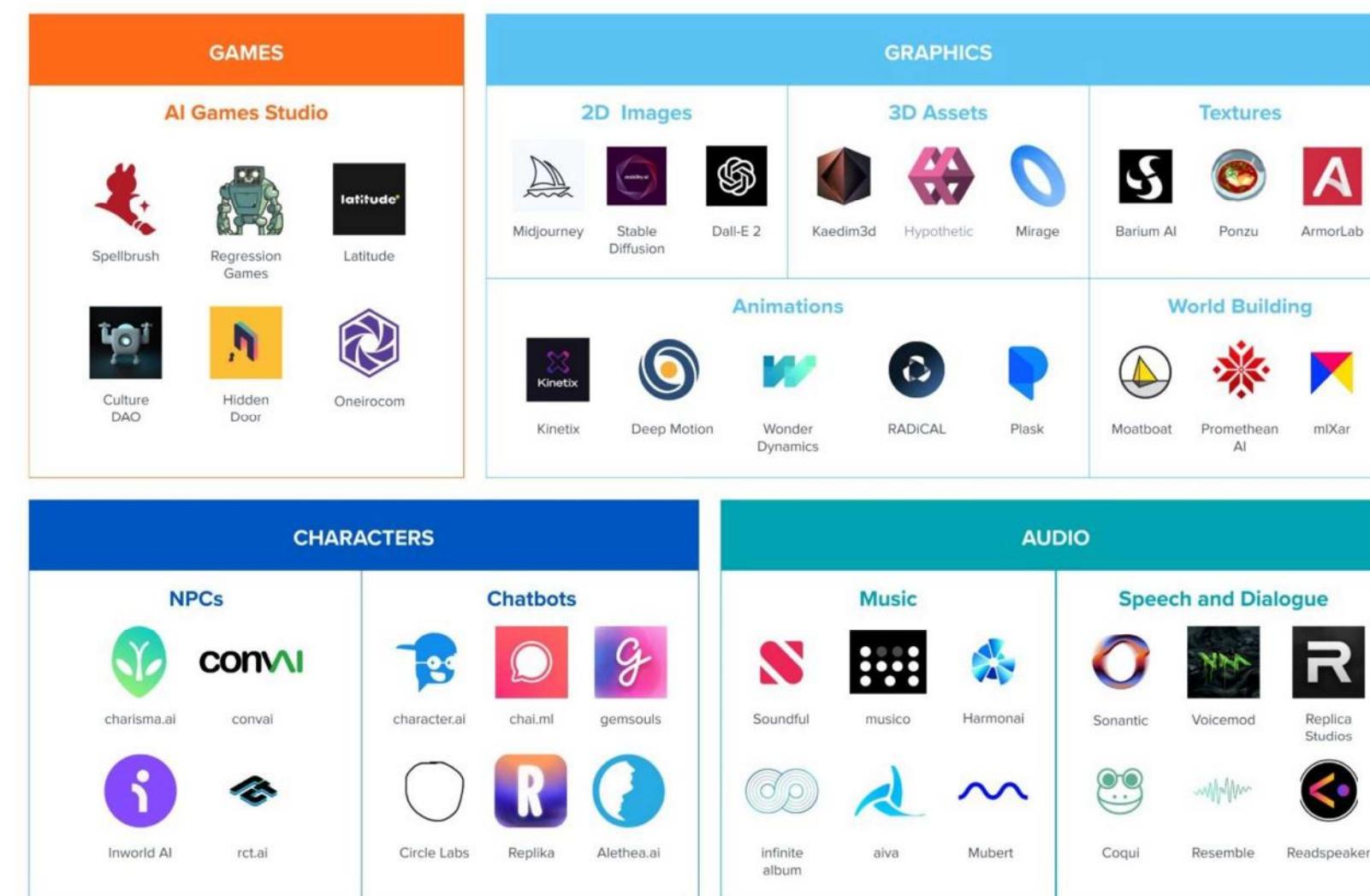
- ◆ **文字產生**：目前非常多人使用的 ChatGPT 是一種使用 AI 來產生文字的技術。例如使用者可以給它一個主題，例如「氣候變化的影響」，之後 AI 就可以自動產生一篇完整的文章，內容可能會包括事實、觀點或建議，就像是人類寫的一樣。
- ◆ **圖片產生**：GANs ( 產生對抗網路 ) 就是一種用於產生圖片的 AI 技術。例如最為人所知的 midjourney(付費) 或是 Leonardo(免費)，給一個描述，如「一個穿黃色毛衣的小孩在學校裡跑步」，它可以產生一個對應的圖片。
- ◆ **音樂生成**：有些 AI 可以生成音樂。您可以給一個主題或者風格，就可以創作出一首歌曲。
- ◆ **影片生成**：AI 也可以生成影片。例如，有些 AI 可以生成深度偽造 ( deepfakes )，這種技術可以將一個人的臉替換成另一個人的臉。



# AIGC 應用範例

- ◆ 總體來說所有的 AIGC 都是 AI，但不是所有的 AI 都被用來生成內容。例如有些 AI 被用來預測股票市場的走勢，或者用來控制自駕車，這些就不屬於 AIGC。主要區別在於 AI 的使用目的和功能：AI 可以用來完成各種任務，而 AIGC 則專注於創建新的內容。

Generative AI for Games Market Map



# AIGC 技術與挑戰

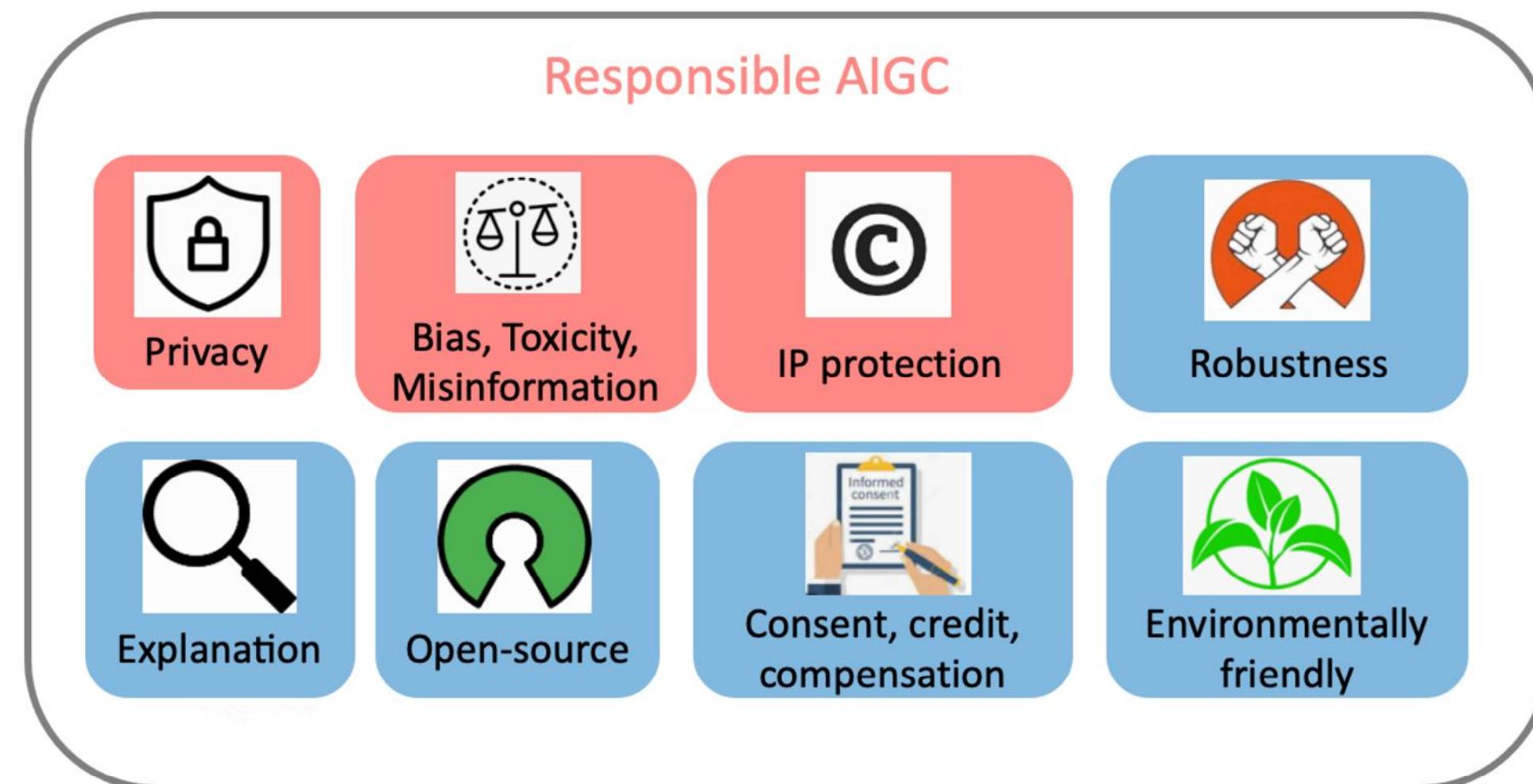
---

- ◆ 儘管 AIGC 技術取得了矚目的成就，但仍然存在一些挑戰和問題需要解決，其中之一是生成內容的真實性和道德問題。
- ◆ 由於機器生成的內容並非由真實人類創作，可能存在知識的錯誤或不準確性，以及倫理和版權的問題，因此，需要建立相應的監管機制和準則，確保 AIGC 的使用是合法和合理的。
- ◆ 還有一些道德和法律問題，例如版權問題、假新聞的傳播、深度偽造的濫用等。因此，雖然 AIGC 具有巨大的潛力，但也需要謹慎地使用它。
- ◆ 此外，AIGC 技術發展還需要更強大的計算能力和運算平台支持，特殊應用晶片（ASIC）和其他高性能硬體設備的發展，對 AIGC 的應用和性能提升起到關鍵作用，能更快速、更準確的生成內容，並推動 AIGC 技術在各個領域的廣泛應用。



# AIGC 未來發展

◆ 展望未來，AIGC 將持續引領人工智能領域的創新浪潮，隨著技術的不斷進步和硬體的不斷優化，AIGC 能夠生成更加逼真、多樣性和具有創造力的內容，繼續改變人們的生活方式，帶來更多的便利和樂趣，AIGC 的崛起將為人工智能領域帶來更多的創新和可能性，這一領域的不斷發展和應用將與人類的創造力和智慧相互融合，共同推動科技的進步，為人類社會帶來更美好的未來。



# 什麼是 GPT？

---

- ◆ GPT ( Generative Pre-trained Transformer ) 是一種基於 Transformer 模型的語言生成模型，廣泛應用於機器翻譯、文本生成、對話系統等各種應用領域。
- **G** : Generative ( 生成式模型 )，根據大量的訓練數據，學習自然語言的結構和模式，生成對話文字。
- **P** : Pre-trained ( 預訓練 )，基於大量數據訓練模型，使得模型有更好的特徵提取能力。
- **T** : Transformer ( 轉換器 )，學習文字結構和上下文關係，產生符合語法和語義的對話。



# 什麼是 GPT？

---

- ◆ GPT 的發展歷史可以追溯到 2018 年，當時由 OpenAI 首次提出。GPT 的訓練方式是基於無監督學習，使用大量的數據進行預訓練，透過預測下一個詞的任務，模型能夠學習到人類語言的結構、語義和邏輯。
- ◆ 下方表格列出 GPT 的發展歷史：

年份	版本	重要發展
2018	GPT-1	由 OpenAI 發佈，是第一個基於深度學習的大型語言模型，可以生成自然語言文本。
2019	GPT-2	具有 1.5 億個參數。GPT-2 能夠生成高品質的自然語言文本，並且具有多項語言處理能力，由於擔憂 GPT-2 可能被用於偽造新聞等危險行為，OpenAI 決定不立即公開。
2020	CPT-3	具有 175 億個參數，被認為是當時最先進的語言模型，可以應用於自然語言生成、語言理解、機器翻譯和對話式應用等領域，OpenAI 開始與各種公司合作，將 GPT-3 的技術應用到各種商業產品和服務中。
2021	GPT-3.5	包含更多的語言模型和更高效的計算方法，提高了模型的性能和效率。
2022	ChatGPT API	開放 GPT-3 的應用接口，讓開發者可以更輕鬆地使用 GPT-3 技術開發各種應用和產品。
2023	CPT-4	模型參數量高達 40 兆，能夠更好地學習和理解自然語言的語義和語境，並能讀取圖片、網頁和文件。

# 什麼是 ChatGPT？

---

- ◆ ChatGPT 是建立在 OpenAI GPT 模型的「對話」生成系統，而 OpenAI 是一家成立於 2015 年的人工智慧研究實驗室，致力於推動人工智慧技術的發展和應用。
- ◆ 在發展過程中，OpenAI 吸引了眾多重要的投資者和合作夥伴，其中一位最重要的投資者是伊隆馬斯克 ( Elon Musk )，他同時也是 OpenAI 的共同創辦人。此外，OpenAI 還吸引了其他包括微軟 ( Microsoft )、亞馬遜 ( Amazon ) 和戴森 ( Dyson ) 等知名科技公司的投資和合作，這些投資者的資金支持扮演了至關重要的作用，不僅替 OpenAI 打開了市場，更促進了技術的應用和商業化。
- ◆ 馬斯克 2018 年離開 OpenAI 後，OpenAI 開始設立了以營利為目的的子公司以及相關產品模式，微軟更加碼提供數十億美元的資助，這也是為什麼馬斯克在 2023 年會在推特上表示與過去的目標「Open」有所不同的原因。



# ChatGPT 可以做什麼？

---

- ◆ 由於 ChatGPT 的建立在 GPT 的基礎之上，可以透過大量的無監督學習，對大規模的數據進行預測任務。
- ◆ ChatGPT 的模型能夠理解對話的語境和連貫性，並學習到常見的對話模式和用語。當模型與人類操作者進行對話並進一步微調，就能提高其回答問題的準確性和生成合理的回應。
- ◆ ChatGPT 的出現將對人類未來產生深遠的影響，下方列出 ChatGPT 的常見應用：

- **客戶服務和支援**：提供自動化的客戶服務和解答常見問題，減輕人工客服的負擔。
- **虛擬助手**：擔任個人助理的角色，進行回答問題、行程管理、和提供建議。
- **程式設計**：提供相關程式碼，或針對程式碼進行除錯和改善。
- **教育**：客製化的虛擬老師，回答學習上的問題、提供解釋和指導。
- **創意寫作**：生成故事、詩歌和劇本等創意作品，為作家和創作者提供靈感和協助。
- **翻譯和語言學習**：協助練習外語對話、提供語法和詞彙建議。
- **專業知識**：提供各個領域的專業知識，回答醫學、法律、科學等方面的問題。
- **市場研究**：分析大量的市場數據和消費者反饋，提供更多商業決策參考。
- **心理輔導**：提供心理支持和輔導，幫助人們處理情緒和壓力。
- **健康和醫療**：提供基本的健康資訊、回答健康問題和提供健康建議。
- **新聞和資訊**：提供即時的新聞資訊、天氣預報、股票報價等，幫助人們保持與世界的聯繫。
- **娛樂和休閒**：進行有趣的對話、玩遊戲、解謎等，提供娛樂和休閒的樂趣。
- **智慧家庭和物聯網**：控制智慧家庭或物聯網設備、回答問題和提供指示。
- **社交媒體和聊天應用**：在社交媒體平台或聊天應用中提供對話和互動。

# ChatGPT 的資料從何而來？

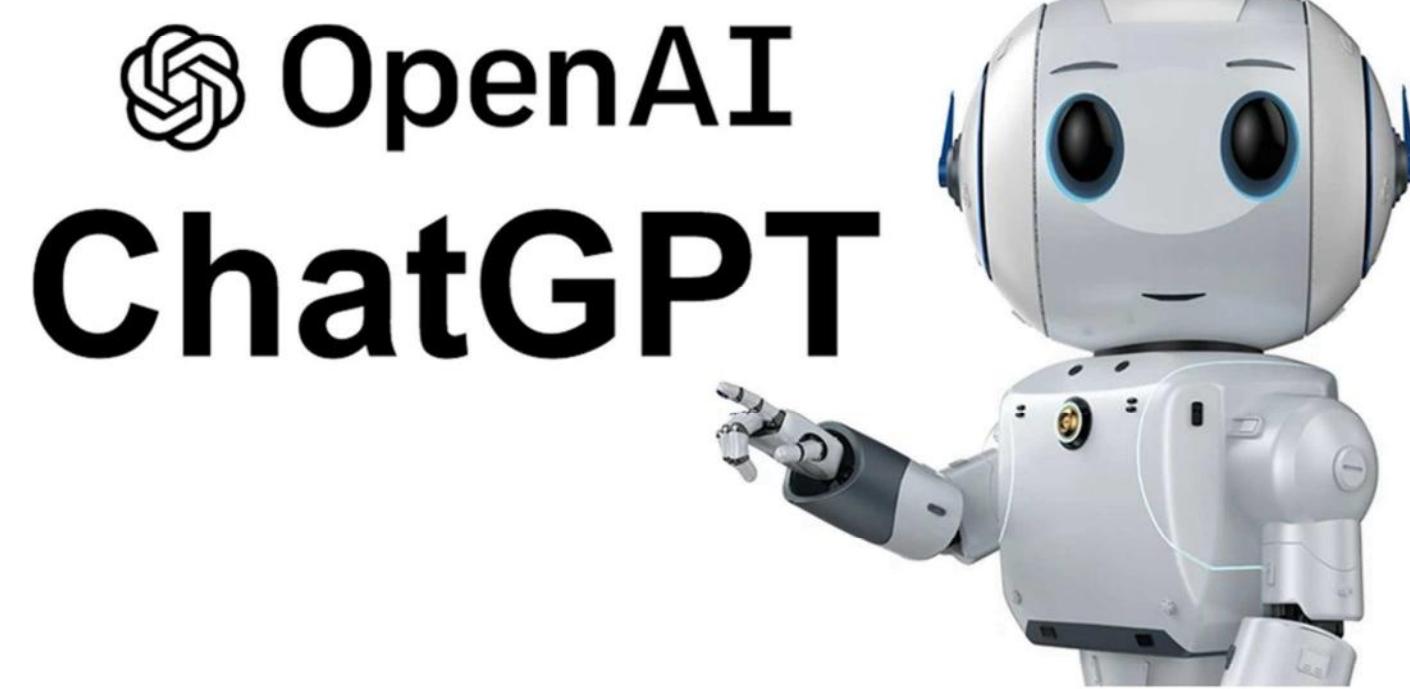
- ◆ ChatGPT 使用了大量的網路文字資料，例如維基百科、網際網路文章、書籍、新聞...等，這些資料被用來訓練模型，使其能夠學習語言的結構、語義和常見的知識。雖然 ChatGPT 在訓練時並沒有人類的介入協助，但在訓練資料的收集和前處理過程中，則會需要人類的協助，包括對資料進行篩選、標記或驗證等工作，然而，模型的訓練本身並不需要人類的指導或監督。
- ◆ ChatGPT 是基於 GPT-3.5 架構的 ChatGPT 模型，使用的資料的年份範圍是截至 2021 年 9 月，所以回答的知識都屬於這個時間範圍內的資訊（GPT-4 可以讀取外部資料），如果超出這個範圍的新聞、事件或科技發展，就無法正確回答。
- ◆ 此外，ChatGPT 沒有自己的主觀判斷、真實理解或記憶，因此對話時只會根據於先前訓練過的模式、資料或紀錄進行相關回應。



# 開始使用 ChatGPT

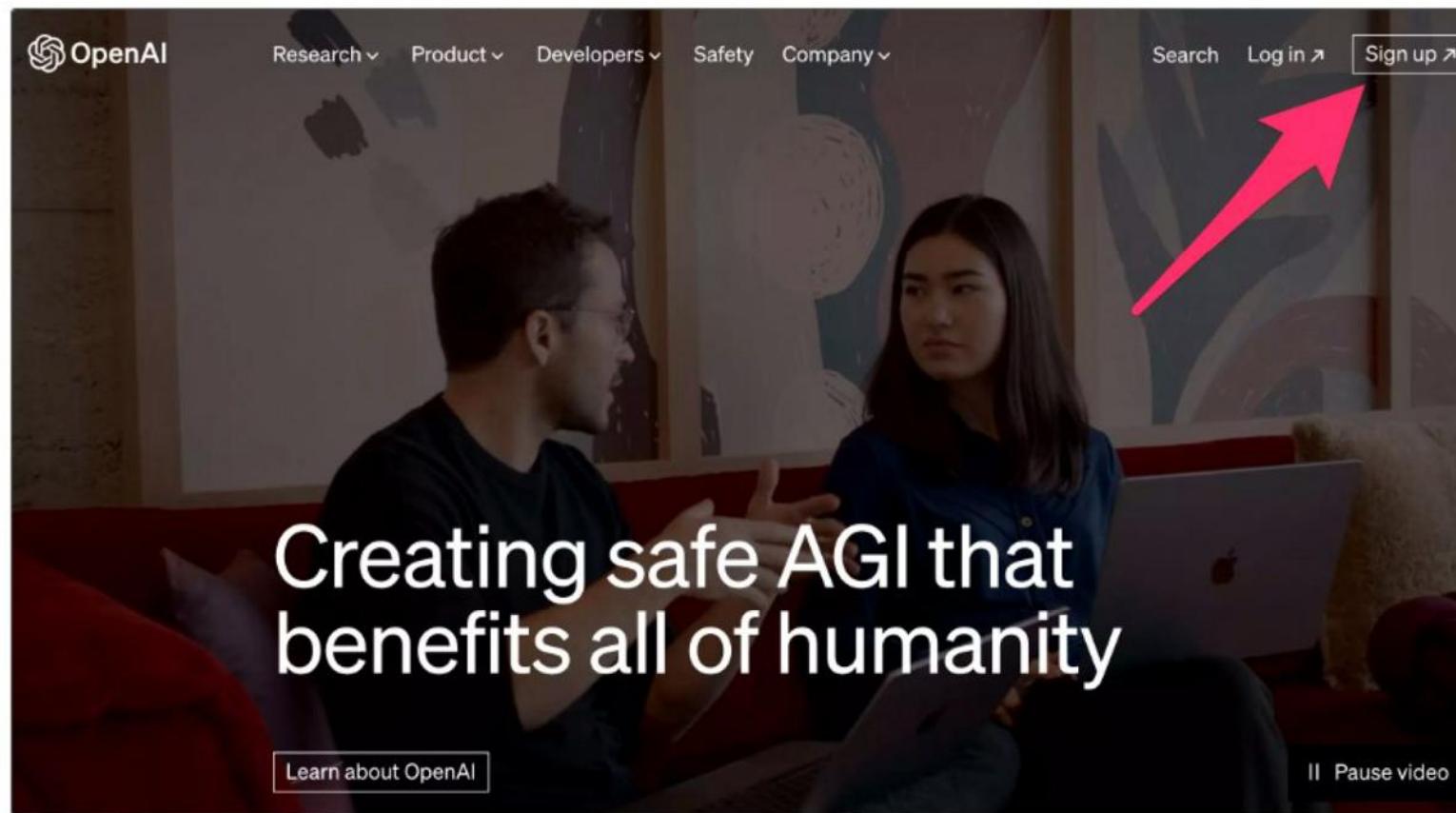
---

- ◆ 接下來我們介紹如何開始使用 ChatGPT，包含註冊 OpenAI、OpenAI 服務使用額度、ChatGPT 對話介面、ChatGPT 歷史紀錄和主題等。



# 註冊 OpenAI

- ◆ 前往 OpenAI 的網站，點擊右上角的 SIGN UP 開始進行註冊。
- ◆ 可以使用 email、Google 帳號、Apple 帳號或微軟帳號的方式註冊。每個帳號都需要綁定手機號碼，一個手機號碼只能對應到一個帳號。



## Create your account

Note that phone verification may be required for signup. Your number will only be used to verify your identity for security purposes.

Email address

Continue

Already have an account? [Log in](#)

---

OR

Continue with Google

Continue with Microsoft Account

Continue with Apple

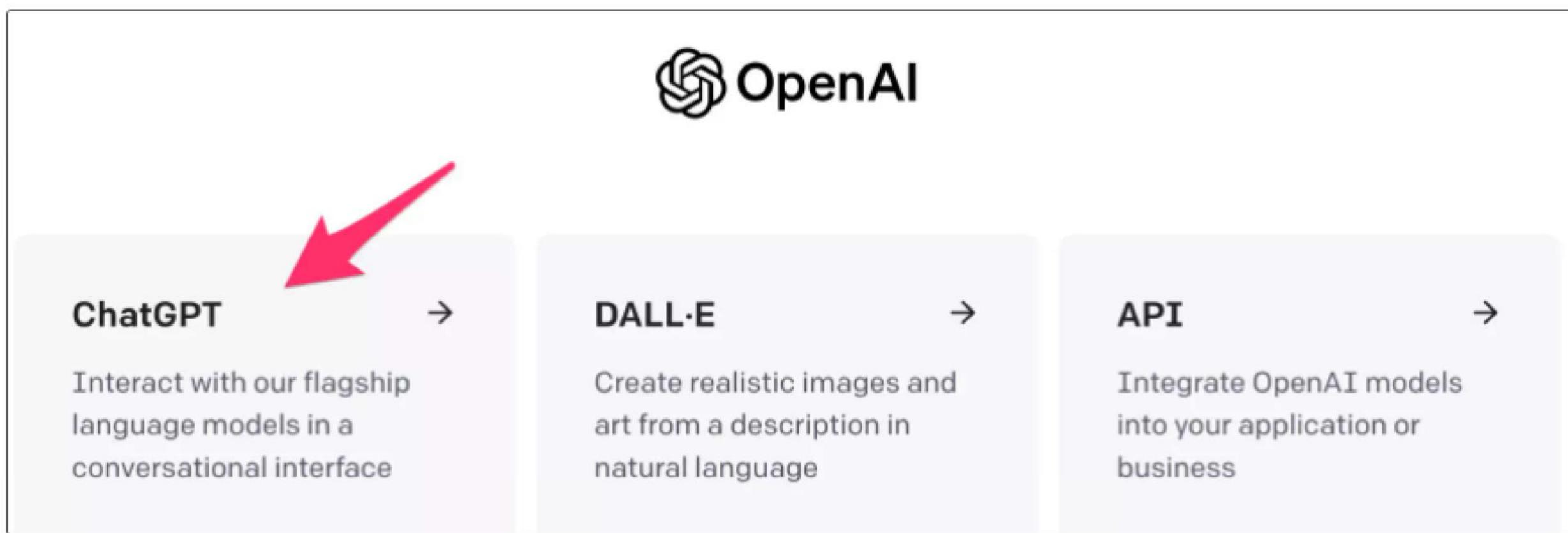
## Verify your phone number

+886

Send code

# 註冊 OpenAI

- ◆ 註冊成功登入後，可以看到 OpenAI 提供三個入口，點擊 ChatGPT 就可以開啟與 ChatGPT 聊天的畫面，而另外二個 DALL·E 負責畫圖，API 負責提供工程師串接。
- ◆ 也可以從這個連結進入：<https://chat.openai.com>



# ChatGPT 對話介面

- ◆ 進入 ChatGPT 聊天畫面後，介面分成左右兩個部分：

• 左側：

- 上方：對話歷史紀錄。
- 下方：帳號資訊、升級方案。

• 右側：

- 上方：聊天對話過程、提示資訊。
- 下方：聊天對話框。



# 開始與 ChatGPT 對話

- ◆ 與 ChatGPT 對話分成兩種，第一種是「新對話」，第二種是「歷史紀錄對話（主題對話）」，以下介紹兩種的差異：

## 新對話

- ◆ 剛開啟 ChatGPT 對話介面時，或點擊右上方「New Chat」，在右下方輸入文字，就會展開全新的對話，同時這份對話也會自動儲存為一個新的「歷史紀錄」，儲存為歷史紀錄後，ChatGPT 會自動為這個歷史紀錄加上主題，點擊後方的「編輯圖示」，就可以自訂這個歷史紀錄的主題。



# 開始與 ChatGPT 對話

## 歷史紀錄對話（主題對話）

- ◆ 如果不想要建立新對話，也可以點擊過去的歷史紀錄（之前建立的主題），就能延續過去的主題進行對話，每個主題只會記得自己的對話內容，彼此之間互不影響，因此可以替每個主題進行各自的設定，例如某些主題負責翻譯（中翻英、英翻中），某些主題具備特定語氣（虛擬上司...等）。



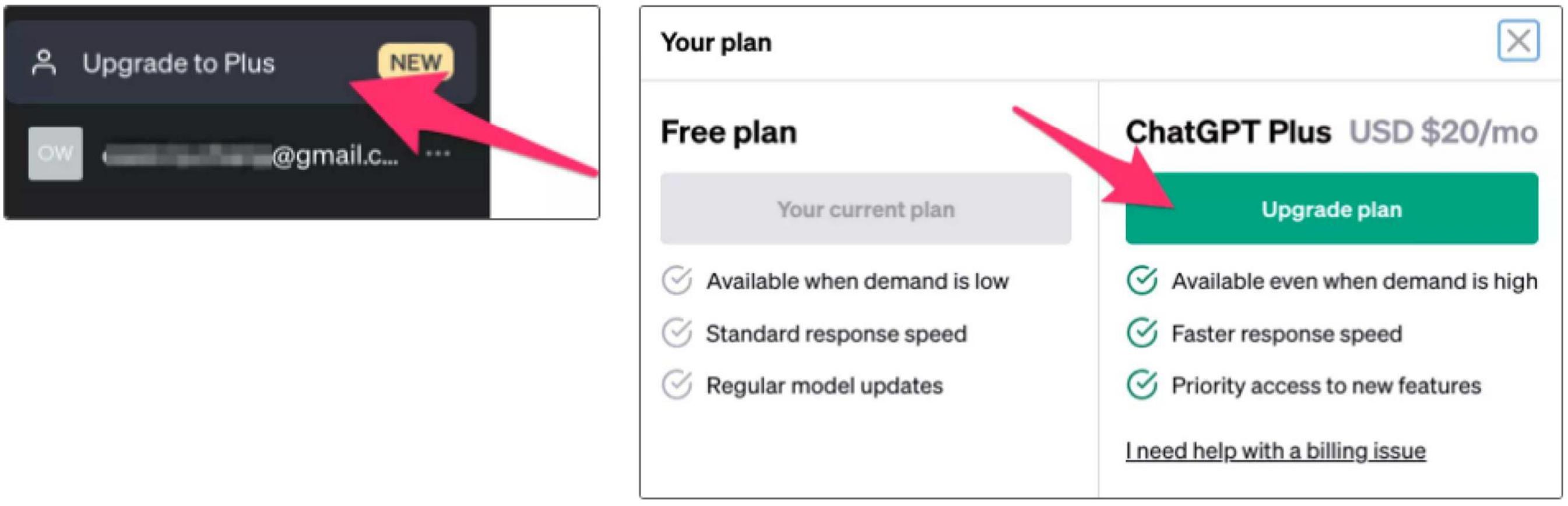
# 使用限制與注意事項

- ◆ ChatGPT 在使用上有下列幾個限制和注意事項：
- 資料庫只到 2021 年底：如果詢問 2022 年以後的資料，可能會發生無法回答或胡亂回答的狀況。
- 資料正確性有待商榷：針對 ChatGPT 所回應的內容，如果不是相關背景，建議再次使用 Google 搜尋，避免錯誤的狀況發生。（有發生使用 ChatGPT 產生的錯誤資料而被告上法院的狀況）
- 對話內容不能太長：因為 ChatGPT 會計算傳輸的 Token 數量，當對話內容過長時，可能會被強制截斷或取消該次對話。
- 對話必須遵守法規：不能夠進行腥羶色等觸犯善良風俗的對話，嚴重可能會被註銷帳號。



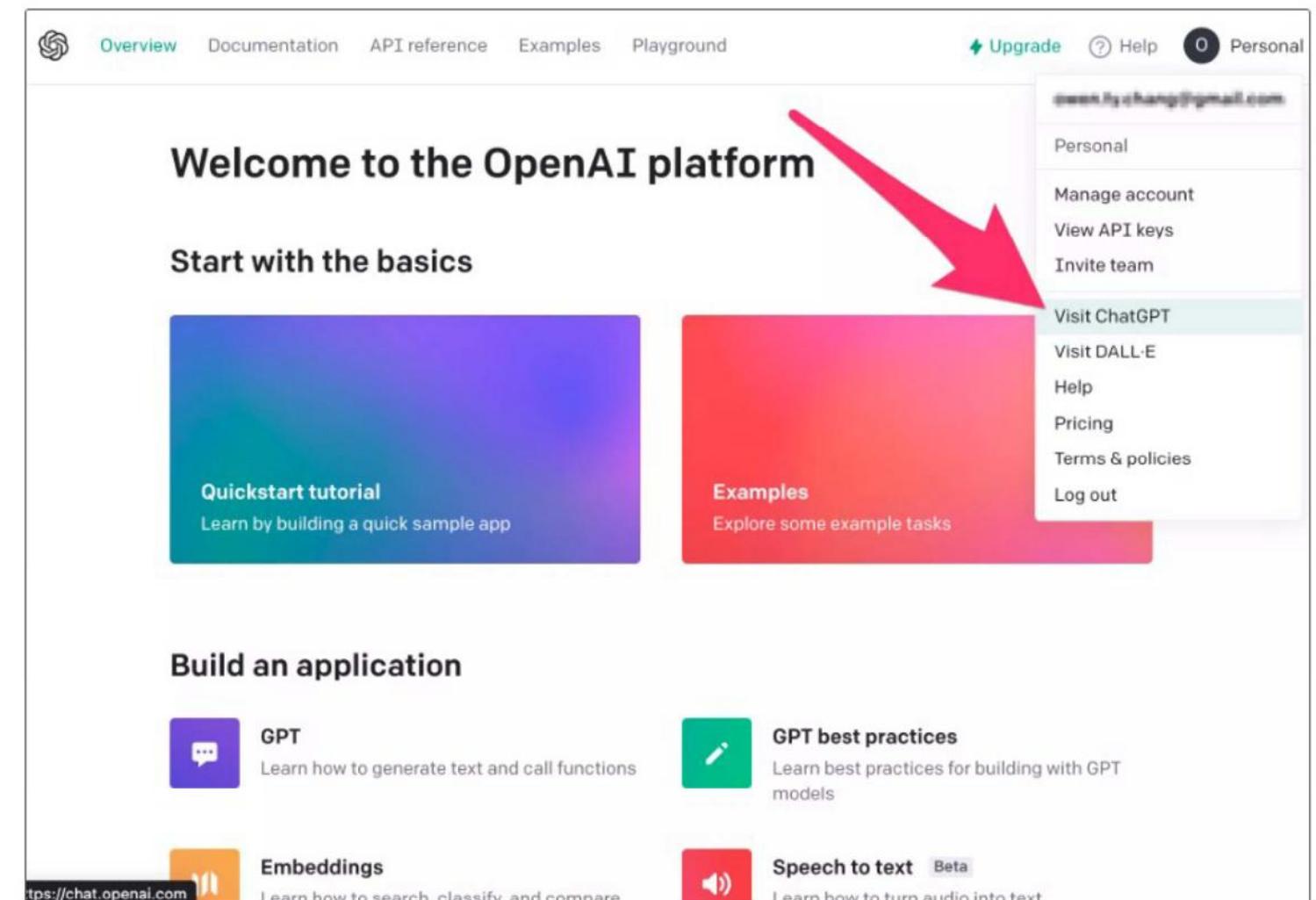
# 升級 ChatGPT Plus

- ◆ 由於免費版的 ChatGPT 有一些使用上的限制（但對輕量型的使用者來說應該都足夠）。如果要讓資料庫更完整，或是擴充對話內容的長度，就必須進行升級到 ChatPlus，點擊左下角個人帳號區域的「Upgrade to Plus」。
- ◆ 選擇「Upgrade plan」，就可以使用每個月花費 20 美金升級到 Plus。



# 其他 OpenAI 功能

- ◆ 除了基本的對話，如果要使用其他 OpenAI 功能，登入後可以前往 OpenAI Platform，Platform 整理了開發者或進階使用者會用到的所有資源，包含教學說明、API 以及更多技術文件，從個人帳號裡的「Visit ChatGPT」，也可以前往 ChatGPT 的對話介面。前往：<https://platform.openai.com>
- ◆ 使用 ChatGPT 其實不困難，不僅免費註冊，還只需要透過聊天就能夠操作，只要掌握一些基本的對話原則，就能夠讓 ChatGPT 化身為得力的小幫手，協助處理各種場合需要。



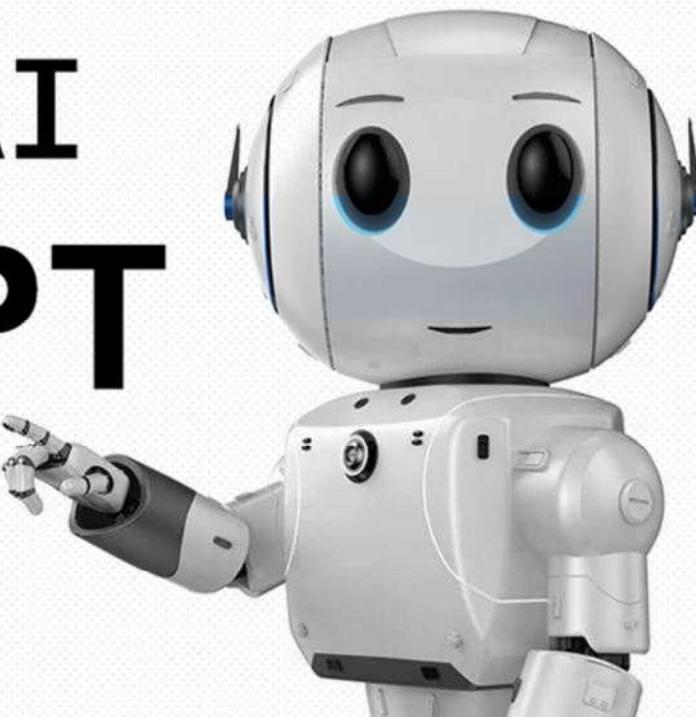
# ChatGPT

## 程式學習輔助與擴充功能應用



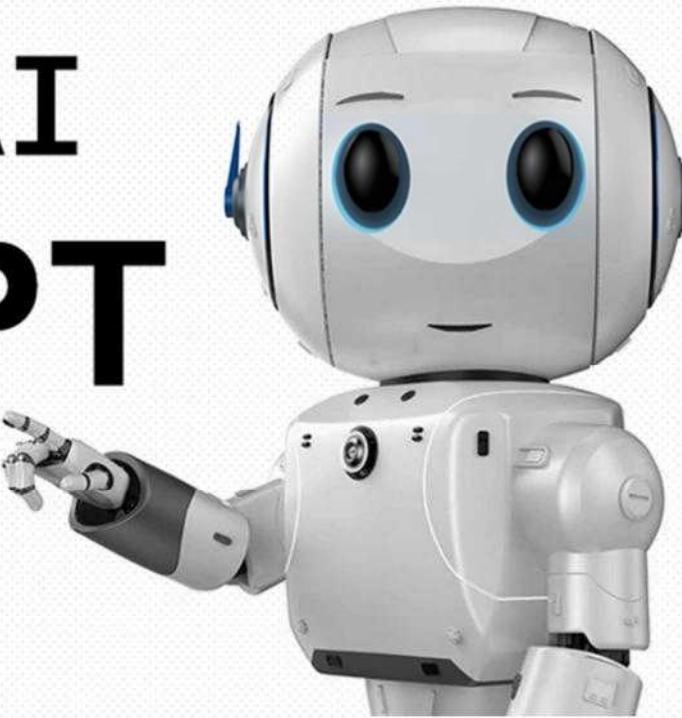
OpenAI

# ChatGPT



# 程式學習輔助

 OpenAI  
**ChatGPT**



# ChatGPT 應用 - 程式學習輔助

---

- ◆ 過去，程式設計師需要花費大量的時間和精力學習特定的程式語言和技術才能在工作中表現出色。  
然而現在有了 ChatGPT，它能夠使用自然語言與你進行對話，並協助快速產生高效優質的程式碼。
- ◆ 此外，ChatGPT 還能協助進行程式碼的重構、註解、除錯、轉換語語、製作說明文件，可以協助的面向非常廣泛。
- ◆ 以下將以問答的方式，讓 ChatGPT 幫助編寫符合需求的 Python 程式，要注意的是每個人生成的答案不一定會相同。



# 程式碼生成

---

- ◆ 當我們想利用 ChatGPT 來生成 Python 程式碼時，我們首先「**輸入提示語**」，例如「請使用 Python 語言寫一個計算平均值的程式」，讓 ChatGPT 理解並產生相應的程式碼。
- ◆ 之後複製程式碼到 Python 編輯器中進行「執行、測試和修正」，直到完成目的為止。



# 程式碼生成

- ◆ 要利用 ChatGPT 生成 Python 程式碼，可以採用下列步驟來完成：

1. 輸入一個清楚明確的提示語，讓 ChatGPT 理解需求，例如：

- 「請用 *Python* 寫一個終極密碼的遊戲」。
- 「寫一個 *Python* 程式，用於輸入計算兩個整數的和」。

- ◆ 下圖為自動生成的「計算兩個整數和」程式碼。(每次產生的 Python 程式碼不一定會相同)

```
python
# 輸入兩個整數
num1 = int(input("請輸入第一個整數："))
num2 = int(input("請輸入第二個整數："))

# 計算兩個整數的和
sum_result = num1 + num2

# 輸出結果
print("兩個整數的和為：" , sum_result)
```

```
python
def main():
    # 輸入第一個整數
    num1 = int(input("請輸入第一個整數："))

    # 輸入第二個整數
    num2 = int(input("請輸入第二個整數："))

    # 計算兩個整數的和
    total = num1 + num2

    # 顯示結果
    print(f"{num1} 和 {num2} 的總和是：{total}")

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

# 程式碼生成

2. 選擇一個 Python 程式碼編輯器，不管是 IDE 編輯器或線上的環境都可以。

這裡我們使用 Google Colab 讓使用者在雲端上編寫和執行程式碼。

- 登入「<https://colab.research.google.com>」，點選「新增筆記本」。
- 點選「Copy code」複製程式碼，貼上至「Google Colab」內，如紅框處所示。

志峰 寫一個 Python 程式，用於輸入計算兩個整數的和

以下是一個簡單的Python程式，可以接受使用者輸入兩個整數並計算它們的和：

```
python
# 輸入兩個整數
num1 = int(input("請輸入第一個整數："))
num2 = int(input("請輸入第二個整數："))

# 計算兩個整數的和
sum_result = num1 + num2

# 輸出結果
print("兩個整數的和為：", sum_result)
```



CO Untitled1.ipynb ☆

檔案 編輯 檢視畫面 插入 執行階段 工具 說明

+ 程式碼 + 文字

```
# 輸入兩個整數
num1 = int(input("請輸入第一個整數："))
num2 = int(input("請輸入第二個整數："))

# 計算兩個整數的和
sum_result = num1 + num2

# 輸出結果
print("兩個整數的和為：", sum_result)
```

# 程式碼生成

3. 由於生成的程式碼可能無法考慮到各種輸入狀況和執行的正確性，因此需要仔細的執行、測試與修正，確保能夠正常運作。

- ◆ 例如當你按下  鈕執行程式時，輸入了非數字的資料，如字母「xxx」，程式就無法計算出結果、並且會顯示類似下圖的錯誤訊息。

4  
秒

```
# 輸入兩個整數
num1 = int(input("請輸入第一個整數: "))
num2 = int(input("請輸入第二個整數: "))

# 計算兩個整數的和
sum_result = num1 + num2

# 輸出結果
print("兩個整數的和為: ", sum_result)
```

請輸入第一個整數: 20  
請輸入第二個整數: 30  
兩個整數的和為: 50

11  
秒

```
# 輸入兩個整數
num1 = int(input("請輸入第一個整數: "))
num2 = int(input("請輸入第二個整數: "))

# 計算兩個整數的和
sum_result = num1 + num2

# 輸出結果
print("兩個整數的和為: ", sum_result)
```

請輸入第一個整數: xxx

-----  
ValueError Traceback (most recent call last)  
<ipython-input-1-7e6434d4a442> in <cell line: 2>()  
 1 # 輸入兩個整數  
----> 2 num1 = int(input("請輸入第一個整數: "))  
 3 num2 = int(input("請輸入第二個整數: "))  
 4  
 5 # 計算兩個整數的和

ValueError: invalid literal for int() with base 10: 'xxx'

搜尋 STACK OVERFLOW

# 程式碼生成

4. 請回到 ChatGPT 告訴它 「**輸入非數字資料時會錯誤**」，讓它重新生成 Python 程式碼並加以解決。
5. ChatGPT 使用 try-except 語句來處理使用者輸入的錯誤。例如當輸入非數字資料時，會提示「錯誤，請輸入整數」。

要注意的是每個人生成的解決方式不一定會相同。

志峰 輸入非數字資料時會錯誤

你可以在此程式中加入錯誤處理，以處理輸入非數字資料的情況。以下是更新後的程式碼：

```
python
while True:
    try:
        # 輸入兩個整數
        num1 = int(input("請輸入第一個整數："))
        num2 = int(input("請輸入第二個整數："))

        # 計算兩個整數的和
        sum_result = num1 + num2

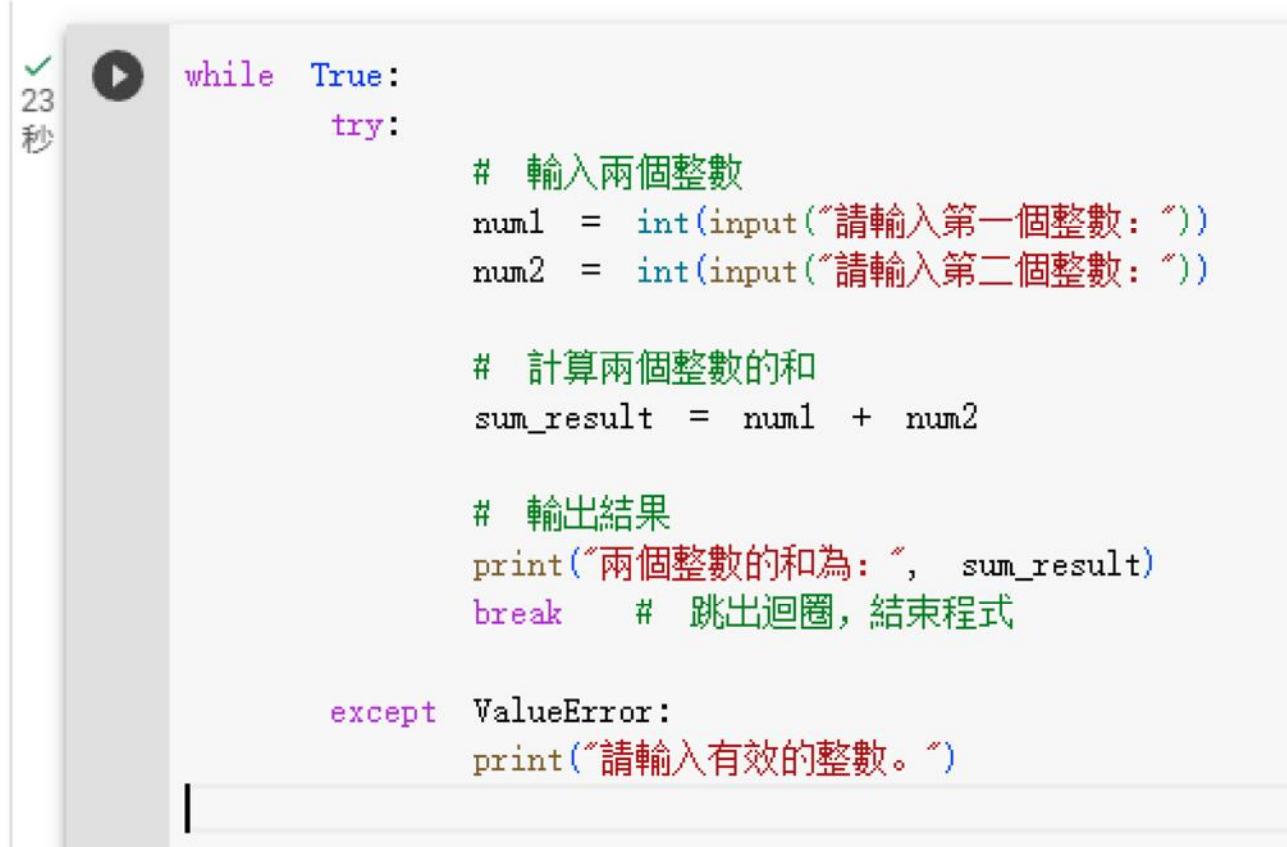
        # 輸出結果
        print("兩個整數的和為：", sum_result)
        break  # 跳出迴圈，結束程式

    except ValueError:
        print("請輸入有效的整數。")
```

這個程式使用了一個無限迴圈(`while True`），當用戶輸入非數字資料時，程式會捕獲 `ValueError` 錯誤並顯示錯誤訊息，然後再次提示用戶輸入。只有在成功輸入兩個整數時，程式才會跳出迴圈並顯示計算結果。

# 程式碼生成

6. 我們再把重新產生的程式碼複製到 Google Colab 上進行執行與測試然後輸入各種可能的資料，例如數字、字母、特殊符號等。如果不會出現任何程式或結果錯誤的情況，表示該問題已經成功解決了。



```
✓ 23 秒
while True:
    try:
        # 輸入兩個整數
        num1 = int(input("請輸入第一個整數: "))
        num2 = int(input("請輸入第二個整數: "))

        # 計算兩個整數的和
        sum_result = num1 + num2

        # 輸出結果
        print("兩個整數的和為: ", sum_result)
        break    # 跳出迴圈，結束程式

    except ValueError:
        print("請輸入有效的整數。")
```

```
▶ 請輸入第一個整數: xxx
請輸入有效的整數。
請輸入第一個整數: !@#2
請輸入有效的整數。
請輸入第一個整數: 123
請輸入第二個整數: 567
兩個整數的和為: 690
```

# 程式碼生成

7. 在修正的過程中，ChatGPT 使用 try… except 語法來幫助我們解決遇到輸入錯誤的問題。

如果我們不懂這個語句，可以透過 ChatGPT 來學習它，如下圖所示。(若不清楚什麼是 try except，提示語可以再使用「什麼是 try except」)。

志峰 什麼是 try except

try 和 except 是 Python 中用於錯誤處理的關鍵字。

try 區塊用於包裹可能會引發錯誤的程式碼塊。當這些程式碼在執行時發生錯誤，Python 會立即退出 try 區塊並轉到 except 區塊。

except 區塊則用於定義在 try 區塊中發生特定類型錯誤時應該執行的程式碼。這樣，你可以在 except 區塊中處理這些錯誤，以避免程式意外終止。

以下是一個簡單的例子：

```
python
try:
    num = int(input("請輸入一個數字："))
    result = 10 / num
    print("結果：", result)

except ZeroDivisionError:
    print("不能除以 0。")

except ValueError:
    print("請輸入有效的數字。")
```

在這個例子中，try 區塊包含了輸入和計算程式碼，並且可能引發 ZeroDivisionError (除以零) 和 ValueError (無法轉換為數字) 這兩種錯誤。如果這些錯誤中的其中一個發生，相應的 except 區塊會被執行，從而處理這些錯誤情況。

# 重構程式

- ◆ 重構程式是專案開發過程中不可或缺的一部分，其目的在於不改變程式原有功能的情況下，對程式碼進行優化和改良，提高其可讀性、可重用性、增加效率、簡化邏輯、減少重複、降低維護成本等等，以下逐一說明。

## 增加可讀性、可重用性：

- ◆ 我們可以使用「重構程式」或「程式碼重構」等類似的提示語，把「計算兩個整數之和」程式碼進行重構。(每個人的重構程式碼不一定會相同)
- ◆ 如果出現「...我需要您提供更具體的問題...」類似的訊息時，表示它不知道你要重構的是哪一段程式碼，請把剛剛的程式碼複製並貼上到 ChatGPT 即可。
- ◆ 重構後，您會發現它把原本的程式碼包裝成函式，並以易於閱讀的方式為函式命名，同時它也將程式模組化、可以單獨執行或被其他程式引用增加可重用性。

志峰 程式碼重構

當然，以下是更加簡化和模組化的程式碼版本：

```
python
def get_integer_input(prompt):
    while True:
        try:
            value = int(input(prompt))
            return value
        except ValueError:
            print("請輸入有效的整數。")

def main():
    num1 = get_integer_input("請輸入第一個整數：")
    num2 = get_integer_input("請輸入第二個整數：")
    print("兩個整數的和為：", num1 + num2)

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

這個版本進一步簡化了程式碼，將計算兩個整數和的部分直接放在 `print` 函式中，省去了 `calculate\_sum` 函式。這是因為計算和的操作非常簡單，所以可以直接在 `print` 中進行。

這個程式碼版本的結構非常簡單，主要邏輯在 `main` 函式中，而 `get\_integer\_input` 函式處理輸入。你可以保存這個程式碼並執行它，這將輕鬆地輸入兩個整數並顯示它們的和。

# 重構程式

## 增加效能：

- ◆ 透過重構程式的技巧、也可以讓我們從中學習到更有效率的程式寫法。
- ◆ 我們使用「寫一個 Python 程式輸入並計算從 1 累加至 n 的和」提示文字讓 ChatGPT 產生程式碼每個人的重構程式碼不一定會相同)。
- ◆ 我們發現原程式中每次相加的次數會隨著數字 n 的增加而增加，造成效率不佳。

志峰 寫一個 Python 程式輸入並計算從 1 累加至 n 的和

當然，以下是一個 Python 程式，用於輸入一個數字 n，然後計算並顯示從 1 累加至 n 的和：

```
python
def calculate_sum(n):
    return sum(range(1, n + 1))

def main():
    try:
        n = int(input("請輸入一個數字 n : "))
        if n < 1:
            print("請輸入大於等於 1 的數字。")
        else:
            result = calculate_sum(n)
            print("從 1 累加至", n, "的和為：", result)
    except ValueError:
        print("請輸入有效的數字。")

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

這個程式會提示你輸入一個數字 n，然後使用 `calculate\_sum` 函式計算從 1 到 n 的和，最後顯示結果。如果輸入的不是數字，程式會捕獲 `ValueError` 錯誤。如果輸入的數字小於 1，程式會顯示錯誤訊息。

你可以將這個程式保存成 `.py` 檔案，然後執行它，按照提示輸入一個數字 n，程式會計算並顯示從 1 累加至 n 的和。

# 重構程式

## 增加效能：

- ◆ 將這個程式碼給 ChatGPT 「重構程式」後，因為採用高斯求和公式，所以計算效率更高，同時也讓程式碼較為簡潔。

志峰 請用高斯求和公式重構程式

當然，以下是使用高斯求和公式來重構程式碼的版本：

```
python
def calculate_sum(n):
    return (n * (n + 1)) // 2

def main():
    try:
        n = int(input("請輸入一個數字 n : "))
        if n < 1:
            print("請輸入大於等於 1 的數字。")
        else:
            result = calculate_sum(n)
            print(f"從 1 累加至 {n} 的和為: {result}")
    except ValueError:
        print("請輸入有效的數字。")

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

在這個版本中，我們只是在 `calculate\_sum` 函式中使用了高斯求和公式，而其餘部分保持不變。這個程式碼版本使用了較高效的方法來計算從 1 累加至 n 的和，特別適用於大數字。

你可以保存這個程式碼並執行它，輸入一個數字 n，程式會計算並顯示從 1 累加至 n 的和，並使用高斯求和公式來進行計算。

# 重構程式

---

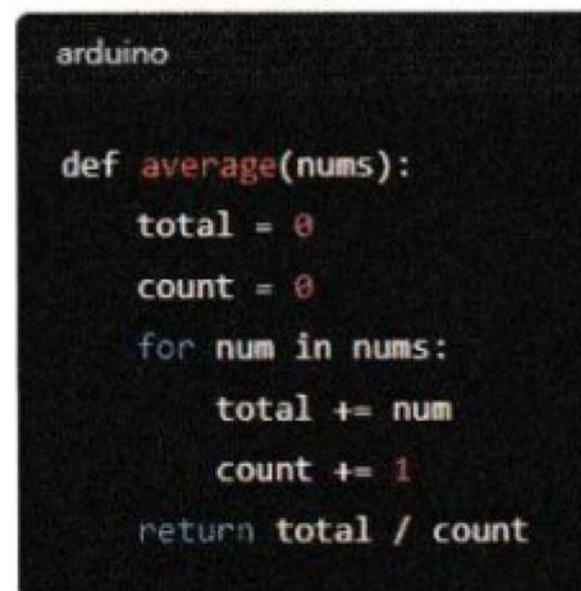
## 簡化邏輯、減少重複：

- ◆ 如果一個程式碼的邏輯很複雜，除了會造成難以閱讀與理解外，也容易出現錯誤和漏洞。

為了提高程式碼的可讀性及維護的效率，我們需要優化及改良程式的邏輯，使其更加簡單易懂。

以下為「計算平均值」的程式碼經過重構前後的差別。

- ◆ 重構前，我們自行輸入程式碼，使用基本的迴圈和變數來計算傳入 `nums` 串列的平均值，透過每一次的「累加」與「加 1」的操作來計算總和與元素的個數，最後將總和除以個數得到平均值。



The screenshot shows a code editor window with a dark theme. The title bar says "arduino". The code is written in Python and defines a function named `average` that takes a list of numbers as input. Inside the function, two variables `total` and `count` are initialized to 0. A `for` loop iterates over each number in the `nums` list, adding it to `total` and incrementing `count` by 1. Finally, the function returns the total divided by count.

```
arduino

def average(nums):
    total = 0
    count = 0
    for num in nums:
        total += num
        count += 1
    return total / count
```

# 重構程式

---

- ◆ 輸入「重構程式」的提示語讓 ChatGPT 重構後，會發現使用了內建函數 `sum()` 和 `len()` 來計算串列的總和跟個數，然後相除得到平均值，這樣就能夠簡化邏輯，減少重複的程式碼，更加易於閱讀和理解。

```
python

def average(nums):
    return sum(nums) / len(nums)
```

# 重構程式

- ◆ 當我們希望 ChatGPT 幫助進行程式碼重構時，應該先測試程式，確保沒有問題後，再使用「重構程式」或「程式碼重構」之類的提示語，讓 ChatGPT 根據程式碼進行自動化的重構。
- ◆ 另外，也可以加上「提高效率」、「提高可讀性」等語句，

讓 ChatGPT 根據特定的需求進行重構及優化。例如以下敘述：

- 重構程式，簡化邏輯。
- 重構程式，增加效能。
- 重構程式，提高可讀性。

志峰 重構程式，簡化邏輯。

當然，以下是更簡化邏輯的程式碼版本：

python

```
def calculate_sum(n):
    return n * (n + 1) // 2

def main():
    try:
        n = int(input("請輸入一個數字 n："))
        result = calculate_sum(n)
        print(f"從 1 累加至 {n} 的和為：{result}")
    except ValueError:
        print("請輸入有效的數字。")

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

Copy code

在這個版本中，我們刪除了不必要的條件檢查，因為高斯求和公式對於  $n$  大於等於 1 的情況是有效的，所以不需要再額外檢查。這樣可以使程式碼更簡潔。

你可以保存這個程式碼並執行它，輸入一個數字  $n$ ，程式會計算並顯示從 1 累加至  $n$  的和，並使用高斯求和公式來進行計算。

# 程式註解

- ◆ 註解在電腦程式中用於解釋程式碼以便日後自己回憶設計過程，或讓其他開發者能夠更容易理解和維護。

- ◆ 如果需要為某段或全部程式碼加上註解，可以使用類似「**幫忙加註解**」

「**加上詳細註解**」的提示，讓 ChatGPT 產生相應的註解，底下是常見

的作法：

1. Python 的程式註解使用「#」符號來表示，符號後面的文字都會被視為註解。我們讓 ChatGPT 先生成一個終極密碼遊戲的程式，如下圖所示。(每個人產生的程式碼不一定會相同)

志峰 終極密碼遊戲

當然，以下是一個簡單的終極密碼遊戲的 Python 程式：

```
python
import random

def main():
    lower_bound = 1
    upper_bound = 100
    secret_number = random.randint(lower_bound, upper_bound)
    attempts = 0

    print("歡迎來到終極密碼遊戲！")
    print(f"我想了一個 {lower_bound} 到 {upper_bound} 之間的數字，試著猜猜看。")

    while True:
        guess = int(input("請輸入你的猜測："))
        attempts += 1

        if guess < secret_number:
            print("太小了，再試試。")
        elif guess > secret_number:
            print("太大了，再試試。")
        else:
            print(f"恭喜你，你猜對了！用了 {attempts} 次。")
            break

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

# 程式註解

- ◆ 使用「加上詳細註解」的提示語，讓 ChatGPT 產生完整的註解，幫助日後的開發與維護。

終極密碼遊戲

```
志峰
```

當然，以下是一個簡單的終極密碼遊戲的 Python 程式：

```
python
Copy code

import random

def main():
    lower_bound = 1
    upper_bound = 100
    secret_number = random.randint(lower_bound, upper_bound)
    attempts = 0

    print("歡迎來到終極密碼遊戲！")
    print(f"我想了一個 {lower_bound} 到 {upper_bound} 之間的數字，試著猜猜看。")

    while True:
        guess = int(input("請輸入你的猜測："))
        attempts += 1

        if guess < secret_number:
            print("太小了，再試試。")
        elif guess > secret_number:
            print("太大了，再試試。")
        else:
            print(f"恭喜你，你猜對了！用了 {attempts} 次。")
            break

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

加上詳細註解

```
志峰
```

當然，以下是加上詳細註解的終極密碼遊戲程式：

```
python
Copy code

import random

def main():
    lower_bound = 1      # 密碼範圍的下限
    upper_bound = 100    # 密碼範圍的上限
    secret_number = random.randint(lower_bound, upper_bound)  # 隨機生成密碼
    attempts = 0         # 紀錄猜測的次數

    print("歡迎來到終極密碼遊戲！")
    print(f"我想了一個 {lower_bound} 到 {upper_bound} 之間的數字，試著猜猜看。")

    while True:
        guess = int(input("請輸入你的猜測："))  # 玩家輸入猜測
        attempts += 1   # 猜測次數加一

        if guess < secret_number:
            print("太小了，再試試。")
        elif guess > secret_number:
            print("太大了，再試試。")
        else:
            print(f"恭喜你，你猜對了！用了 {attempts} 次。")
            break

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

# 程式註解

2. 對於一般的程式註解，我們可以使用「**協助加上註解**」的功能；如果是針對函數的註解，建議使用「**使用 docstrings 註解**」的功能，以生成更完整的程式說明。

◆ 使用 **docstrings** 註解是一種特殊的字串，通常以三個單引號 ("""或三個雙引號 (""")括住註解文字的前後。

這種註解用於描述函數的使用方式、參數、回傳值等資訊，並且會放在程式碼的開頭、結束處或特定函數之前。

◆ 透過 **docstrings** 開發人員可以更容易地理解程式的功能和用途。

志峰 兩數相加的函數



當然，以下是一個簡單的 Python 函數，用於將兩個數字相加：

```
python
def add_numbers(a, b):
    return a + b

num1 = float(input("請輸入第一個數字："))
num2 = float(input("請輸入第二個數字："))

result = add_numbers(num1, num2)
print("兩數相加的結果為：" , result)
```

這個程式定義了一個 `add\_numbers` 函數，該函數接受兩個參數 `a` 和 `b`，然後將它們相加並返回結果。你可以根據需要輸入兩個數字，然後使用這個函數計算它們的和。

# 程式註解

- ◆ 以下是一個兩數相加的函數及其註解。使用「**使用 docstrings 註解**」的提示文字、讓 ChatGPT 產生完整的說明，能夠清楚地知道這個函數接受哪些參數、返回什麼結果、進而更容易理解這個函數的作用。

志峰 兩數相加的函數

當然，以下是一個簡單的 Python 函數，用於將兩個數字相加：

```
python
def add_numbers(a, b):
    return a + b

num1 = float(input("請輸入第一個數字："))
num2 = float(input("請輸入第二個數字："))

result = add_numbers(num1, num2)
print("兩數相加的結果為：", result)
```

這個程式定義了一個 `add\_numbers` 函數，該函數接受兩個參數 `a` 和 `b`，然後將它們相加並返回結果。你可以根據需要輸入兩個數字，然後使用這個函數計算它們的和。

志峰 使用 docstrings 註解

當然，這是使用 docstrings 註解的兩數相加函數：

```
python
def add_numbers(a, b):
    """
    將兩個數字相加並返回結果。
    """

    Parameters:
        a (float): 第一個數字。
        b (float): 第二個數字。

    Returns:
        float: 兩個數字的和。
    """

    return a + b
```

這個函數定義了一個 docstring，它提供了對函數用途、參數和返回值的說明。你可以將這個函數儲存到 `.py` 檔案中，然後在你的程式中使用它。當你查看這個函數的文檔時，這些註解會幫助你理解函數的功能和使用方式。

志峰 加入一般註解

當然，這是加入一般註解和使用 docstrings 註解的兩數相加函數：

```
python
def add_numbers(a, b):
    """
    將兩個數字相加並返回結果。
    """

    Parameters:
        a (float): 第一個數字。
        b (float): 第二個數字。

    Returns:
        float: 兩個數字的和。
    """

    result = a + b # 將兩個數字相加
    return result

# 主程式
num1 = 5
num2 = 7
sum_result = add_numbers(num1, num2) # 呼叫 add_numbers 函數計算和
print(f"兩數 {num1} 與 {num2} 的和為：{sum_result}")
```

在這個程式中，`add\_numbers` 函數的 docstring 提供了對函數用途、參數和返回值的說明。此外，在函數內部的一般註解則說明了相加的操作。在主程式部分，我們使用這個函數來計算兩個數字的和，並將結果顯示出來。

這些註解可以幫助你和其他人更好地理解程式的運作和邏輯。你可以根據需要添加更多註解來解釋程式的各個部分。

# 程式碼除錯(Debug)

- ◆ 當寫好的程式測試執行時，可能會出現各種問題，例如程式執行錯誤、程式當掉卡住、輸出不是我們要的結果等等。  
此時我們需要找出其中的問題並修復它，這個過程就叫做「Debug」，中文翻譯為「除錯」或「偵錯」。
- ◆ 一般而言程式的錯誤分為「語法錯誤」與「邏輯錯誤」兩種。



# 程式碼除錯(Debug)

1. 語法錯誤 (Syntax Error)：是指程式碼有錯無法被直譯器或編譯器正確解析，通常是因為指令打錯、資料型態不對、缺少括號、冒號或引號等。

◆ 這類錯誤只要將程式碼提供給 ChatGPT，它會自動找出錯誤的地方並進行修復，如下圖所示，根據原始程式提出建議及修正後的程式碼。

志峰

```
sum=0
for i in range(1,51)
    sum+=i
print("1加到50的結果是:", sum)
```

你提供的程式碼有一些小錯誤，我幫你修正一下並加上註解：

python

```
sum_result = 0 # 初始化總和變數為0

# 使用迴圈計算1加到50的總和
for i in range(1, 51): # 注意冒號的使用
    sum_result += i # 將當前數字加到總和中

# 顯示計算結果
print("1加到50的結果是:", sum_result)
```

這個程式碼使用了迴圈計算從1加到50的總和，並且在計算完畢後顯示了結果。我加上了註解以說明程式碼的每一部分。你可以將這個程式碼保存成 `\*.py` 檔案，然後執行它，你將會看到計算結果。

# 程式碼除錯(Debug)

2. **邏輯錯誤 (Logical Error)**：是指程式可以正常運作，但是執行結果與預期不合，通常是因為程式邏輯有誤或者演算法有問題等。

- ◆ 這類程式錯誤通常需要透過執行程式，並觀察結果或錯誤訊息來找出問題所在。
- ◆ 當我們無法找到問題點時，可以請 ChatGPT 幫忙分析錯誤原因，並提供解決方式。
- ◆ 底下示範將「執行狀況」與「錯誤訊息」提供給 ChatGPT，協助除錯。



當然，這是一個簡單的 Python 程式，用於輸入並計算兩個整數的和：

```
python
def main():
    num1 = int(input("請輸入第一個整數："))
    num2 = int(input("請輸入第二個整數："))

    sum_result = num1 + num2

    print("兩個整數的和為：" , sum_result)

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

# 程式碼除錯(Debug)

- ◆ 提供「執行狀況」：例如之前在計算兩個整數的和時，我們可以向 ChatGPT 提供「輸入非數字資料會錯誤」這個問題，請它分析錯誤原因、並提供修正建議。

當然，這是一個簡單的 Python 程式，用於輸入並計算兩個整數的和：

```
python
def main():
    num1 = int(input("請輸入第一個整數："))
    num2 = int(input("請輸入第二個整數："))

    sum_result = num1 + num2

    print("兩個整數的和為：", sum_result)

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

志峰 輸入非數字資料會錯誤

當然，在程式中加入錯誤處理機制可以避免輸入非數字資料時的錯誤。以下是帶有錯誤處理的程式碼版本：

```
python
def main():
    try:
        num1 = int(input("請輸入第一個整數："))
        num2 = int(input("請輸入第二個整數："))

        sum_result = num1 + num2

        print("兩個整數的和為：", sum_result)
    except ValueError:
        print("請輸入有效的整數。")

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

在這個版本的程式碼中，我們使用了 `try` 和 `except` 來捕獲 `ValueError`，這是由於輸入非數字資料導致的錯誤。如果用戶輸入了非數字資料，程式將顯示錯誤訊息，而不會崩潰。

你可以保存這個程式碼為 `.py` 檔案，然後執行它。如果輸入非數字資料，程式將提示錯誤訊息，否則將計算並顯示兩個整數的和。

# 程式碼除錯(Debug)

- ◆ 提供「錯誤訊息」：或者將執行時的錯誤訊息提供給 ChatGPT，讓它分析錯誤原因，並提供修正建議。

志峰

請輸入第一個整數：xxx

```
-----  
ValueError          Traceback (most recent call last)  
<ipython-input-4-7e6434d4a442> in <cell line: 2>()  
      1 # 輸入兩個整數  
----> 2 num1 = int(input("請輸入第一個整數 : "))  
      3 num2 = int(input("請輸入第二個整數 : "))  
      4  
      5 # 計算兩個整數的和
```

ValueError: invalid literal for int() with base 10: 'xxx'



非常抱歉，你遇到了 `ValueError`，這是因為你輸入了非數字的資料。這是預期行為，因為 `int()` 函數只能將合法的數字轉換成整數。在你輸入非數字資料時，它無法進行轉換，因此引發了 `ValueError`。

如果你希望在輸入非數字資料時得到更友好的提示，你可以使用前面所提供的程式碼中的 `try` 和 `except` 機制來處理這種情況，並向用戶顯示錯誤訊息。這樣用戶會知道輸入的資料無效並得到指示。

如果你有其他問題，歡迎隨時提問！

# 生成說明文件

- ◆ 程式的說明文件類似於產品說明書的形式，其主要目的是為了協助其他開發者或使用者更好地理解系統。
- ◆ ChatGPT 提供了 Markdown 標記語言，可以快速編寫出具有結構化的文件，而這種語言也被廣泛應用於編寫說明文件、發表訊息、論文等多種場合。
- ◆ 為了生成具有良好可讀性的說明文件，可以使用類似「[請用 Markdown 生成說明文件](#)」的提示語，讓 ChatGPT 自動產生說明文件。
- ◆ Markdown 是一種輕量級的標記語言、可以幫助您清晰明瞭地呈現程式碼的功能、使用方式及注意事項。



The screenshot shows a Markdown editor window titled "readme.md". The code area contains the following content:

```
1 # Markdown
2 You can emphasize text with **bold**, *italic*, or ***both***.
3
4 ## Subheading
5
6 You can also ~~strike-through~~ words and denote inline `code`.
```

To the right of the editor, there are two sections of generated text:

## Markdown

You can emphasize text with **bold**, *italic*, or ***both***.

## Subheading

You can also ~~strike-through~~ words and denote inline `code`.

# 生成說明文件

- ◆ 我們使用前面「**輸入並計算兩個整數和**」的例子讓 ChatGPT 生成說明文件。

- ◆ 當您完成程式 Debug 把「語法或邏輯」的錯誤都解決後，為了讓日後的開發人員更容易接手，建議使用 Markdown 製作程式碼的說明文件。

志峰 輸入並計算兩個整數和

當然，這是一個輸入並計算兩個整數和的 Python 程式，同時處理非數字輸入的情況：

```
python
Copy code

def add_numbers(num1, num2):
    return num1 + num2

def main():
    while True:
        try:
            num1 = int(input("請輸入第一個整數："))
            num2 = int(input("請輸入第二個整數："))

            result = add_numbers(num1, num2)

            print("兩個整數的和為：", result)
            break # 成功輸入並計算，退出迴圈
        except ValueError:
            print("請輸入有效的整數。")

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

志峰 請用 Markdown 生成說明文件

當然，以下是使用 Markdown 格式生成的說明文件範例：

markdown

# 兩數相加的程式

這個程式是用於輸入兩個整數，然後計算它們的和。

## 使用說明

1. 執行程式後，它會要求你輸入兩個整數。
2. 輸入兩個整數後，程式會計算它們的和並顯示結果。

如果輸入的是非數字資料，程式會顯示錯誤訊息並要求重新輸入。

## 程式碼示例

```
```python
def add_numbers(num1, num2):
    """
    將兩個整數相加並返回結果。
    """

    Parameters:
        num1 (int): 第一個整數。
        num2 (int): 第二個整數。

    Returns:
        int: 兩個整數的和。
    """
    return num1 + num2

def main():
    while True:
        try:
            num1 = int(input("請輸入第一個整數："))
            num2 = int(input("請輸入第二個整數："))

            result = add_numbers(num1, num2)

            print("兩個整數的和為：", result)
            break # 成功輸入並計算，退出迴圈
        except ValueError:
            print("請輸入有效的整數。")

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

```

# 轉換程式語言

- ◆ 每個程式語言都有其用途與特性，我們可以使用「**轉換成 [程式語言]**」的提示、讓 ChatGPT 達到不同程式語言的轉換。
- ◆ 以下是一個計算兩個整數和的 Python 程式碼範例，分別轉換成 Java、C# 的程式碼。

The image displays three side-by-side code snippets generated by ChatGPT, each showing a different programming language's version of a simple addition program.

**Left Panel (Python):**

志峰 寫一個 Python 程式，用於輸入計算兩個整數的和

當然，以下是一個簡單的 Python 程式，用於輸入兩個整數並計算它們的和：

```
python
def main():
    try:
        num1 = int(input("請輸入第一個整數："))
        num2 = int(input("請輸入第二個整數："))

        sum_result = num1 + num2

        print("兩個整數的和為：" , sum_result)
    except ValueError:
        print("請輸入有效的整數。")

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

**Middle Panel (Java):**

志峰 轉換成 Java

當然，以下是一個簡單的 Java 程式，用於輸入兩個整數並計算它們的和：

```
java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("請輸入第一個整數：");
            int num1 = scanner.nextInt();

            System.out.print("請輸入第二個整數：");
            int num2 = scanner.nextInt();

            int sumResult = num1 + num2;

            System.out.println("兩個整數的和為：" + sumResult);
        } catch (java.util.InputMismatchException e) {
            System.out.println("請輸入有效的整數。");
        }
    }
}
```

**Right Panel (C#):**

志峰 轉換成 C#

當然，以下是一個簡單的 C# 程式，用於輸入兩個整數並計算它們的和：

```
csharp
using System;

namespace SumCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("請輸入第一個整數：");
                int num1 = int.Parse(Console.ReadLine());

                Console.WriteLine("請輸入第二個整數：");
                int num2 = int.Parse(Console.ReadLine());

                int sumResult = num1 + num2;

                Console.WriteLine("兩個整數的和為：" + sumResult);
            }
            catch (FormatException)
            {
                Console.WriteLine("請輸入有效的整數。");
            }
        }
    }
}
```

# Q & A