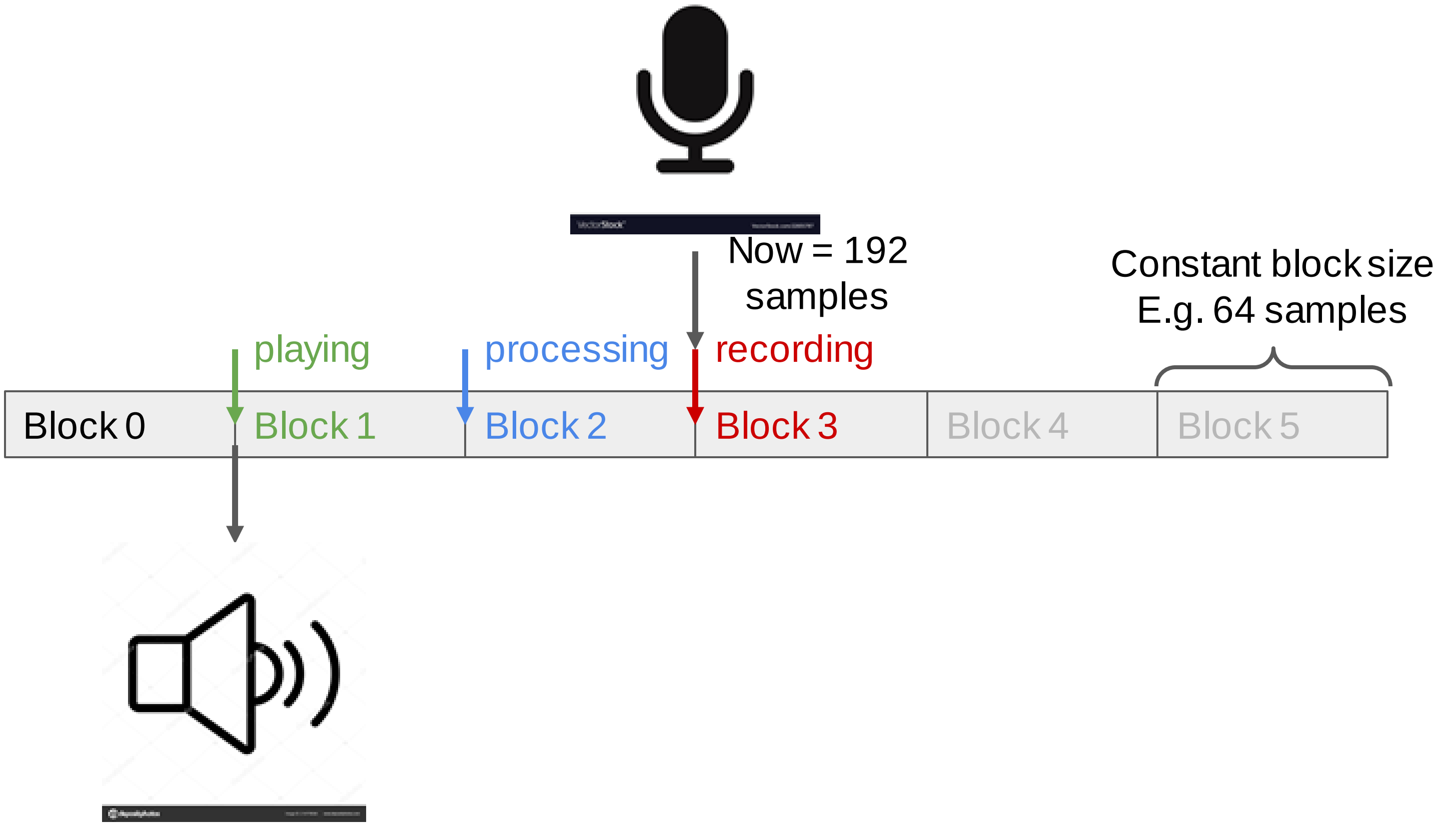
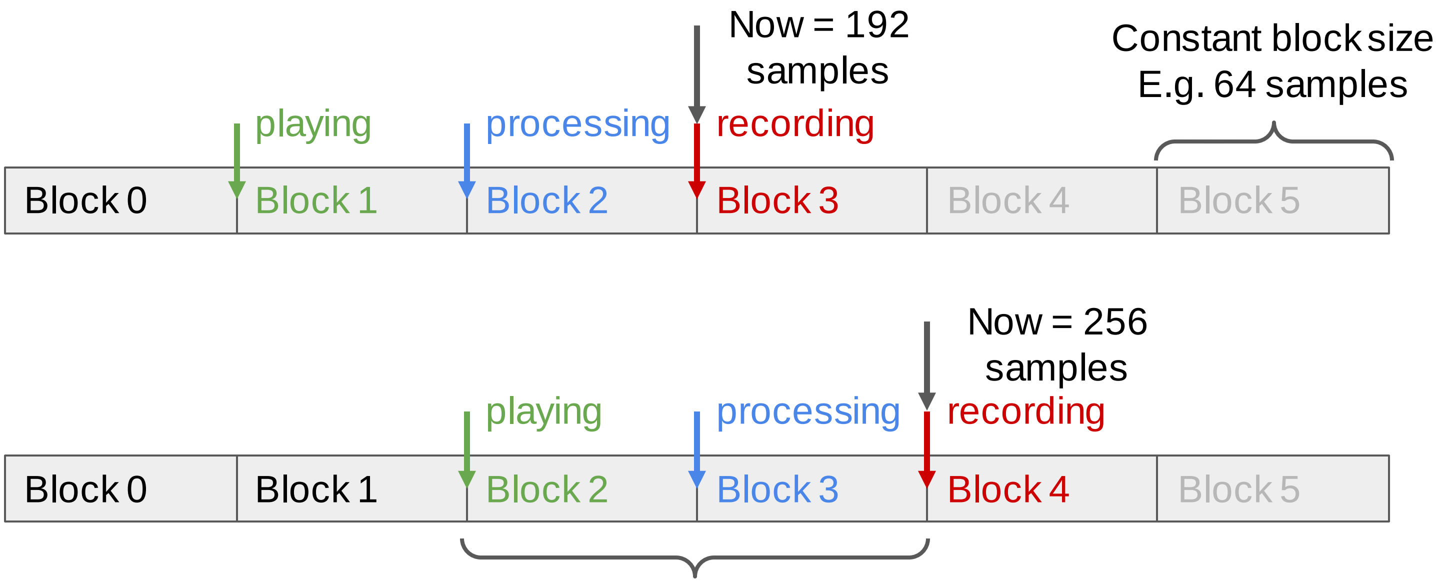
Real-time audio

Block-based audio processing in python

Block-based processing

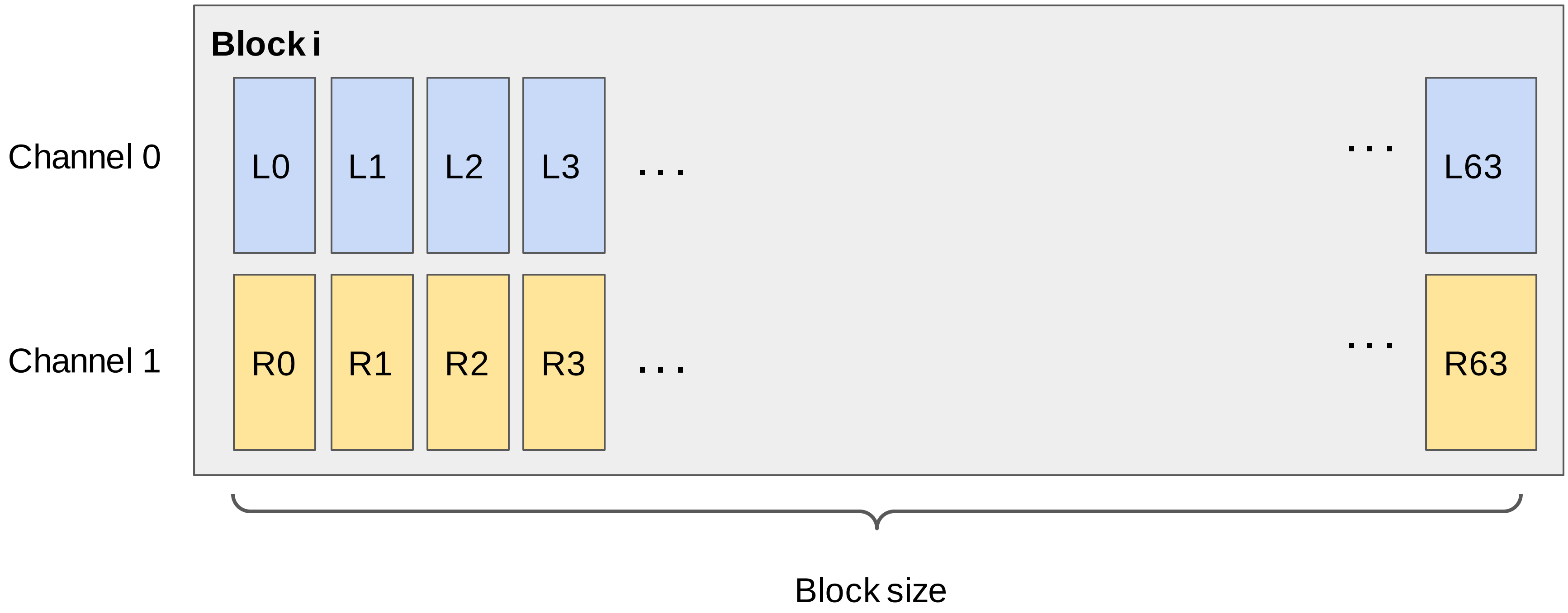


Block-based processing (latency)



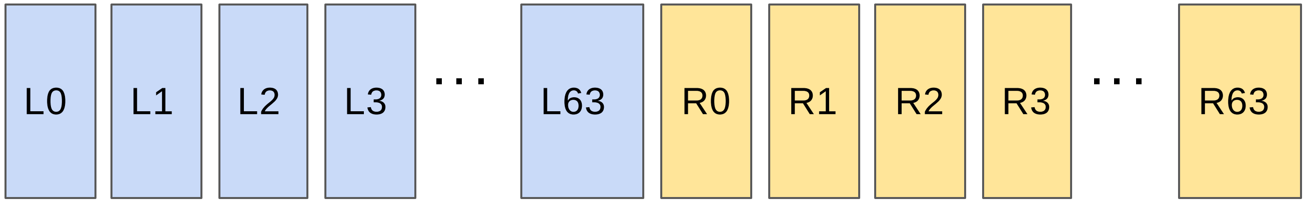
Latency = 2*block size

Inside a block

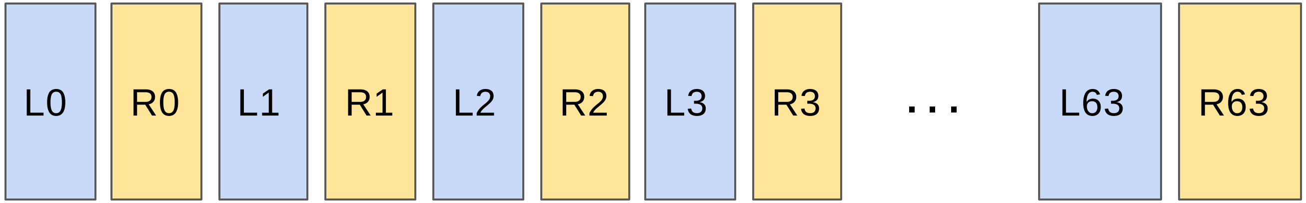


Block in computer memory

Non-interleaved

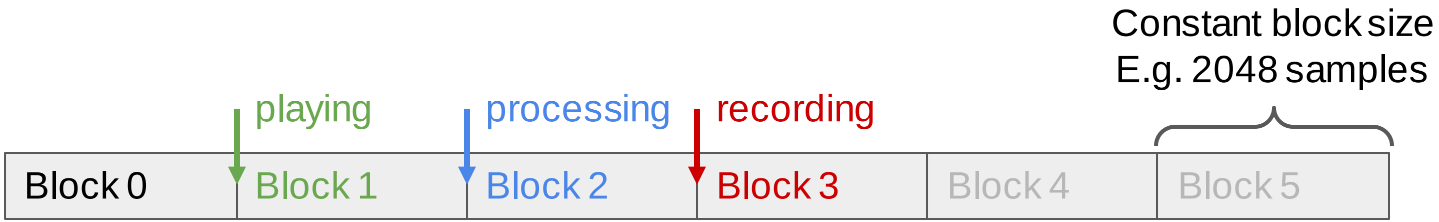


Interleaved



This information depends on the sound card and / or the audio files themselves. In class we will be using either 16bit monophonic files or monophonic microphone input (no need for panning information).

Block-based processing in Python




```
import pyaudio

WINDOW_SIZE = 2048 # or block size
CHANNELS = 1 # for stereo we would need 2
RATE = 44100
```

```
p = pyaudio.PyAudio()
output = p.open(format=pyaudio.paInt16, # 16bit
                channels=CHANNELS,
                rate=RATE,
                output=True,
                input=True, # default is False
                frames_per_buffer=WINDOW_SIZE,
                stream_callback=callback)
```

```
def callback( in_data, frame_count, time_info, status):
    n = np.frombuffer( in_data , dtype='int16' )
    to_play = np.zeros( (n.size , CHANNELS) , dtype='int16' )
    # process in_data ...
    # 0 is left, 1 is right speaker / channel
    to_play[:,0] = n
    return (to_play, pyaudio.paContinue)
```

input=True produces a block of data from the microphone that eventually becomes in_data, which is the block under processing and this eventually becomes the block to_play, when the previous (processed) block has finished playing.