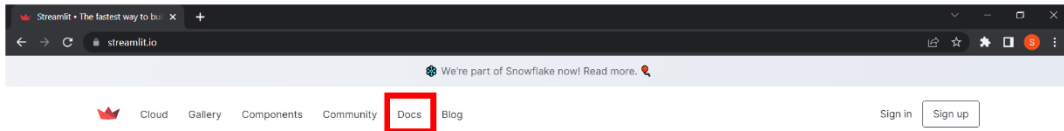


ไลบรารี Streamlit

เว็บไซต์ <https://streamlit.io>

เมื่อเข้ามาที่หน้าเว็บไซต์ ให้กดที่เมนู Docs



A faster way to build and share data apps

Streamlit turns data scripts into shareable web apps in minutes.
All in pure Python. No front-end experience required.

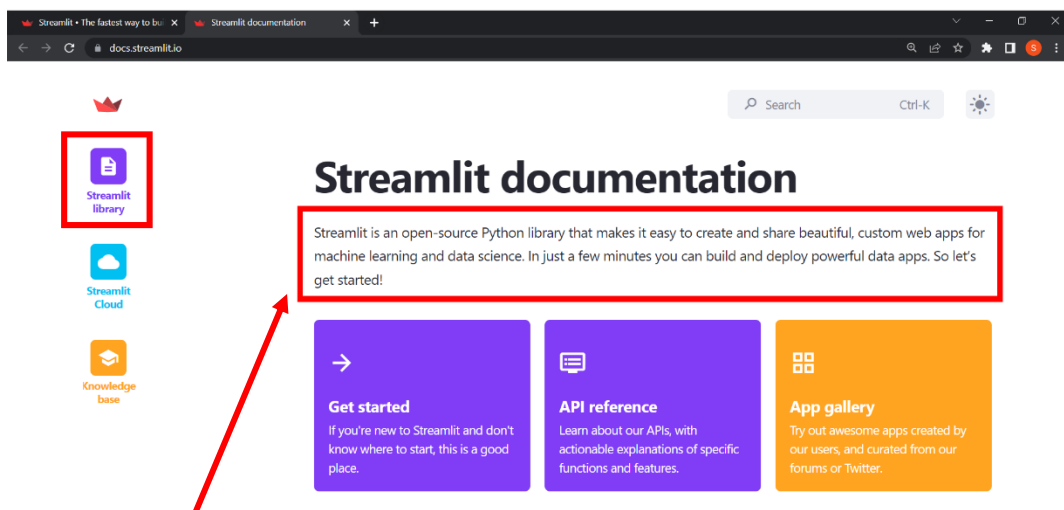
Try Streamlit now

Sign up for Streamlit Cloud

Streamlit เป็นเครื่องมือสำหรับสร้างเว็บแอปพลิเคชัน (web applications) ในภาษา Python อย่างง่าย โดยมีการสร้างอินเทอร์เฟซผู้ใช้แบบสตรีมไลน์

Streamlit

ที่หน้า Streamlit Documentation ให้กดที่เมนู Streamlit library



Streamlit เป็นไลบรารี Python โอเพนซอร์สที่ทำให้ง่ายต่อการสร้างและแบ่งปันเว็บแอปพลิเคชันที่สวยงามและที่ปรับแต่งได้ตามต้องการ

Streamlit

ที่ API reference แสดงคำสั่งต่าง ๆ สำหรับใช้ในการสร้าง Web Application

Documentation

Streamlit library

Get started (+)

- API reference (selected)
- Write and magic (+)
- Text elements (+)
- Data display elements (+)
- Chart elements (+)
- Input widgets (+)
- Media elements (+)
- Layouts and containers (+)
- Status elements (+)
- Control flow (+)
- Utilities (+)
- Mutate charts

API reference

Streamlit makes it easy for you to visualize, mutate, and share data. The API reference is organized by activity type, like displaying data or optimizing performance. Each section includes methods associated with the activity type, including examples.

Browse our API below and click to learn more about any of our available commands! 🔍

Display almost anything

st.write

Write arguments to the app.

```
st.write("Hello **world**!")
st.write(my_data_frame)
st.write(my_mpl_figure)
```

Magic

Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using `st.write`.

```
"Hello **world**!"
my_data_frame
my_mpl_figure
```

42

Write and magic

Documentation

Streamlit library

Get started (+)

- API reference
- Write and magic (selected)
- st.write
- magic
- Text elements (+)
- Data display elements (+)
- Chart elements (+)
- Input widgets (+)
- Media elements (+)
- Layouts and containers (+)
- Status elements (+)
- Control flow (+)
- Utilities (+)
- Mutate charts
- State management
- Performance (+)

st.write and magic commands

Streamlit has two easy ways to display information into your app, which should typically be the first thing you try: `st.write` and magic.

st.write

Write arguments to the app.

```
st.write("Hello **world**!")
st.write(my_data_frame)
st.write(my_mpl_figure)
```

Magic

Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using `st.write`.

```
"Hello **world**!"
my_data_frame
my_mpl_figure
```

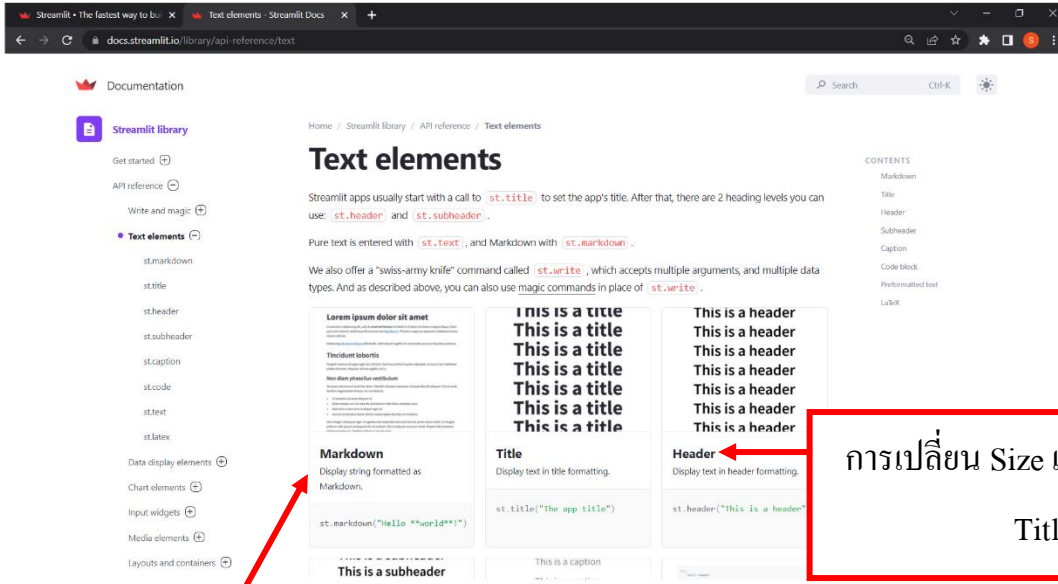
Was this page helpful? Yes No [Suggest edits](#)

Still have questions?

การแสดงผลข้อความเป็น Streamlit.write ก็จะเป็นการนำค่าจากตัวแปรขึ้นจอที่หน้าเว็บไซต์

43

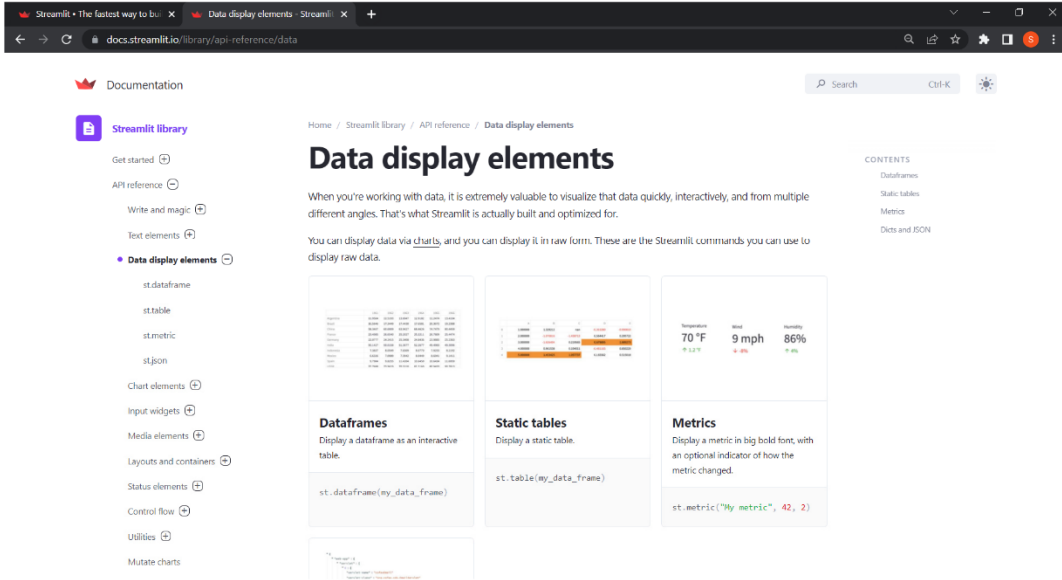
Text elements



การเปลี่ยนสี, เปลี่ยนการจัดการหน้า ชิดซ้าย ชิดขวา อยู่ตรงกลางจอ จะใช้คำสั่งที่ชื่อ (Markdown)

การเปลี่ยน Size เปลี่ยนขนาดตัวหนังสือ
Title, Header

Data display elements



การแสดงผลหน้าจอ: ก็จะมีการแสดงผลแบบตาราง แสดงผลค่าสถิติ แสดงผลปบ Metrics

Chart elements

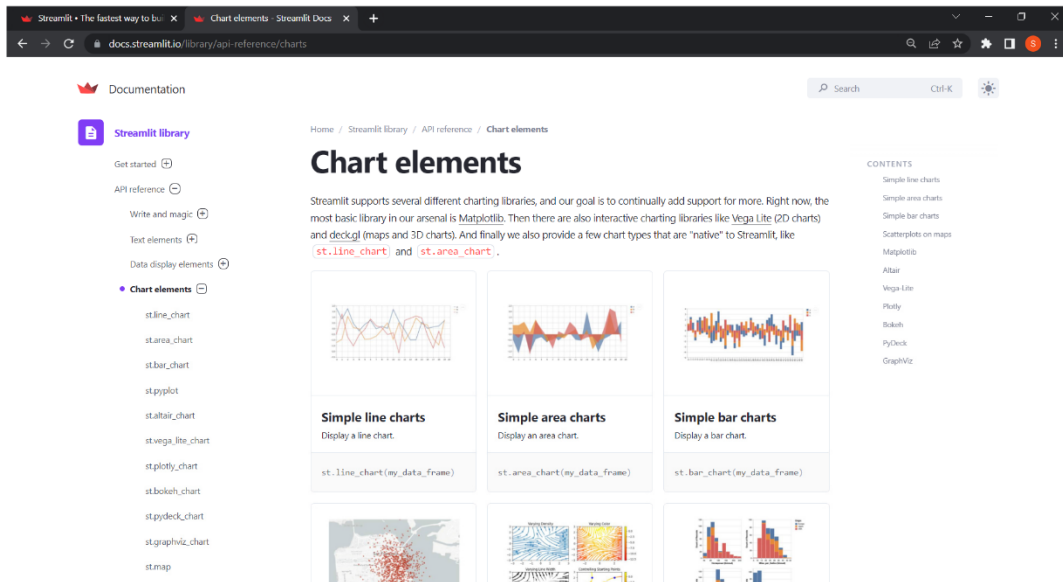
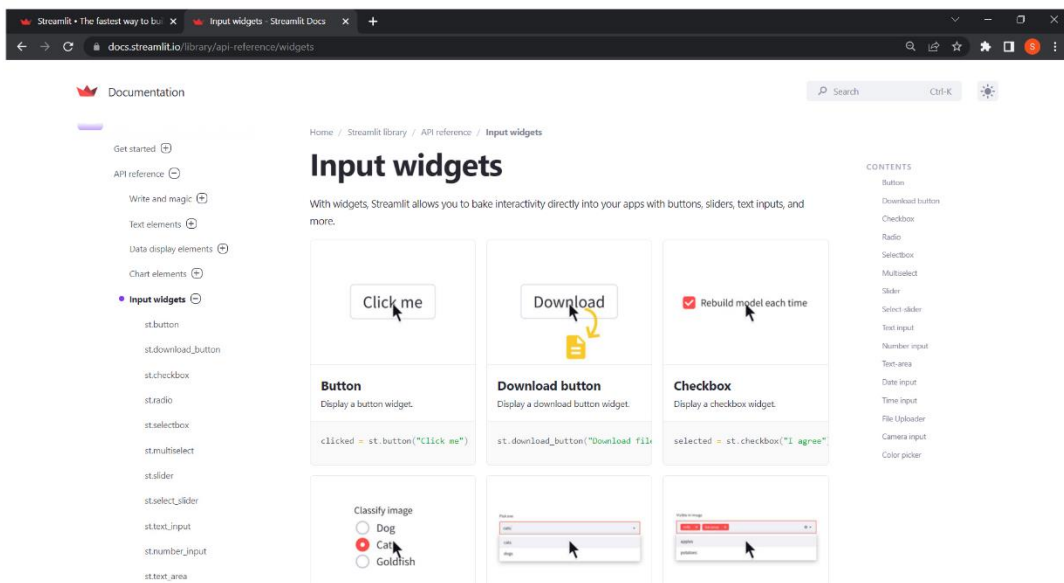


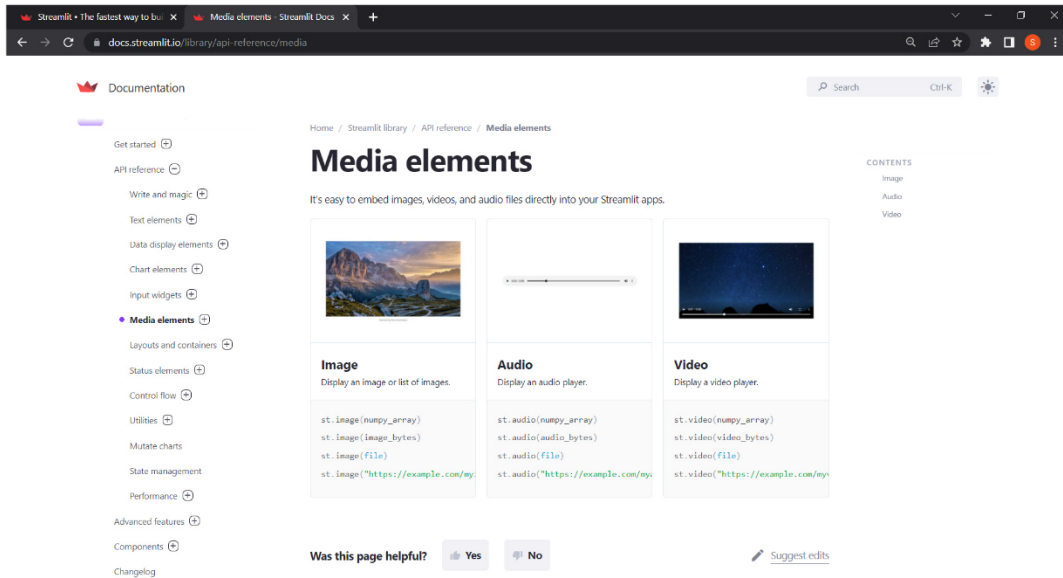
Chart สามารถ Port Graph ได้หลายกราฟ หลายแบบ เราสามารถ Port Graph ที่หน้าเว็บไซค์ได้หลายแบบ

Input widgets



Input = ก็จะมีการรับค่าจากการกดปุ่ม, Checkbox (ช่องทำเครื่องหมาย), radio button (ปุ่มตัวเลือก), Read box (กล่องอ่าน), Multi List box (กล่องรายการหลายรายการ)

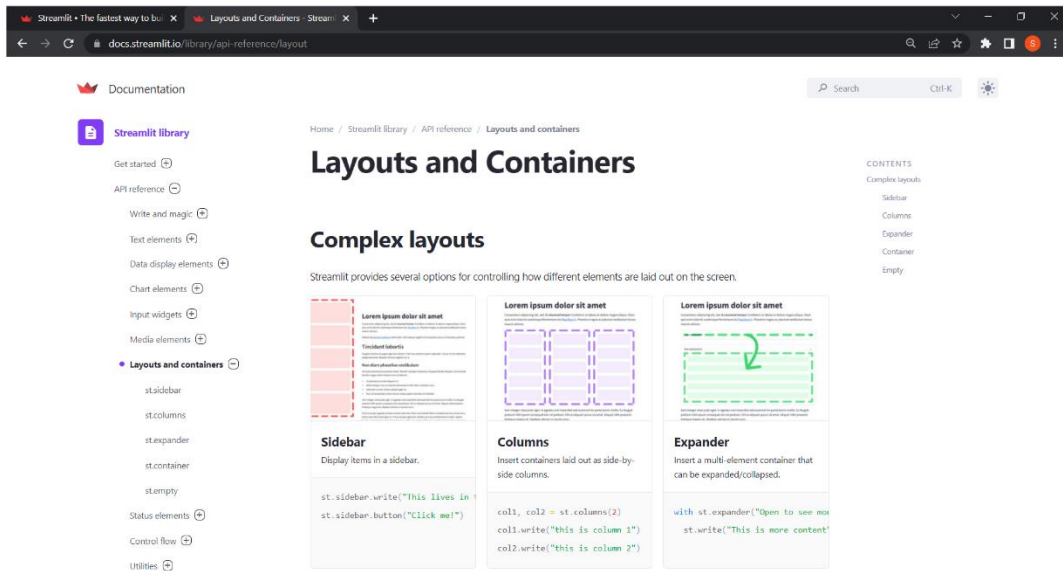
Media elements



The screenshot shows the Streamlit documentation page for 'Media elements'. The page title is 'Media elements' and the subtitle is 'It's easy to embed images, videos, and audio files directly into your Streamlit apps.' There are three columns of content: 'Image', 'Audio', and 'Video'. Each column shows a visual example and the corresponding code snippet. The 'Image' section shows a landscape photo and code for `st.image()`. The 'Audio' section shows a progress bar and code for `st.audio()`. The 'Video' section shows a video player and code for `st.video()`. A sidebar on the left lists navigation options, and a 'Was this page helpful?' feedback form is at the bottom.

จัดการเกี่ยวกับภาพที่เราจะใช้กัน คือ **Image** รวมถึง Audio และ Video ด้วยในการประมวลผลเสร็จแล้วไปขึ้นหน้าจอ

Layouts and Containers



The screenshot shows the Streamlit documentation page for 'Layouts and Containers'. The page title is 'Layouts and Containers' and the subtitle is 'Complex layouts'. The page explains that Streamlit provides several options for controlling how different elements are laid out on the screen. There are three columns of content: 'Sidebar', 'Columns', and 'Expander'. Each column shows a visual example and the corresponding code snippet. The 'Sidebar' section shows a sidebar with a button and code for `st.sidebar.write()` and `st.sidebar.button()`. The 'Columns' section shows two columns of text and code for `st.columns()`. The 'Expander' section shows an expandable container and code for `st.expander()` and `st.write()`. A sidebar on the left lists navigation options, and a 'Was this page helpful?' feedback form is at the bottom.

การจัดการคอลัมน์ = การจัดนำข้อมูลมาจัดเป็นคอลัมน์ 3 คอลัมน์ 4คอลัมน์ ก็ทำได้

Display progress and status

The screenshot shows the Streamlit documentation page for 'Display progress and status'. The page title is 'Display progress and status' and the subtitle is 'Streamlit provides a few methods that allow you to add animation to your apps. These animations include progress bars, status messages (like warnings), and celebratory balloons.' The page is divided into three columns, each showing a different status element with a visual example and a code snippet.

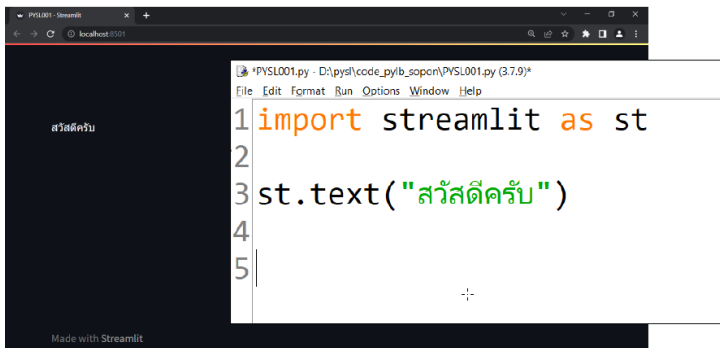
Progress bar	Spinner	Balloons
Display a progress bar.	Temporarily displays a message while executing a block of code.	Display celebratory balloons!
<pre>for i in range(101): st.progress(i) do_something_slow()</pre>	<pre>with st.spinner("Please wait..."): do_something_slow()</pre>	<pre>do_something() # Celebrate when all done! st.balloons()</pre>

Progress กรณีที่เราประมวลผลอยู่ก็จะมีแถบเลื่อนให้ หรือจะโชว์เป็น
ลักษณะนาฬิกา หรือจะแสดงความยินดีเป็นลูกโป่ง หรือบอลลูก

การเรียกใช้ฟังก์ชัน และคำสั่งต่างๆ ของ Library Streamlit

ตัวอย่างที่ 1 การแสดงข้อความ ด้วย st.text (PYSL001.py)

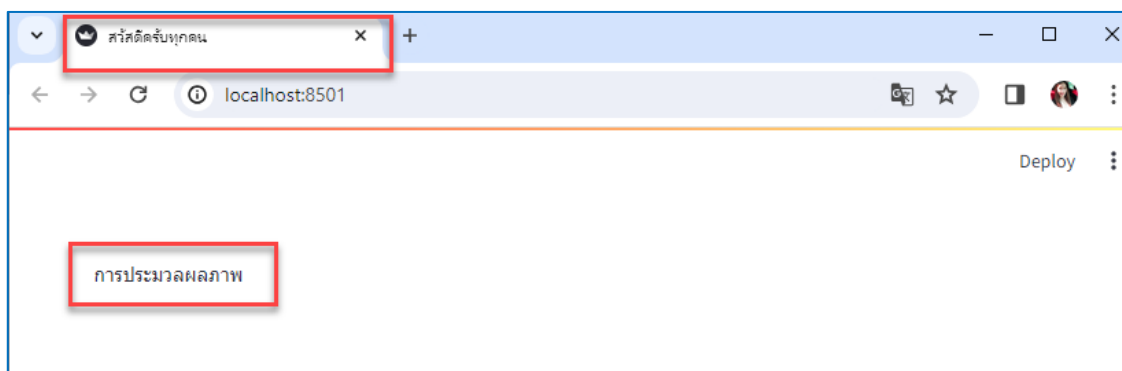
```
PYSL001.py
1 import streamlit as st
2
3 st.text("สวัสดีครับ/สวัสดีค่ะ")
```



- ❖ คำสั่งนี้ใช้ฟังก์ชัน text ของ Streamlit เพื่อแสดงข้อความธรรมดาบนหน้าเว็บแอปพลิเคชัน
- ❖ import streamlit as st : การนำเข้าของไลบรารี Streamlit และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้ชื่อ st.

ตัวอย่างที่ 2 การกำหนด title ของหน้าเว็บเพจ

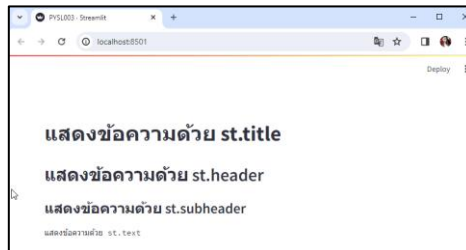
```
PYSL002.py
1 import streamlit as st
2
3 st.set_page_config(page_title="สวัสดีครับทุกคน")
4
5 st.text("การประมวลผลภาพ")
6
```



- ❖ ฟังก์ชัน set_page_config กำหนดการตั้งค่าชื่อเพจหรือหัวข้อหน้าเว็บในแท็บของเบราว์เซอร์

ตัวอย่างที่ 3 การแสดงข้อความ กำหนดขนาดข้อความ) ด้วย st.title , st.header และ st.subheader

```
PYSL003.py
1 import streamlit as st
2
3 st.title("แสดงข้อความด้วย st.title")
4 st.header("แสดงข้อความด้วย st.header")
5 st.subheader("แสดงข้อความด้วย st.subheader")
6 st.text("แสดงข้อความด้วย st.text")
```



❖ หัวข้อใหญ่ / หัวข้อรอง / หัวข้อย่อย / ข้อความธรรมดา

ตัวอย่างที่ 4 การทำตัวอักษรให้เป็น ตัวหนา ตัวเอียง และขีดเส้นใต้ ด้วย st.markdown

```
PYSL007.py > ...
1 import streamlit as st
2
3 text1 = "ข้อความที่ 1 ตัวปกติ"
4 st.markdown(f'<p>{text1}</p>', unsafe_allow_html=True)
5
6 text2 = "ข้อความที่ 2 ตัวหนา"
7 st.markdown(f'<b>{text2}</b>', unsafe_allow_html=True)
8
9 text3 = "ข้อความที่ 3 ตัวเอียง"
10 st.markdown(f'<i>{text3}</i>', unsafe_allow_html=True)
11
12 text4 = "ข้อความที่ 4 ขีดเส้นใต้"
13 st.markdown(f'<u>{text4}</u>', unsafe_allow_html=True)
14
15 text5 = "ข้อความที่ 5 ตัวหนาและขีดเส้นใต้"
16 st.markdown(f'<u><b>{text5}</b></u>', unsafe_allow_html=True)
17
18 text6 = "ข้อความที่ 6 ตัวเอียงตัวหนาและขีดเส้นใต้"
19 st.markdown(f'<u><b><i>{text6}</i></b></u>', unsafe_allow_html=True)
```


ข้อความที่ 1 ตัวปกติ

ข้อความที่ 2 ตัวหนา

ข้อความที่ 3 ตัวเอียง

ข้อความที่ 4 ขีดเส้นใต้

ข้อความที่ 5 ตัวหนาและขีดเส้นใต้

ข้อความที่ 6 ตัวเอียงตัวหนาและขีดเส้นใต้

❖ `import streamlit as st` : การนำเข้าของไลบรารี Streamlit และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้ชื่อ `st`.

`f` คือ `f-string` หรือ "**Formatted String Literals**" ซึ่งเป็นวิธีการจัดรูปแบบสตริงใน Python

❖ ฟังก์ชัน `st.markdown` คือ ใช้เพื่อแสดงข้อความที่มีการกำหนดรูปแบบด้วย HTML/CSS

`unsafe_allow_html=True` อนุญาตให้ใช้ HTML ภายใน `st.markdown`

ข้อความที่ 1: แสดง "ข้อความที่ 1 ตัวปกติ" ในรูปแบบปกติ.

ข้อความที่ 2: แสดง "ข้อความที่ 2 ตัวหนา" ในรูปแบบตัวหนา. HTML `...` ใช้สำหรับตัวหนา.

ข้อความที่ 3: แสดง "ข้อความที่ 3 ตัวเอียง" ในรูปแบบตัวเอียง. HTML `<i>...</i>` ใช้สำหรับตัวเอียง.

ข้อความที่ 4: แสดง "ข้อความที่ 4 ขีดเส้นใต้" ในรูปแบบขีดเส้นใต้. HTML `<u>...</u>` ใช้สำหรับขีดเส้นใต้.

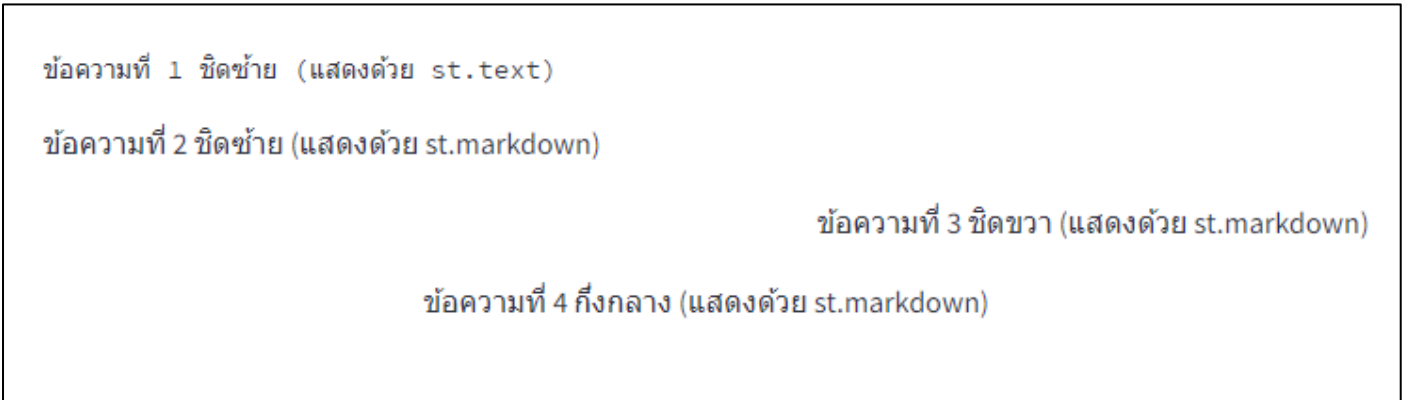
ข้อความที่ 5: แสดง "ข้อความที่ 5 ตัวหนาและขีดเส้นใต้" ในรูปแบบทั้งตัวหนาและขีดเส้นใต้.

ข้อความที่ 6: แสดง "ข้อความที่ 6 ตัวเอียงตัวหนาและขีดเส้นใต้" ในรูปแบบที่รวมตัวเอียง, ตัวหนา และขีดเส้นใต้.

โค้ดนี้ใช้ HTML เพื่อกำหนดรูปแบบของข้อความที่แสดงบนเว็บ. มันช่วยให้สามารถควบคุมรูปแบบข้อความได้มากกว่าที่ `st.text` ทำได้.

ตัวอย่างที่ 5 การทำให้ตัวอักษรอยู่ชิดซ้าย ชิดขวา และกึ่งกลาง ด้วย st.markdown แสดงข้อความที่มีการกำหนดรูปแบบด้วย HTML/CSS

```
PYSL008.py > ...
1 import streamlit as st
2
3 text1 = "ข้อความที่ 1 ชิดซ้าย (แสดงด้วย st.text)"
4 st.text(text1)
5
6 text2 = "ข้อความที่ 2 ชิดซ้าย (แสดงด้วย st.markdown)"
7 st.markdown(f'<p style="text-align:left">{text2}</p>', unsafe_allow_html=True)
8
9 text3 = "ข้อความที่ 3 ชิดขวา (แสดงด้วย st.markdown)"
10 st.markdown(f'<p style="text-align:right">{text3}</p>', unsafe_allow_html=True)
11
12 text4 = "ข้อความที่ 4 กึ่งกลาง (แสดงด้วย st.markdown)"
13 st.markdown(f'<p style="text-align:center">{text4}</p>', unsafe_allow_html=True)
14
```



- ❖ **import streamlit as st** : การนำเข้าของไลบรารี Streamlit และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้ชื่อ st.
- ❖ ใช้ฟังก์ชัน text เพื่อแสดง text1 บนหน้าเว็บ st.text มักจะแสดงข้อความโดยการชิดซ้ายตามค่าเริ่มต้น

แสดงข้อความชิดซ้ายด้วย st.text:

- ❖ **text1 = "ข้อความที่ 1 ชิดซ้าย (แสดงด้วย st.text)"**: st.text จะแสดงข้อความในรูปแบบชิดซ้ายเป็นฟังก์ชัน text เพื่อแสดง text1 บนหน้าเว็บ st.text มักจะแสดงข้อความโดยการชิดซ้ายตามค่าเริ่มต้น

แสดงข้อความชิดซ้ายด้วย HTML ใน st.markdown:

- ❖ `text2 = "ข้อความที่ 2 ซิดซ้าย (แสดงด้วย st.markdown)":` กำหนดข้อความให้กับตัวแปร `text2`.
- ❖ `st.markdown(f'<p style="text-align:left">{text2}</p>', unsafe_allow_html=True):` แสดงข้อความ `text2` ในรูปแบบ HTML โดยใช้ฟังก์ชัน `st.markdown`. `unsafe_allow_html=True` อนุญาตให้ใช้ HTML ใน Markdown.

แสดงข้อความซิดขวาด้วย HTML ใน st.markdown:

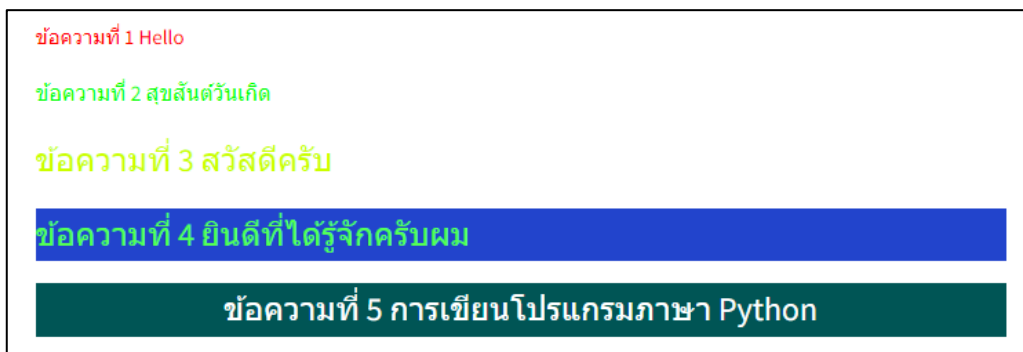
- ❖ `text3 = "ข้อความที่ 3 ซิดขวา (แสดงด้วย st.markdown)":` กำหนดข้อความให้กับตัวแปร `text3`.
- ❖ `st.markdown(f'<p style="text-align:right">{text3}</p>', unsafe_allow_html=True):` แสดงข้อความ `text3` ในรูปแบบ HTML โดยใช้ `st.markdown` และตั้งค่าการจัดตำแหน่งข้อความเป็นซิดขวา.

แสดงข้อความกึ่งกลางด้วย HTML ใน st.markdown:

- ❖ `text4 = "ข้อความที่ 4 กึ่งกลาง (แสดงด้วย st.markdown)":` กำหนดข้อความให้กับตัวแปร `text4`.
- ❖ `st.markdown(f'<p style="text-align:center">{text4}</p>', unsafe_allow_html=True):` แสดงข้อความ `text4` ในรูปแบบ HTML โดยใช้ `st.markdown` และตั้งค่าการจัดตำแหน่งข้อความเป็นกึ่งกลาง.
- ❖ `style="text-align` ช่วยกำหนดการจัดเรียงข้อความภายในแท็ก
- ❖ `unsafe_allow_html=True:` อนุญาตให้ใช้ HTML ในฟังก์ชัน `st.markdown`

ตัวอย่างที่ 6 การเปลี่ยนสีและขนาดของตัวอักษร ด้วย st.markdown (ใช้เพื่อแสดงข้อความที่มีการกำหนดรูปแบบด้วย HTML/CSS)

```
PYSL009.py > ...
1 import streamlit as st
2
3 text1 = "ข้อความที่ 1 Hello"
4 st.markdown(f'<p style="color:#ff0000;font-size:15px;">{text1}</p>', unsafe_allow_html=True)
5
6 text2 = "ข้อความที่ 2 สุขสันต์วันเกิด"
7 st.markdown(f'<p style="color:#00ff00;font-size:100%;">{text2}</p>', unsafe_allow_html=True)
8
9 text3 = "ข้อความที่ 3 สวัสดีครับ"
10 st.markdown(f'<p style="color:#ccff00;font-size:150%;">{text3}</p>', unsafe_allow_html=True)
11
12 text4 = "ข้อความที่ 4 ยินดีที่ได้รู้จักครับผม"
13 st.markdown(f'<p style="background-color:#2244cc;color:#55ff55;font-size:24px;">{text4}</p>', unsafe_allow_html=True)
14
15 text5 = "ข้อความที่ 5 การเขียนโปรแกรมภาษา Python"
16 st.markdown(f'<p style="background-color:#005555;color:#ffffff;font-size:150%;text-align:center;">{text5}</p>', unsafe_allow_html=True)
17
```



- ❖ **import streamlit as st** : การนำเข้าของไลบรารี Streamlit และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้ชื่อ st.
- ❖ **color, font-size, background-color, และ text-align** ใน style ช่วยให้สามารถกำหนดสี, ขนาดฟอนต์, สีพื้นหลัง, และการจัดวางข้อความได้ตามต้องการในรูปแบบของ HTML
- ❖ **unsafe_allow_html=True**: อนุญาตให้ใช้ HTML ในฟังก์ชัน st.markdown

ตัวอย่างที่ 7 การแสดงข้อความแบบ 2 ข้อความ จากค่าในตัวแปร

```
PYSL015.py > ...
1  import streamlit as st
2  from time import sleep
3
4  text_out1 = st.empty()
5  text_out2 = st.empty()
6
7  x = ["แอปเปิ้ล", "มะละกอ", "กล้วย", "ส้ม"]
8  y = ["ราดหน้า", "ข้าวผัด", "ไข่เจียว", "หมูทอด"]
9
10 for i in range(0,4): # i = 0 1 2 3
11     text_out1.text("แสดงชื่อผลไม้ : " + x[i])
12     text_out2.text("แสดงชื่ออาหาร : " + y[i])
13     sleep(2)
14
15 st.text("แสดงข้อมูลครบถ้วนแล้ว")
```

```
แสดงชื่อผลไม้ : ส้ม
แสดงชื่ออาหาร : หมูทอด
แสดงข้อมูลครบถ้วนแล้ว
```

- ❖ โค้ดนี้ใช้ Streamlit และ โมดูล time ของ Python เพื่อแสดงข้อความที่อัปเดตตามลำดับในหน้าเว็บแอป
- ❖ `import streamlit as st`: นำเข้าไลบรารี Streamlit. และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้ชื่อ st.
- ❖ `from time import sleep`: นำเข้าฟังก์ชัน sleep จากโมดูล time
- ❖ `text_out1 = st.empty()`: สร้างอ็อบเจกต์ว่างที่จะใช้สำหรับแสดงข้อความที่ 1.
- ❖ `text_out2 = st.empty()`: สร้างอ็อบเจกต์ว่างที่จะใช้สำหรับแสดงข้อความที่ 2.
- ❖ `x = ["แอปเปิ้ล", "มะละกอ", "กล้วย", "ส้ม"]`: สร้างรายการ x ที่ประกอบด้วยชื่อผลไม้.
- ❖ `y = ["ราดหน้า", "ข้าวผัด", "ไข่เจียว", "หมูทอด"]`: สร้างรายการ y ที่ประกอบด้วยชื่ออาหาร.
- ❖ `for i in range(0,4)`: สร้างลูปที่ทำงานสี่รอบ, โดย i จะมีค่าตั้งแต่ 0 ถึง 3.
- ❖ `text_out1.text("แสดงชื่อผลไม้ : " + x[i])`: อัปเดตข้อความใน text_out1 ด้วยชื่อผลไม้จากรายการ x.
- ❖ `text_out2.text("แสดงชื่ออาหาร : " + y[i])`: อัปเดตข้อความใน text_out2 ด้วยชื่ออาหารจากรายการ y.

❖ `sleep(2)`: หน่วงเวลา 2 วินาทีก่อนที่จะทำรอบถัดไปของลูป

แสดงข้อความท้ายสุด:

❖ `st.text("แสดงข้อมูลครบถ้วนแล้ว")`

ตัวอย่างที่ 8 การรับข้อความอินพุตด้วย `st.text_input`

```
PYSL020.py > ...
1  import streamlit as st
2
3  #name = st.text_input("กรุณาป้อนชื่อ : ", "พิมพ์ตรงนี้นะครับ")
4  name = st.text_input("กรุณาป้อน ชื่อ : ")
5  surname = st.text_input("กรุณาป้อน นามสกุล : ")
6  st.text("สวัสดีครับ คุณ " + name + ' ' + surname)
7
8
```

กรรณาป้อน ชื่อ :

กรรณาป้อน นามสกุล :
สวัสดีครับ คุณ พิศุทธา ใจธรรม

❖ `import streamlit as st`: นำเข้าไลบรารี Streamlit. และให้เรียกใช้งานฟังก์ชันต่างๆ ภายใต้อชื่อ `st`.

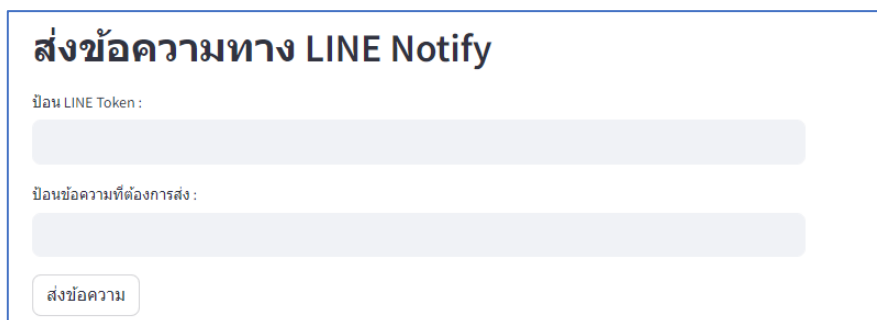
สร้างฟิลด์ป้อนข้อมูลต่างๆ:

- ❖ `name = st.text_input("กรุณาป้อน ชื่อ : ")`: สร้างฟิลด์ป้อนข้อมูล (text input) ที่ให้ผู้ใช้ป้อนชื่อ ค่าที่ผู้ใช้ป้อนจะถูกเก็บในตัวแปร `name`.
- ❖ `surname = st.text_input("กรุณาป้อน นามสกุล : ")`: สร้างฟิลด์ป้อนข้อมูลอีกอันหนึ่งที่ให้ผู้ใช้ป้อนนามสกุล. ค่าที่ผู้ใช้ป้อนจะถูกเก็บในตัวแปร `surname`.
- ❖ `st.text("สวัสดีครับ คุณ " + name + ' ' + surname)`: คำสั่งนี้ใช้เพื่อแสดงข้อความที่รวมชื่อและนามสกุลที่ผู้ใช้ป้อนไว้

*** โค้ดนี้มีประโยชน์ในการสร้างฟอร์มหรือแบบสอบถามง่ายๆ บนเว็บแอปพลิเคชัน Streamlit ซึ่งผู้ใช้สามารถป้อนข้อมูลและได้รับการตอบกลับแบบโต้ตอบบนหน้าเดียวกัน.

ตัวอย่างที่ ๑ การส่งข้อความทาง LINE Notify

```
1 import streamlit as st
2 import line_api
3
4 st.header("ส่งข้อความทาง LINE Notify")
5 mytoken = st.text_input("ป้อน LINE Token : ")
6 text_send = st.text_input("ป้อนข้อความที่ต้องการส่ง : ")
7
8 if st.button("ส่งข้อความ"):
9     try:
10         line_api.lineNotify(text_send,mytoken)
11         st.text("ส่งข้อความของคุณแล้ว")
12     except:
13         st.text("ไม่สามารถส่งข้อความได้")
```



The screenshot shows a web application titled "ส่งข้อความทาง LINE Notify". It features two text input fields: the first is labeled "ป้อน LINE Token :" and the second is labeled "ป้อนข้อความที่ต้องการส่ง :". Below these fields is a button labeled "ส่งข้อความ".

- ❖ **import streamlit as st:** นำเข้าไลบรารี Streamlit
- ❖ **import line_api:** นำเข้าไลบรารีที่สมมติว่าชื่อ line_api, ซึ่งคาดว่าให้ความสามารถในการโต้ตอบกับ LINE Notify.

สร้างหัวข้อหลักบนหน้าเว็บ:

- ❖ **st.header("ส่งข้อความทาง LINE Notify"):** ใช้ฟังก์ชัน header เพื่อแสดงหัวข้อ "ส่งข้อความทาง LINE Notify" บนหน้าเว็บ.

รับข้อมูลจากผู้ใช้:

- ❖ `mytoken = st.text_input("ป้อน LINE Token : ")`: สร้างช่องใส่ข้อมูลสำหรับผู้ใส่ LINE Token.
- ❖ `text_send = st.text_input("ป้อนข้อความที่ต้องการส่ง : ")`: สร้างช่องใส่ข้อมูลสำหรับผู้ใส่ข้อความที่ต้องการส่ง.

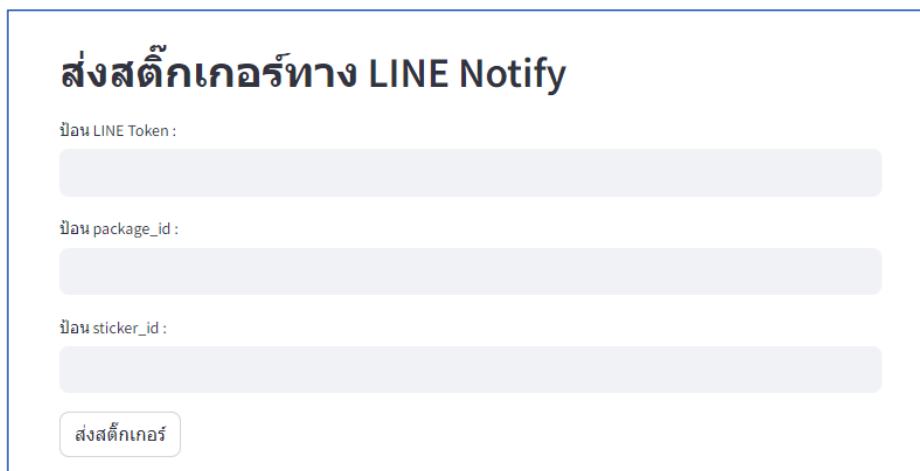
ปุ่มกดสำหรับส่งข้อความและตรวจสอบการดำเนินการ:

- ❖ `if st.button("ส่งข้อความ")`: ตรวจสอบว่าปุ่ม "ส่งข้อความ" ถูกกดหรือไม่.
- ❖ `try`: บล็อก `try` ใช้สำหรับเรียกใช้งานที่อาจเกิดข้อผิดพลาด.
- ❖ `line_api.lineNotify(text_send,mytoken)`: เรียกใช้ฟังก์ชัน `lineNotify` จากไลบรารี `line_api` พร้อมข้อความที่ต้องการส่งและ `token`.
- ❖ `st.text("ส่งข้อความของคุณแล้ว")`: ถ้าไม่มีข้อผิดพลาด, แสดงข้อความนี้บนหน้าเว็บ.
- ❖ `except`: ถ้าเกิดข้อผิดพลาดในบล็อก `try`, บล็อก `except` จะดำเนินการ.
- ❖ `st.text("ไม่สามารถส่งข้อความได้")`: แสดงข้อความนี้บนหน้าเว็บหากไม่สามารถส่งข้อความได้.

*** ผลลัพธ์คือเว็บแอปที่ผู้ใช้สามารถป้อน LINE Token และข้อความที่ต้องการส่ง และเมื่อกดปุ่ม "ส่งข้อความ", จะมีการพยายามส่งข้อความไปยัง LINE ผ่าน LINE Notify. ถ้าการส่งสำเร็จ, จะแสดงข้อความยืนยันบนหน้าเว็บ.

ตัวอย่างที่ 10 การส่งสติ๊กเกอร์ทาง LINE Notify

```
1 import streamlit as st
2 import line_api
3
4 st.header("ส่งสติ๊กเกอร์ทาง LINE Notify")
5 mytoken = st.text_input("ป้อน LINE Token : ")
6 package_id = st.text_input("ป้อน package_id : ")
7 sticker_id = st.text_input("ป้อน sticker_id : ")
8
9 if st.button("ส่งสติ๊กเกอร์"):
10     try:
11         line_api.notifySticker(int(sticker_id),int(package_id),mytoken)
12         st.text("ส่งสติ๊กเกอร์ของคุณแล้ว")
13     except:
14         st.text("ไม่สามารถส่งสติ๊กเกอร์ได้")
15
```



- ❖ `import streamlit as st`: นำเข้าไลบรารี Streamlit
- ❖ `import line_api`: นำเข้าไลบรารีที่สมมติว่าชื่อ `line_api`. สมมติว่าไลบรารีนี้มีฟังก์ชันที่ใช้สำหรับส่งสติ๊กเกอร์ผ่าน LINE Notify.

สร้างหัวข้อบนหน้าเว็บ:

- ❖ `st.header("ส่งสติ๊กเกอร์ทาง LINE Notify")`: ใช้ `header` เพื่อแสดงหัวข้อ "ส่งสติ๊กเกอร์ทาง LINE Notify" บนหน้าเว็บ.

รับข้อมูลจากผู้ใช้:

- ❖ `mytoken = st.text_input("ป้อน LINE Token : ")`: สร้างช่องใส่ข้อมูลสำหรับผู้ใช้ใส่ LINE Token.
- ❖ `package_id = st.text_input("ป้อน package_id : ")`: ช่องใส่ข้อมูลสำหรับผู้ใช้ใส่ package_id ของสติ๊กเกอร์.
- ❖ `sticker_id = st.text_input("ป้อน sticker_id : ")`: ช่องใส่ข้อมูลสำหรับผู้ใช้ใส่ sticker_id ของสติ๊กเกอร์.

ปุ่มกดสำหรับส่งสติ๊กเกอร์และตรวจสอบการดำเนินการ:

- ❖ `if st.button("ส่งสติ๊กเกอร์")`: ตรวจสอบว่าปุ่ม "ส่งสติ๊กเกอร์" ถูกกดหรือไม่.
- ❖ `try`: บล็อก `try` ใช้สำหรับเรียกใช้งานที่อาจเกิดข้อผิดพลาด.
- ❖ `line_api.notifySticker(int(sticker_id), int(package_id), mytoken)`: เรียกใช้ฟังก์ชัน `notifySticker` จากไลบรารี `line_api` ด้วย `sticker_id`, `package_id` ที่แปลงเป็นจำนวนเต็ม, และ `token`.
- ❖ `st.text("ส่งสติ๊กเกอร์ของคุณแล้ว")`: ถ้าไม่มีข้อผิดพลาด, แสดงข้อความนี้บนหน้าเว็บ.
- ❖ `except`: ถ้าเกิดข้อผิดพลาดในบล็อก `try`, บล็อก `except` จะดำเนินการ.
- ❖ `st.text("ไม่สามารถส่งสติ๊กเกอร์ได้")`: แสดงข้อความนี้บนหน้าเว็บหากไม่สามารถส่งสติ๊กเกอร์ได้.

*** ผลลัพธ์คือเว็บแอปที่ให้ผู้ใช้ป้อน LINE Token และรายละเอียดของสติ๊กเกอร์ (`package_id` และ `sticker_id`) และสามารถส่งสติ๊กเกอร์ผ่าน LINE Notify โดยกดปุ่ม "ส่งสติ๊กเกอร์".

ตัวอย่างที่ 1 การแสดงภาพนิ่งด้วย St.image

PYSL042.py - D:\WDO WepApp\code_pycvwebapp2\PYSL042.py (3.7.9)

File Edit Format Run Options Window Help

```
1 import streamlit as st
2 import cv2
3
4 img = cv2.imread("zebra.jpg")
5
6 st.image(img, caption='ม้าลาย', channels="BGR") #OpenCV BGR  RGB
7
```



ม้าลาย

ตัวอย่างที่ 2 การแสดงภาพนิ่งพร้อมด้วยความกว้างและความสูง

```
PYSL043.py - D:\VDO WepApp\code_pycvwebapp2\PYSL043.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3
4 img = cv2.imread("bird.jpg")
5 (h,w,c) = img.shape
6
7 st.image(img, caption="นก", channels="BGR")
8 st.text("ภาพนี้ กว้าง " + str(w) + " สูง " + str(h))
9
```



นก

ภาพนี้ กว้าง 400 สูง 267

ตัวอย่างที่3 การแสดงภาพนิ่งแบบ 3 คอลัมน์ด้วยst.columns

PYSL044.py - D:\VDO WepApp\code_pycvwebapp2\PYSL044.py (3.7.9)

File Edit Format Run Options Window Help

```
1 import streamlit as st
2 import cv2
3
4 img1 = cv2.imread("zebra.jpg")
5 img2 = cv2.imread("duck.jpg")
6 img3 = cv2.imread("bird.jpg")
7
8 col1, col2, col3 = st.columns(3)
9
10 col1.image(img1, caption='ม้าลาย', channels="BGR")
11 col2.image(img2, caption='เป็ด', channels="BGR")
12 col3.image(img3, caption='นก', channels="BGR")
13
```



ม้าลาย



เป็ด



นก

ตัวอย่างที่4 การเปิดไฟล์ภาพนิ่งด้วย st.file_uploader

```
PYSL045.py - D:\WDO WepApp\code_pycvwebapp2\PYSL045.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3
4 img_file = st.file_uploader("เปิดไฟล์ภาพ")
5
6 if img_file is not None:
7     st.image(img_file,channels="BGR")
8
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

Browse files

car.jpg 30.1KB



ตัวอย่างที่ 5 การเปิดไฟล์ภาพนิ่งด้วย st.file_uploader แล้วแปลงให้อยู่ในรูปแบบของ OpenCV

```
PYSL046.py - D:\VDO WepApp\code_pycvwebapp2\PYSL046.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 img_file = st.file_uploader("เปิดไฟล์ภาพ")
6
7 if img_file is not None:
8     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
9     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
10    st.image(img ,caption="OpenCV Format",channels="BGR")
11
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

duck.jpg 13.2KB



OpenCV Format

ตัวอย่างที่ 6 การเปิดไฟล์ภาพหนึ่งด้วย st.file_uploader พร้อมแสดงชื่อไฟล์และขนาดภาพ

PYSL047.py - D:\VDO WepApp\code_pycvwebapp2\PYSL047.py (3.7.9)

File Edit Format Run Options Window Help

```
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 img_file = st.file_uploader("เปิดไฟล์ภาพ")
6
7 if img_file is not None:
8     print(img_file)
9     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
10    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
11    (h,w,c) = img.shape
12    st.image(img ,channels="BGR")
13    st.text("ชื่อไฟล์ : " + img_file.name)
14    st.text("ขนาดภาพ : " + str(w) + "," + str(h) + "," + str(c) )
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file [Browse files](#)

fw2.jpg 18.2KB



ชื่อไฟล์ : fw2.jpg

ขนาดภาพ : 240,240,3

ตัวอย่างที่ 7 การแปลงภาพ Color ไปเป็นภาพ Grayscale

```
PYSL048.py - D:\VDO WepApp\code_pycvwebapp2\PYSL048.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 st.subheader("แปลงภาพ Color ไปเป็นภาพ Grayscale")
6 img_file = st.file_uploader("เปิดไฟล์ภาพ")
7
8 if img_file is not None:
9     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
10    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
11    #-----
12    img_out = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
13    #-----
14    col1, col2 = st.columns(2)
15    col1.image(img, caption='ภาพ Color', channels="BGR")
16    col2.image(img_out, caption='ภาพ Grayscale')
```

แปลงภาพ Color ไปเป็นภาพ Grayscale

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

 fw1.jpg 24.9KB



ภาพ Color



ภาพ Grayscale

ตัวอย่างที่ 8 การแปลงไปเป็นภาพ Binary

```
PYSL049.py - D:\VDO WepApp\code_pycvwebapp2\PYSL049.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 st.subheader("แปลงภาพไปเป็น Binary")
6 img_file = st.file_uploader("เปิดไฟล์ภาพ")
7
8 if img_file is not None:
9     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
10    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
11    #-----
12    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
13    ret, BW = cv2.threshold(img_gray, 120, 255, cv2.THRESH_BINARY_INV)
14    #-----
15    col1, col2, col3 = st.columns(3)
16    col1.image(img, caption='ภาพ Color', channels="BGR")
17    col2.image(img_gray, caption='ภาพ Grayscale')
18    col3.image(BW, caption='ภาพ Binary')
```

แปลงภาพไปเป็น Binary

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

Browse files

padthai.jpg 35.4KB



ภาพ Color



ภาพ Grayscale



ภาพ Binary

ตัวอย่างที่ 9 การใส่ข้อความลงบนภาพ

```
PYSL052.py - D:\VDO WepApp\code_pycvwebapp2\PYSL052.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 img_file = st.file_uploader("เปิดไฟล์ภาพ")
6
7 if img_file is not None:
8     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
9     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
10     #-----
11     img_out = img.copy()
12     text = "Hello"
13     font = cv2.FONT_HERSHEY_SIMPLEX
14     cv2.putText(img_out, text, (20, 100), font, 3.0, (0, 255, 0), 3)
15     #-----
16     col1, col2 = st.columns(2)
17     col1.image(img, caption='ภาพ Input', channels="BGR")
18     col2.image(img_out, caption='ภาพ Output', channels="BGR")
19
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

Browse files

elephant.jpg 35.0KB



ภาพ Input



ภาพ Output

ตัวอย่างที่ 10 การใส่ข้อความลงบนภาพแบบป้อนข้อความได้

```
PYSL053.py - D:\VDO WepApp\code_pycvwebapp2\PYSL053.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 img_file = st.file_uploader("เปิดไฟล์ภาพ")
6
7 def put_text(img,text):
8     #-----
9     img_out = img.copy()
10    font = cv2.FONT_HERSHEY_SIMPLEX
11    cv2.putText(img_out,text,(20,100),font,1.5,(0,255,0),5)
12    #-----
13    col1, col2 = st.columns(2)
14    col1.image(img, caption='ภาพ Input',channels="BGR")
15    col2.image(img_out, caption='ภาพ Output',channels="BGR")
16
17 if img_file is not None:
18    file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
19    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
20    text_in = st.text_input("ป้อนข้อความ","Hello")
21    put_text(img,text_in)
22
23
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file Browse files

duck.jpg 13.2KB ×

ป้อนข้อความ

Hello



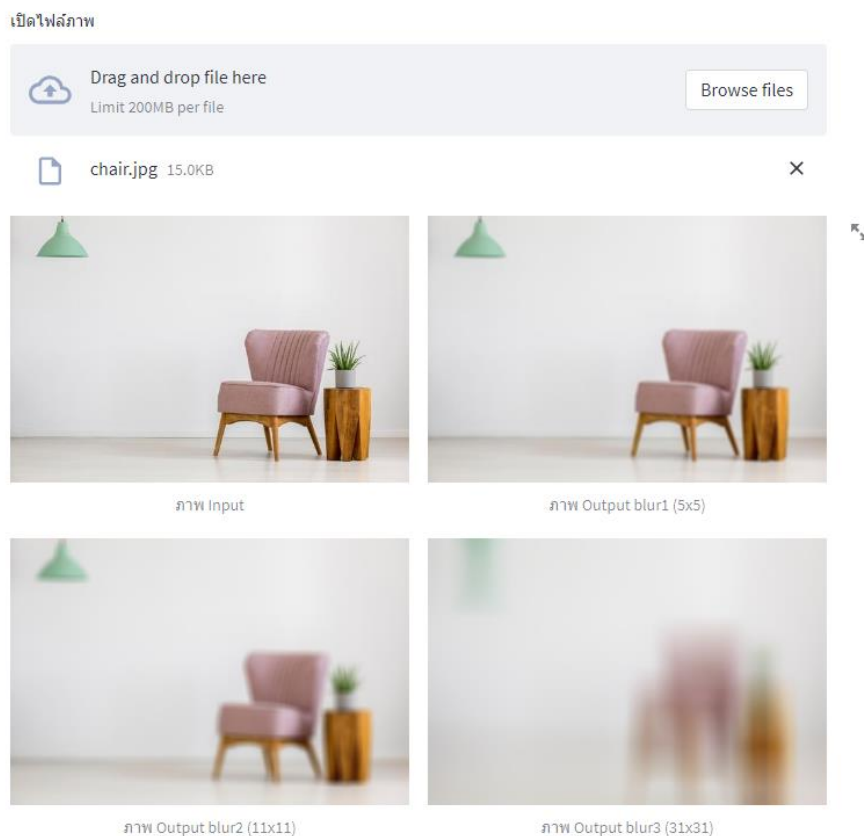
ภาพ Input



ภาพ Output

ตัวอย่างที่ 11 การเบลอภาพด้วย Average Filter

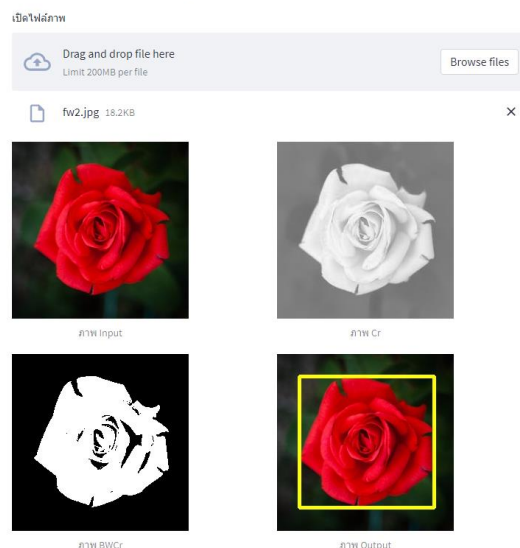
```
PYSL054.py - D:\VDO WepApp\code_pycvwebapp2\PYSL054.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 img_file = st.file_uploader("เปิดไฟล์ภาพ")
6
7 if img_file is not None:
8     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
9     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
10    #-----
11    img_blur1 = cv2.blur(img, (5,5))
12    img_blur2 = cv2.blur(img, (11,11))
13    img_blur3 = cv2.blur(img, (31,101))
14    #-----
15    col1, col2 = st.columns(2)
16    col1.image(img, caption='ภาพ Input',channels="BGR")
17    col2.image(img_blur1, caption='ภาพ Output blur1 (5x5)',channels="BGR")
18
19    col1, col2 = st.columns(2)
20    col1.image(img_blur2, caption='ภาพ Output blur2 (11x11)',channels="BGR")
21    col2.image(img_blur3, caption='ภาพ Output blur3 (31x31)',channels="BGR")
22
23
```



ตัวอย่างที่ 12 การตรวจจับวัตถุสีแดงในภาพนิ่งโดยใช้ Cr

```
PYSL057.py - D:\VDO WepApp\code_pycvwebapp2\PYSL057.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 st.title("ตรวจจับวัตถุสีแดง")
6 img_file = st.file_uploader("เปิดไฟล์ภาพ")
7
8 if img_file is not None:
9     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
10    img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
11    #-----
12    imgYCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCR_CB)
13
14    channels = cv2.split(imgYCrCb) # Y 0 Cr 1 Cb 2
15    Cr = channels[1]
16
17    ret, BW = cv2.threshold(Cr, 190, 255, cv2.THRESH_BINARY)
18
19    contours, hierarchy = cv2.findContours(BW, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIM
20
21    if len(contours) > 0: #เจอ
22        areas = [cv2.contourArea(c) for c in contours]
23        max_index = np.argmax(areas) #[ 20000 350000 456 ] 0 1 2
24        cnt = contours[max_index]
25
26        x, y, w, h = cv2.boundingRect(cnt)
27
28        img_out = img.copy()
29        cv2.rectangle(img_out, (x, y), (x+w, y+h), (0, 255, 255), 4) #BGR
30        #-----
31        col1, col2 = st.columns(2)
32        col1.image(img, caption='ภาพ Input', channels="BGR")
33        col2.image(Cr, caption='ภาพ Cr')
34
35        col1, col2 = st.columns(2)
36        col1.image(BW, caption='ภาพ BWCr')
37        col2.image(img_out, caption='ภาพ Output', channels="BGR")
38
39
40
```

ตรวจจับวัตถุสีแดง



ตัวอย่างที่ 13 การตรวจจับวัตถุโดยใช้ YOLO Model ex20obj.onnx กับภาพนิ่ง

```
PYSL059.py - D:\VDO WepApp\code_pycvwebapp2\PYSL059.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4 from yolo_predictions import YOLO_Pred
5
6 yolo = YOLO_Pred('ex20obj.onnx', 'ex20obj.yaml')
7
8 img_file = st.file_uploader("เปิดไฟล์ภาพ")
9
10 if img_file is not None:
11     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
12     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
13     #-----
14     pred_image, obj_box = yolo.predictions(img)
15     #-----
16     st.image(pred_image, caption='ภาพ Output', channels="BGR")
17
18
19
20
```

เปิดไฟล์ภาพ



Drag and drop file here
Limit 200MB per file

Browse files



cat_dog.jpg 25.3KB



ภาพ Output

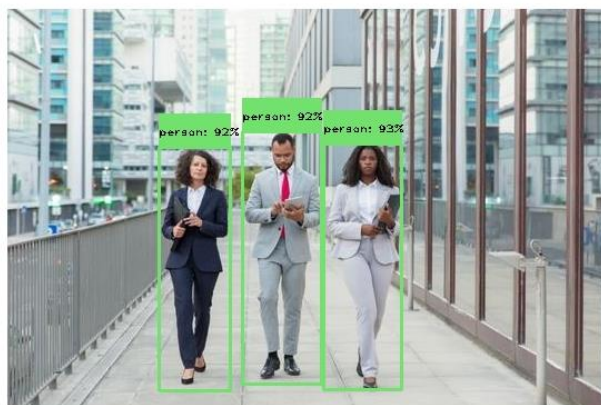
ตัวอย่างที่ 14 การตรวจจับวัตถุโดยใช้ YOLO Model ex20obj.onnx กับภาพนิ่ง พร้อมแสดงชื่อวัตถุ

```
PYSL060.py - D:\VDO WepApp\code_pycvwebapp2\PYSL060.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4 from yolo_predictions import YOLO_Pred
5
6 yolo = YOLO_Pred('ex20obj.onnx', 'ex20obj.yaml')
7
8 img_file = st.file_uploader("เปิดไฟล์ภาพ")
9
10 if img_file is not None:
11     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
12     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
13     #-----
14     count = 0  ***
15     pred_image, obj_box = yolo.predictions(img)
16     print(obj_box)
17     if len(obj_box) > 0: #เจอวัตถุใน 20 อย่างนั้น
18         obj_names = ''
19         for obj in obj_box:
20             obj_names = obj_names + obj[4] + ' '
21             if obj[4] == "person":  ***
22                 count = count + 1  ***
23         text_obj = 'ตรวจพบ ' + obj_names
24     else:
25         text_obj = 'ไม่พบวัตถุ'
26     #-----
27     st.image(pred_image, caption='ภาพ Output', channels="BGR")
28     st.text(text_obj)
29     st.text("พบ person " + str(count) + " คน")  ***
30
31
32
```

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file Browse files

person.jpg 39.1KB ×



ภาพ Output

ตรวจพบ person person person

พบ person 3 คน

ตัวอย่างที่15 การจำแนกภาพขยะ Recycle ได้แก่ cardboard, glass, metal, plastic โดยใช้ CNN Model recycle.h5 กับภาพหนึ่ง

```
PYSL062.py - D:\VDO WepApp\code_pycvwebapp2\PYSL062.py (3.7.9)
File Edit Format Run Options Window Help
1 import streamlit as st
2 import cv2
3 import numpy as np
4
5 from tensorflow.keras.preprocessing.image import load_img, img_to_array
6 from tensorflow.keras.models import load_model
7
8 from tensorflow.keras.applications.resnet50 import preprocess_input
9
10 model = load_model('recycle.h5', compile=False)
11
12 target_img_shape=(128,128)
13
14 st.subheader("การจำแนกภาพขยะ Recycle")
15 img_file = st.file_uploader("เปิดไฟล์ภาพ")
16
17 if img_file is not None:
18     file_bytes = np.asarray(bytearray(img_file.read()), dtype=np.uint8)
19     img = cv2.imdecode(file_bytes, cv2.IMREAD_COLOR)
20     #-----
21     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB )
22     test_image = cv2.resize(img,target_img_shape)
23
24     test_image = img_to_array(test_image)
25     test_image = preprocess_input(test_image)
26
27     test_image = np.expand_dims(test_image,axis=0) # (1, 128, 128, 3)
28
29     result = model.predict(test_image)
30     st.write(result)
31
32     class_answer = np.argmax(result,axis=1)
33     if class_answer == 0:
34         predict = 'cardboard'
35     elif class_answer == 1:
36         predict = 'glass'
37     elif class_answer == 2:
38         predict = 'metal'
39     elif class_answer == 3:
40         predict = 'plastic'
41     st.write("predict = "+predict)
42     #-----
43     st.image(img ,caption=predict,channels="RGB")
44
45
46
```

การจำแนกภาพขยะ Recycle

เปิดไฟล์ภาพ

Drag and drop file here
Limit 200MB per file

Browse files

tb3.jpg 14.9KB ×

0	1	2	3
0	0.0003	0.0002	0.9995

predict = plastic



plastic

การจัดการกล้อง Webcam

การเชื่อมต่อกล้อง Webcam

```
PYSL064.py
1 import streamlit as st
2 from streamlit_webrtc import webrtc_streamer
3
4 st.title("ทดสอบกล้อง")
5
6 webrtc_streamer(key="test",
7                 media_stream_constraints={"video": True, "audio": False})
8
```

นำเข้าไลบรารี streamlit เพื่อใช้ในการสร้างแอปพลิเคชันเว็บ และ streamlit_webrtc เพื่อใช้ในการสร้างสตรีมเว็บ

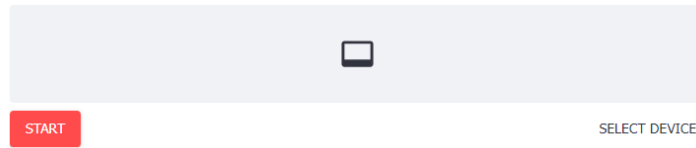
แคม

st.title() ใช้สร้างหัวข้อในแอปพลิเคชันเว็บ โดย "ทดสอบกล้อง" เป็นข้อความที่จะแสดงเป็นหัวข้อ

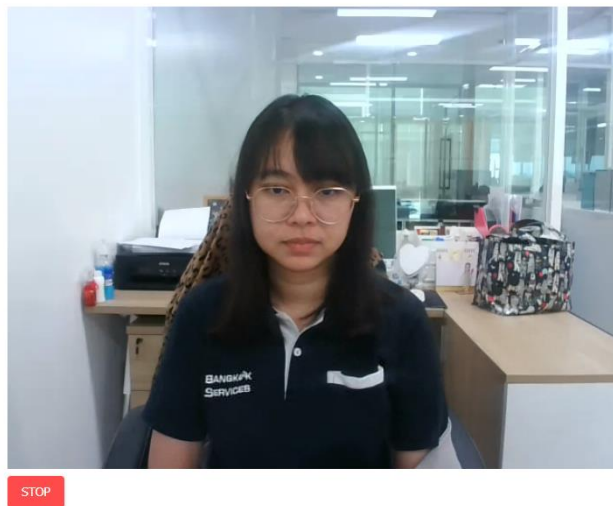
`webrtc_streamer()` เป็นฟังก์ชันที่ใช้ในการสตรีมวิดีโอและเสียงผ่านเว็บ โดยรับพารามิเตอร์หลักเป็น `key` ซึ่งเป็นรหัสที่ใช้เพื่อแยกและระบุวิดีโอและพารามิเตอร์ `media_stream_constraints` ซึ่งกำหนดเงื่อนไขสำหรับสตรีมวิดีโอและเสียง ในที่นี้กำหนดให้สตรีมวิดีโอเท่านั้น (ไม่มีเสียง) ด้วยการให้ค่า `{"video": True, "audio": False}`

ผลการรันโปรแกรม

ทดสอบกล้อง



ทดสอบกล้อง



การ Flip ภาพจากกล้อง

```

PYSL065.py > ...
1  import streamlit as st
2  from streamlit_webrtc import webrtc_streamer
3  import av
4  import cv2
5
6  st.title("การกลับภาพ")
7
8  class VideoProcessor: #สร้าง class สำหรับประมวลผลภาพ
9      def recv(self, frame):
10         img = frame.to_ndarray(format="bgr24") #24bit 8B 8G 8R
11         #-----
12         img = cv2.flip(img,1) #แกน x 0 บน-ล่าง แกน y 1 ซ้าย-ขวา
13         #-----
14         return av.VideoFrame.from_ndarray(img, format="bgr24")
15
16  webrtc_streamer(key="test",
17                 video_processor_factory=VideoProcessor, #คลาส
18                 media_stream_constraints={"video": True, "audio": False})
19

```

นำเข้าไลบรารี streamlit เพื่อใช้ในการสร้างแอปพลิเคชันเว็บ และ streamlit_webrtc เพื่อใช้ในการสร้างสตรีมเว็บแคม นอกจากนี้ยังนำเข้าไลบรารี av และ cv2 เพื่อใช้ในการจัดการวิดีโอและการประมวลผลภาพ

กำหนดหัวข้อของแอปพลิเคชันเว็บเป็น "การ Flip ภาพ"

สร้างคลาส VideoProcessor สำหรับการประมวลผลวิดีโอ ในคลาสนี้ เรากำหนดเมธอด recv() เพื่อรับเฟรมวิดีโอที่สตรีมจากกล้องเว็บ ซึ่งจะนำเข้าเป็นรูปแบบ bgr24 (Blue-Green-Red 24-bit) และทำการ flip ภาพ ใช้ฟังก์ชัน cv2.flip() ของ OpenCV เพื่อทำการกลับด้านภาพ โดยในที่นี้กำหนดให้ flip แกน y (ซ้าย-ขวา)

* flip แกน x (บน-ล่าง) `img = cv2.flip(img,0)`

* flip แกน y (ซ้าย-ขวา) `img = cv2.flip(img,1)`

ใช้ webrtc_streamer() เพื่อเริ่มสตรีมวิดีโอผ่านเว็บ ระบุ key="test" เพื่อระบุวิดีโอเรีกรอร์ค และ video_processor_factory=VideoProcessor เพื่อระบุคลาส VideoProcessor ที่ใช้ในการประมวลผลวิดีโอ นอกจากนี้ยังระบุ media_stream_constraints={"video": True, "audio": False} เพื่อเฉพาะการสตรีมวิดีโอเท่านั้น

ดังนั้น โค้ดนี้จะสร้างเว็บแอปพลิเคชันที่มีหัวข้อ "การ Flip ภาพ" และจะแสดงวิดีโอจากกล้องเว็บ โดยภาพจะถูกกลับด้านในแกน y (ซ้าย-ขวา) โดยใช้ OpenCV ก่อนที่จะแสดงผลออกทางเว็บไชต์

ผลการรันโปรแกรม

การ Flip ภาพ



START

SELECT DEVICE

การ Flip ภาพ



STOP

```

PYSL075.py > ...
1 import streamlit as st
2 from streamlit_webrtc import webrtc_streamer
3 import av
4 import cv2
5 import numpy as np
6
7 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
8
9 st.title("ตรวจจับใบหน้า")
10
11 class VideoProcessor:
12     def recv(self, frame):
13         img = frame.to_ndarray(format="bgr24")
14         #-----
15         img = cv2.flip(img,1)
16         img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
17         faces = face_cascade.detectMultiScale(img_gray, 1.3, 5)
18         for (x,y,w,h) in faces:
19             cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0),5)
20         #-----
21         return av.VideoFrame.from_ndarray(img, format="bgr24")
22
23 webrtc_streamer(key="test",
24                 video_processor_factory=VideoProcessor,
25                 media_stream_constraints={"video": True, "audio": False})
26

```

นำเข้าไลบรารี streamlit เพื่อใช้ในการสร้างแอปพลิเคชันเว็บ และ streamlit_webrtc เพื่อใช้ในการสร้างสตรีมเว็บแคม นอกจากนี้ยังนำเข้าไลบรารี av และ cv2 เพื่อใช้ในการจัดการวิดีโอและการประมวลผลภาพ

การตรวจจับใบหน้าด้วย Haar Cascade Classifier ซึ่งเป็นวิธีการที่พบบ่อยและเป็นที่นิยมในการตรวจจับใบหน้าในภาพ เริ่มต้นจากการกำหนด Haar Cascade Classifier ด้วยการโหลดไฟล์ XML

haarcascade_frontalface_default.xml ซึ่งเป็น โมเดลสำหรับตรวจจับใบหน้าที่ถูกฝึกสอนมาอย่างดี

กำหนดหัวข้อของแอปพลิเคชันเว็บเป็น "การ Flip ภาพ"

สร้างคลาส VideoProcessor ในเมธอด recv() เราทำการแปลงภาพเป็นภาพขาว-ดำ (grayscale) และใช้ detectMultiScale() เพื่อตรวจจับใบหน้าในภาพ หากตรวจพบใบหน้า เราจะวาดสี่เหลี่ยมรอบใบหน้าโดยใช้ cv2.rectangle() ที่มีสีเขียว และความหนาของเส้น 5 เพื่อให้เด่นขึ้น

ใช้ webrtc_streamer() เหมือนเดิมเพื่อเริ่มสตรีมวิดีโอ กำหนด key="test" เพื่อระบุวิดีโอเรียกคอร์ด และระบุ video_processor_factory=VideoProcessor เพื่อใช้คลาส VideoProcessor ที่เราสร้างไว้ในการประมวลผลวิดีโอ

ดังนั้น แอปพลิเคชันเว็บนี้จะแสดงวิดีโอจากกล้องเว็บ และตรวจจับใบหน้าในภาพ และวาดสี่เหลี่ยมรอบใบหน้าในเฟรมวิดีโอที่ตรวจพบใบหน้า

ผลการรันโปรแกรม

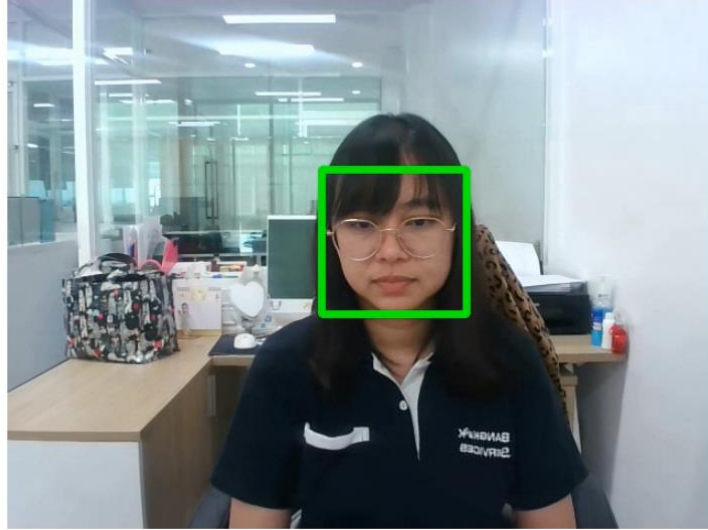
ตรวจจับใบหน้า



START

SELECT DEVICE

ตรวจจับใบหน้า



STOP