

# Live de Python #14

## Threads

# Roteiro

- Processos
- Concorrencia
- Scheduler
- Paralelismo
- GIL
- Classe Threads
- CODE!!!!



# Processos (linux)

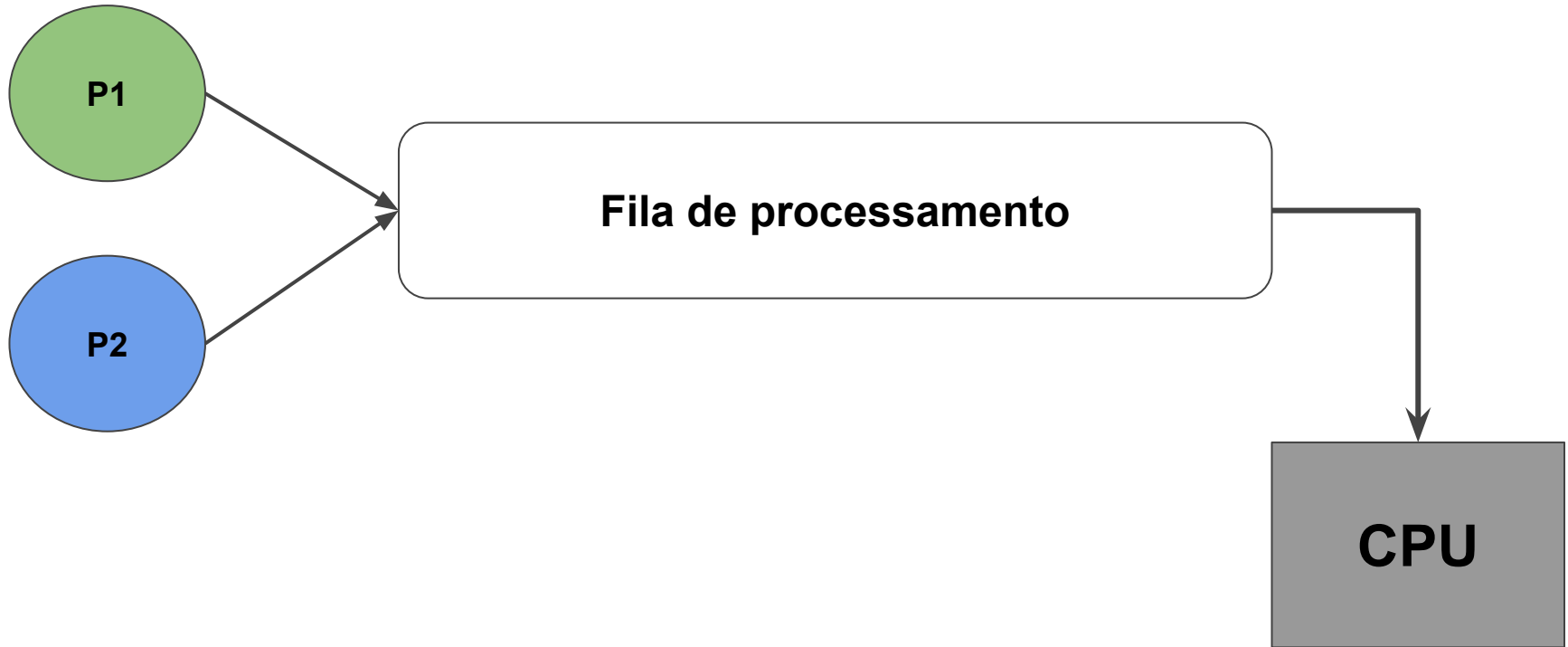
- Estado
  - Rodando
  - Esperando
  - Parado
  - zumbi
- Informação do scheduler (-20/+20)
  - Prioridade, por exemplo
- PID (identificação do processo)
  - Um número que o identifica
- (...)



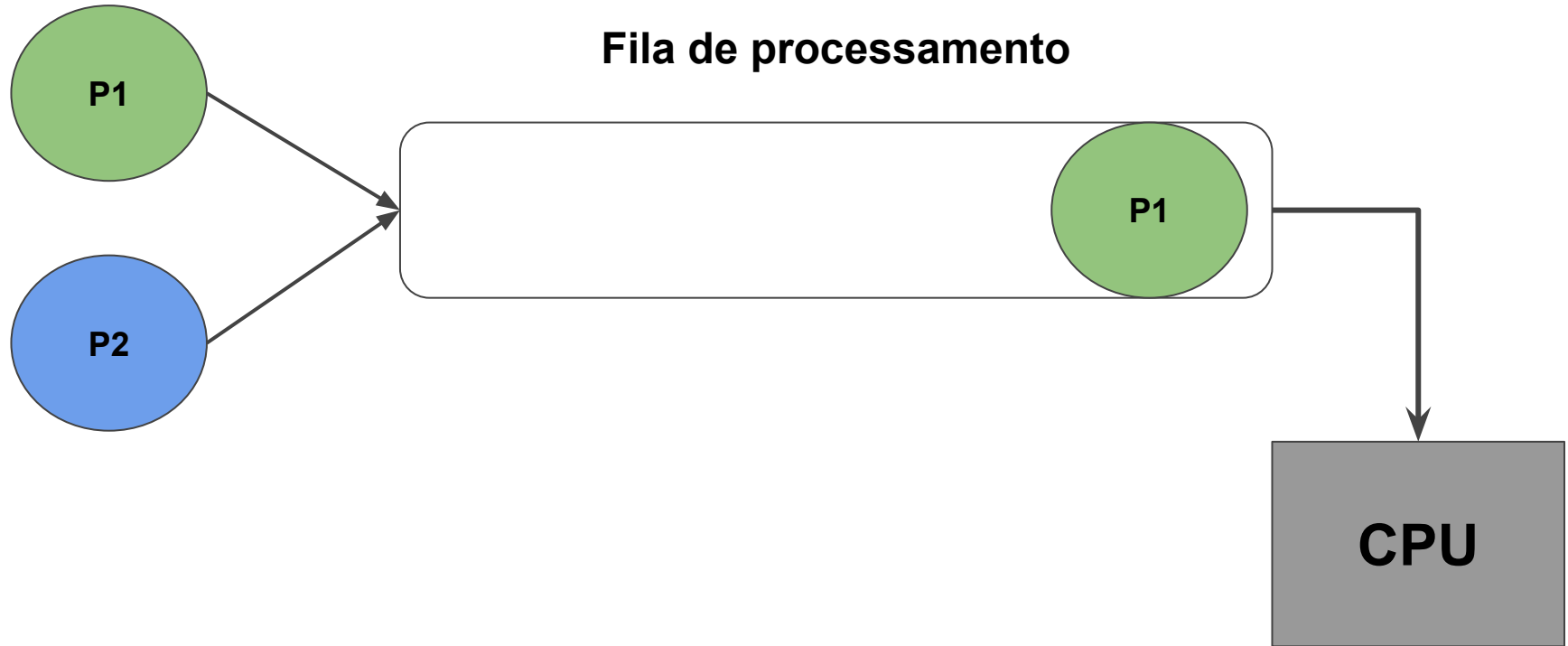
# Processos (linux)

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+
1398	z4r4tu5tr	20	0	451M	57304	19220	S	0.0	0.7	0:00.00
1283	z4r4tu5tr	20	0	258M	5028	4348	S	0.0	0.1	0:00.01
1286	z4r4tu5tr	20	0	258M	5028	4348	S	0.0	0.1	0:00.00
1284	z4r4tu5tr	20	0	258M	5028	4348	S	0.0	0.1	0:00.00
1216	z4r4tu5tr	20	0	350M	10976	9516	S	0.0	0.1	0:00.08
1218	z4r4tu5tr	20	0	350M	10976	9516	S	0.0	0.1	0:00.01
1217	z4r4tu5tr	20	0	350M	10976	9516	S	0.0	0.1	0:00.00
1205	z4r4tu5tr	9	-11	1638M	14892	11372	S	0.7	0.2	0:04.98
1213	z4r4tu5tr	-6	0	1638M	14892	11372	S	0.0	0.2	0:00.00
1212	z4r4tu5tr	-6	0	1638M	14892	11372	S	0.7	0.2	0:02.98
1172	z4r4tu5tr	20	0	181M	5388	4172	S	0.0	0.1	0:00.29

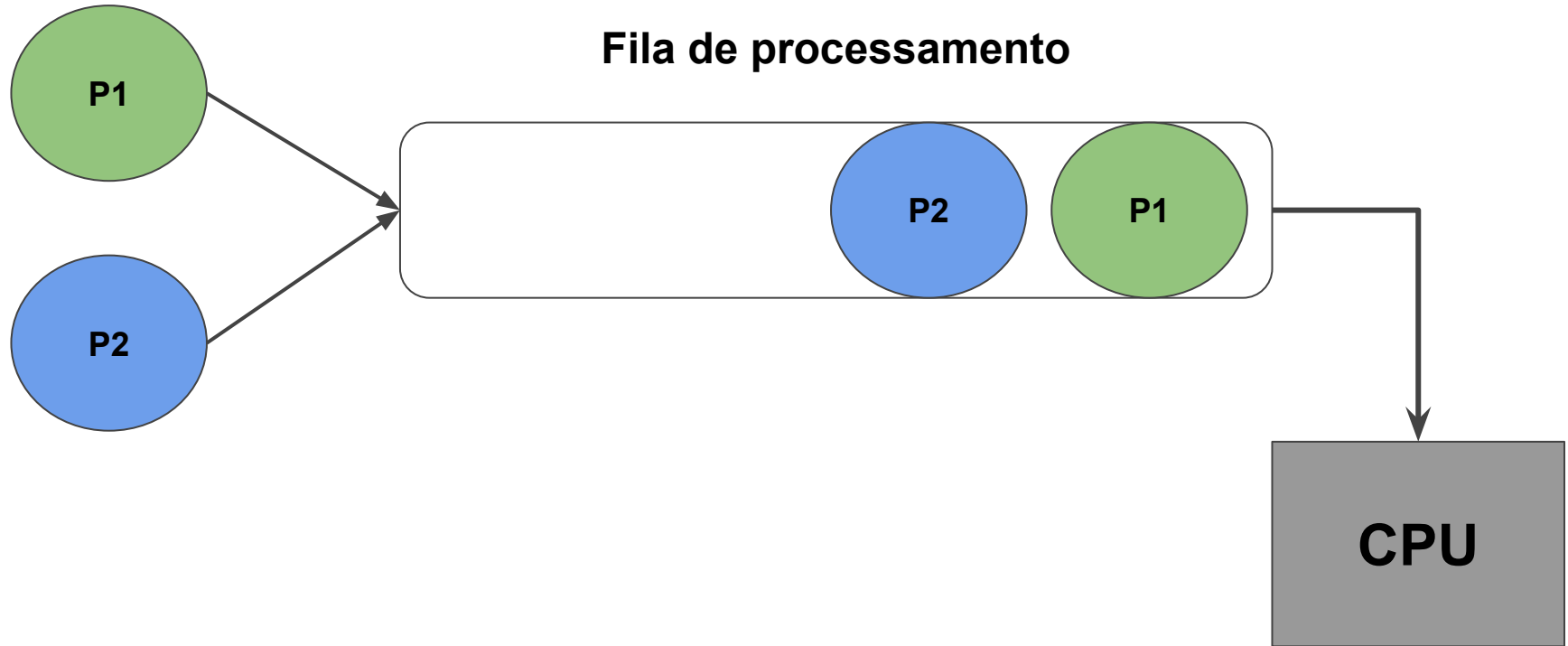
# Concorrência [0]



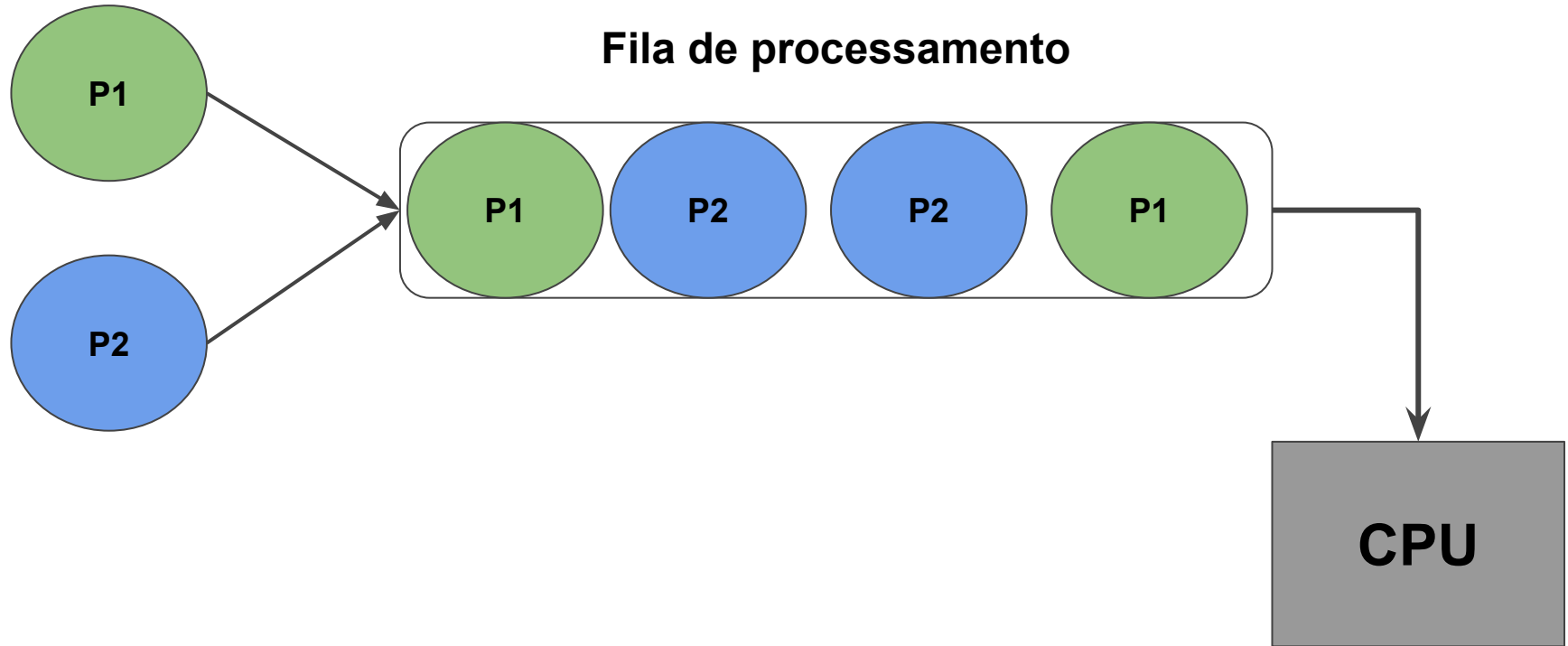
# Concorrência [1]



# Concorrência [2]

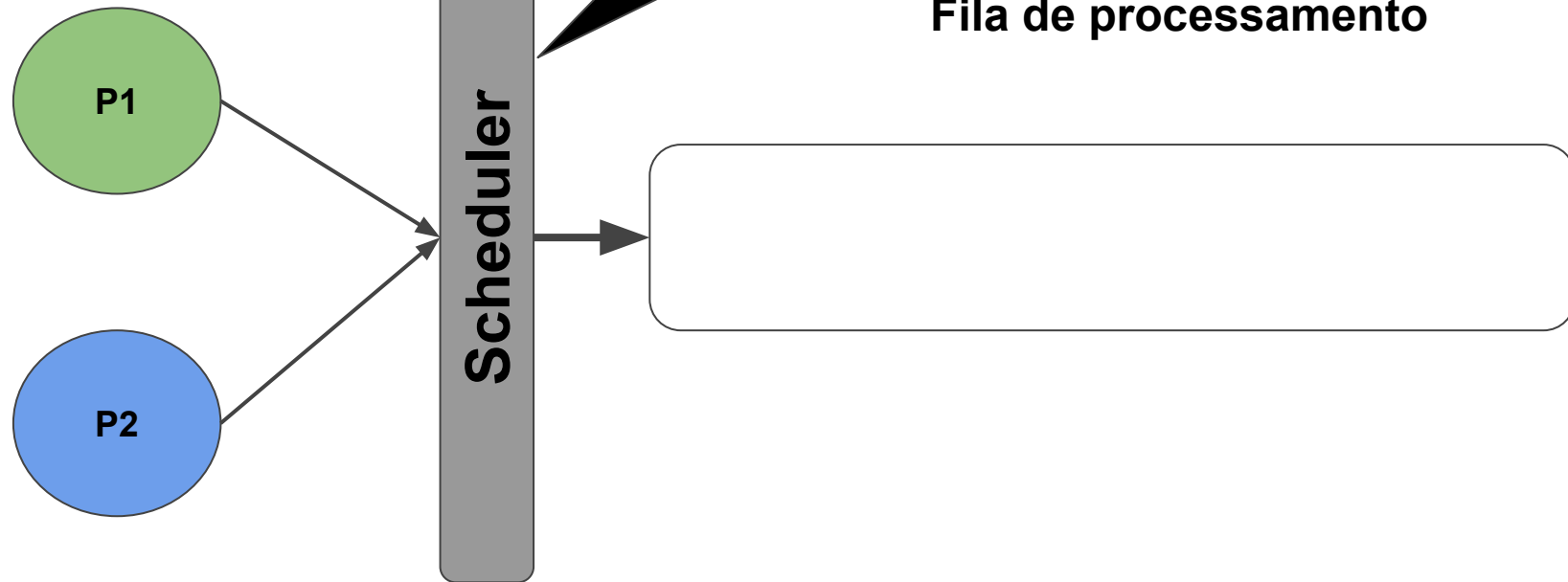


# Concorrência [3]

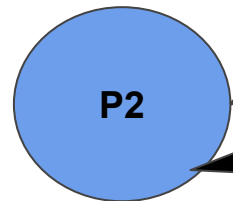
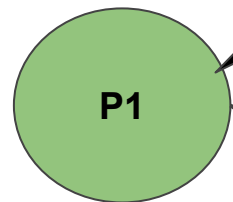




Scheduler [0]



Schedul +20



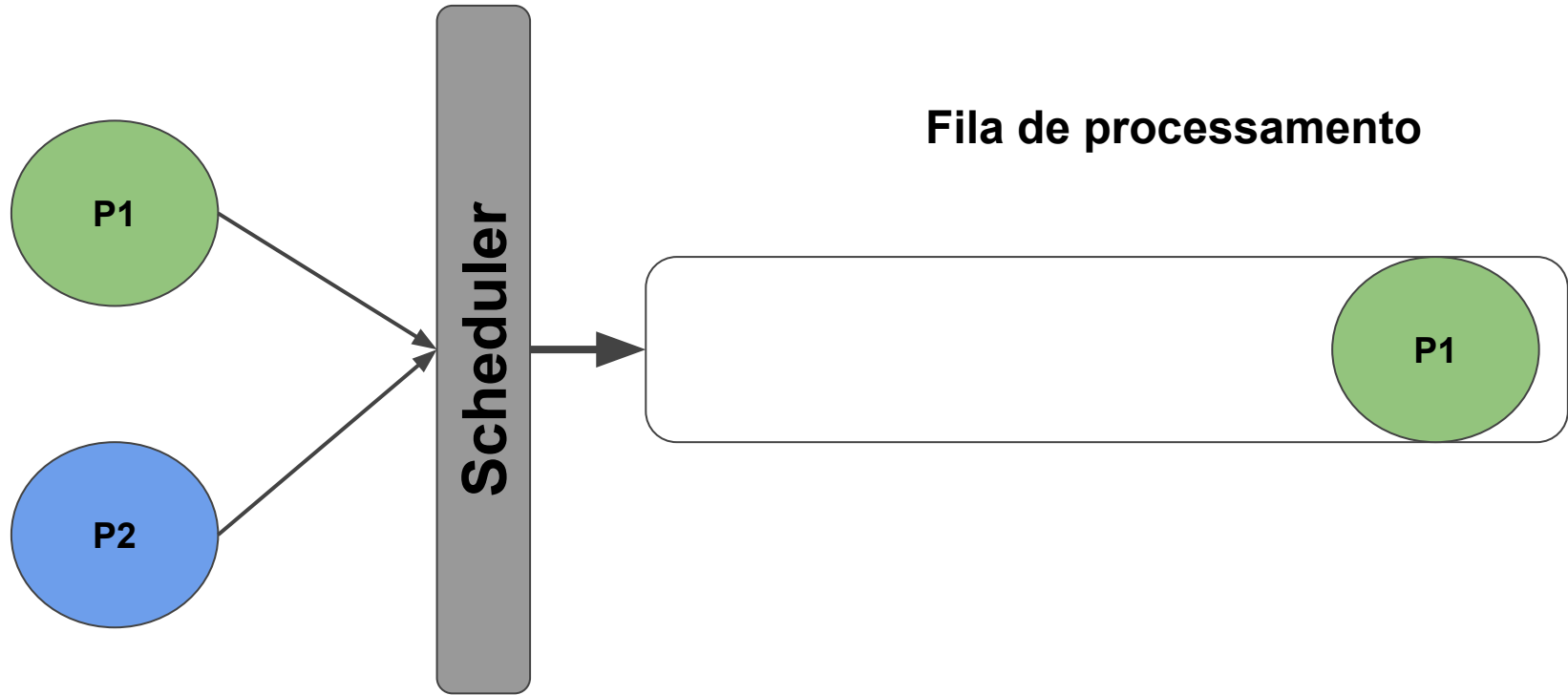
Scheduler

Fila de processamento

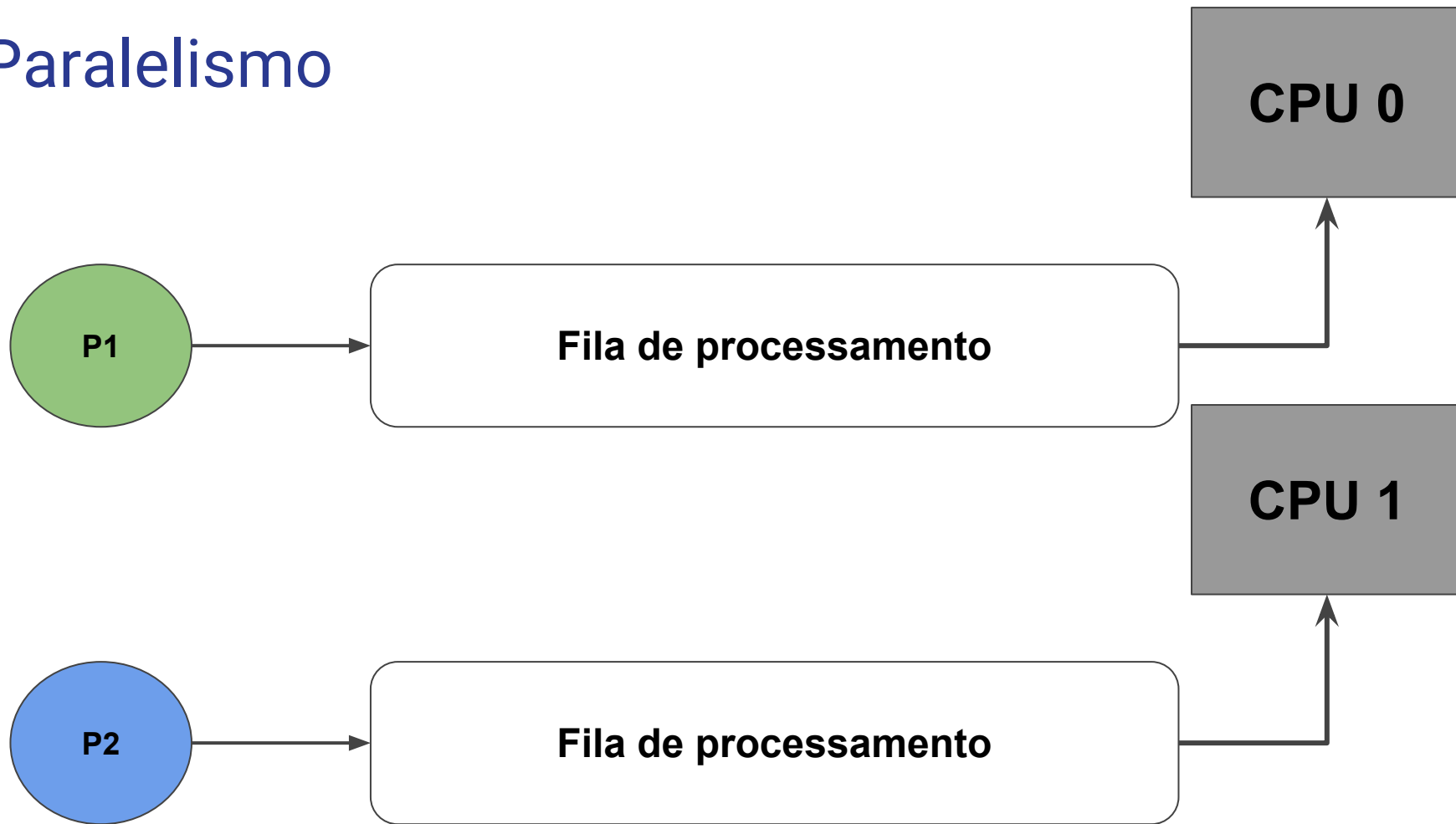


0

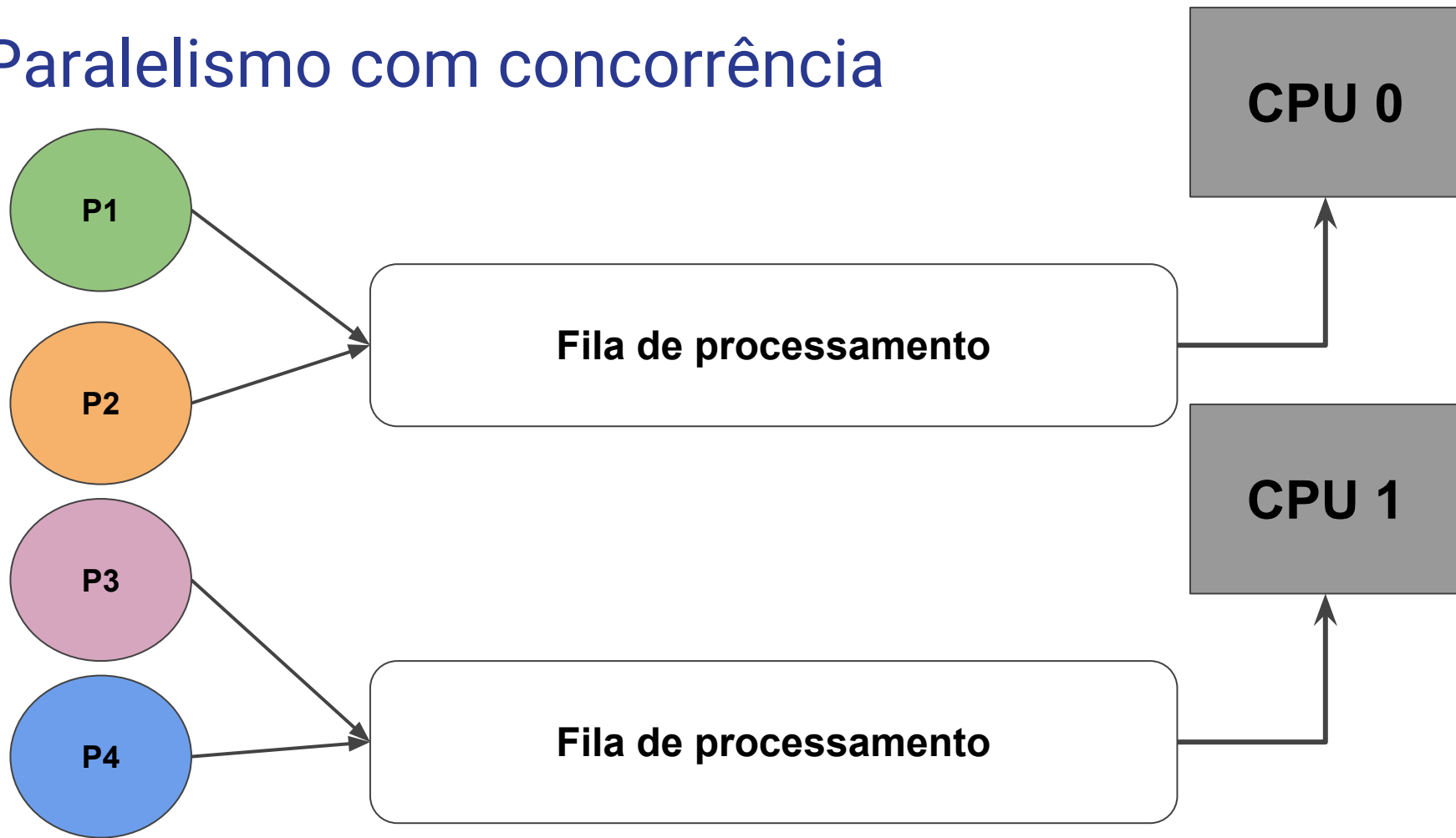
## Scheduler [2]



# Paralelismo



# Paralelismo com concorrência



# GIL

Global Interpreter Lock

## Regras sobre o GIL

1. Você nunca deve falar sobre o GIL
2. Você NUNCA deve falar sobre o GIL
3. Você nunca deve nem se quer mencionar o GIL

---

# GIL

- Trava mutuamente excludente (mutex)????
- Impedir que o recurso de multitarefa preemptiva do sistema operacional (não deixar que as threads tomem conta do processo pai)
- Ou seja, uma Thread por vez (Concorrência)



# Exemplo (Slatkin - 2016) [0]

```
numbers = [2139079, 1214759, 1516637, 1852285]

• def factorize(number):
    for i in range(1, number + 1):
        if number % i == 0:
            yield i

start = time()
for number in numbers:
    list(factorize(number))
end = time()

print(f'{end-start}') # ~ 0.4634
```



# Exemplo (Slatkin - 2016) [0]

```
• class FactorizeThread(Thread):  
•     def __init__(self, number):  
        super().__init__()  
        self.number = number  
  
•     def run(self):  
        self.factors = list(factorize(self.number))
```

```
start = time()  
threads = []  
for number in numbers:  
    thread = FactorizeThread(number)  
    thread.start()  
    threads.append(thread)  
  
for thread in threads:  
    thread.join()  
  
end = time()  
|  
print(f'{end-start}') # ~ 0.7339
```

??????

- **Sequencial** ->  $\sim 0.45$
- **“Paralelo”** ->  $\sim 0.73$



# Threads

`threading.Thread()`

- Representa uma atividade que será executada em um fluxo separado
- Pode ser iniciada sobrescrevendo o método 'run()' ou chamada por um invocável (função ou `__call__`)
- Para iniciar uma thread chamamos o método 'start()'
- (3.3) A thread pode ser um daemon ou não (finalizar quando o processo python é finalizado)



# Threads

`threading.Thread()`

- `is_alive`
  - Determina se a thread está em atividade ou não
- `join`
  - Bloqueia a thread (não é uma finalização)



CODE !!!!