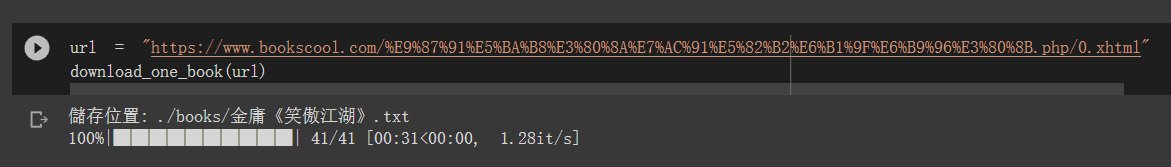
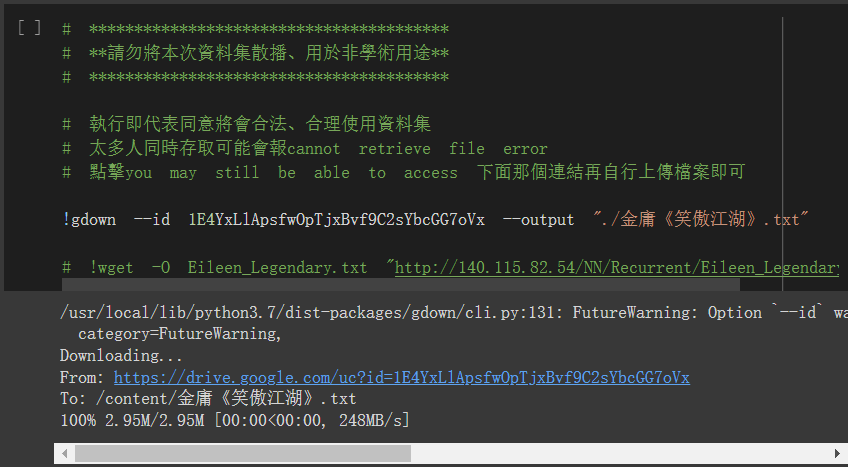
**Hw2 – 109403021**

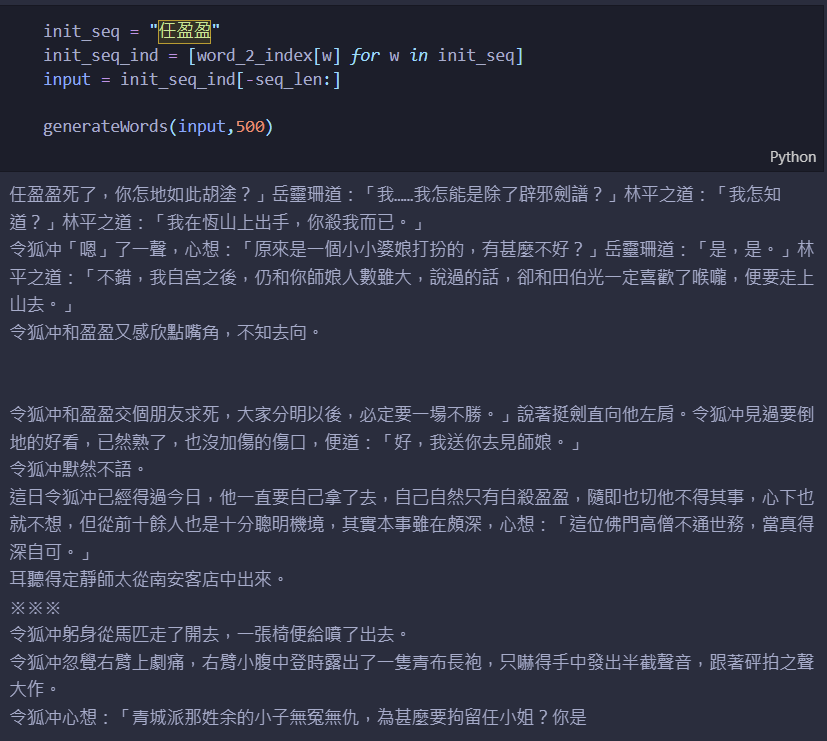
**Text Generation連結:** [**連結**](https://colab.research.google.com/drive/1Pw09oZ-Fd0t86_rAR1LE-TM9diz0L-B-?usp=sharing)

**StockRNN 連結:** [**連結**](https://colab.research.google.com/drive/1k_FeT9rnysEKcwS_GIsDiNE3DS9CcxWK?usp=sharing)

1. **Text Generation**

先利用提供的爬蟲程式下載了金庸的笑傲江湖來做為此次更改使用的資料集，然後載入資料集

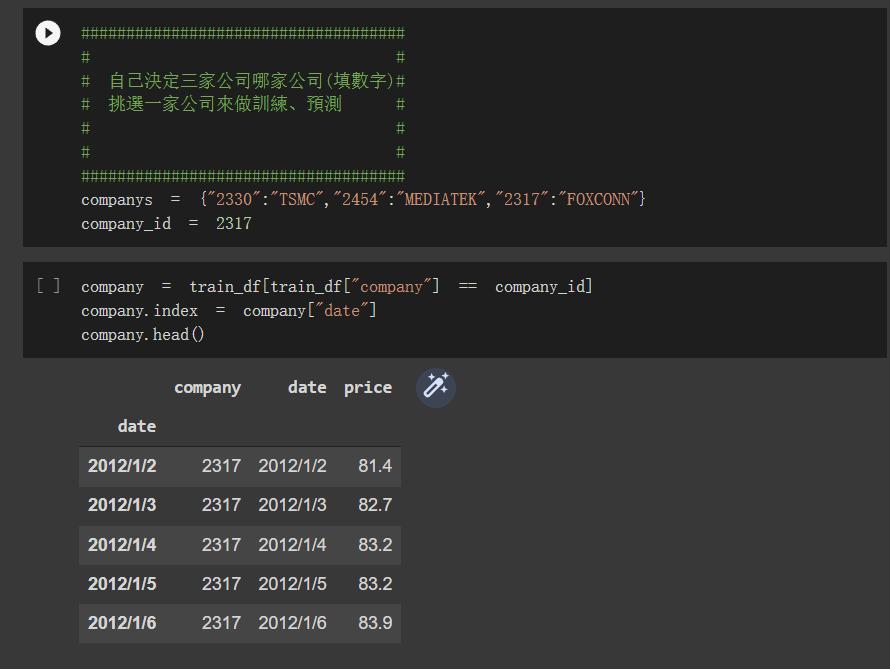




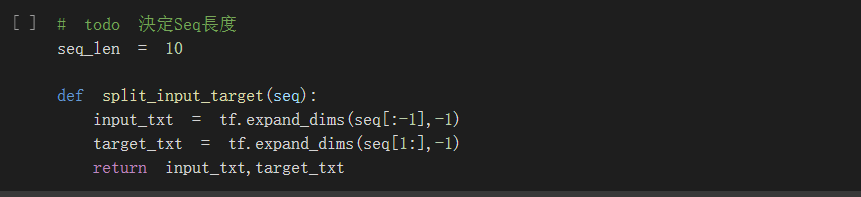
最後我使用”任盈盈”做為開頭產生出以上的文章，雖然一些文法如上下引號的使用有一些錯誤，但是可以看出仍有一些字詞的正確對應如各角色的姓名、林平之知道自己自宮過。

1. **StockRNN**

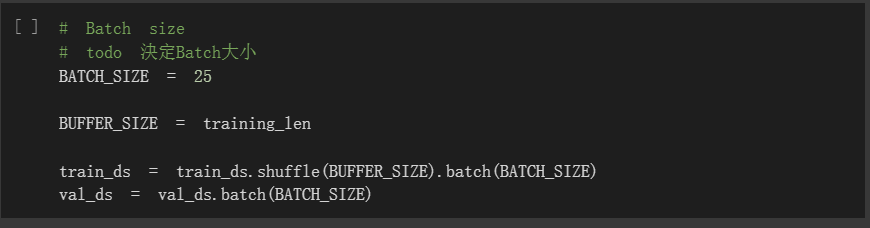
我選擇了2317:FOXCONN這家公司來做訓練、預測



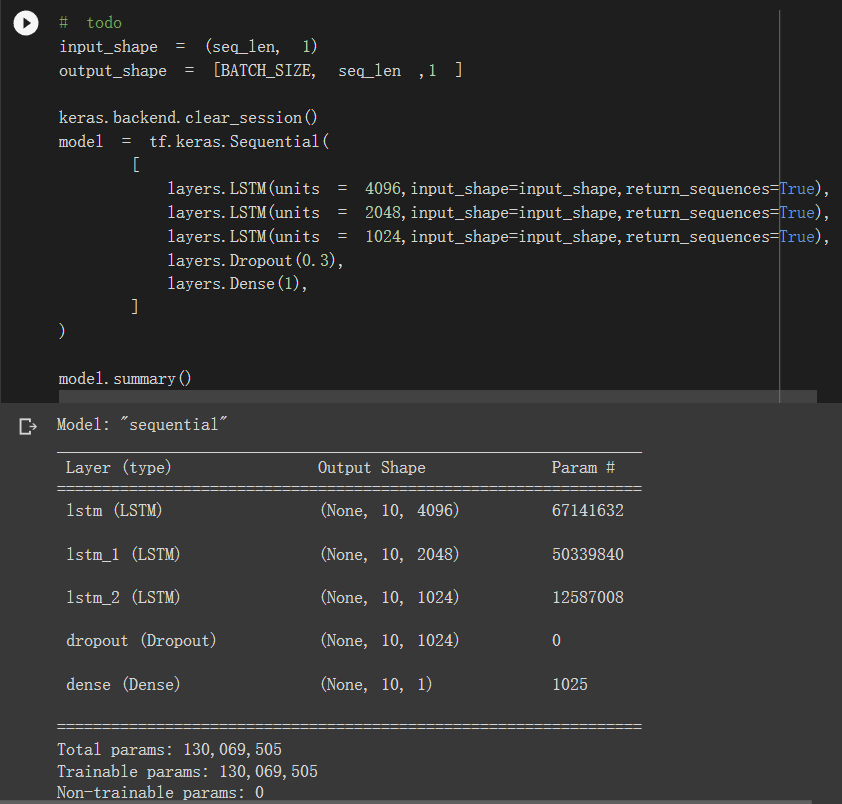
將seq\_len修改為10，因為覺得增加觀察和預測的天數能較不擬合



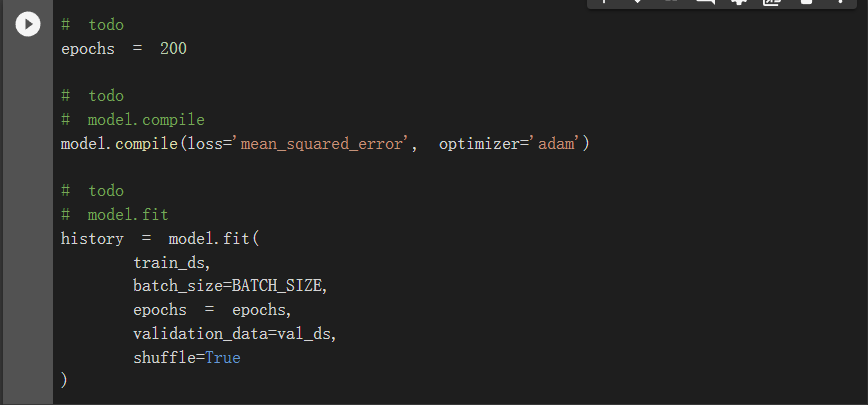
來回訓練調整batch size為25



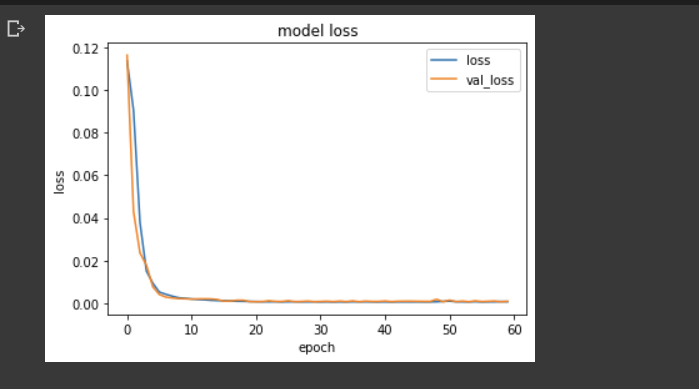
建立模型。參考TextGeneration的模型最終疊了3層LSTM，並加上dropout防止過度擬合，但效果其實不顯著



訓練了200次。相比上次的作業，這次使用mse計算loss，而optimizer依然使用adam。model.fit中shuffle設為true打亂資料集



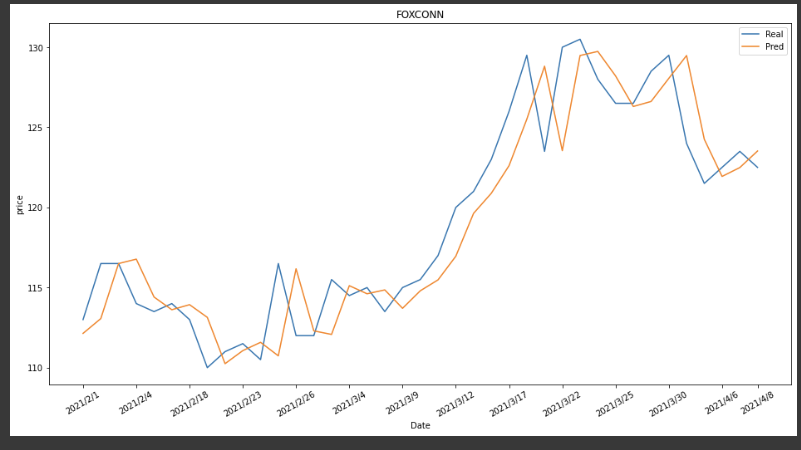
結果只訓練了不到十次就收斂到頗小的loss



根據提示先將input的維度轉換為next\_input，再取出最後一個output值加入到pred\_prices列表中，再將real\_price轉為小數點形式並加入input列表，最後為了維持input的長度去掉了第一個值



預測和真實的結果如下，走勢大致相同但預測的會有一些延遲感



1. **心得**

這次的作業相較上次的確難度有所降低，但我覺得比上次有趣很多，文字生產器看到能產生自己喜歡的小說風格之文章，就是前後語意毫無邏輯甚至有點搞笑。由此我認為這種文字生產器基本上應該很難應用在小說這種帶有大量對話和劇情邏輯的地方，畢竟RNN只是藉由前一個字延續預測、產生下去，不可能顧及到文句前後的邏輯。所以這種技術應該較於適合應用在一些抽象、藝術，前後文較無強烈邏輯連結的文章，諸如新詩之類的。

股票走勢預測的作業因為我還對股市沒有接觸所以較無深刻感受，雖然預測出來的走勢和真實的十分相近，但由於是參考前面的資料做預測而有難以避免的延遲效果。而且我認為這種單純由之前的數據來預測的方式仍然不夠現實，畢竟真實的情況要考慮的層面還多得很多。不過如果能將其他層面的影響也化為參數考慮進來，可能真的就可以製造出精準度很高的股票預測器。

RNN這種「記憶」模式的預測感覺還有很多可以實現的應用，十分有可能讓我未來想要預測或是呈現某種序列時，嘗試使用RNN實作、實現，因此在思考層面上這次作業激發了我不少。