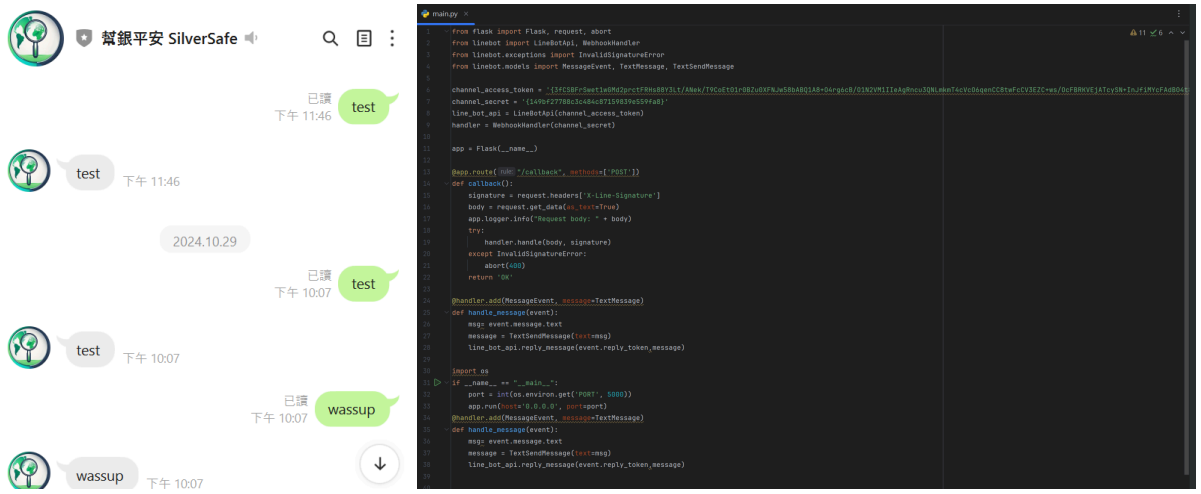
At first, I actually wanted to create an app, but It is too difficult and there are few online tutorials dedicated to creating an app. So instead I created a Line bot, which there are basic tutorials on how to create a simple one first.

Step 1: I watched these youtube videos to familiarize with python and first create a Line bot that replies whatever I texted it:

- This step took about a month, familiarizing with basic python commands was arduous, but creating the first bot wasn't that hard since the tutorials are clear.

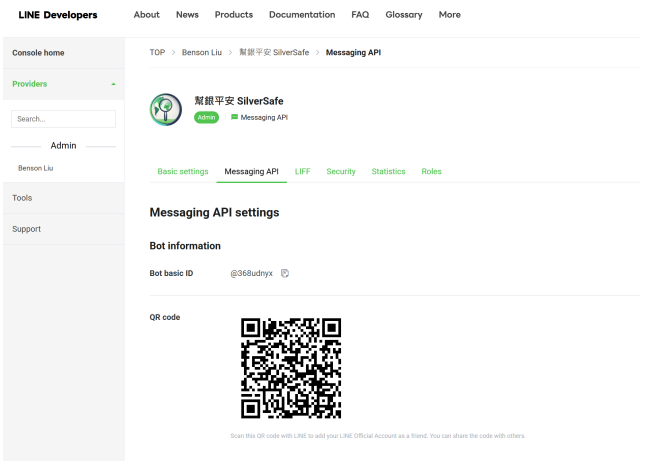


Python 6 小時初學者課程 (2023) #python教學 #python入門 #完整課程

10分鐘用Python做LineBot聊天機器人｜程式設計｜提醒:heroku已經在2022/11/28公告…

Also, need to create a Line developers account, I basically just followed the youtube videos, or look at github examples

ex: https://github.com/yaoandy107/line-bot-tutorial



Step 2: Expanding on the basic line bot created in step 1, I eventually created the first prototype

- took me about 1-2 month to finish this step as it was really complicated at first, and oftentimes the codes required to build this bot was too hard for my current level

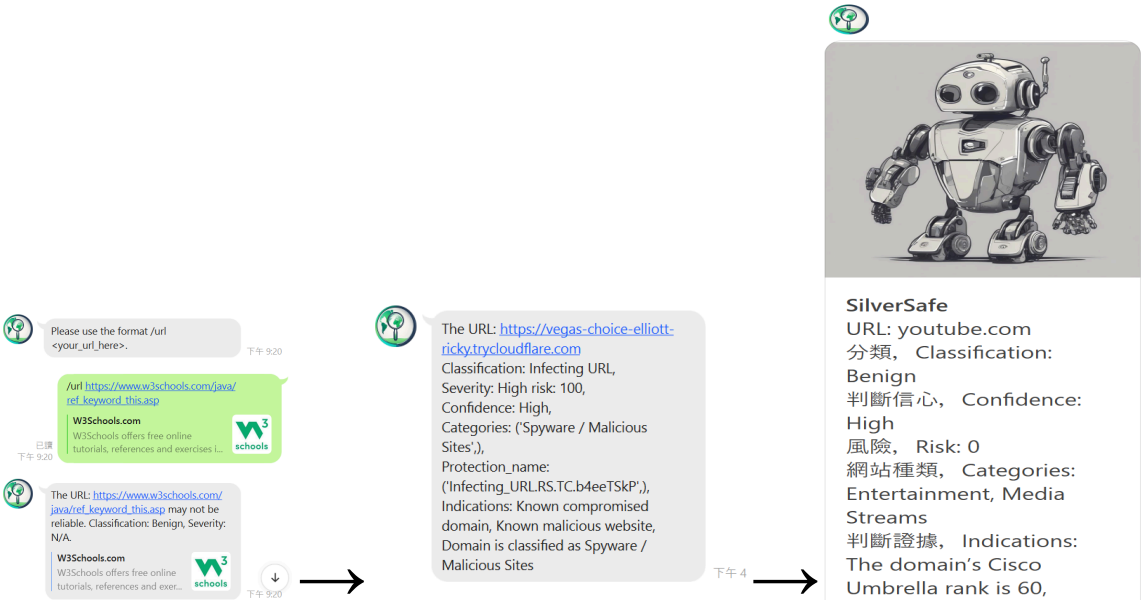
Things that helped me in this step

1. W3Schools when I forgot what python command to use
2. Github similar tutorials of calling on API's
3. Chat gpt plus for general suggestions and debugging
4. If I really get stuck and even chat gpt can't find the problem(usually dealing with environment setting problems or cmd command issues)I'll Use Anydesk to connect with my Dad's coder friend and he will look and teach me what I did wrong.

Thing I learned in this step:

Although this version can detect URL reliability and classify it, the format's are too compacted and not visibly clear. Which is why I again searched on youtube tutorials and learned how to change the flex messages using Line's official flex message simulator: https://developers.line.biz/flex-simulator/
And I also added sub categories such as confidence ratings, url's categories, indications to make the result more precise.

Here's the evolution so far



Step 3: Improve on the bot's function

- This step took another 1 month

As the prototype was ready to use, I introduced this bot to about 200 people (relatives, friends, parents' friends…etc)to collect user feedback. The feedback can be concluded in the following points.

1. Everytime the user wants to search up any link or url, they need to type the keyword "/url" first.
    a. This was especially hard to fix since this keyword is core logic for my code: it is how the code determines if the user wants to start a search. If I didn't include this keyword determination factor, the bot will fail to determine if the user provided text that's not a functional link.
    b. If I didn't remember wrong, this was what the code looks like:
    @handler.add(MessageEvent, message=TextMessage)
    def handle_text(event):
        text = event.message.text.strip()
        if text.lower().startswith("/url "):
            query = text[5:]
            do_search(query)
2. Many users complained that they don't really care about the additional information such as indication, categories…etc, they preferred a response that can immediately inform them if the url or link is reliable or not.

How did I fix these problems?

1. I looked on reddit discussions and asked chat gpt for suggestions, and I tried to switch from a prefix-based trigger to a global regex search so that the bot can pick up any valid link or domain and process it without the need of a certain keyword.

```
url_pattern = re.compile(
    r'\b(?:(?:[a-z][a-z0-9+\-.]*://)?(?:www\.)?)'
    r'(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,}'
    r'(?:/\S*)?\b',
    re.IGNORECASE
)
match = url_pattern.search(user_message)

if match:
    url = match.group(0).strip()
```

I also added logic to prepend bare domains so when the matched domain has no protocol the API call always sees a fully qualified URL

```
if not re.search(r'://', url):
    url = "http://" + url
url_info = cp_get_url_rep(url)
```

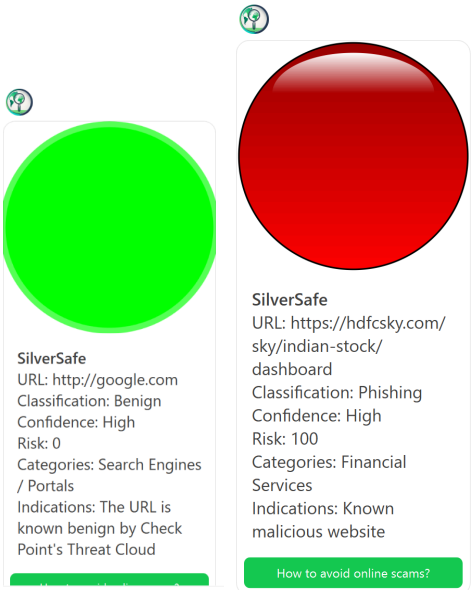
2. To resolve problem 2, I simply use if statements in the following way: if the url or link is classified as unreliable it shows a huge red light image, if its classified as N/A then it shows a yellow light image, if its classified as benign then it shows a huge green light image

```
def get_light_color(classification, risk):
    risk = risk.lower()
    if "Infecting URL" in classification or "CnC Server" in classification or
"Compromised Website" in classification or "Phishing" in classification or
"Infecting Website" in classification or "Spam" in classification or
"Cryptominer" in classification or "Volatile Website" in classification:
        return "red"
    elif "Web Hosting" in classification or "File Hosting" in classification
or "Parked" in classification or "Unclassified" in classification:
```

```
        return "yellow"
    else:
        return "green"
// and these codes for loading the image.
if url_info:
    light_color = get_light_color(url_info['classification'],
url_info['risk'])

    light_images = {
        "red":
"https://www.pngall.com/wp-content/uploads/14/Red-Light-PNG-Cutout.png",
        "yellow":
"https://png.pngtree.com/png-clipart/20201029/ourmid/pngtree-circle-clipart-
orange-yellow-circle-png-image_2381941.jpg",
        "green":
"https://www.pngall.com/wp-content/uploads/14/Green-Circle-PNG-Images.png"
    }
```

After resolving these problems, this is the final form.



Future plans

1. For some reason, the API only shows the risk level as 0, 64, or 100. I want to fix it so that it can reflect the actual risk values instead of the thresholds that each classification belongs to.
2. I also want to try to add a function where the bot can detect phishing emails.

Why do I enjoy this project?

This project is an interesting initiative because this is the first thing that I made by myself that can have some impact in others' lives. And I especially enjoyed the process where I had to adjust my code, the function of the bot, and even its aesthetics; although it is easy to get frustrated when debugging, struggling to find solutions that can meet users expectations, resolving all these problems using my own way made it enjoyable and entertaining. I like the feeling of discovering how to create or fix the features of this bot by myself, whether it is searching online, asking others, or even exploiting AI, especially since there is really no wrong answer.