Smart Factory실습

Multi-threading

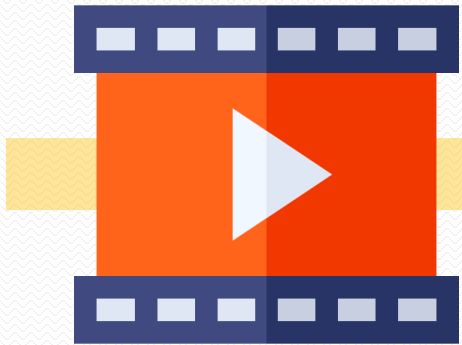# Agenda

- *How to use tool?*

- *Parallelism*
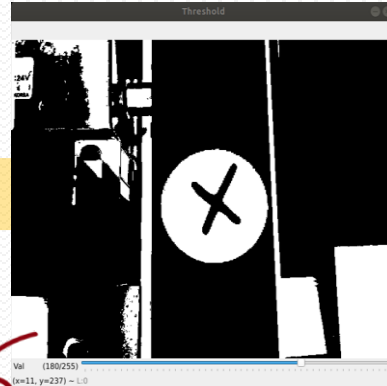
How to use tool?

Motion Detect

Camera Input
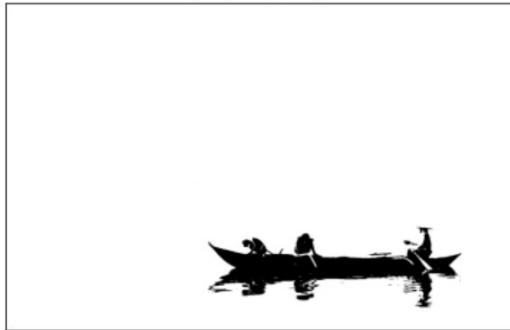
Binarization

ROI Setting

Motion Detect

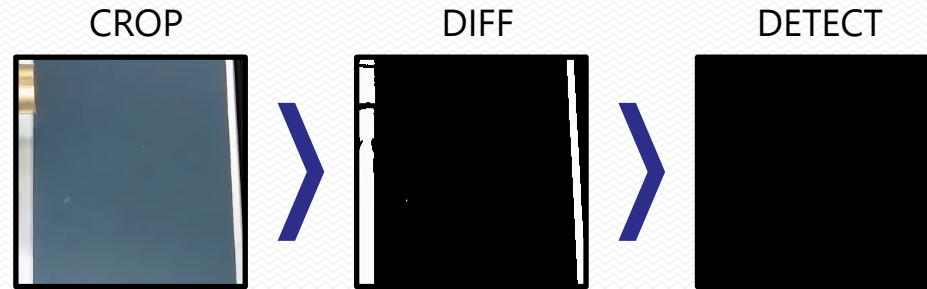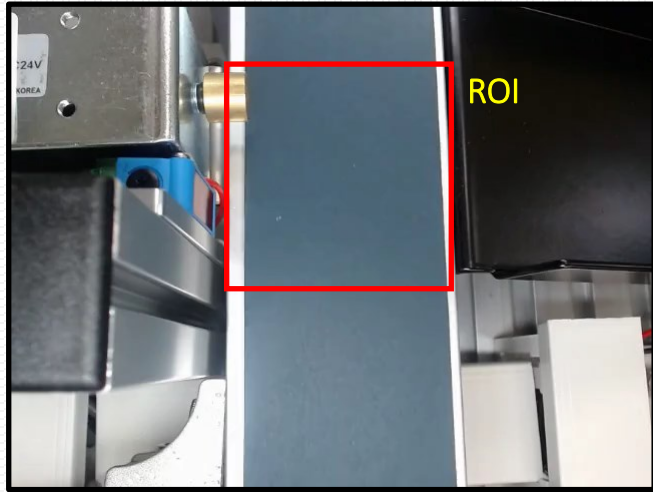To drastically reduce the amount of information you have to work with
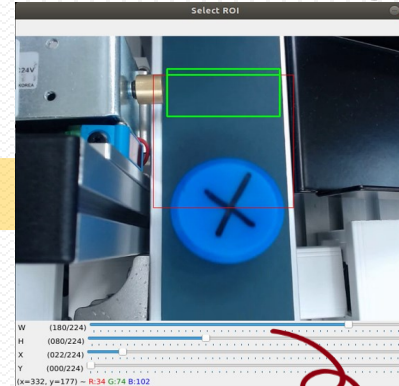
Video Stream
with 640x480 resolution

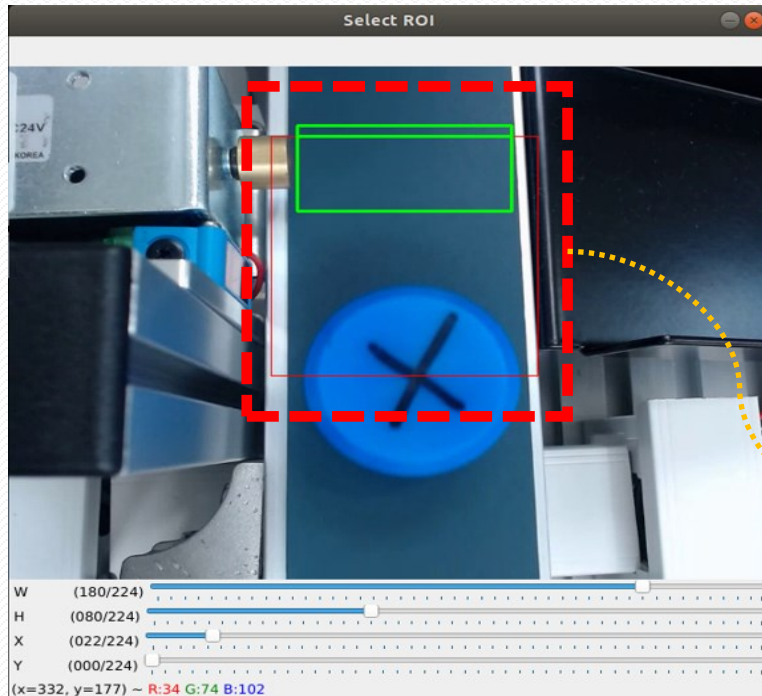224x224 image

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 -I ./resource/factory/conveyor.mp4

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))*

Pre-Implemented Tool

motion.cfg

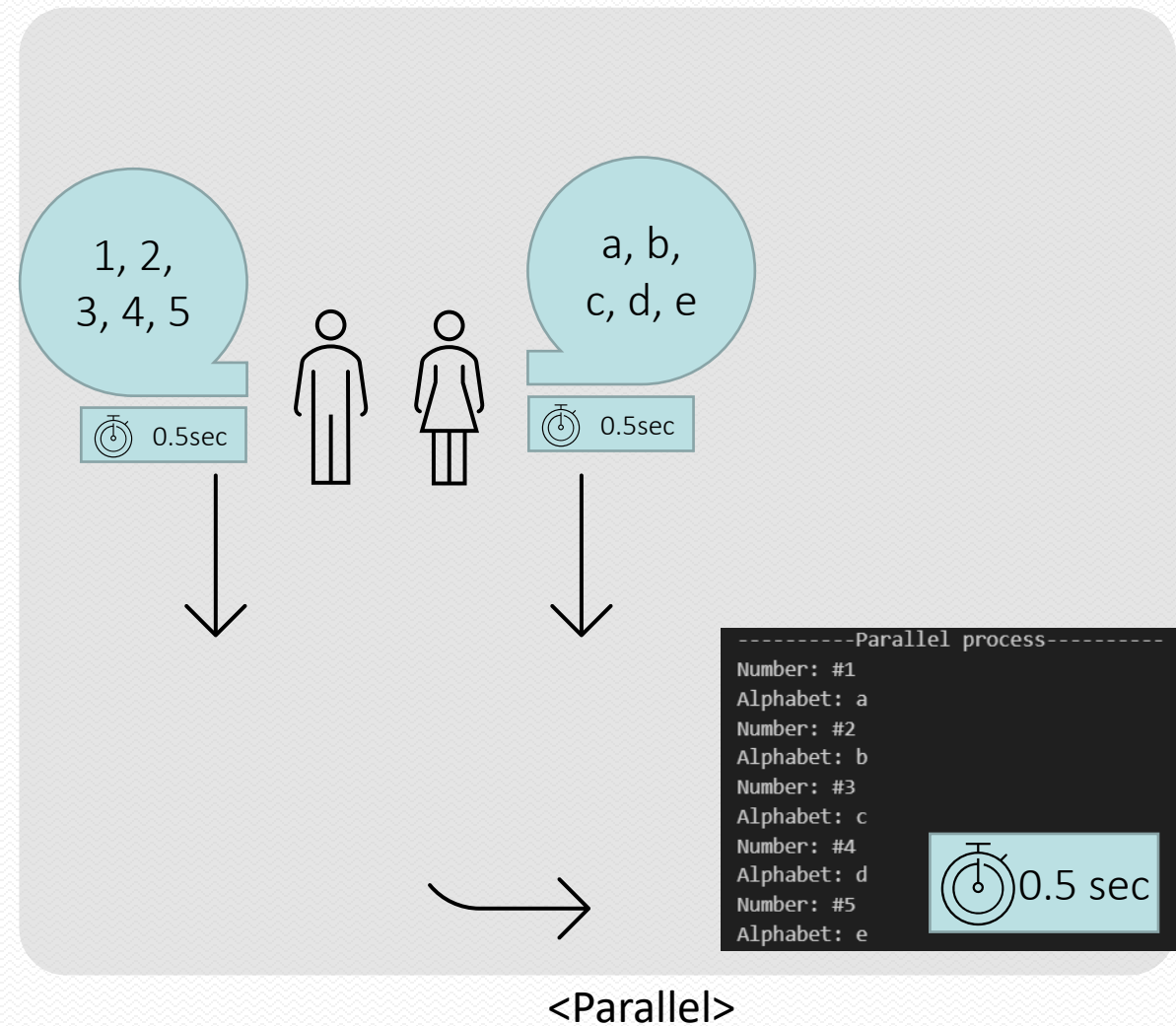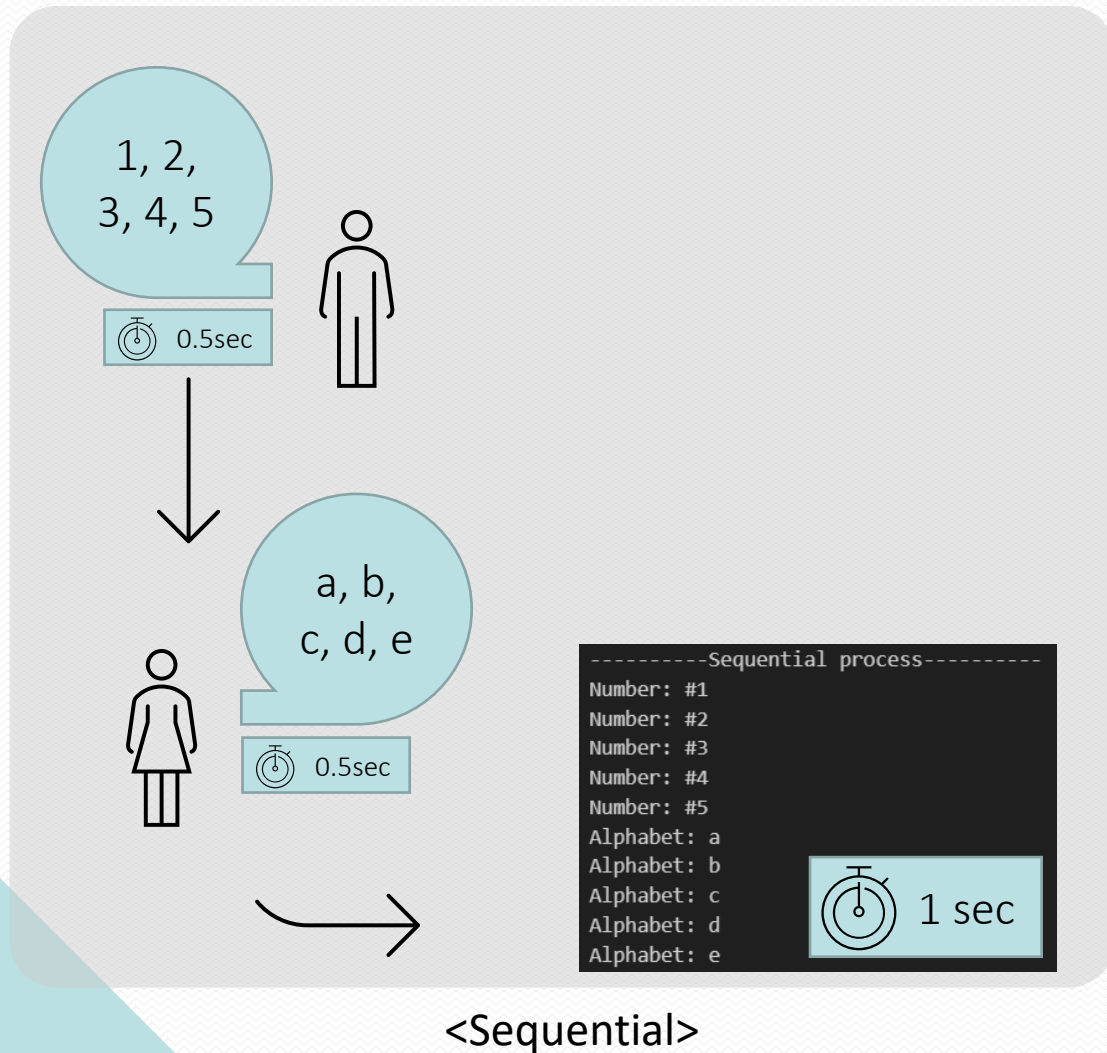Cropped Image

Parallelism

Parallelism

- Sequential vs Parallel

\<Sequential\>

\<Parallel\>

Practice Preparation

- Virtual environment setup
  $ cd Class02/python-parallelism
  $ python3 -m venv .venv
  $ source .venv/bin/activate
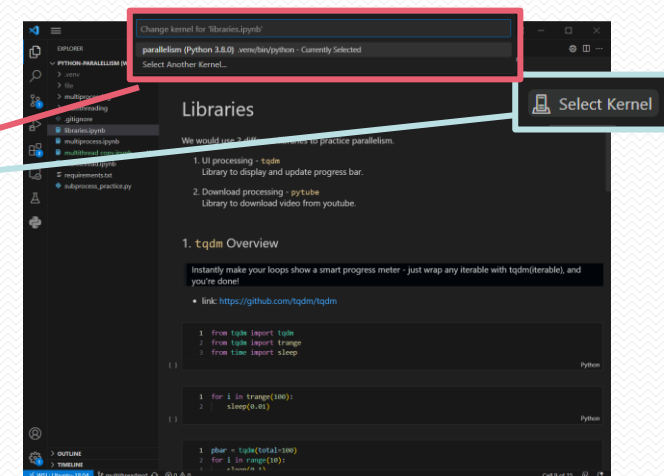 (.venv)$ pip install --upgrade pip
  Open **libraries.ipynb**
  Click 'Select Kernel'
  Selecte 'Install/Enable suggested extensions'
  (.venv)$ pip install -r ./requirements.txt
  (.venv)$ ipython kernel install --user --name=parallelism

  # 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

process	process
Code Data Files	Code Data Files
Register Stack	Register Stack

<Multi-processing>
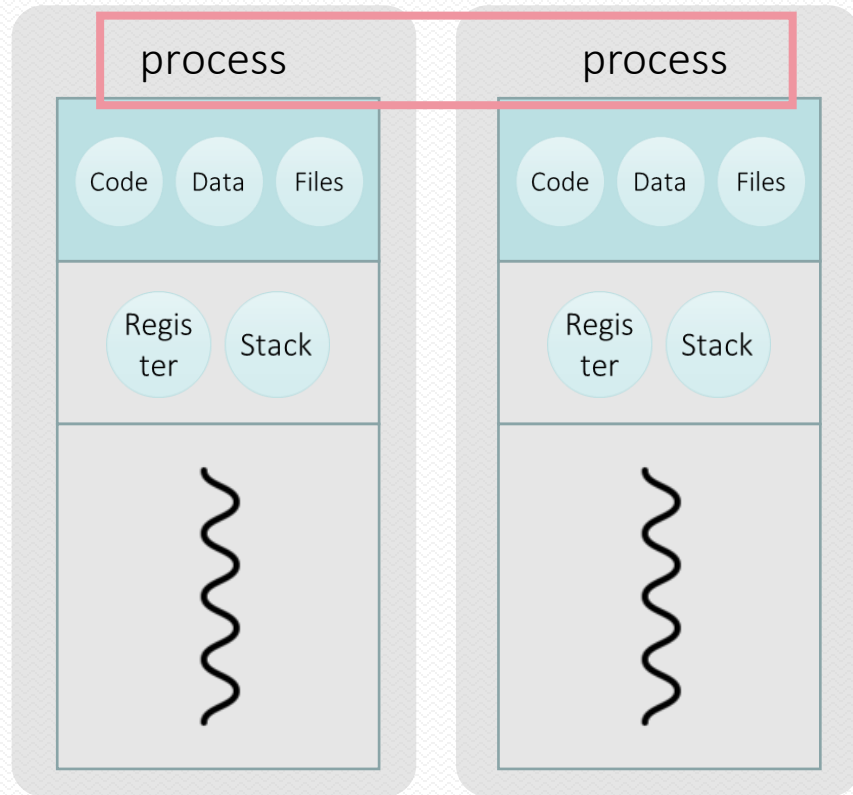
Use **single process**,
but run **multiple threads** inside

process

Shared data
Code   Data   Files

Non-shared data
Register Stack   Register   Stack

Thread

<Multi-threading>
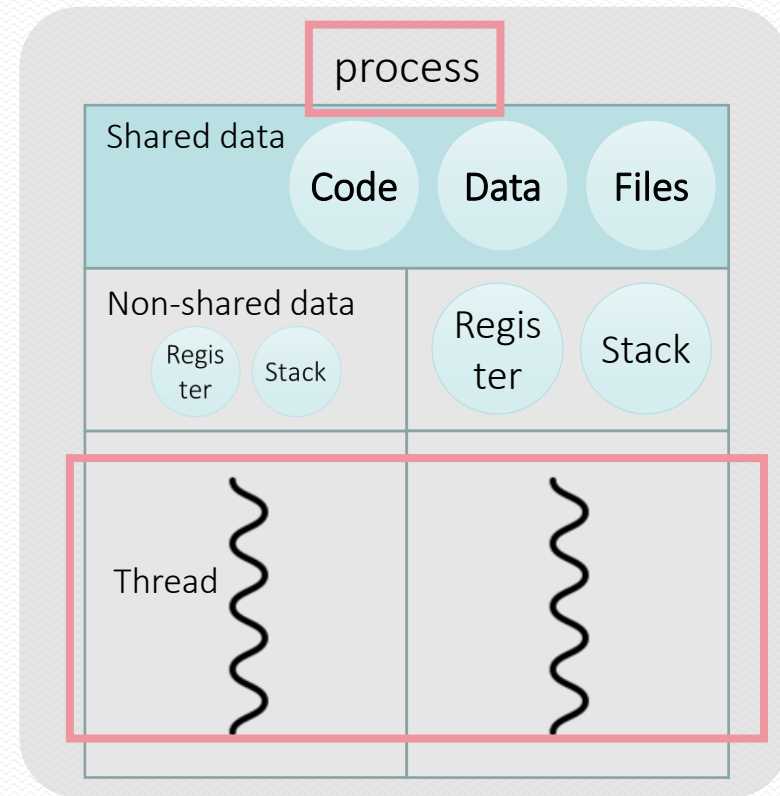
Multi-processing: Concept

- 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

process

| Code | Data | Files |

| Register | Stack |

process

| Code | Data | Files |

| Register | Stack |

\<Multi-processing\>
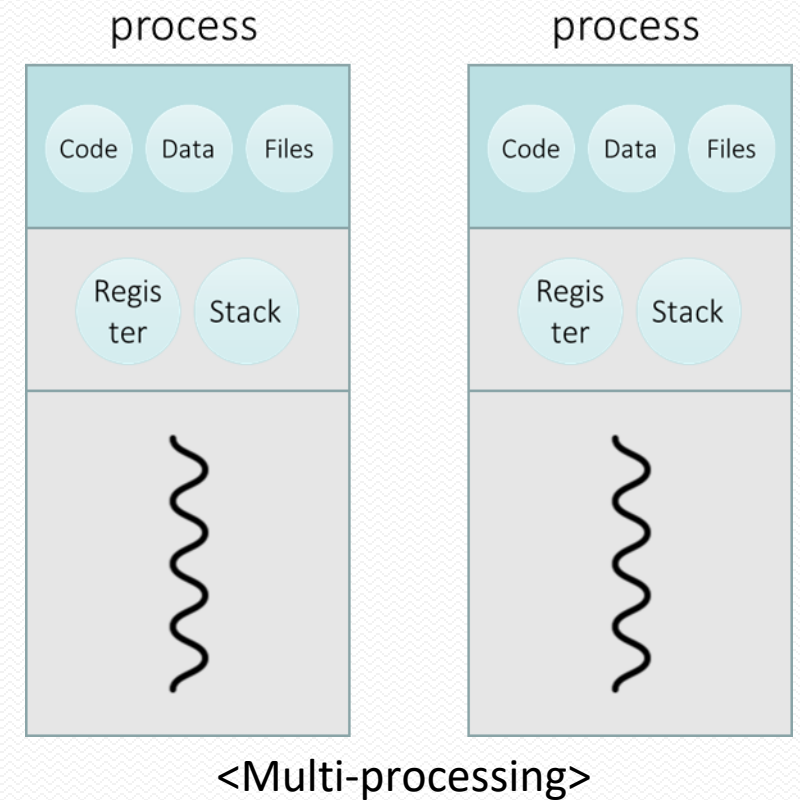
Multi-processing: Inter-process Communication(IPC)

<User space>

process
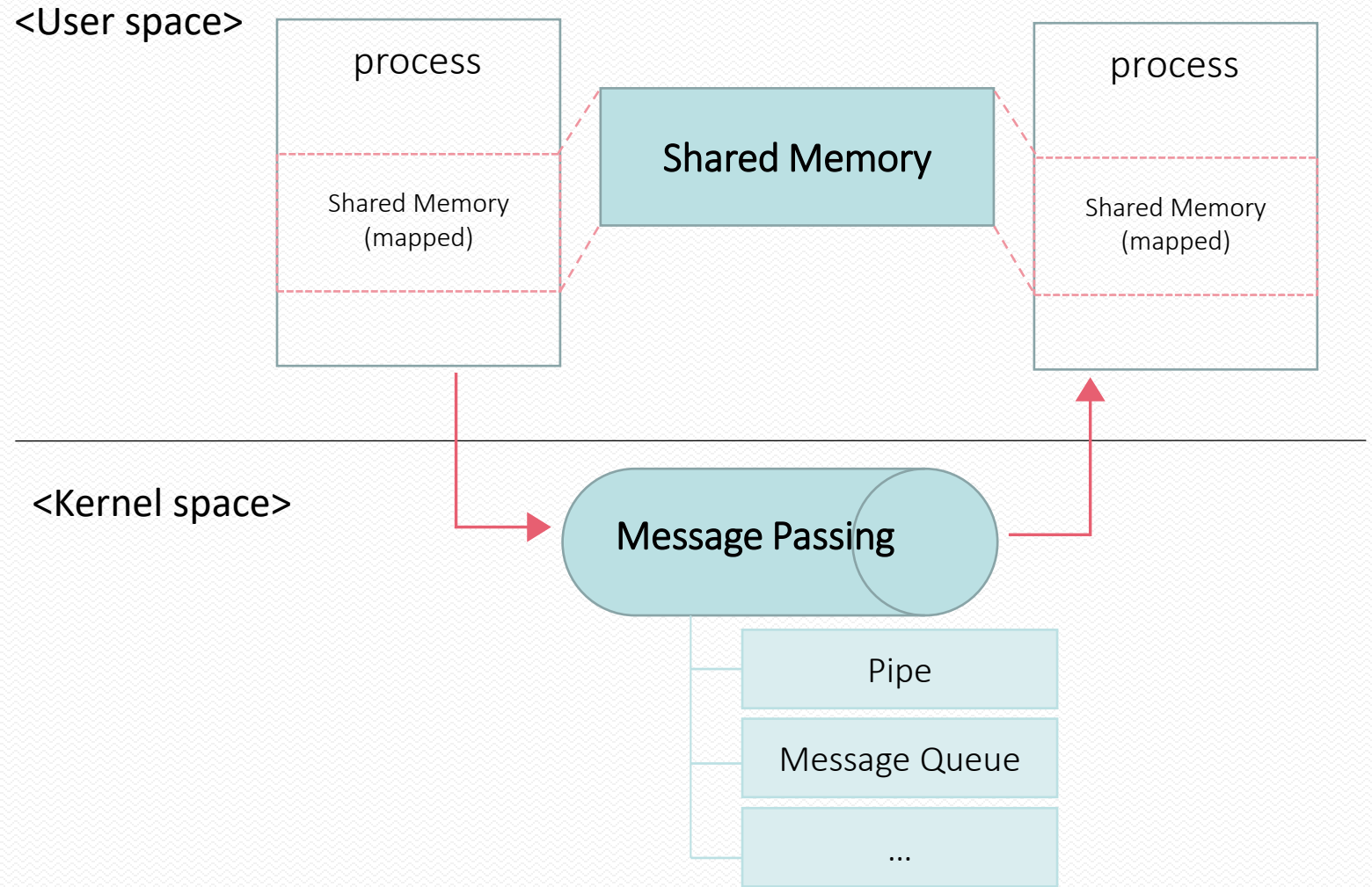
Shared Memory
(mapped)

Shared Memory

process

Shared Memory
(mapped)

<Kernel space>

Message Passing

| Pipe |
| Message Queue |
| ... |
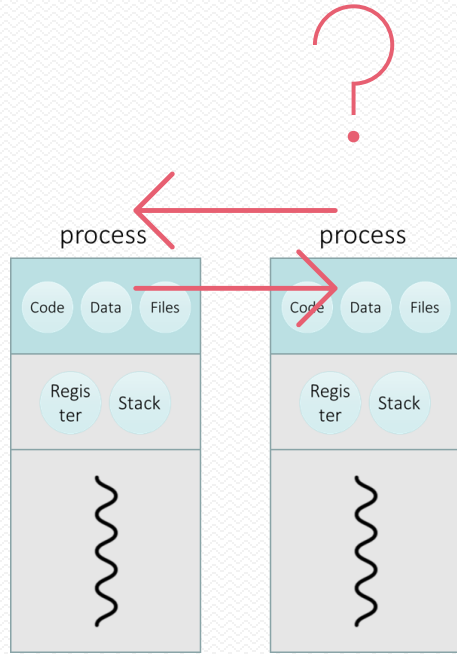
Multi-processing: Lifecycle of Process(multiprocessing.Process class)

start()

starts
the process

join()

waits until
the process
terminates

terminates
the process

terminate()

- Join을 걸지 않는다면?

New Process

run()

Running Process

wait()

(optional)
Blocked Process

return

return

Terminated Process

\<Parent Process\>

\<Child Process\>
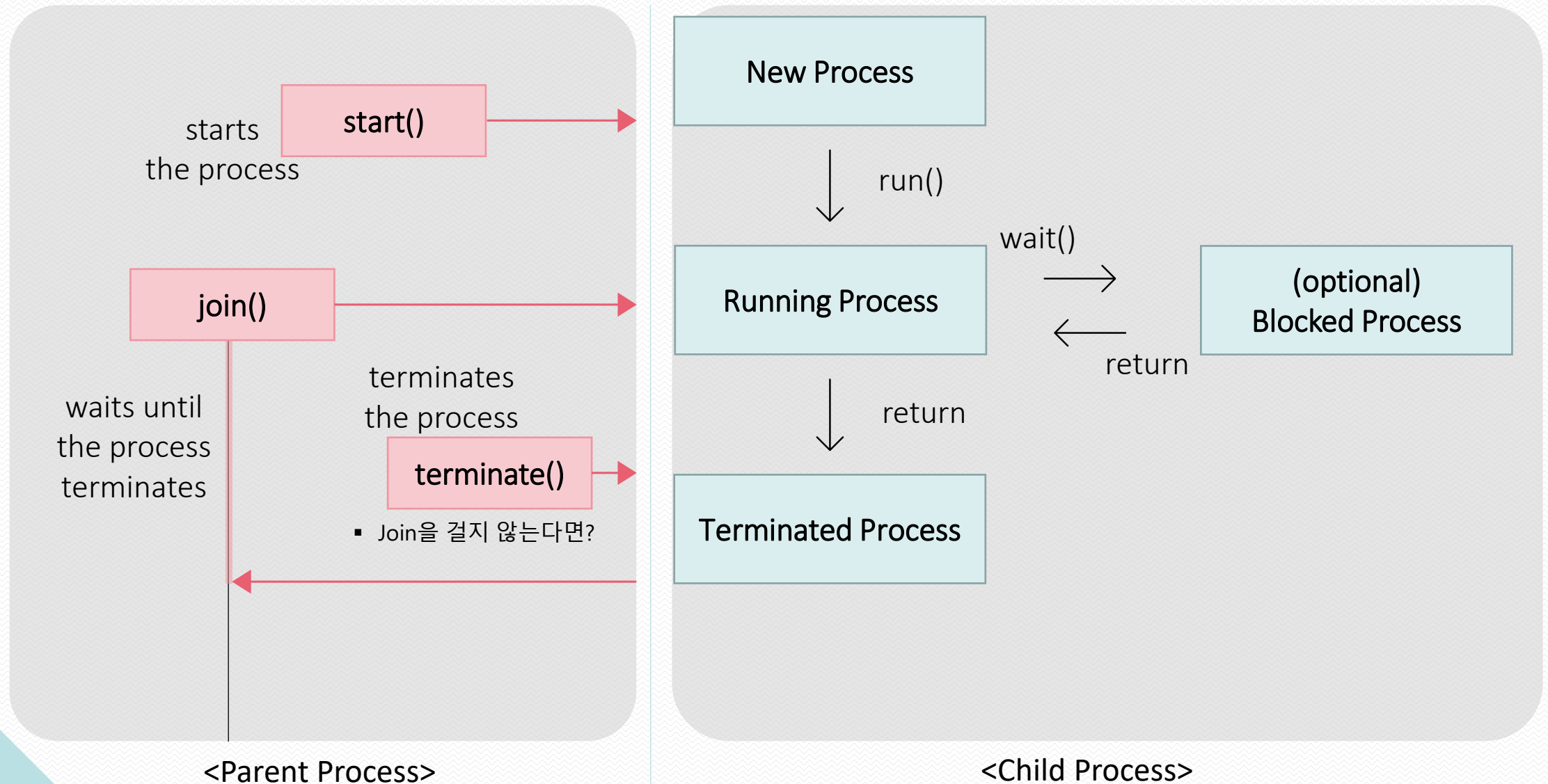
Multi-processing: example

- `pytube`, `tqdm` example

Reference: https://docs.python.org/3/library/multiprocessing.html#multiprocessing-programming
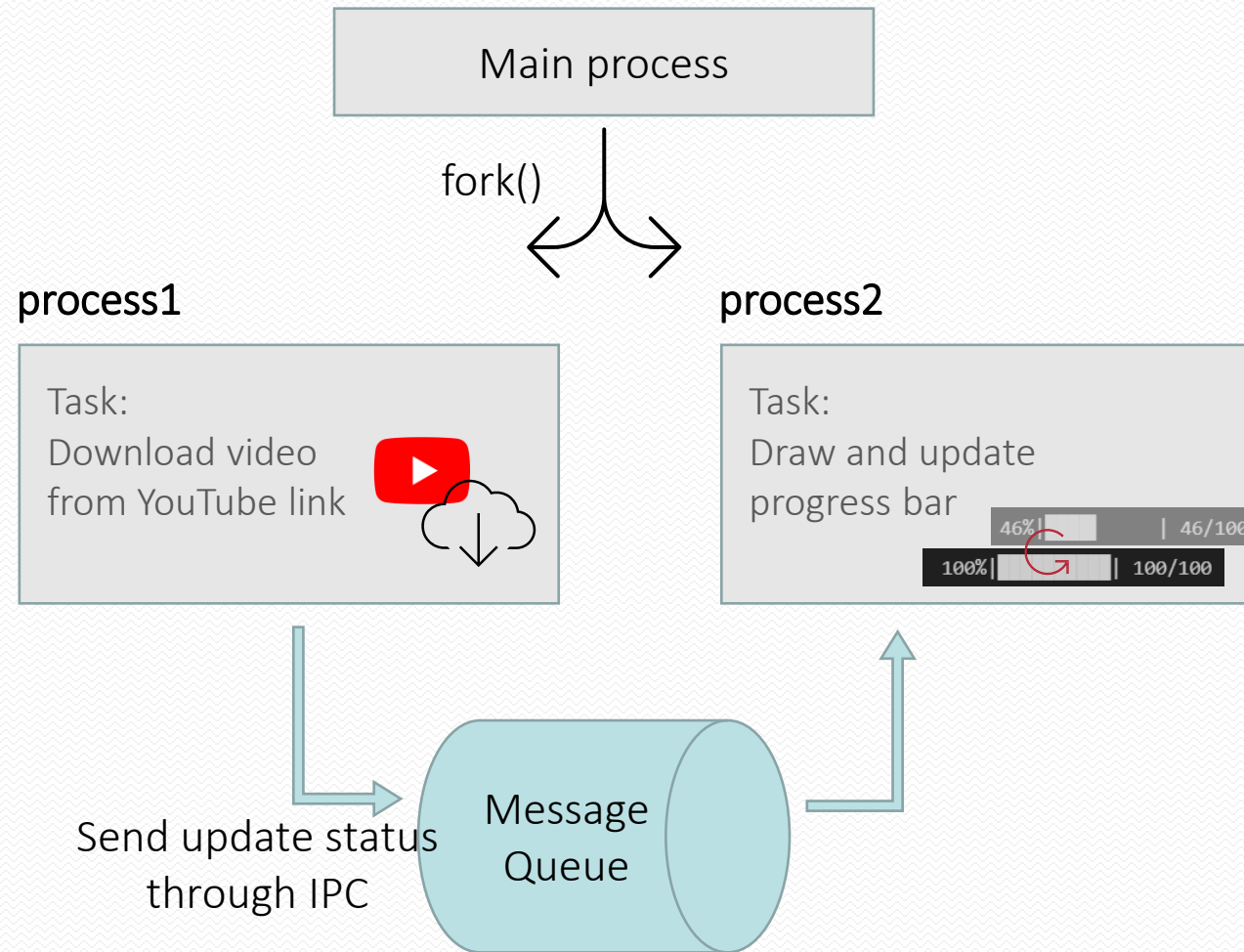
Multi-threading: Concept

- A light-weight process

- Utilize **single process** to execute multiple tasks
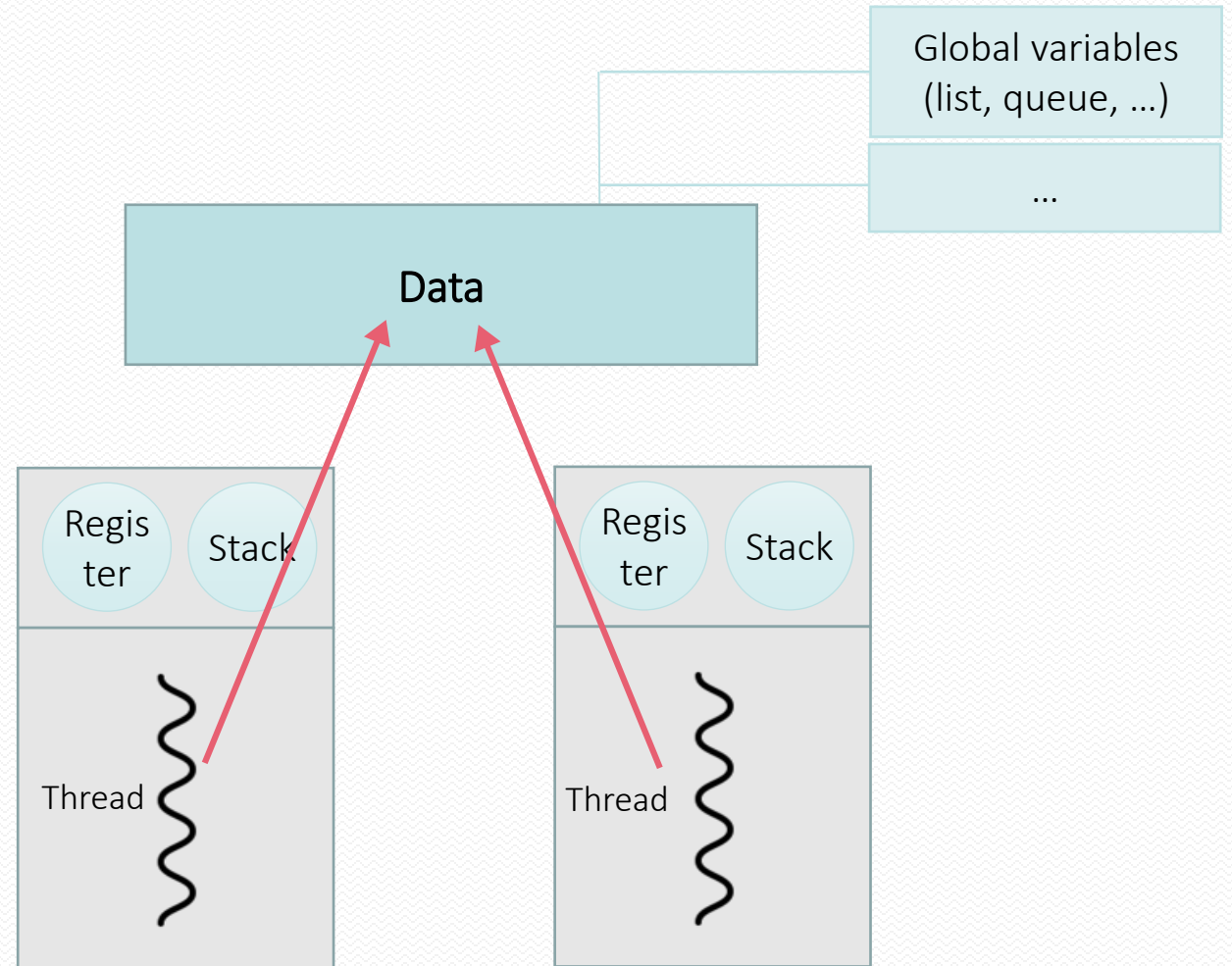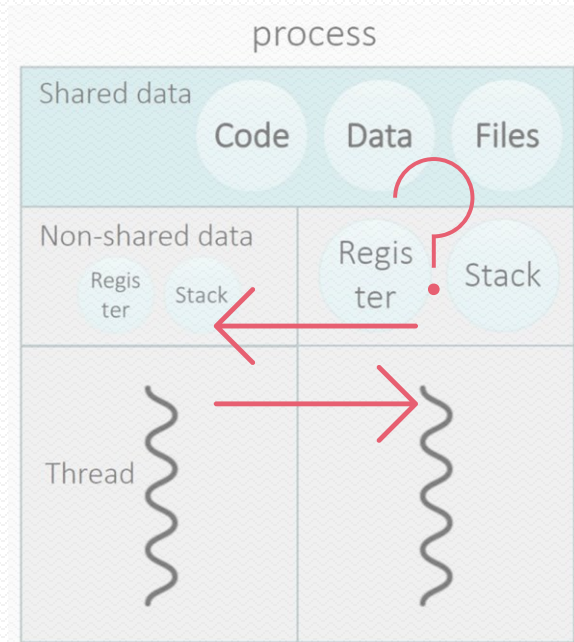
- Process consists of **multiple threads**, where each threads represents each tasks

- Threads in one process share memory area, in terms of code, data segments and heap

process

Shared data			
	Code	Data	Files

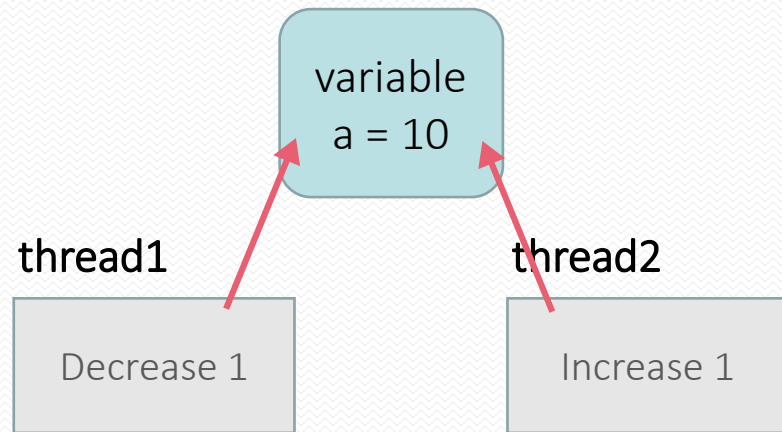| Non-shared data			
Register	Stack	Register	Stack

Thread	

<Multi-threading>
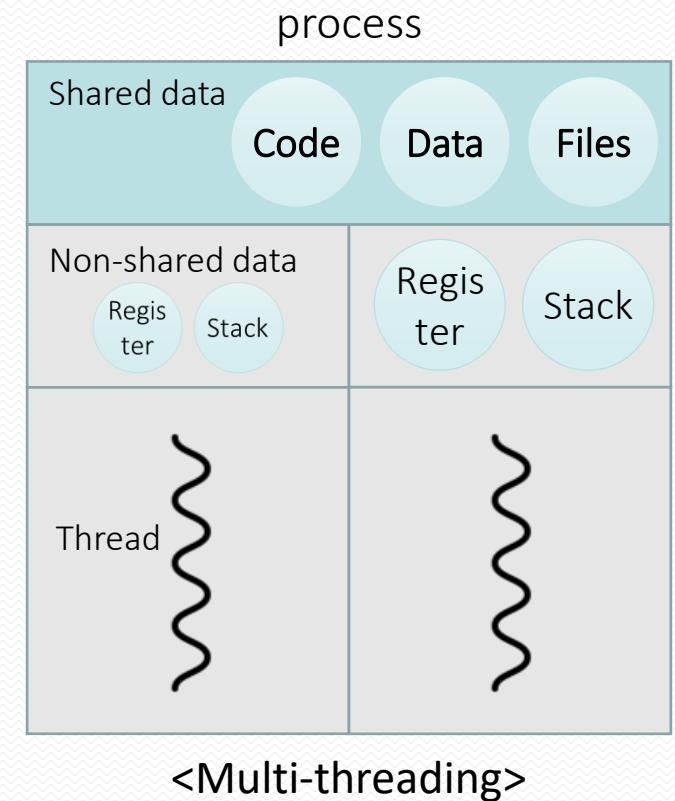
Multi-threading: Communication between threads

GIL(Global Interpreter Lock)

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

variable
a = 10

thread1

thread2

Decrease 1

Increase 1

<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
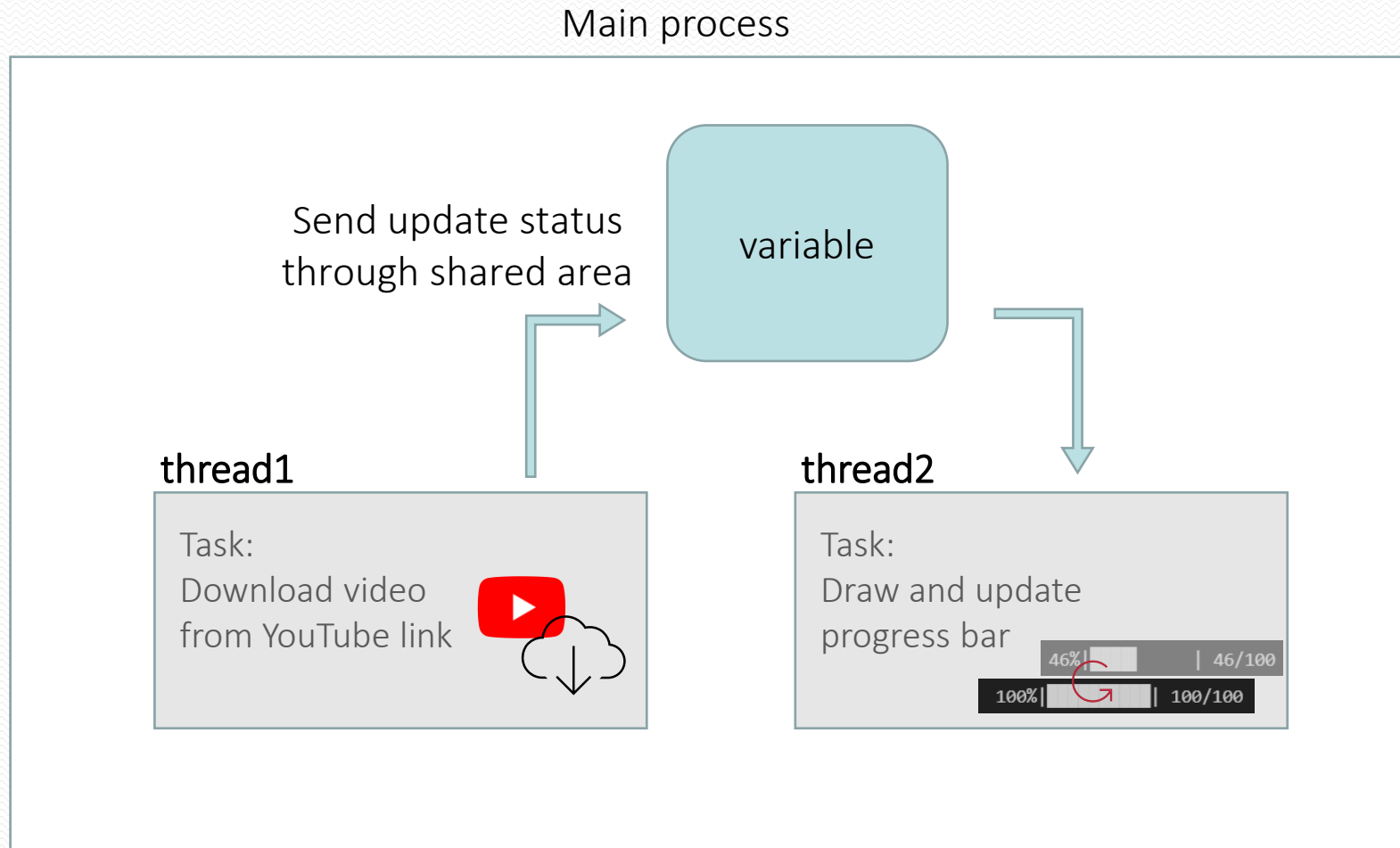
Multi-threading: Problem & Solution

Shared Data

Regis
ter
Stack

Regis
ter
Stack
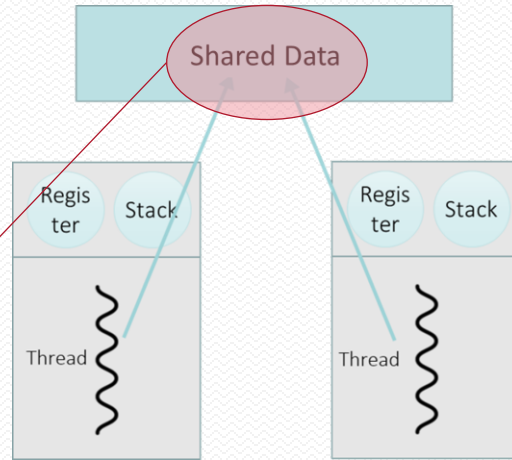
Thread

Thread

- Synchronization Problem
  - When multiple threads attempt to access to shared area at the same time

- Solution
  - Mutex(Mutual Exclusion)

Thread1   Mutex   Thread2

lock

attempt: fail

wait…

unlock   lock   wake()

unlock

<Mutex>
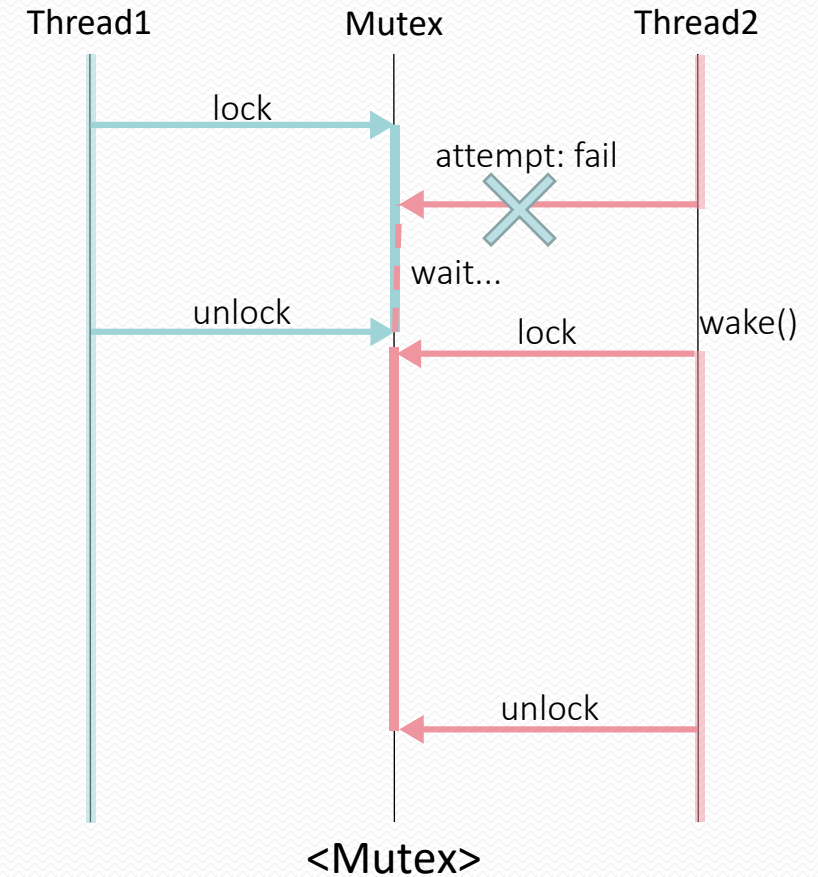
Multi-threading: Example

- `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

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

THREAD 1
(Video1)

THREAD 2
(Video2)

*Queue the camera stream,
receive it from the main
and process it.
Running 'imshow' on each thread
can cause crash issues!*

QUEUE
BUFFER

MAIN
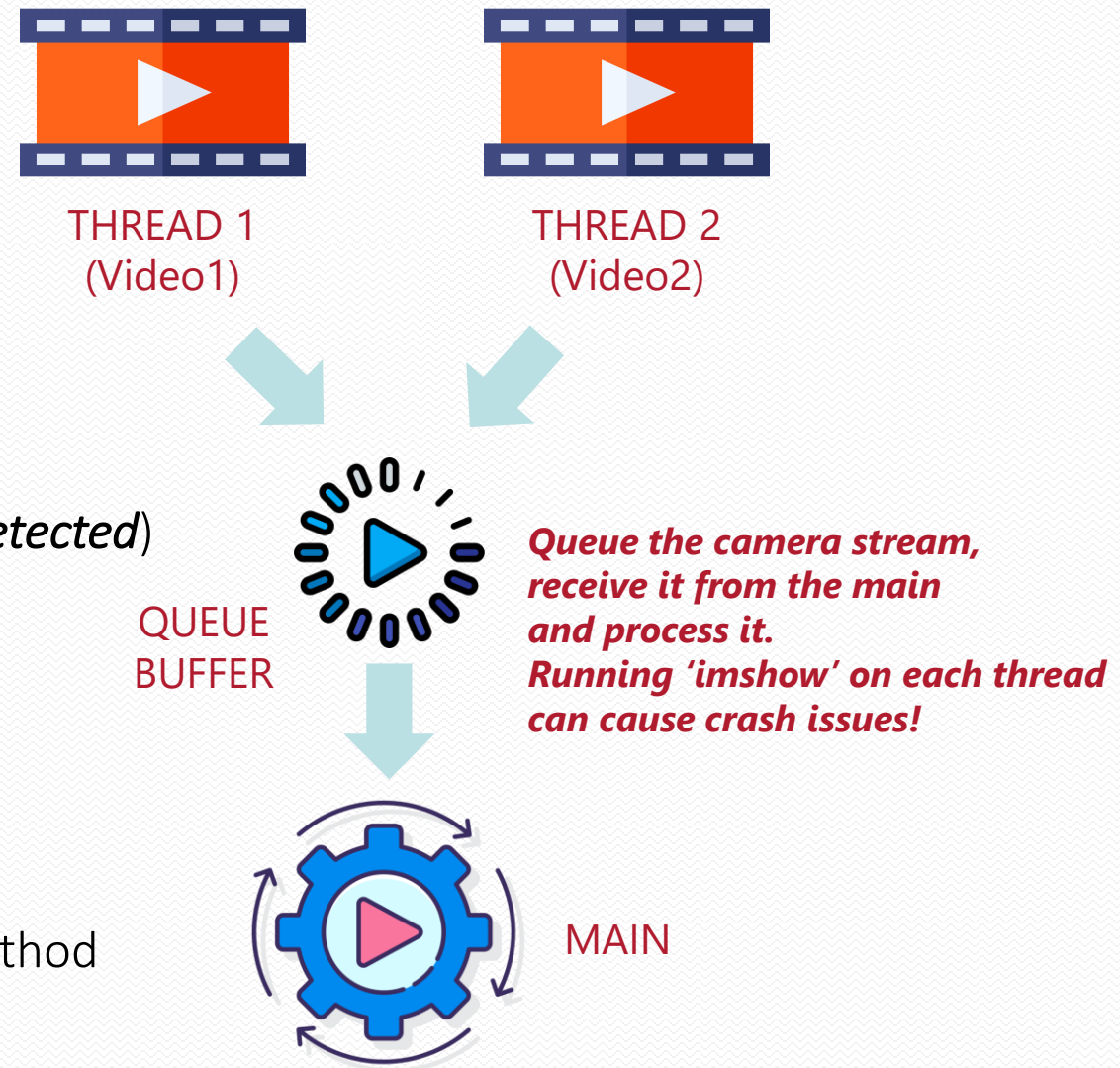
Multi-thread Hands-on (cont,)

THANK YOU