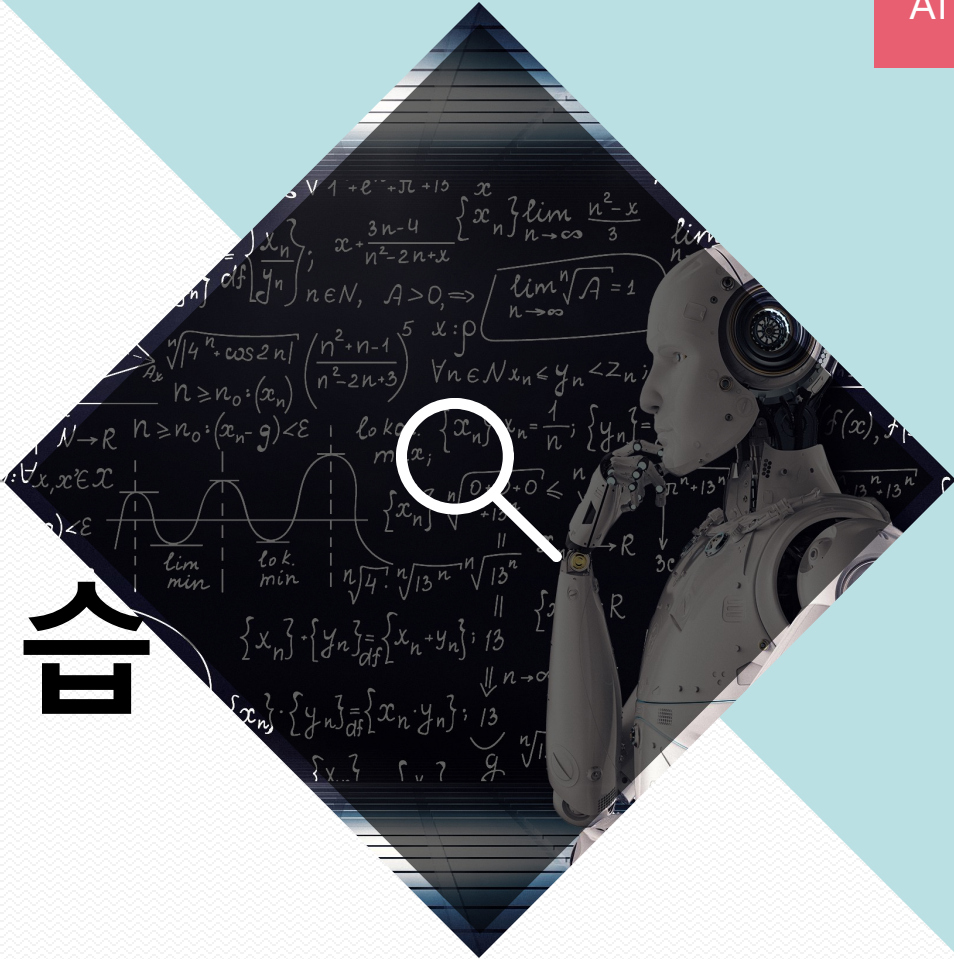


Smart Factory실습

Multi-threading



Agenda

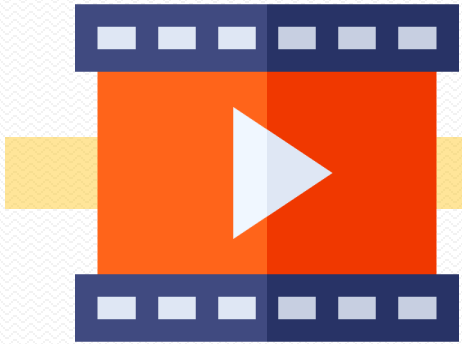
- *How to use tool?*
- *Parallelism*



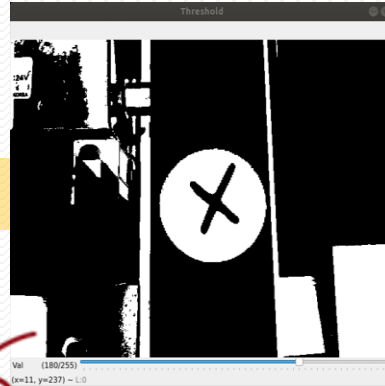
How to use tool?

Motion Detect

Camera Input



Binarization

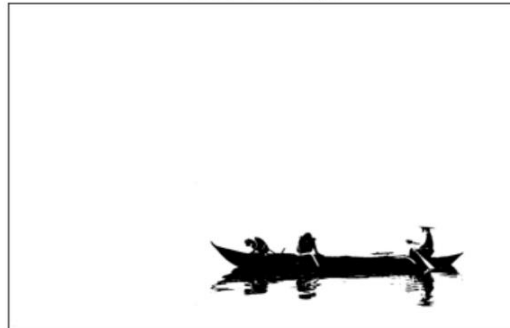


To drastically reduce the amount of information you have to work with

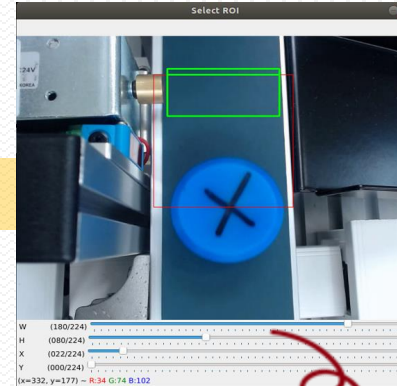
original image



filtered image



Region of Interesting
ROI Setting



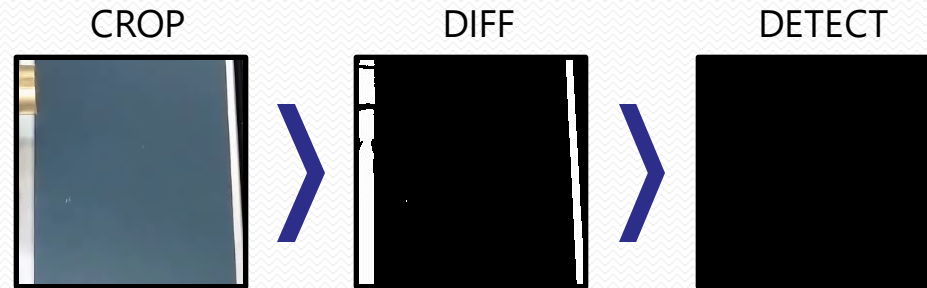
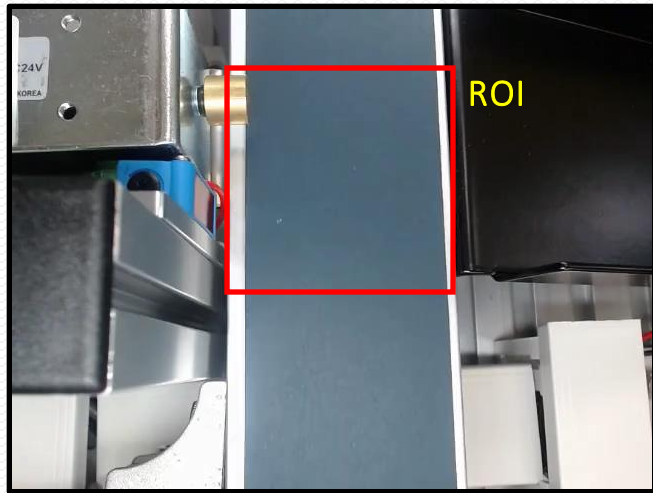
Video Stream
with 640x480 resolution

Motion Detect



224x224 image

Detailed Flow

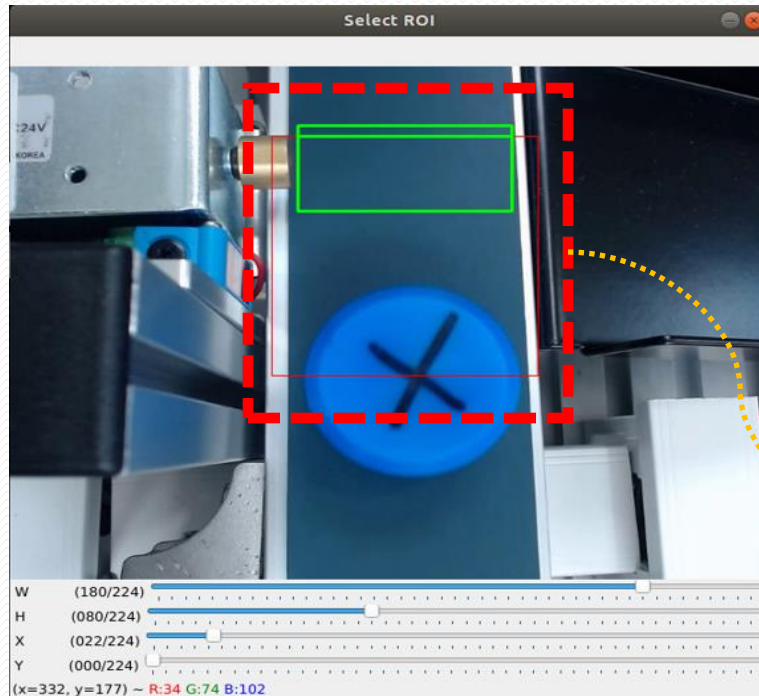


- *Crop the frame per selected ROI*
- *Calculate the frame difference*
 - Apply custom threshold with brightness value
- *Choose the best frames to pass*

Select ROI to Detect Motion

Python Tool (*iotdemo-motion-detector*)

iotdemo-motion-detector -l
./resource/factory/conveyor.mp4



Pre-Implemented Tool

Python Main (*factory.py*)

Detect the Frame (each Thread)

- *detected* = *det.detect(frame)*

if *detected* is None:
 continue

Enqueue (each Thread)

- *q.put(('VIDEO: Cam1 detected', *detected*))*

- *q.put(('VIDEO: Cam2 detected', *detected*))*

```
[default]
inverted = False
flipped = False
top_margin = 10
threshold = 127
top_ratio = 0.6
mid_ratio = 0.4
skip_time = 0.099
roi_top_left = (12, 0)
roi_bottom_right = (212, 90)
crop_top_left = (0, 10)
crop_bottom_right = (224, 234)
```

motion.cfg

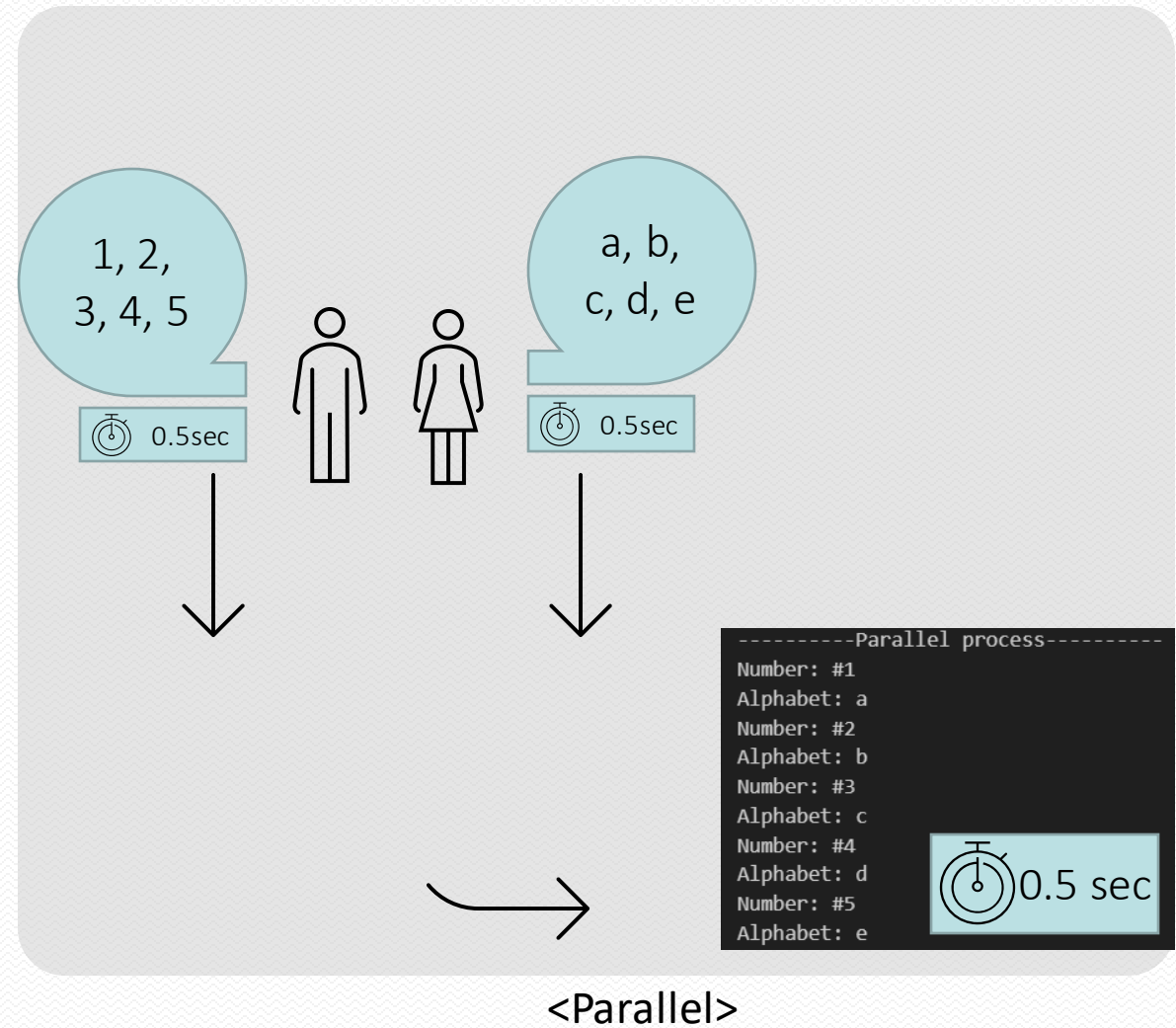
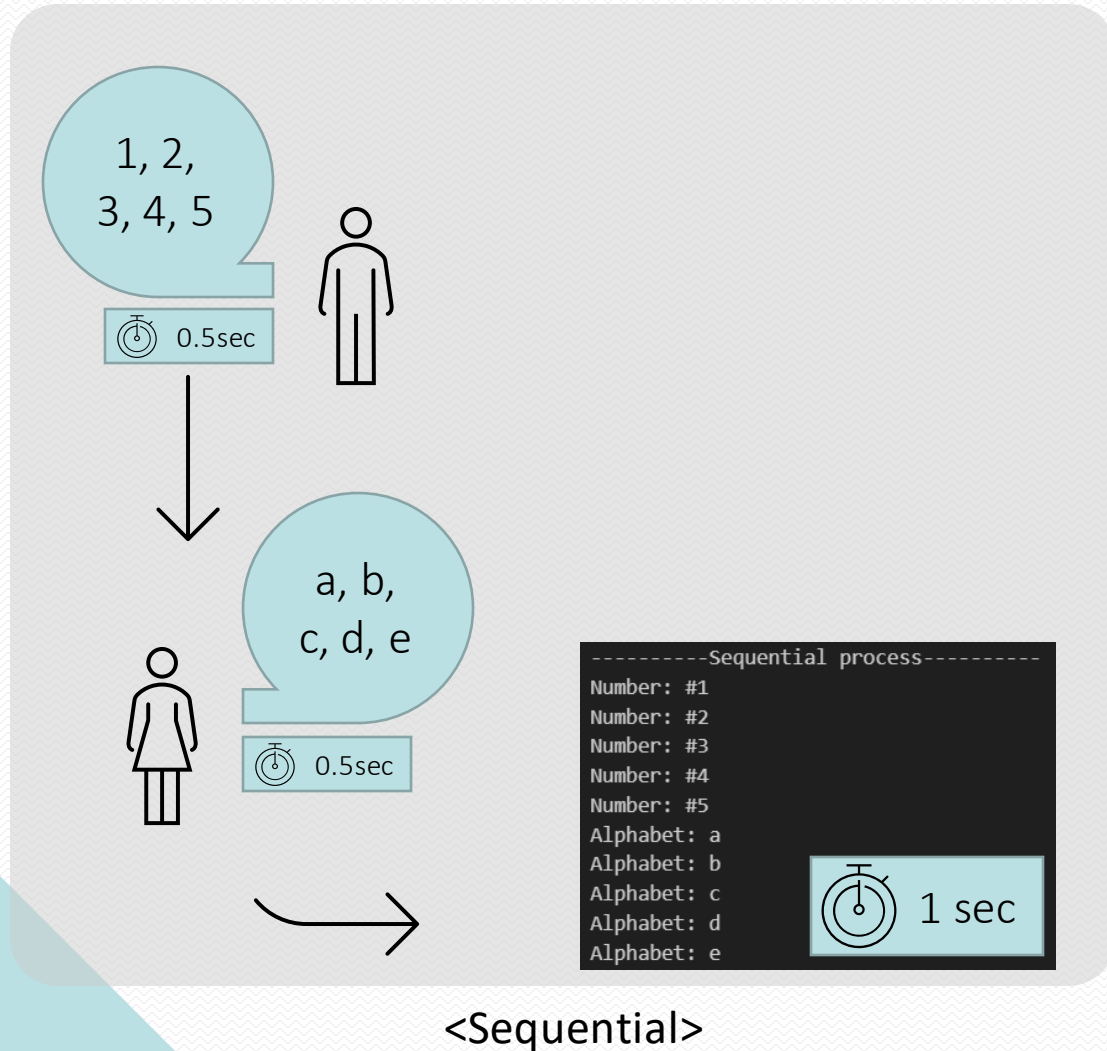


Cropped Image

Parallelism

Parallelism

- Sequential vs Parallel



Practice Preparation

- Virtual environment setup

```
$ cd Class02/python-parallelism
```

```
$ python3 -m venv .venv
```

```
$ source .venv/bin/activate
```

```
(.venv)$ pip install --upgrade pip
```

Open **libraries.ipynb**

Click **'Select Kernel'**

Selecte 'Install/Enable suggested extensions'

```
(.venv)$ pip install -r ./requirements.txt
```

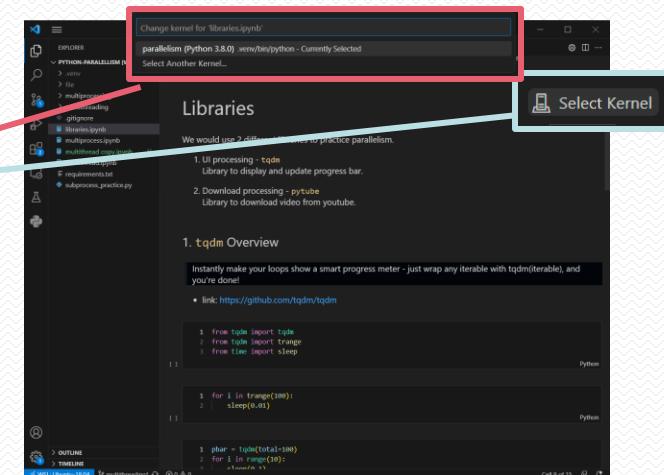
```
(.venv)$ ipython kernel install --user --name=parallelism
```

Verification

```
(.venv)$ jupyter-kernelspec list
```

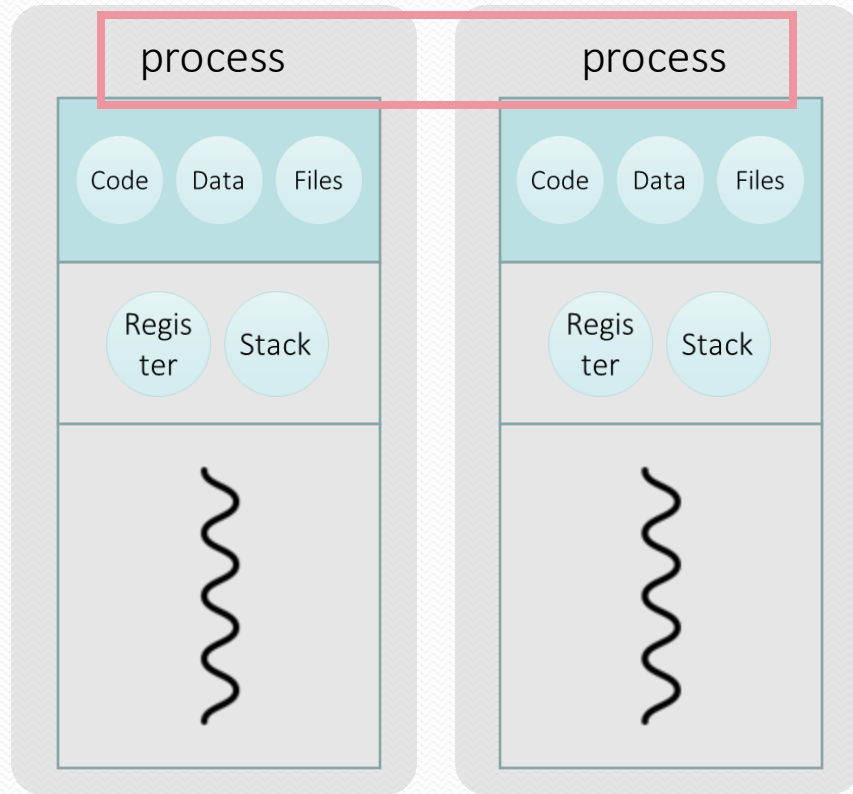
- Select IPython kernel from VSCode UI

- Click `Select Kernel`
- Choose `parallelism`



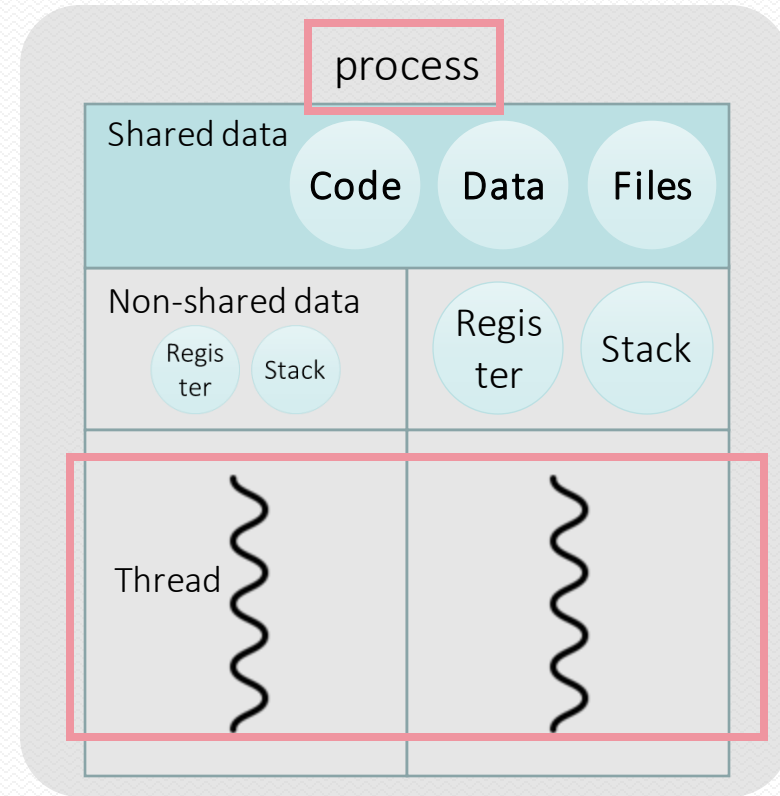
Multi-processing vs Multi-threading

Use **multiple processes**
to execute multiple tasks



<Multi-processing>

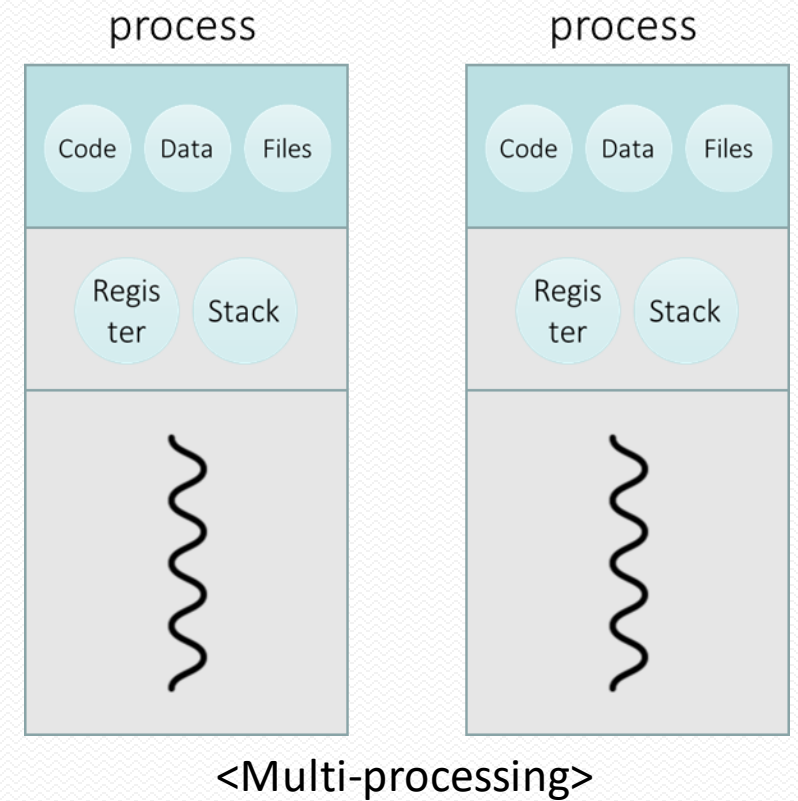
Use **single process**,
but run **multiple threads** inside



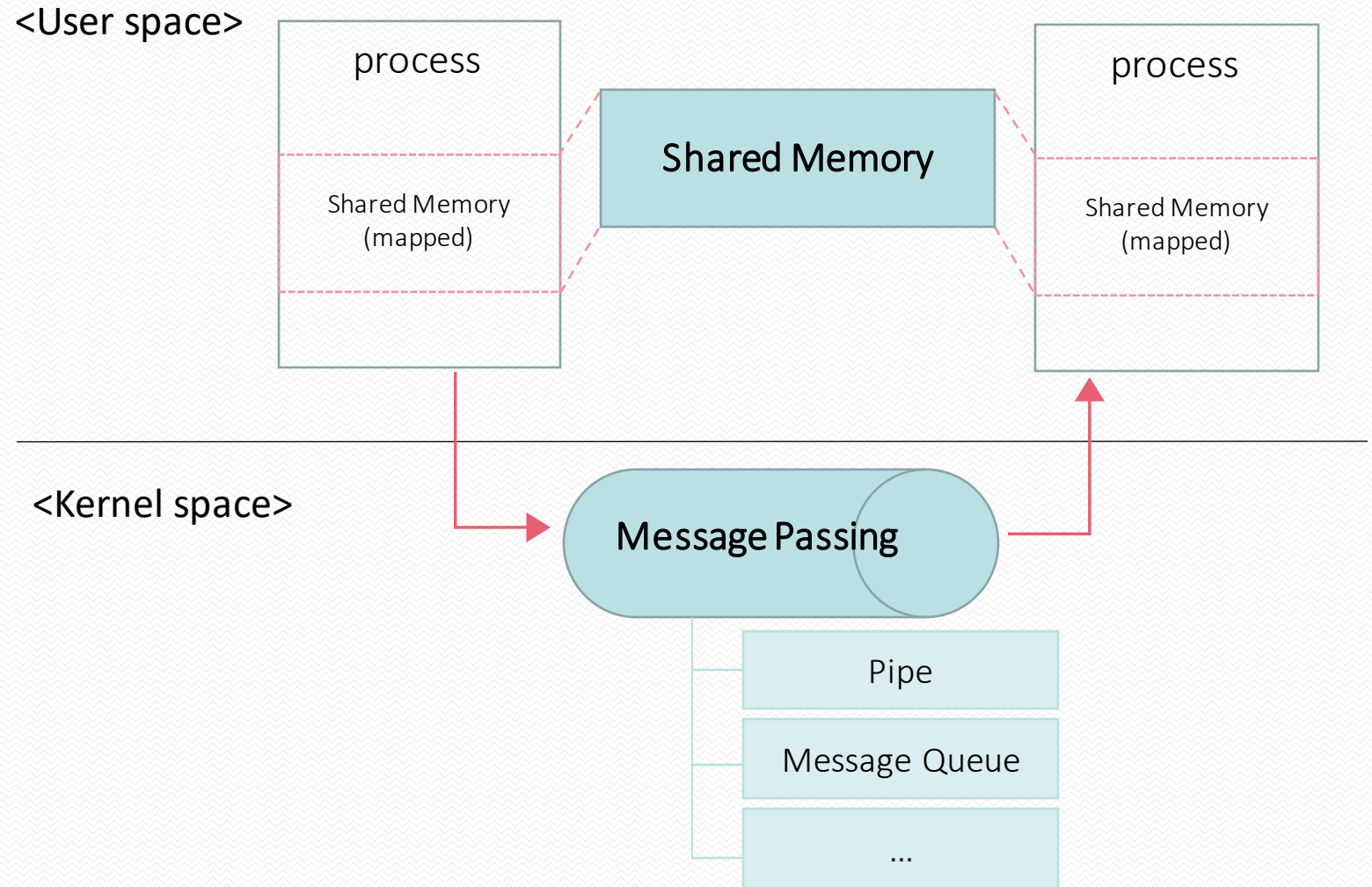
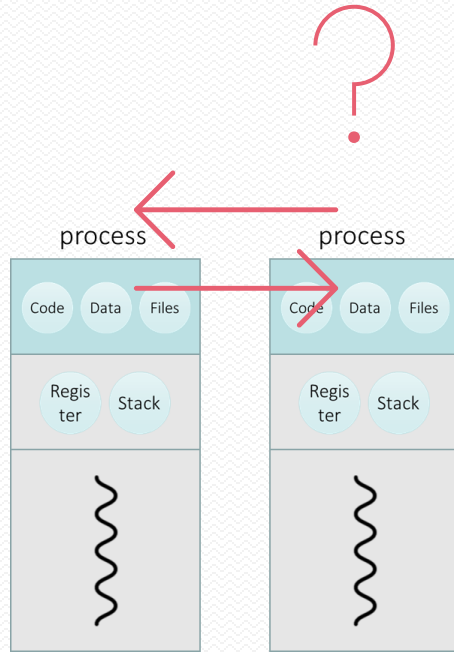
<Multi-threading>

Multi-processing: Concept

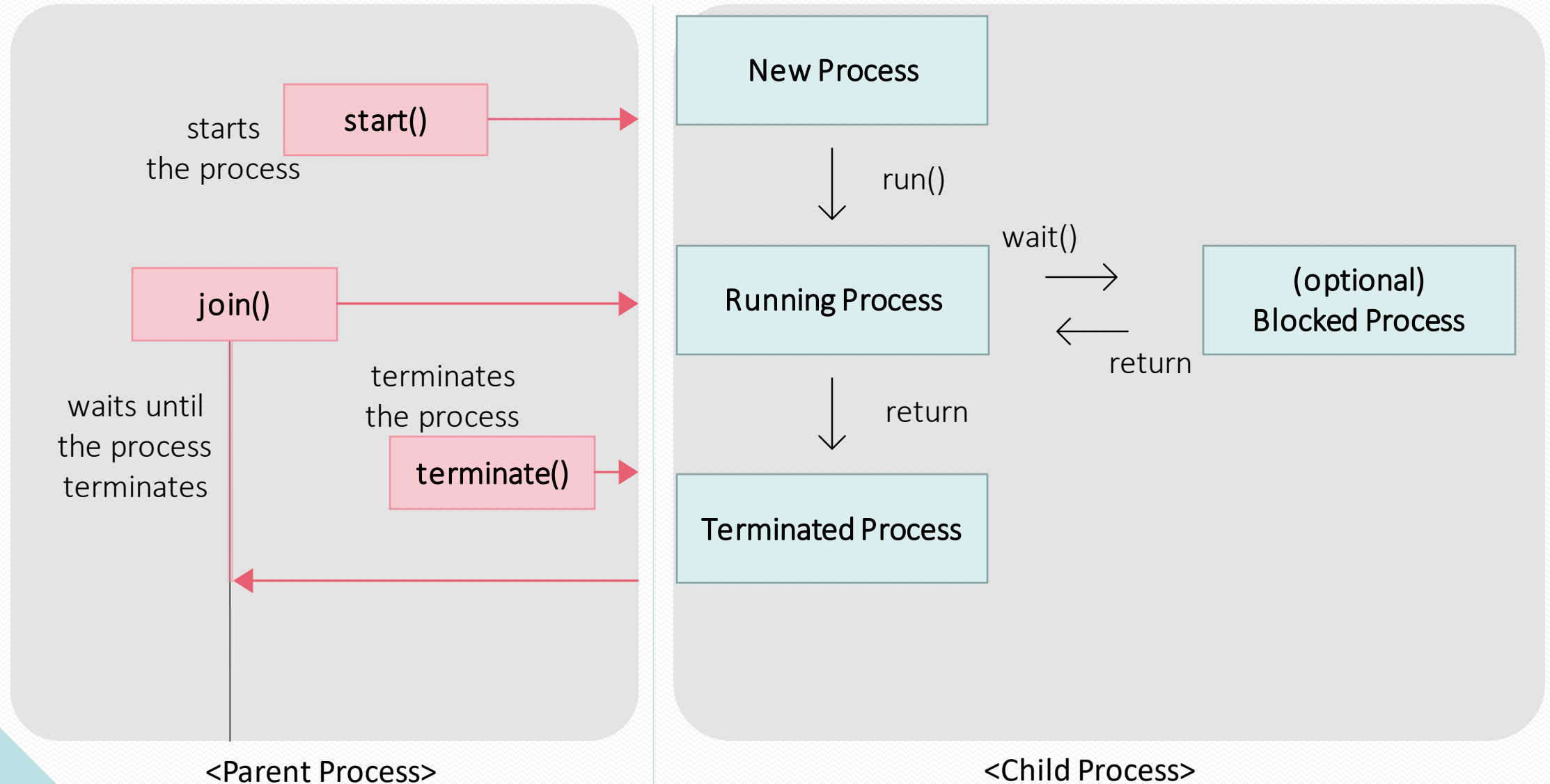
- Each process has code, data, stack and heap areas
- Every processes includes at least 1 thread(main thread)
- Processes are being separated in terms of memory usages



Multi-processing: Inter-process Communication(IPC)

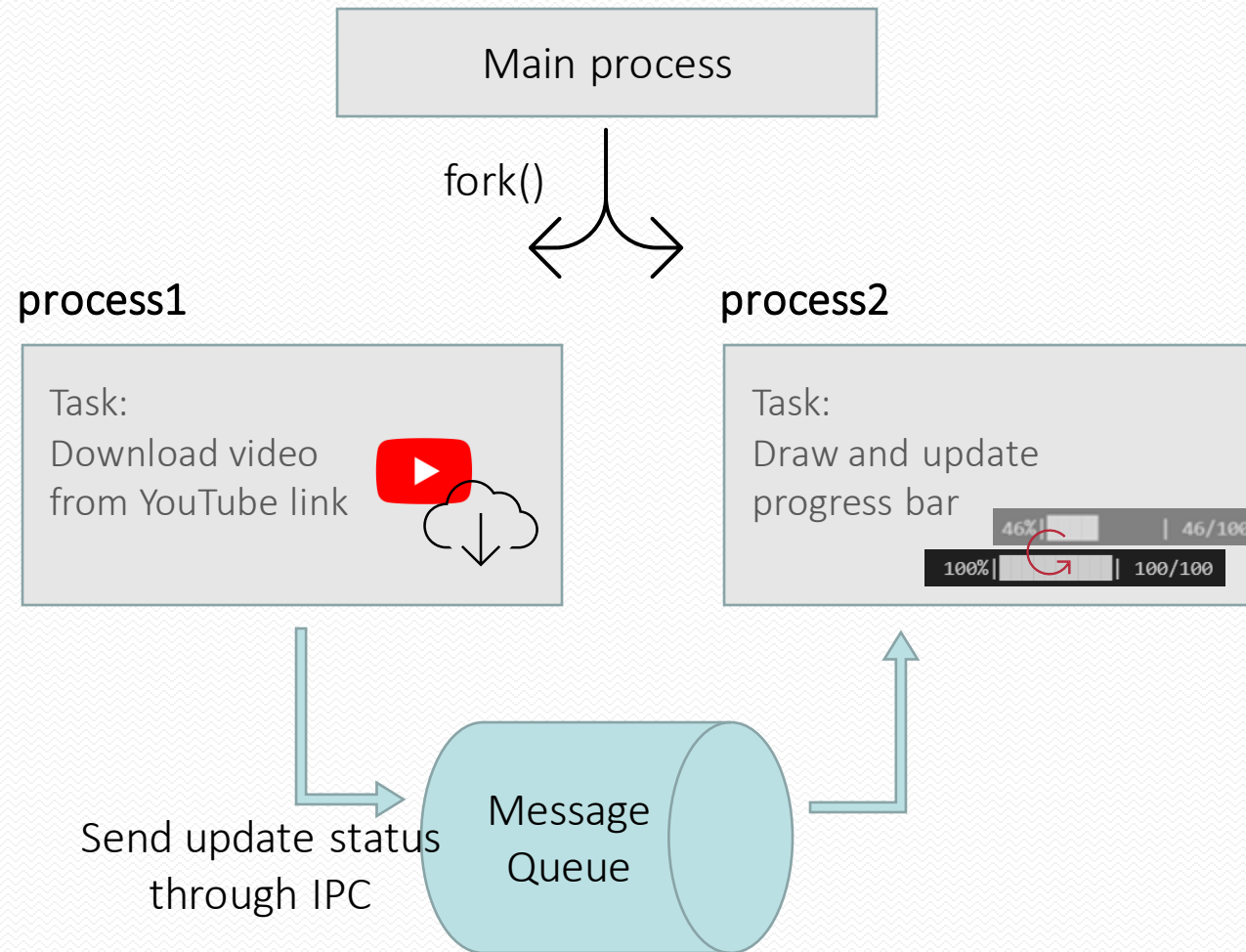


Multi-processing: Lifecycle of Process (multiprocessing.Process class)



Multi-processing: example

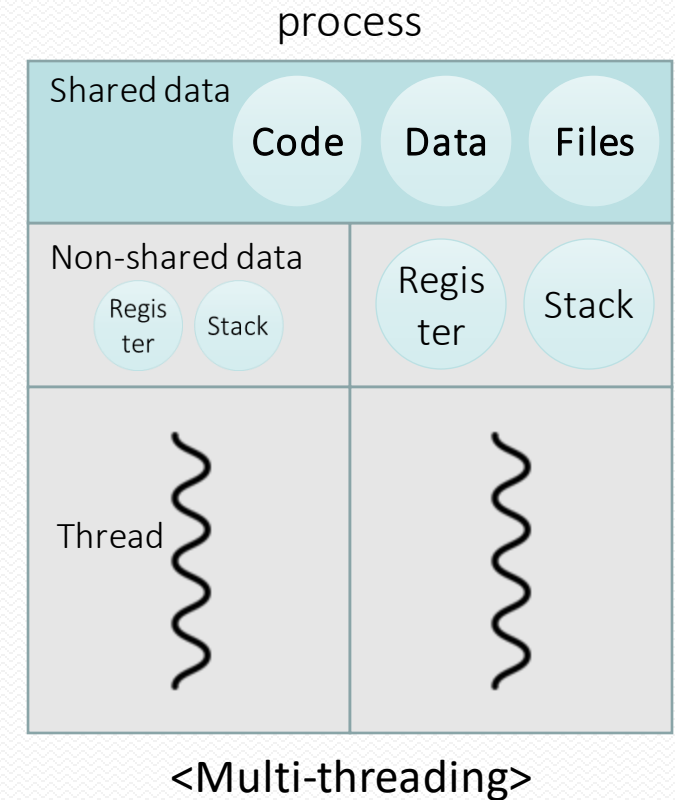
- `pytube`, `tqdm` example



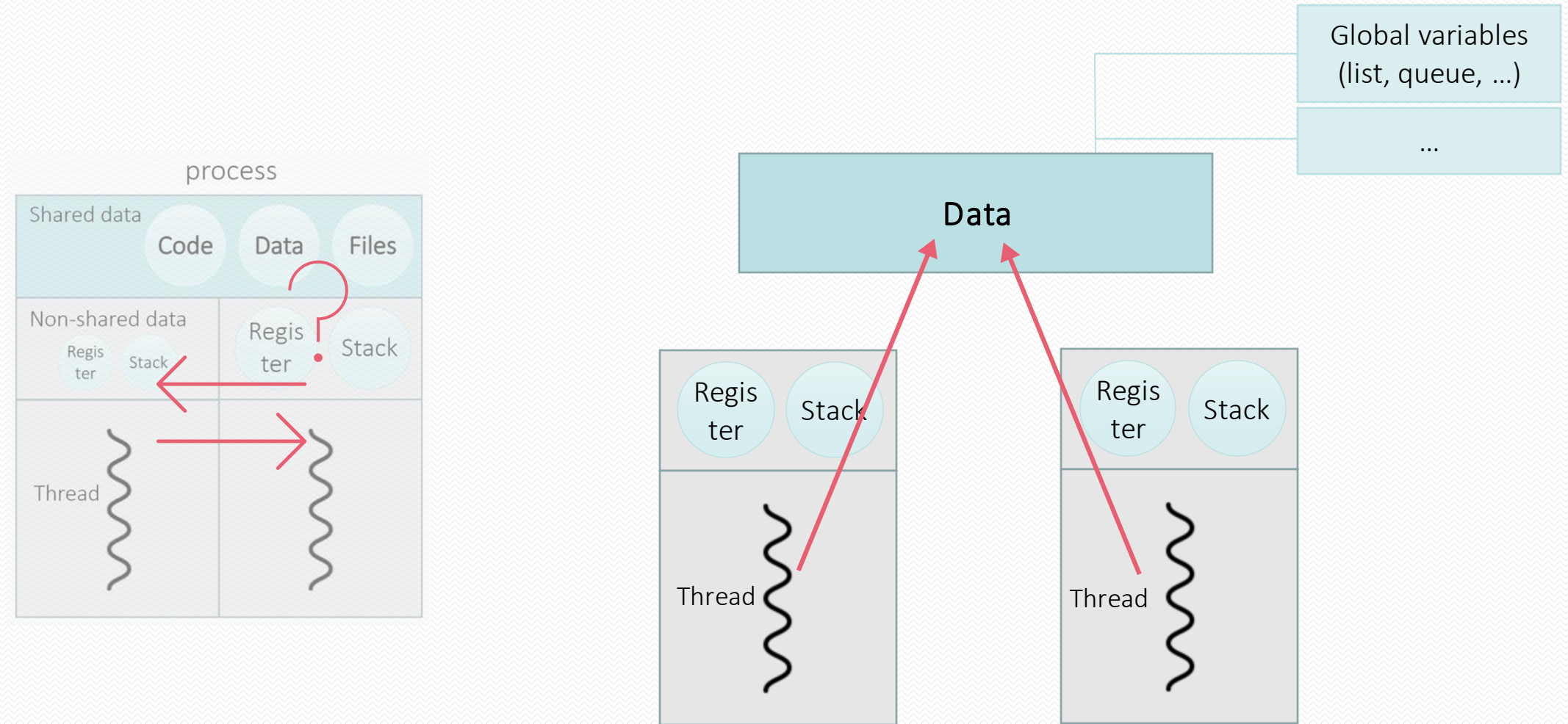
Reference: <https://docs.python.org/3/library/multiprocessing.html#multiprocessing-programming>

Multi-threading: Concept

- A light-weight process
- Utilize **single process** to execute multiple tasks
- Process consists of **multiple threads**, where each threads represents each tasks
- Threads in one process share memory area, in terms of code, data segments and heap

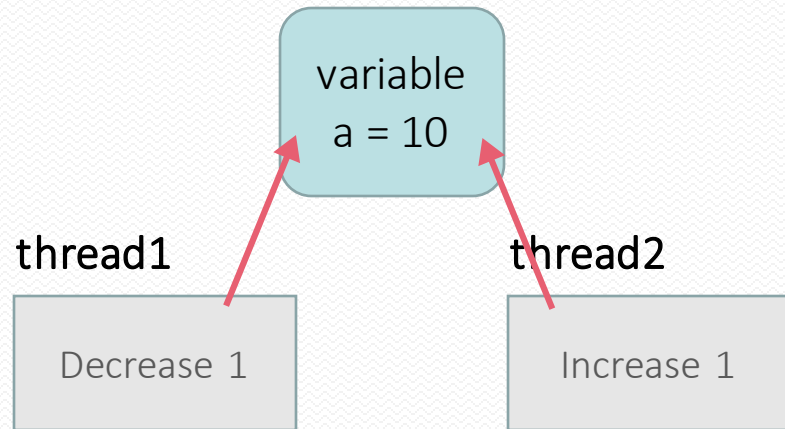


Multi-threading: Communication between threads



GIL(Global Interpreter Lock)

- GIL in python, is a mutex which ensures to run only one thread at once

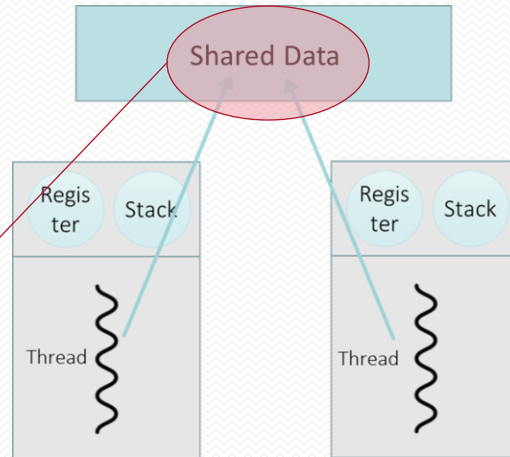


<Race condition>

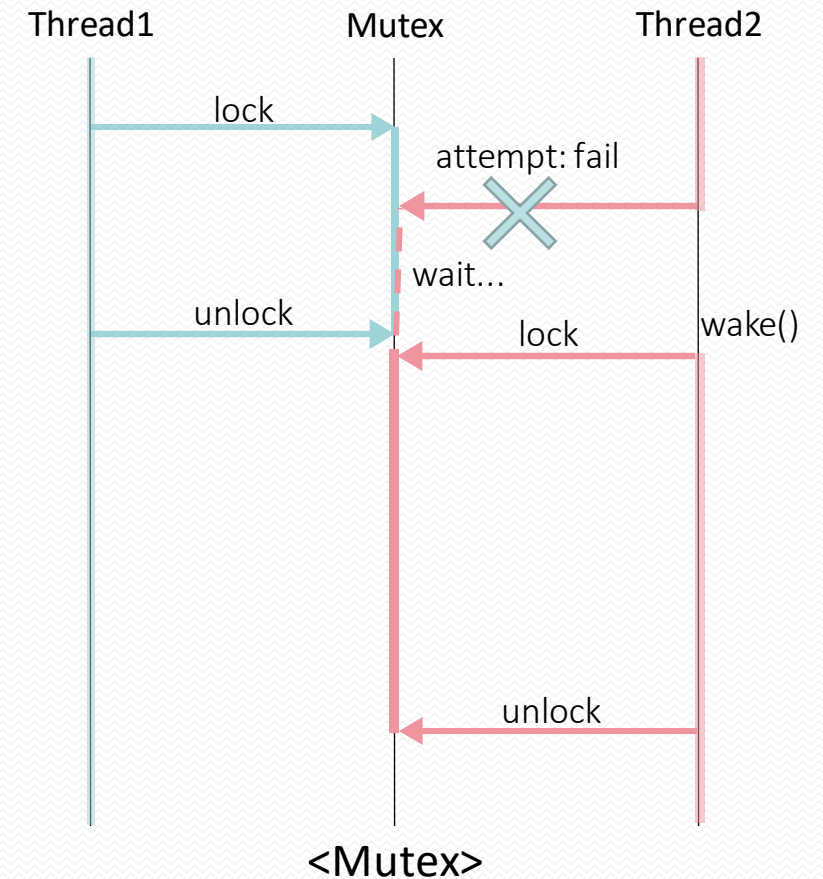
thread1:	thread2:
Read: a	Read: a
Decrease 1	Increase 1
Write: a	Write: a
Result: a?	Result: a?

- Thread-safety
 - Prevents race condition
- Performance issue:
 - Takes more time than actual multi-threaded program

Multi-threading: Problem & Solution

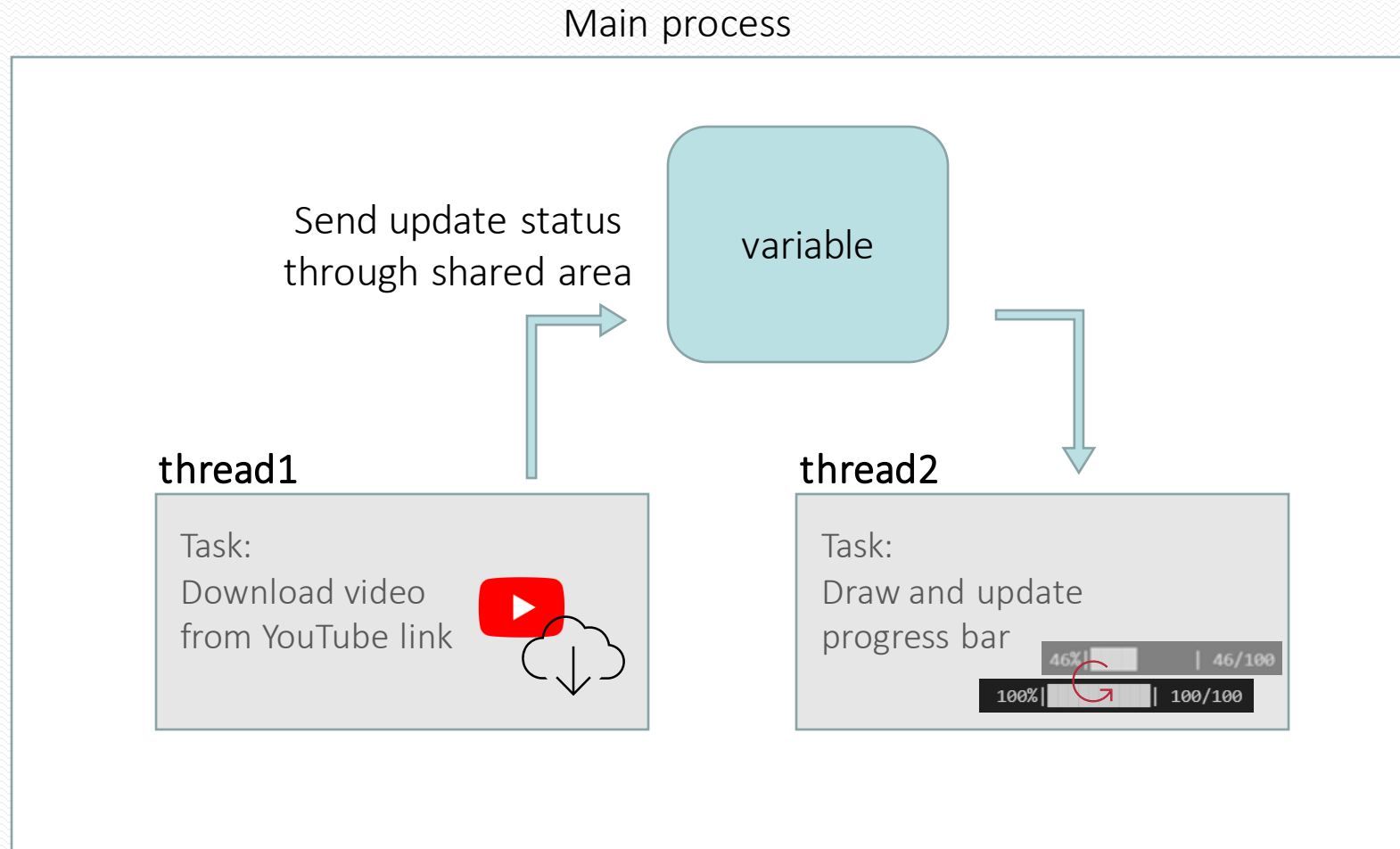


- Synchronization Problem
 - When multiple threads attempt to access to shared area at the same time
- Solution
 - Mutex(Mutual Exclusion)



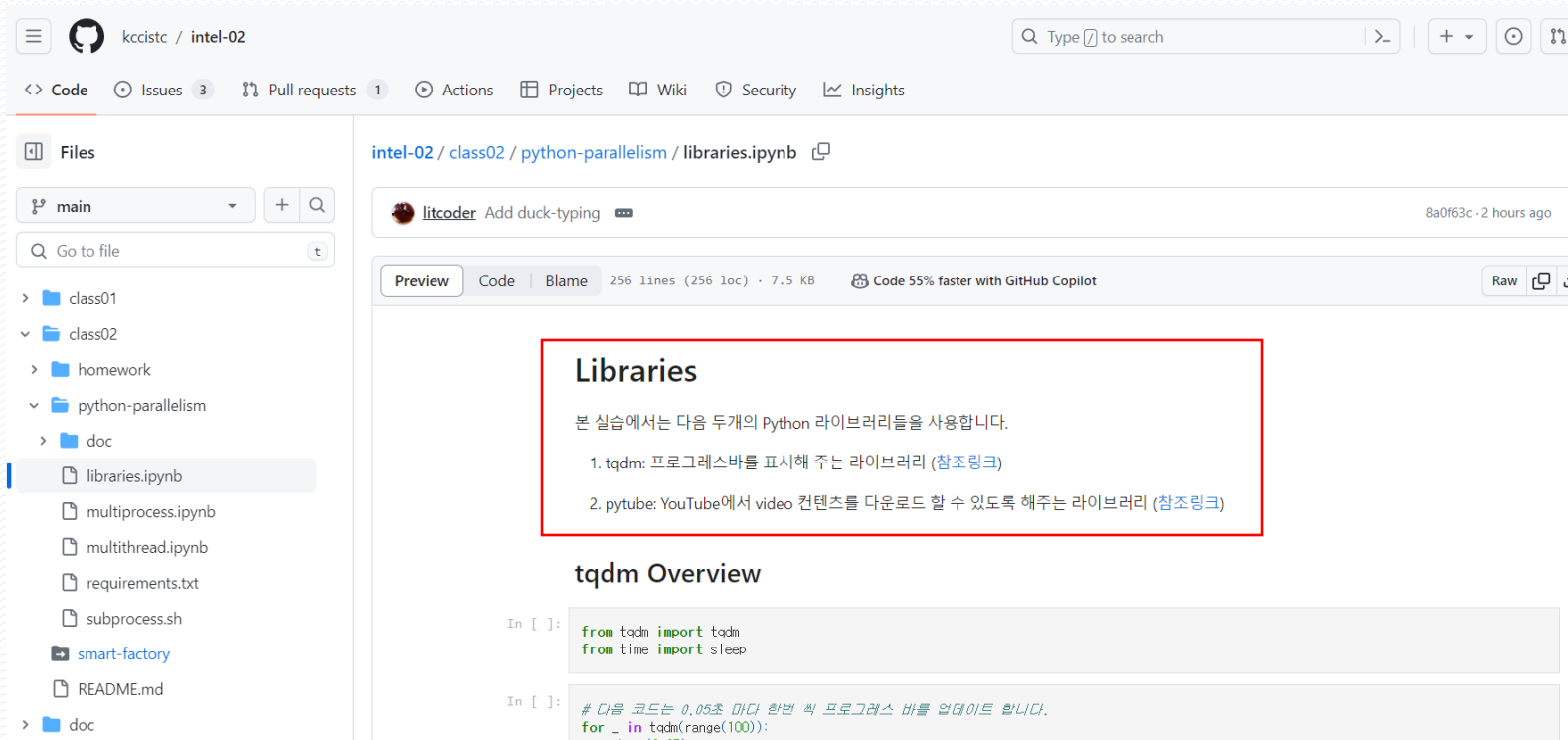
Multi-threading: Example

- `pytube`, `tqdm` example



- `.venv/bin/pip install -r python-parallelism/requirements.txt`
- `.venv/bin/pip freeze | wc -l`

<https://github.com/kccistc/intel-02/blob/main/class02/python-parallelism/libraries.ipynb>



The screenshot shows the GitHub interface for the repository 'kccistc / intel-02'. The file 'libraries.ipynb' is selected, showing its content. The content is a Jupyter Notebook cell with the following text:

```
Libraries

본 실습에서는 다음 두개의 Python 라이브러리들을 사용합니다.

1. tqdm: 프로그레스바를 표시해 주는 라이브러리 (참조링크)
2. pytube: YouTube에서 video 콘텐츠를 다운로드 할 수 있도록 해주는 라이브러리 (참조링크)

tqdm Overview

In [ ]:
from tqdm import tqdm
from time import sleep

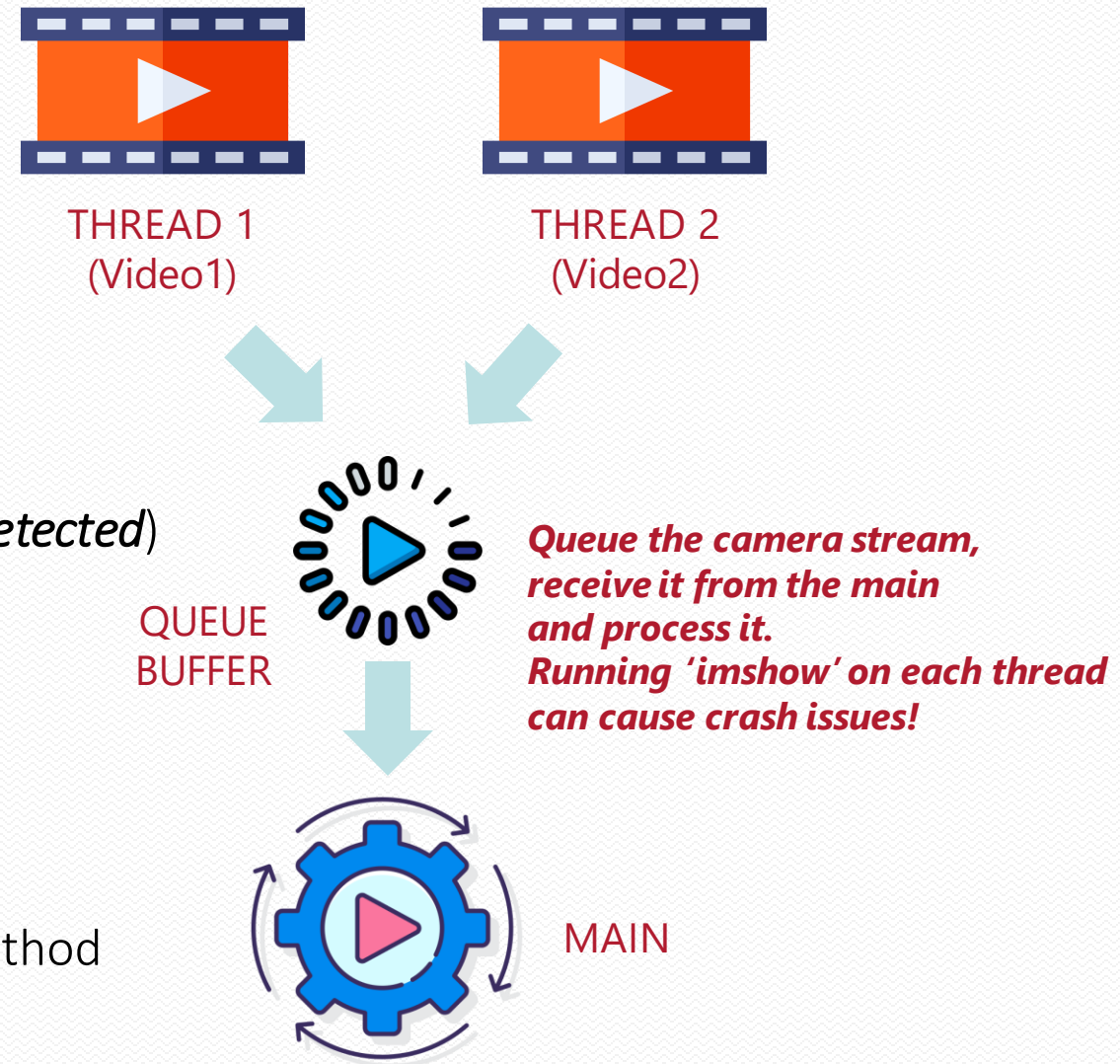
In [ ]:
# 다음 코드는 0.05초 마다 한번씩 프로그레스 바를 업데이트 합니다.
for _ in tqdm(range(100)):
```

Multi-processing vs Multi-threading

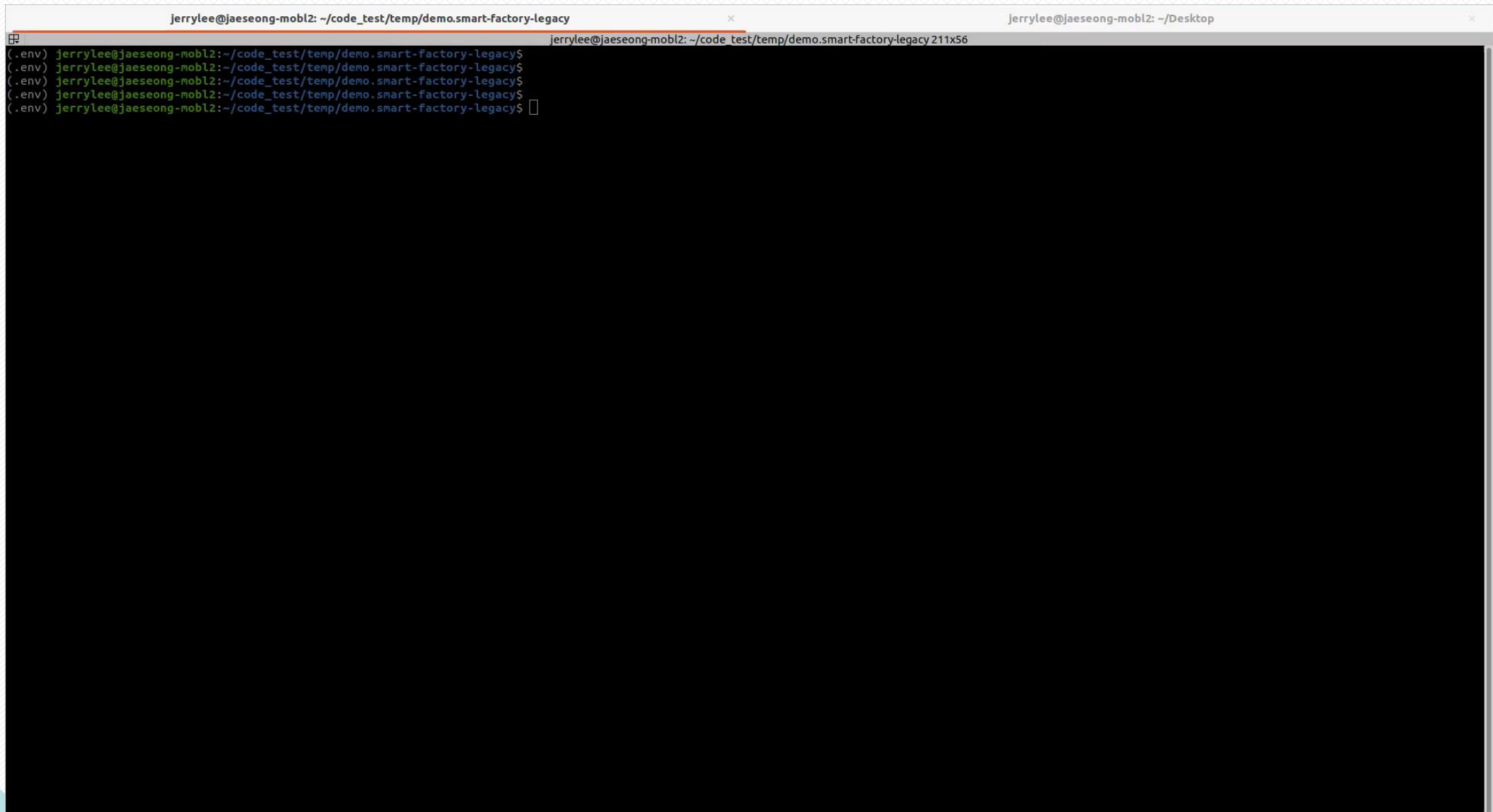
	Reliability	Communication	Execution time	Memory usage
Multi-processing	More reliable: Memory isolation	Slower: IPC mechanism	Slower: Consuming more time to start process	Less effective: Consuming more memory space
Multi-threading	Less reliable: Sharing memory space, need additional protection (i.e. Mutex)	Faster: Using shared memory space	Faster: Consuming less time to start thread	More effective: Consuming less memory space

Multi-thread Hands-on (factory.py)

- Implement the two threads, one queue for Smart Factory
- Thread 1/2
 - Video open/close and Call **MotionDetector**
 - In the while loop,
Enqueue read video frame
(as a tuple, name: *'Video:Cam1/2 live', data:frame*)
When a motion is detected, Enqueue the frame
(as a tuple, name: *'Video:Cam1/2 detected', data:detected*)
- Main
 - Create the thread1/2 and start that threads
 - In **FactoryController** loop,
get the event(name, data) from queue
 - If the name has *'Video:'*,
the data display with *'name[6:]'* using "imshow" method



Multi-thread Hands-on (cont,)



The screenshot shows a terminal window with a dark background and light green text. The window title is "jerrylee@jaeseong-mobl2: ~/code_test/temp/demo.smart-factory-legacy". The terminal output shows a multi-threaded application running, with the prompt "(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy\$" repeated five times. The application is running in a loop, and the output is not visible in the screenshot.

```
jerrylee@jaeseong-mobl2: ~/code_test/temp/demo.smart-factory-legacy
(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy$
(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy$
(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy$
(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy$
(.env) jerrylee@jaeseong-mobl2:~/code_test/temp/demo.smart-factory-legacy$
```



THANK YOU