Code :

**#Cài đặt từng gói thư viện**

**#proxy: Thư viện này được sử dụng để kết nối với proxy server.**

**#dplyr: Thư viện này được sử dụng để thao tác dữ liệu trong R.**

**#ggplot2: Thư viện này được sử dụng để tạo đồ thị trong R.**

**#tm: Thư viện này được sử dụng để xử lý văn bản trong R.**

**#corrplot: Thư viện này được sử dụng để vẽ biểu đồ tương quan trong R.**

install.packages("proxy")

install.packages("dplyr")

install.packages("ggplot2")

install.packages("tm")

install.packages("corrplot")

**#Gọi thư viện**

library(tm)

library(proxy)

library(dplyr)

library(ggplot2)

library(corrplot)

print("The dataset: each sentence is one docoment")

doc <- c( "The sky is blue.", "The sun is bright today.", "The sun in the sky is bright.", "We can see the shining sun, the bright sun.", "The moon is full, the sky full of stars.", "The sky was dark, the stars plentiful and bright.", "The sun is but a morning star.")

# Ma trận tf-idf là một ma trận được sử dụng trong xử lý ngôn ngữ tự nhiên để đo lường tầm quan trọng của một từ trong một văn bản. Ma trận tf-idf được tính toán bằng cách nhân ma trận tần suất từ (tf) với ma trận tần suất nghịch đảo tài liệu (idf).

#Ma trận tần suất từ (tf) là một ma trận được sử dụng để đếm số lần xuất hiện của một từ trong một văn bản.

#Ma trận tần suất nghịch đảo tài liệu (idf) là một ma trận được sử dụng để đo lường mức độ phổ biến của một từ trong một tập hợp các văn bản.

# Dòng code này tạo một corpus từ văn bản doc. Corpus là một tập hợp các văn bản đã được xử lý. Trong trường hợp này, văn bản doc có thể là một tập hợp các câu, đoạn văn hoặc thậm chí là toàn bộ tài liệu.

#Hàm Corpus() lấy một đối tượng VectorSource() làm đầu vào. Đối tượng VectorSource() tạo một vector từ văn bản. Trong trường hợp này, vector được tạo bằng cách chia văn bản thành các từ và đếm số lần mỗi từ xuất hiện.

**corpus <- Corpus( VectorSource(doc) )**

# Dòng code này tạo một danh sách các tham số kiểm soát để sử dụng khi xử lý văn bản. Các tham số kiểm soát này bao gồm:

#removePunctuation: Loại bỏ dấu chấm câu.

#stopwords: Loại bỏ các từ dừng.

#tolower: Chuyển đổi tất cả các từ thành chữ thường.

#Các tham số này được sử dụng để cải thiện độ chính xác của phép đo tf-idf.

**controlList <- list(removePunctuation = TRUE, stopwords = TRUE, tolower = TRUE)**

# Dòng code này tính toán ma trận tần suất từ (tf) cho corpus. Ma trận tf là một ma trận cho biết số lần xuất hiện của mỗi từ trong corpus.

#Hàm TermDocumentMatrix() lấy một đối tượng corpus và một danh sách các tham số kiểm soát làm đầu vào. Hàm này trả về một ma trận, với mỗi hàng đại diện cho một văn bản trong corpus và mỗi cột đại diện cho một từ. Phần tử i, j của ma trận chứa số lần từ j xuất hiện trong văn bản i.

**print("computing the term-frequency matrix: ")**

**(tf <- as.matrix(TermDocumentMatrix(corpus, control = controlList) ) )**

# Dòng code này vẽ biểu đồ ma trận tf.

#Hàm corrplot() lấy một ma trận làm đầu vào và vẽ biểu đồ ma trận. Trong trường hợp này, chúng ta sử dụng phương pháp "number" để vẽ biểu đồ ma trận tf dưới dạng biểu đồ đường. Chúng ta cũng sử dụng tùy chọn is.corr = FALSE để cho biết rằng ma trận tf không phải là ma trận tương quan. Cuối cùng, chúng ta sử dụng tùy chọn cl.pos = "n" để cho biết rằng không nên hiển thị nhãn cột.

**corrplot(tf, method = "number", is.corr = FALSE, cl.pos = "n")**

# Dòng code này tính toán ma trận tần suất nghịch đảo tài liệu (idf) cho corpus. Ma trận idf là một ma trận cho biết mức độ phổ biến của mỗi từ trong corpus.

#Hàm log() tính toán hàm logarit của một số. Trong trường hợp này, chúng ta sử dụng hàm logarit tự nhiên.

#Hàm diag() tạo một ma trận chéo, với các phần tử chính bằng các phần tử đã cho. Trong trường hợp này, chúng ta sử dụng hàm diag() để chuyển đổi ma trận idf thành một ma trận chéo.

**print("computing the idf, and then converting into a diagonal matrix (used later)")**

**(idf <- log(ncol(tf) / (1 + rowSums(tf != 0) ) ) )**

**(idf <- diag(idf) )**

print("calculating the final tf-idf matrix")

# Dòng code này tính toán ma trận tf-idf bằng cách nhân ma trận tf với ma trận idf.

#Hàm crossprod() tính toán tích chéo của hai ma trận. Trong trường hợp này, ma trận tf là ma trận tần suất từ và ma trận idf là ma trận tần suất nghịch đảo tài liệu.

#Kết quả của hàm crossprod() là một ma trận, với mỗi hàng đại diện cho một văn bản trong corpus và mỗi cột đại diện cho một từ. Phần tử i, j của ma trận chứa số lần từ j xuất hiện trong văn bản i, nhân với idf của từ j.

**tf\_idf <- crossprod(tf, idf)**

# Dòng code này đổi tên các cột của ma trận tf-idf thành tên của các hàng của ma trận tf.

#Hàm colnames() đặt tên cho các cột của một ma trận. Trong trường hợp này, chúng ta đặt tên cho các cột của ma trận tf-idf bằng tên của các hàng của ma trận tf. Điều này giúp chúng ta dễ dàng đọc hiểu ma trận tf-idf hơn.

**colnames(tf\_idf) <- rownames(tf)**

# Dòng code này chuẩn hóa ma trận tf-idf bằng cách chia mỗi phần tử của ma trận cho căn bậc hai của tổng bình phương các phần tử trong ma trận.

#Hàm rowSums() tính tổng của các phần tử trong một hàng của một ma trận. Trong trường hợp này, chúng ta sử dụng hàm rowSums() để tính tổng bình phương của các phần tử trong mỗi hàng của ma trận tf-idf.

#Hàm sqrt() tính căn bậc hai của một số. Trong trường hợp này, chúng ta sử dụng hàm sqrt() để tính căn bậc hai của tổng bình phương các phần tử trong mỗi hàng của ma trận tf-idf.

**(tf\_idf <- tf\_idf / sqrt(rowSums(tf\_idf^2) ) )**

# Dòng code này vẽ biểu đồ ma trận tf-idf đã được chuẩn hóa.

#Hàm corrplot() lấy một ma trận làm đầu vào và vẽ biểu đồ ma trận. Trong trường hợp này, chúng ta sử dụng phương pháp "number" để vẽ biểu đồ ma trận tf-idf dưới dạng biểu đồ đường.

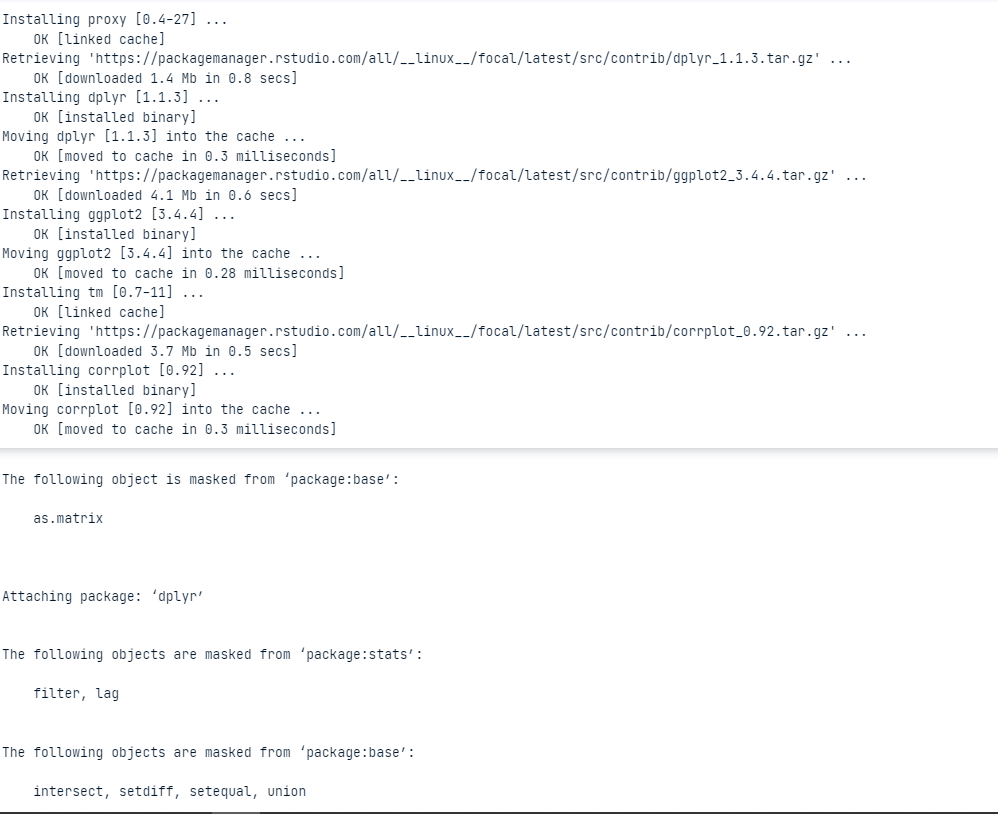
#Kết quả của hàm corrplot() là một biểu đồ hiển thị giá trị của các phần tử trong ma trận tf-idf. Giá trị của các phần tử trong ma trận tf-idf cho biết tầm quan trọng của một từ trong một văn bản.

**corrplot(tf\_idf, method = "number")**

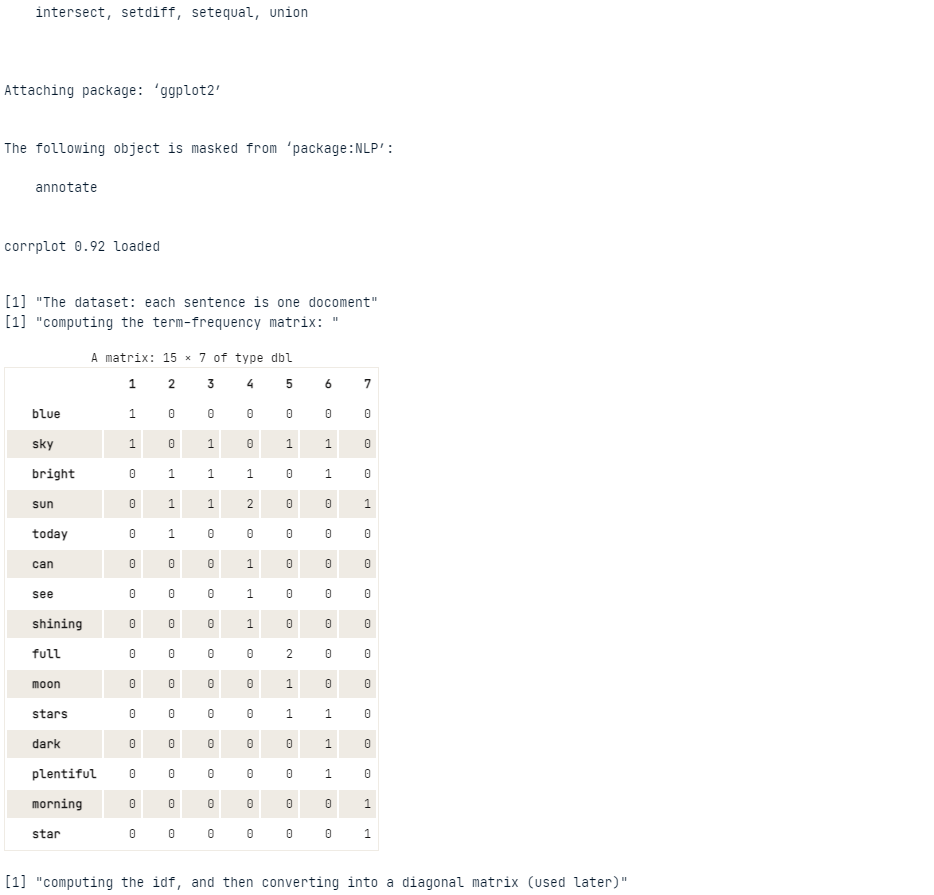


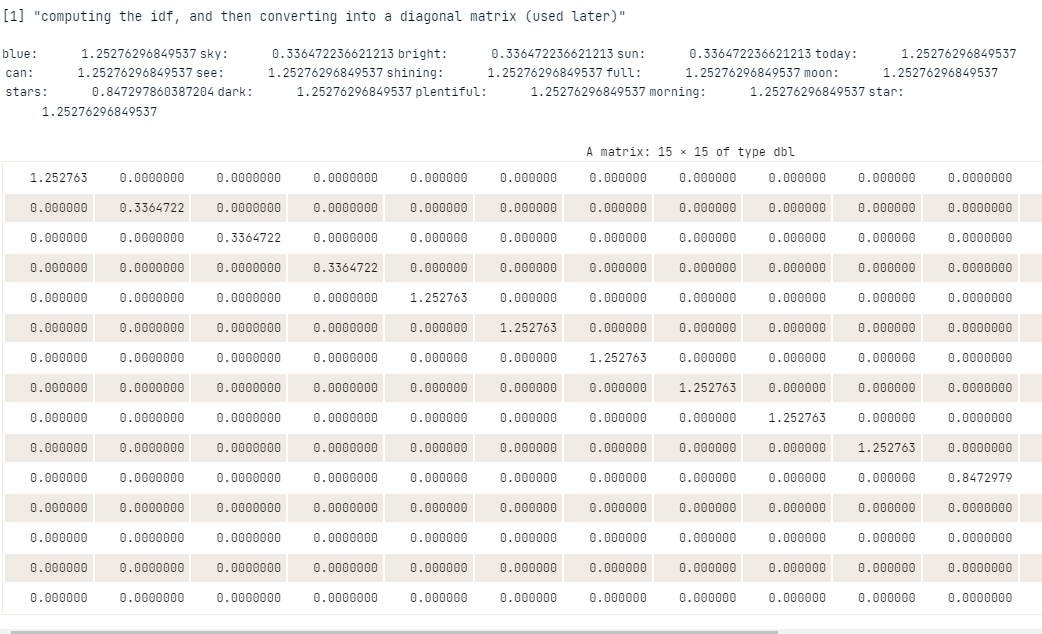
**Output :**

**Install – cài dặt các thư viện**

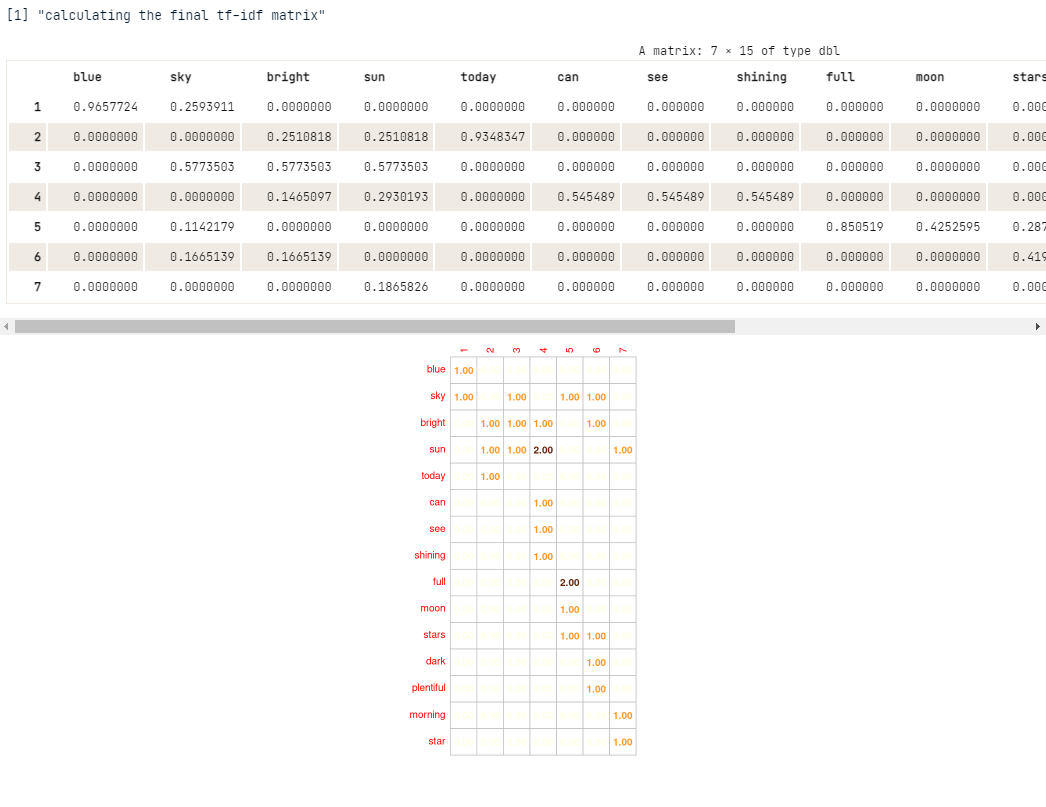


**Tạo ra một ma trận : 15 X 7**

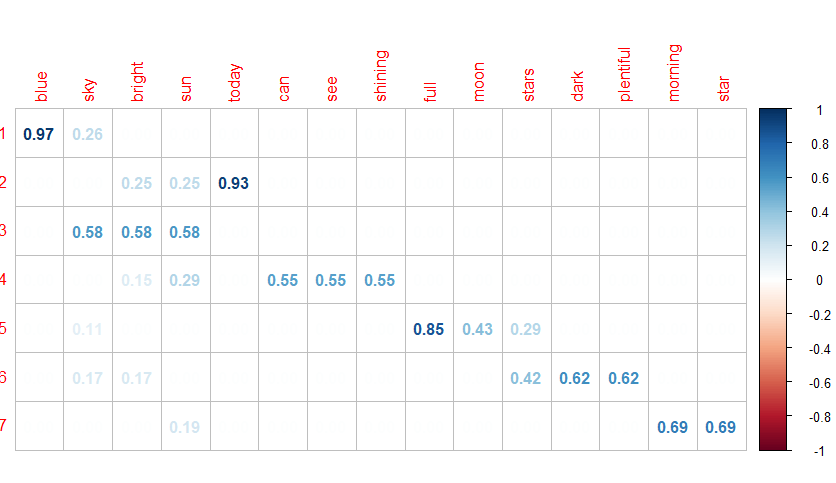




**Output : Tính toán Ma trận tf-idf**

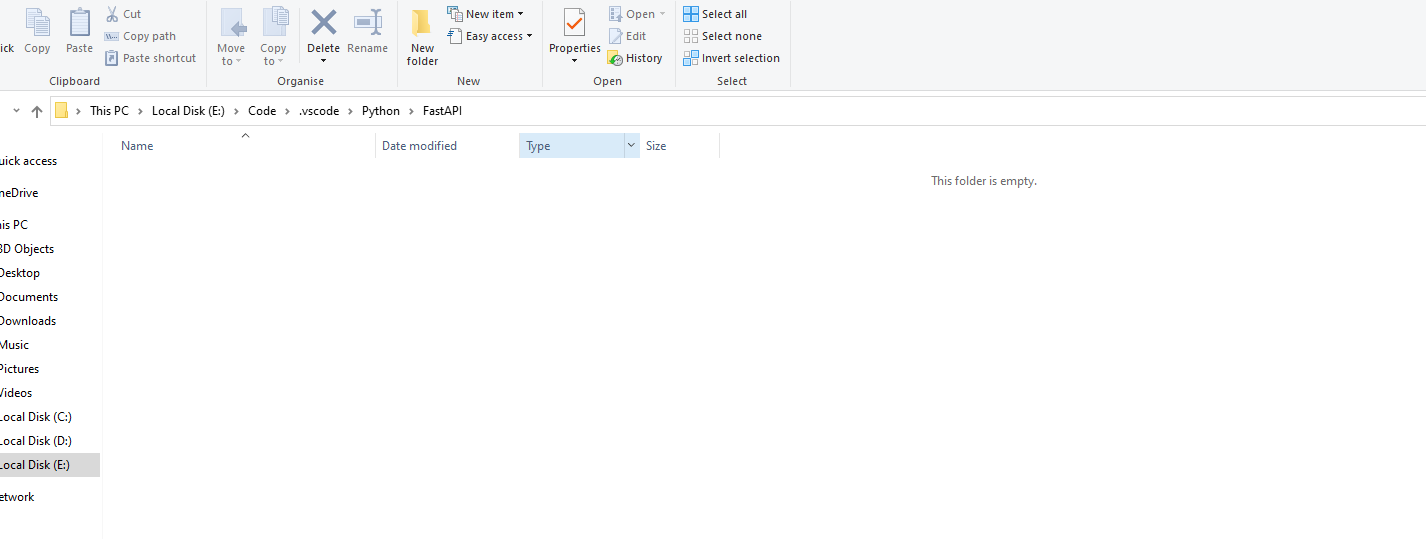


**Kết quả cuối cùng :**

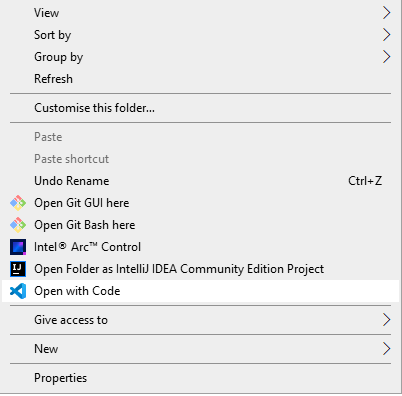


**Cài đặt FastAPI**

Tạo thư muc tên FastAPI



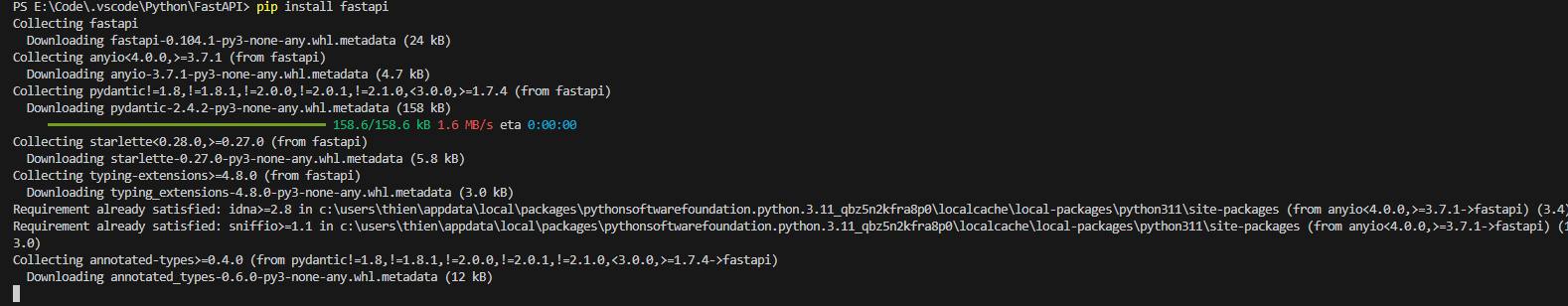
Bấm Open with Code ( Visual Studio )



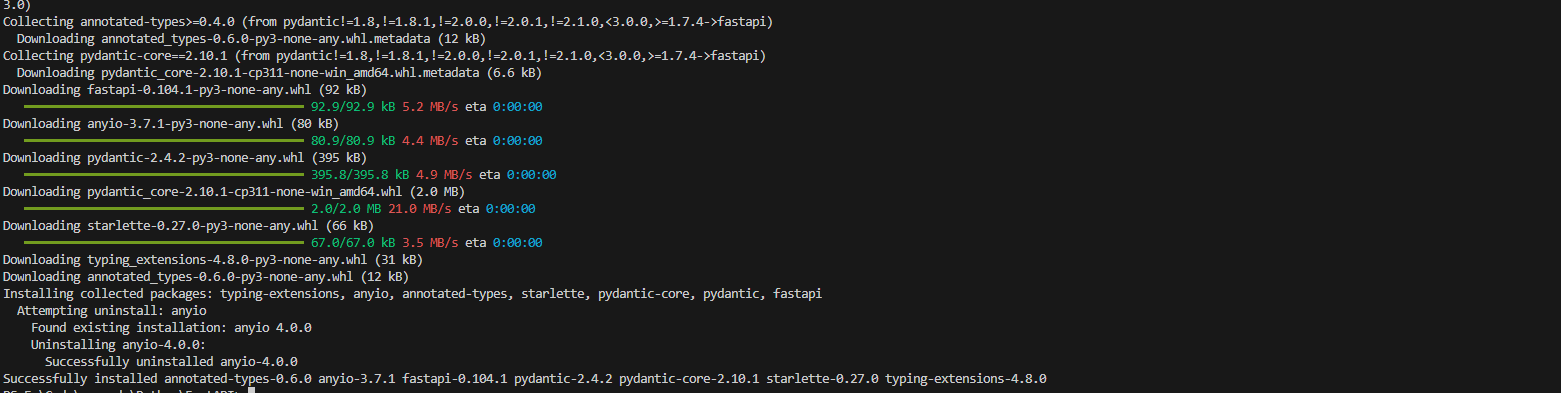
Mở ra và bấm terminal

Nhập lệnh : pip install fastapi để cài fastapi

Quá trình chạy

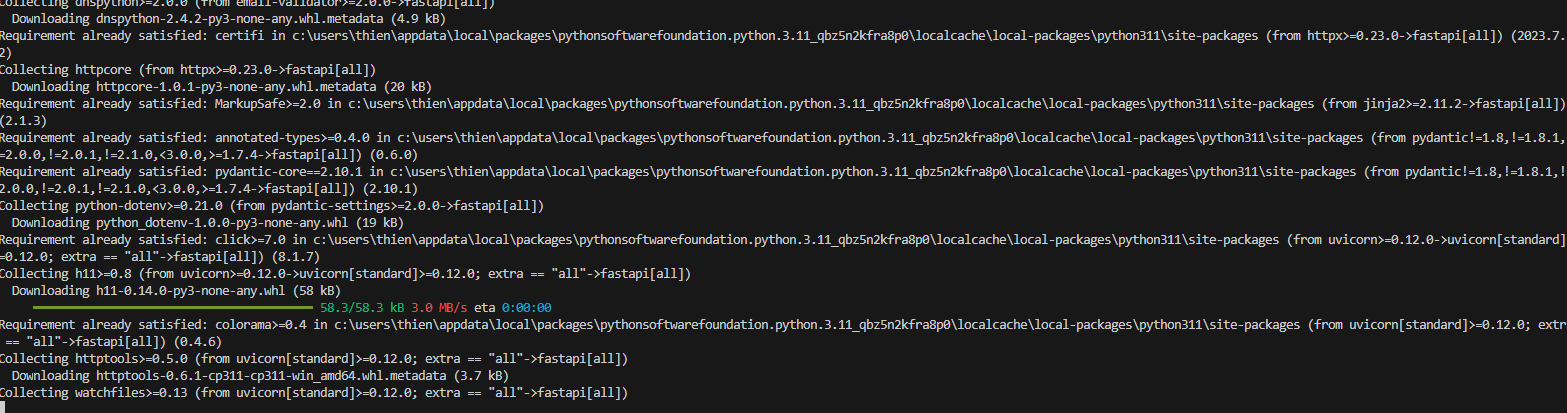


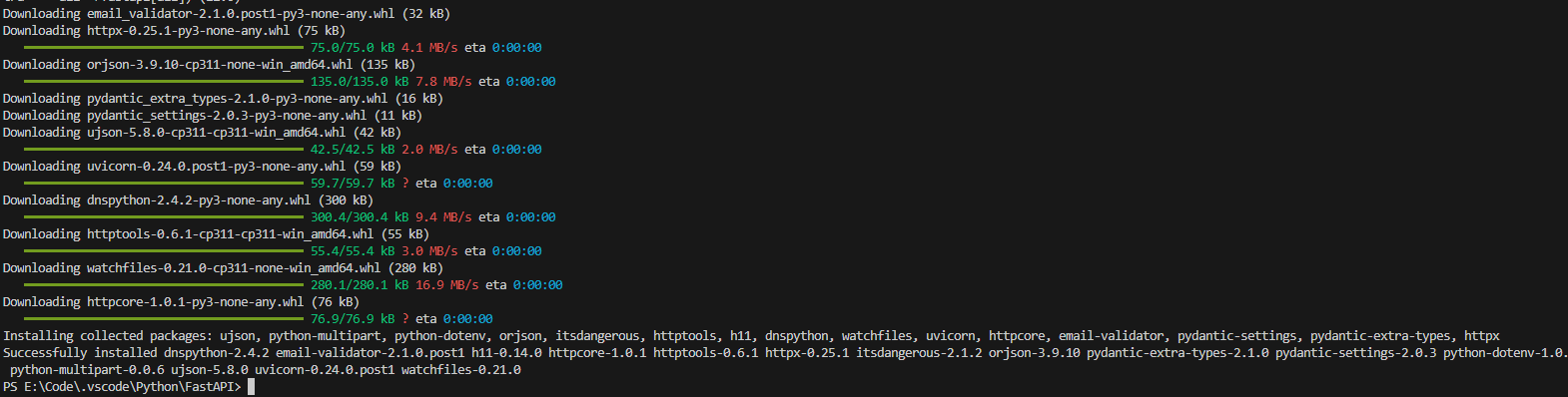
Cài đặt hoàn tất :



Sau đó tiếp tục : dùng pip install fastapi[all] để cài tất cả







Dùng lệnh : pip install "uvicorn[standard]"

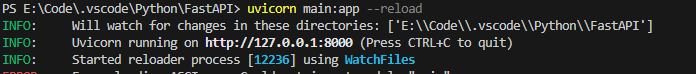




Dùng lệnh : uvicorn main:app --reload



Dùng <http://127.0.0.1:8000/docs> để vào :



Kết quả : giao diện FastAPI

