Google Developer Group
Editable Location

與AI協同工作

將所有平凡無奇的圖片轉成梗圖阿哈哈

Build with AI

讀書會主持人: 顏榕嶙(Bernie)、李准恩(CipCap)
日期: **03/18 燕巢場**

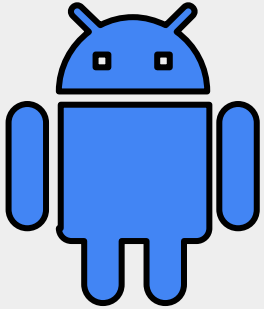
Google Developer Group
Editable Location

這堂課主要會涉及技術的講解，以及帶各位去進行簡單的實作。

API技術由Google提供，此演示只是其潛在功能的一部分。

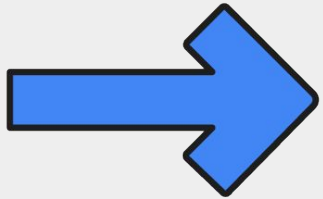
Build with AI

第一章

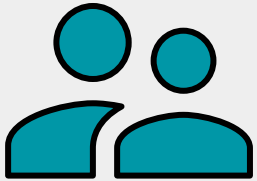


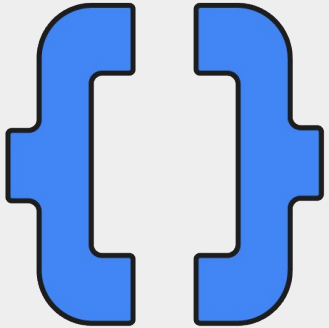
多樣性的AI生態

很多人都忽略了AI是可擴展式工具這一點



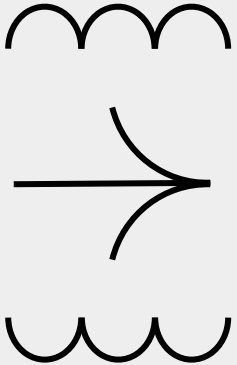
AI 不 AI?





我們的日常會碰到各式各樣的 AI工具 – 這得歸功於技術變遷的速度。

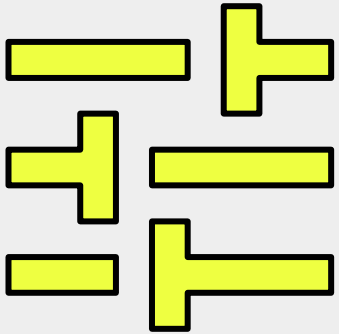
所有領域的企業、研究人員都在試著將這些新穎的技術給融入各式各樣的產品當中，為的就是提升使用者的生活品質。



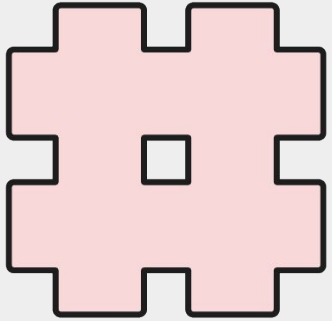
直至兩年前，也許人們都還沒辦法想像一個沒了大型語言模型就生活不下去的社會，然而這已經成為了我們如今的常態。

當然，使用AI本身是個學問，我們學習著各式各樣的新型智慧 產品，但有沒有人曾經想過去成為創造 AI技術的一員呢？

從一到一+



生成式AI是非常強大的工具，很多基礎技術加入了生成式 AI的框架之後，躍身成為現代社會的頂流之一，簡單舉幾個例子：



客服 -> AI 服務系統：使用聊天機器人來協助使用者了解資訊。

數據分析 -> AI 報表生成與統計：省去了很多繁雜的計算，使用可信的預測演算法以及檢索加強生成來協助進行分析。

內容創作 -> AI 輔助數位創作：讓許多藝術家、創作者能 夠用更加迅速的工作流去完成它們的工作。

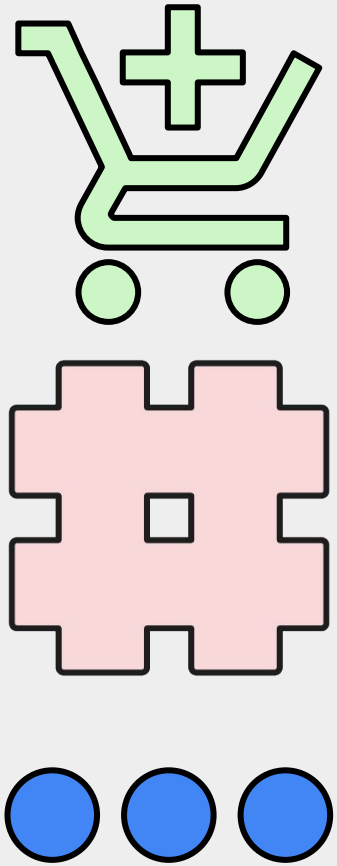
一個創造的契機

身為開發者一員的我們，應該要比起普羅大眾更曉得當前科技趨勢，並善加利用才是屬於自己的優勢。

但創新畢竟是極為困難且複雜的過程，況且我們也並沒有太多的資金或者技術來為之支撐，單單只是一介工程師的我們又怎麼要去鬥爭呢 ？

很簡單，我們只需要知道一個道理：創新的本質並不是去創造一個完全新穎的東西，而是將已知的知識以不同的角度撮合在一起。

仍然懵懂？那麼我們先熟知一下手邊的工具吧。



Google 所提供的 GCP 就是一個非常容易上手的開發平台。

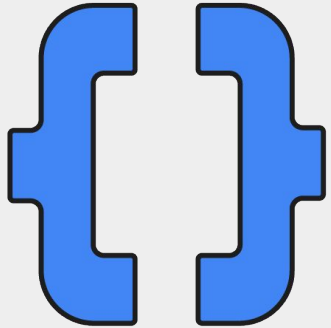
各式API都可以在這裡找到並啟用。

“

很慶幸的，今天各位
都有機會來實際使用
這些工具。

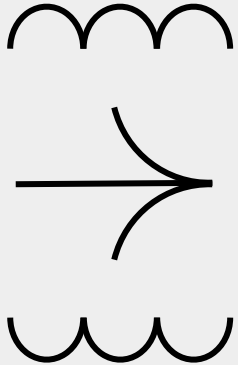
”

一個崇高的願景?



這門課並不是無聊的的概念解 說，那會太過反戲劇化。

如同一開始的副標題所述，我們想要帶領各位去看看一個有趣的 AI實現方式 – 使用 VertexAI 以及 Gemini 去幫圖片做包裝。

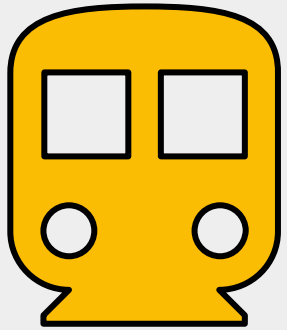


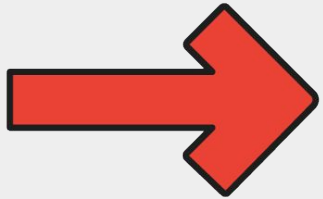
在接下來的過程中會讓各位去實際操作一個 Google Cloud帳戶，並且從新增專案到使用 VertexAI 的 API，從而製作出一個簡易的應用程式。

由於這些Credits是由Google贊助的研究學習經費，因此千萬不要錯過喔!

第二章

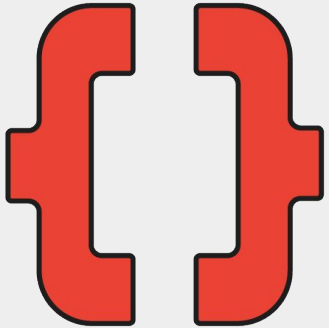
一套流程千萬隨





我們使用服務，並創造服務。

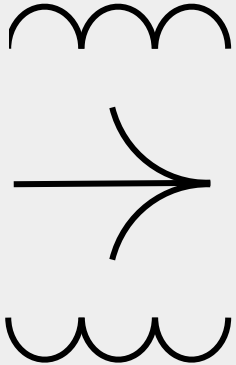
寄宿在屋頂之下



我們得先來了解一下 GCP要怎麼去使用，以及我們要使用甚麼這兩個部分，以今天的課程來說，各位可以先使用以下連結來獲得自己的試用研究帳戶。

https://trygcp.dev/e/build-ai-NKN01

Google的平台提供的服務其實挺多元的，從寄放單元的 Artifact Registry，到虛擬機執行的Compute Engine和Kubernete都有。



當然，VertexAI是他們正再向外推廣的技術，也是我們今天的主題，讓所有人都可以簡簡單單的 Build With AI。

要把資源本地化

我們需要三個步驟來正確使用他們的API。

專案創立

在Cloud Console新建一個專案，在主頁面就可以看到該專案的ID，這個等等就會在我們程式裡面使用。

這東西就是你的金鑰，記得不要外洩給別人看喔。

啟用API

直接在Console內搜尋對應的服務就可以找到API頁面，某些服務會需要你連接一個付款帳 戶。

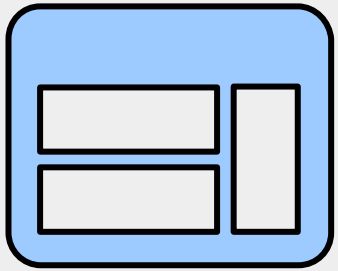
VertexAI是其中之一，但是我們有Credits可以用!

使用程式呼叫

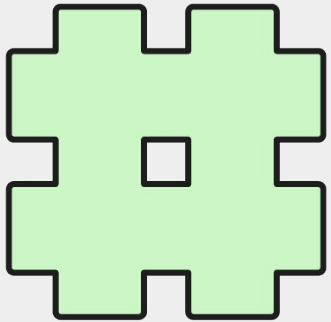
接著就是我們自己本地端的部分了，在安裝相關套件、進行了憑證之後，這些服務就對我們敞開。

今天會用Python作為框架去帶大家實際使用看看。

GCloud CLI



我們會需要GCP的本地環境來進行相關專案的開發，因此尚未安裝的同學可能要先去下載並安裝。

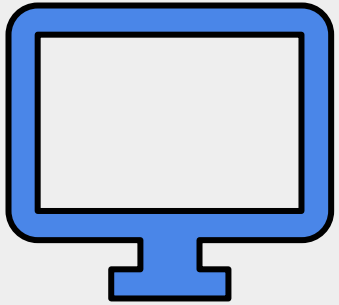


官方下載連結：https://cloud.google.com/sdk/docs/install#windows

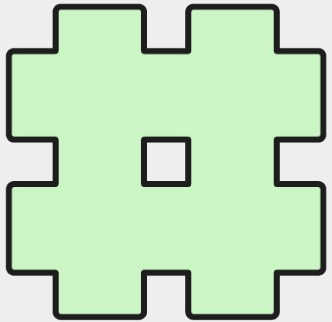
這個環境是基於Python的，支援版本是3.8到3.13，因此也要確保你們的電腦當中有正確的版本存在，否則可能會出現不可預期的問題。

完成之後，你的電腦裡面就有 CLI了，之後也會用它來進行一些 Google服務的串接、以及相關應用程式的開發。

Gemini的影像處理



我們可以用VertexAI的GenerativeModels去呼叫Gemini來為我們做一些資訊處理 – 這些資訊當然也不僅限於文字。



當然，要做一些簡易的 AI聊天機器人用這個框架非常輕易就做出來了，我們倒不如來試試看一些不同的 ...更加特別的角度。

我們今天會用gemini-2.0-flash-001作為我們的模型去進行所有操作，主要是Prompt Engineering和輸入/輸出的部分。

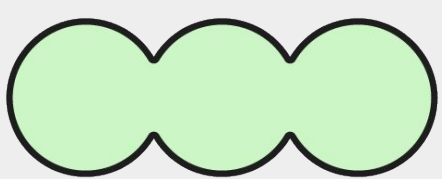
那麼究竟要怎麼去進行實作呢？

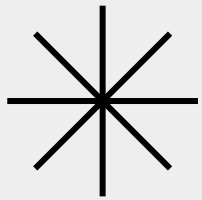
環境 & 秘密管理

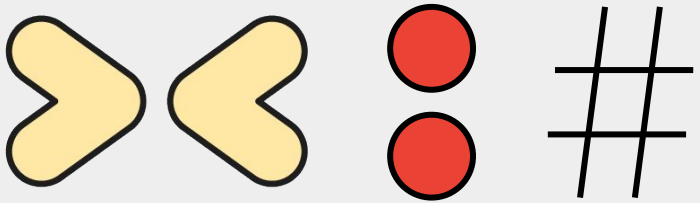
Python venv

我們的小小專案會在 Python的虛擬環境中執行，這樣安裝依賴件也會比較簡單。

```
>py -m venv env | $python3 -m venv env
```



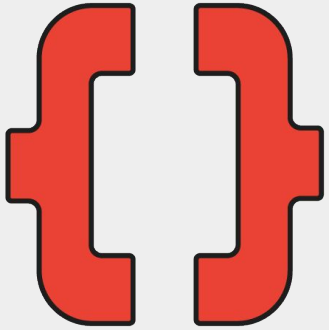




python-dotenv

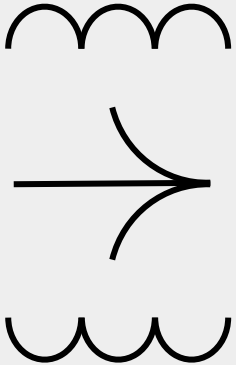
用.env來做憑證的控管比較安全，尤其是如果之後要發佈到Github/Azure Repos時，重要的鑰匙才不會被洩漏。

我們提供了一個範本



在我們組織的GitHub上，你可以去Clone下來一個簡易的VertexAI使用案例，也準備了requirements.txt方便依賴件的下載。

https://github.com/GDG-on-Campus-NKNU/Build-With-AI-NKNU



不過在實際運行之前，我們要在剛剛安裝好的 Google Cloud CLI上進行登入憑證，使用`gcloud auth login`來開啟驗證。

<這裡就看有沒有必要新增預設 啟動ID>

vertexai-demo.py

```
import vertexai
from vertexai.generative_models import GenerativeModel, Part, Image
from dotenv import load_dotenv
import os

load_dotenv()

vertexai.init(project=os.getenv('PROJECTID'), location=os.getenv('REGION'))

model = GenerativeModel(model_name="gemini-2.0-flash-001")

vertex_image = Image.load_from_file("./example.jpg")
prompt = "幫我簡單敘述一下這張圖片。"

response = model.generate_content([Part.from_image(vertex_image), prompt])
result_text = response.text.strip()
print("分析結果:", result_text)
```

功能

這段程式碼的效果就是將那張example.jpg給上傳，我就們會得到一段簡單的敘述文字。

更改Prompt裡面的內容可以得到不同的回覆，就像是你在詢問 Gemini一樣，只是我們在用程式達成這些事。

原理

我們呼叫了VertexAI的服務，將圖片轉換成了Part資料，連同簡易的Prompt丟給Gemini去做分析，並將其回傳的結果印出來，是一個基礎的API通訊的實踐。

當然這個服務要依據於你的專案 ID上。

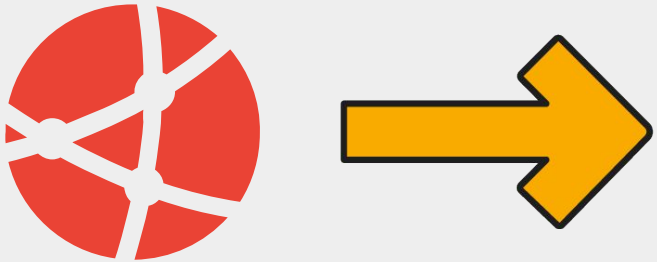
接下來，我們會進入更深層的實踐

準備做點有趣的東西了。

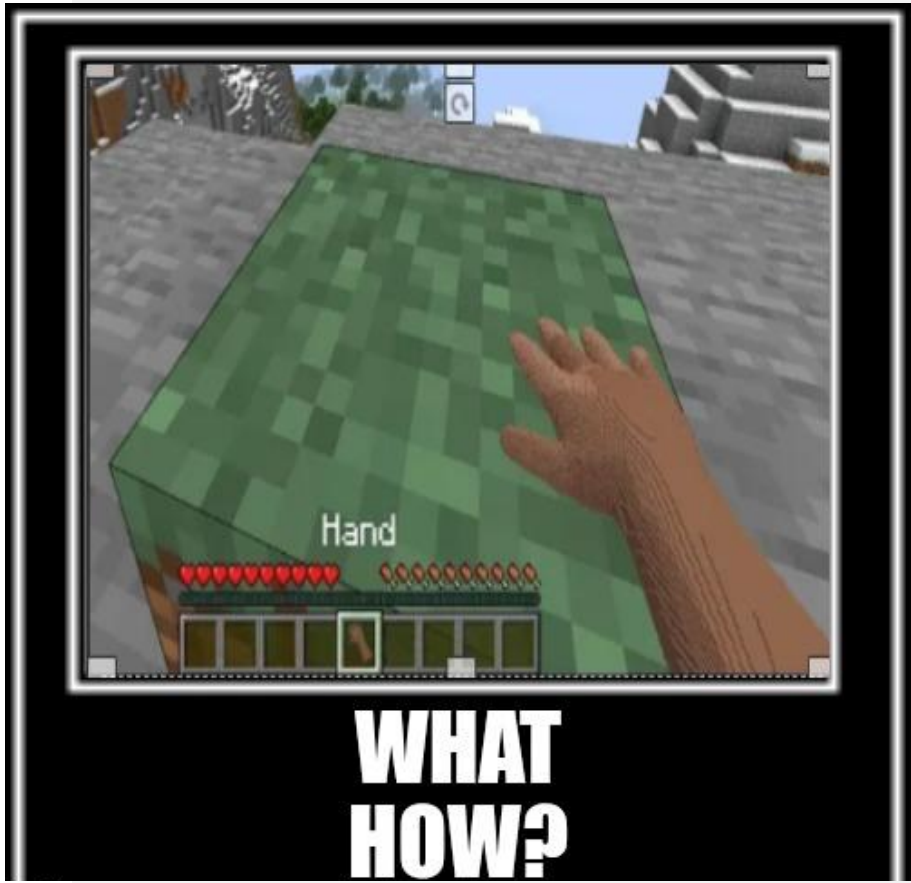
第三章

創意輸出零與一

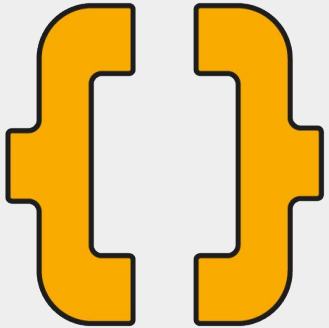
流水線的從零開始。



現在的迷因形式撲朔迷離 – 基本上任何東西都很好笑。



我們可以創造什麼?

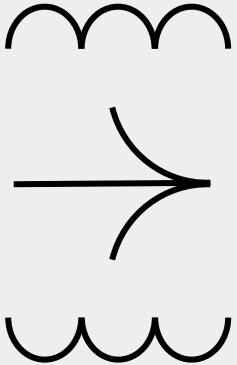


影像技術歸影像技術，生成式 AI歸生成式AI，我們可以從中去得到甚麼樣的啟發，或者來點更哲學的問題：我們可以創造甚麼？

你知道，很多人去嘗試過 AI的幽默感，而結果往往超出它們的想像。

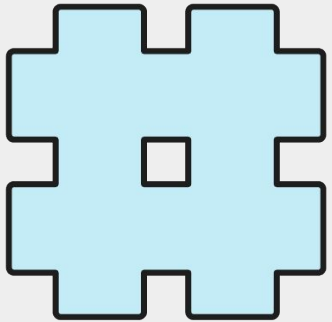
“In the future, humor will be randomly generated.” 這句話在網路上被調侃了許久，如今彷彿就像是赤裸裸的現實。

好吧，我們得承認人工智慧比我們更「有料」多了，那我們何不利用這點來做些東西呢？



解析幽默

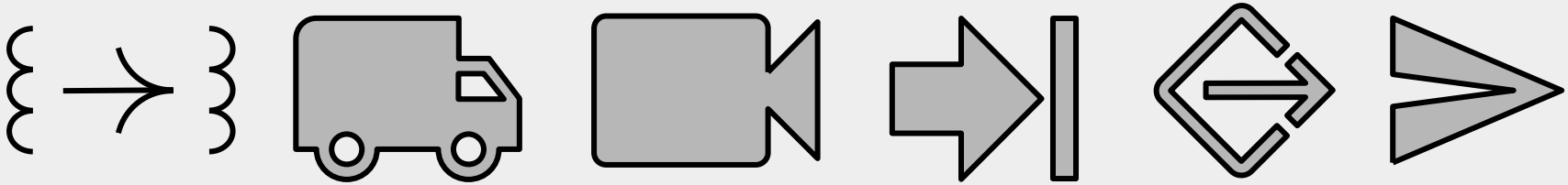
一個還算常見的表現形式是黑框 +文字包著一張圖，其中的 內涵在於那段文字會怎麼去敘述，以達到一個能戳到笑點的幽默。



黑框以及文字遷入的部分可以用 Pillow來輕鬆做到 (OpenCV也行啦，看妳們的喜好)，而文字本身可以請 Gemini來給我們想法。

概念有了，那麼就得動手去試試。

讓我們來思考一下，有甚麼核心的組件是需要我們去實作進去的 ？



服務架設

我們可以將應用程式塑造成一個服務，並用簡單的伺服器來架設API，在這裡使用的是Python的套件FastAPI。

圖片輸入/輸出

讓使用者輸入圖片路徑太不直觀了，因此我們可以試試用tkinter的filedialog實現簡單的選取介面，無論是上傳下載。

與語言模型溝通

用google-cloud-aiplatform去跟VertexAI進行溝通，我們就可以用Gemini來做為整個程式的核心，model也可以自選。

圖像處理

使用Pillow來進行簡易的影像處理，我們要在圖片外部加上一個邊框，以及可以放置文字的預留空間。

伺服器與呼叫程式

所有的程式邏輯都用server.py來實現，並且用另一個caller.py去進行API的呼叫，這樣子比較容易辨認。

容易忽略的細節

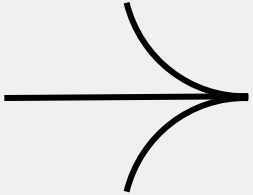
與伺服器的傳輸是使用圖片的Raw Data，然而VertexAI接受的必須是VertexImage格式，因此要先行進行轉換。

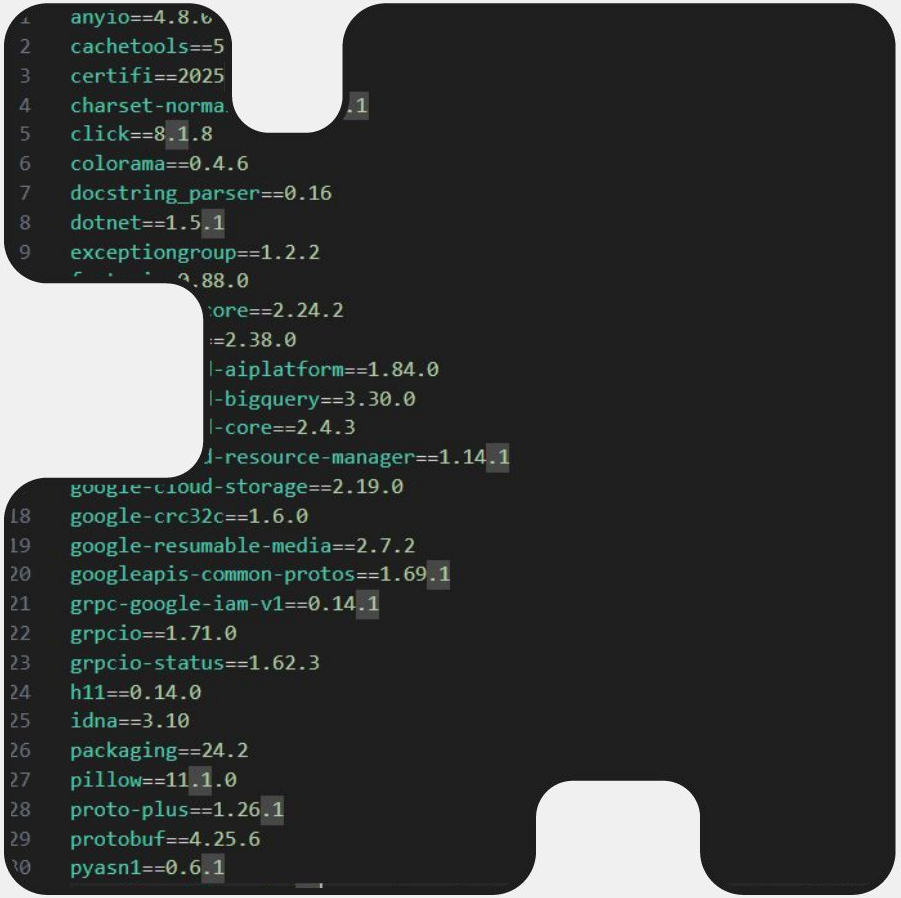
Source: xxxxx

拆解

在存放庫中我有另外附上 memify_demo.py, fastapi_demo.py 等等檔案來協助你們更快速的了解整體實作流程。

我們來看看整體工作流應該要怎麼去實現會比較容易。





MVP 的組成 (server)

我們需要有一個負責處理請求的路由，而在其中要能 夠實現圖片的處理 `(/process-image)`。

使用Pillow進行圖片byte的接收後，轉成pil_image供其他方法所使用，我們會有一個 `create_meme_image(input_image: Image.Image, text: str) -> Image.Image` 來進行整體梗圖化的操作。

而在那之前會先有一個 `generate_meme_text(image_data: bytes) -> str`將圖片資訊餵給Gemini，並獲得其分析之後的結果，以字串回傳，當然這個地方會有適當的圖片資料以及提示詞的組合傳入。

.png和.jpg有基本上的不同(尤其是alpha值的有無)，我們會在經過處理之後以正確的格式回傳圖片 資料 (RGBA 必要時轉回 RGB)。

小小的補充: 建議大家用外部的entry檔案去開啟伺服器，可以用uvicorn

MVP 的組成 (caller)

第一件事當然是連接到伺服器的 url，我們使用request來進行請求 (我在我的實踐當中，為了增加回饋性使用了request_toolbelt的multipart encoder以及tqdm來做進度條，這些其實都可以省略)。

開啟一個隱形的tkinter Root窗口，讓我們可以使用 filedialog的askopenfilename功能，這會開啟一個檔案瀏覽起讓使用者去選擇圖片檔案。

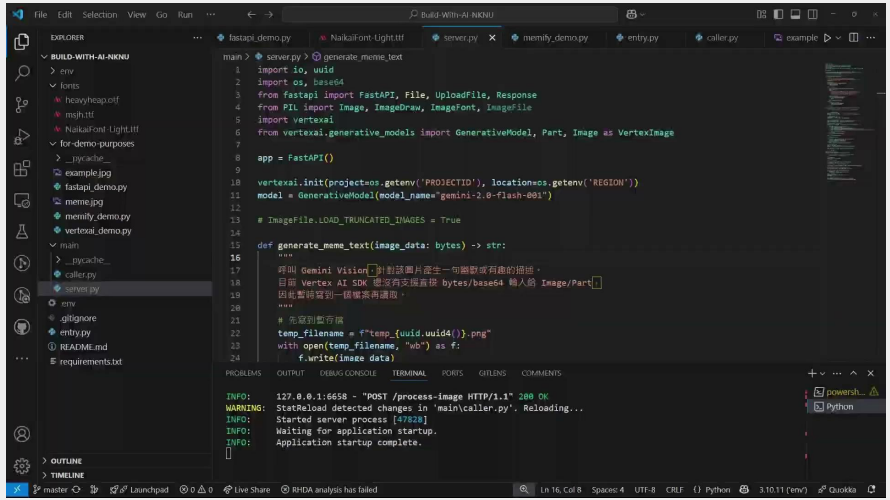
從回傳的content-type獲取圖片格式，詢問使用者下載的位置，然後再進行存放就可以了 (要做進度條的話可以用response.iter_content，但通常都是一瞬間完成)。

相較於伺服器，呼叫的部分比較直觀也簡單很多。

借用的創意

有了整體架構之後

我們可以來看看實際操作起來的感覺如何，最後產生的成果是不是我們所想要的。



單純只是幾十分鐘的功夫，一個應用就完成了。

總會有可以進步的空間。

終章

你所期待的樣子

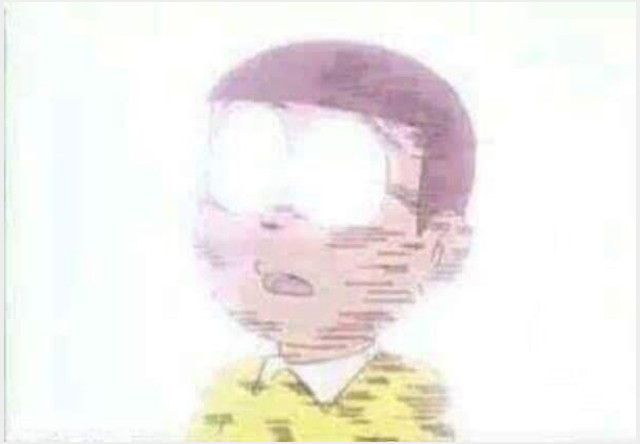
與AI協同工作的重點是你對於過程的享受。

一對一

{ }

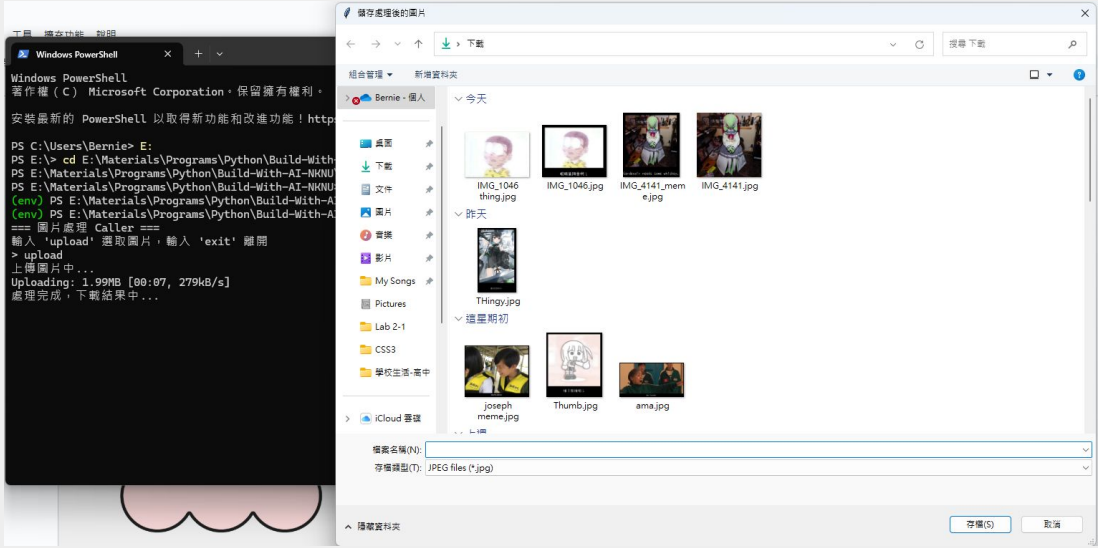
經過我們的調教，有了 AI的幽默生成以及簡單的影像處理技術，現在任何一張正常的圖片都可以變成一個有趣（? 的梗圖來讓大家笑笑。

當然，如果想要做成其他的迷因模板也完全沒有問題，只要做一些相對應的調整就可以了。



運作的樣子

```
(env) PS E:\Materials\Programs\Python\Build-With-AI-NKNU> py entry.py
INFO:     Will watch for changes in these directories: ['E:\\Materials\\Programs\\Python\\Build-With-AI-NKNU']
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [38944] using StatReload
INFO:     Started server process [10660]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```



和諧運作著

程式設計師最希望看到的景象就是開發出來的東西可以正確的運作，非常 值得慶幸的是我們並沒有遇到災難性的問題！

擔心配額?

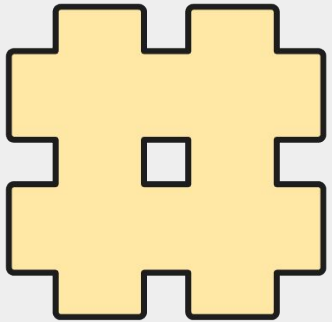
最後是很多人擔心也是最現實的問題: 一直用Google的服務, 錢會不會消失的很快?

以下是我這幾天開發的時候在 VertexAI上的帳戶統計, 其實使用的配額會比大部分人想像的要少很多, 尤其是像我們這種個人開發者, 不會在短時間內使用大量請求的情況下。

所以說如果是因為擔心自己的毛囊而遲遲不敢踏步的, 真的不用怕, 尤其 Google還對開發GenAI的新用戶有所補助。

更深一步的願景

到此，我們Build with AI的理念也算是徹底傳達了 – 只要想到什麼，想去嘗試甚麼，都可以找到工具或者方法去實作。



重點在於你是否有那個實踐力，願意花費寶貴的時間去完成一件突如其來的想法所產生的傻傻專案。

這個所謂的傻傻專案對你自己的回饋可能也意想不到。

Google提供了各式各樣的服務供你去使用 – 即使某些需要付費，你可以斟酌利益，並從你喜歡的方面下手。

工具永遠都在那裡，只是等待著適合的人去將其撿起來罷了。

小禮物與回饋環節！

有跟著今天的課程待到最後的同學 (現場的) 會拿到Google提供的小禮物喔!

課堂上演示的完整程式也會在結束之後被傳到GitHub上，有興趣的人也可以去做參考!

Google Developer Group

Editable Location

感謝你們的參與～

有甚麼問題都可以問講師喔！

Build with AI