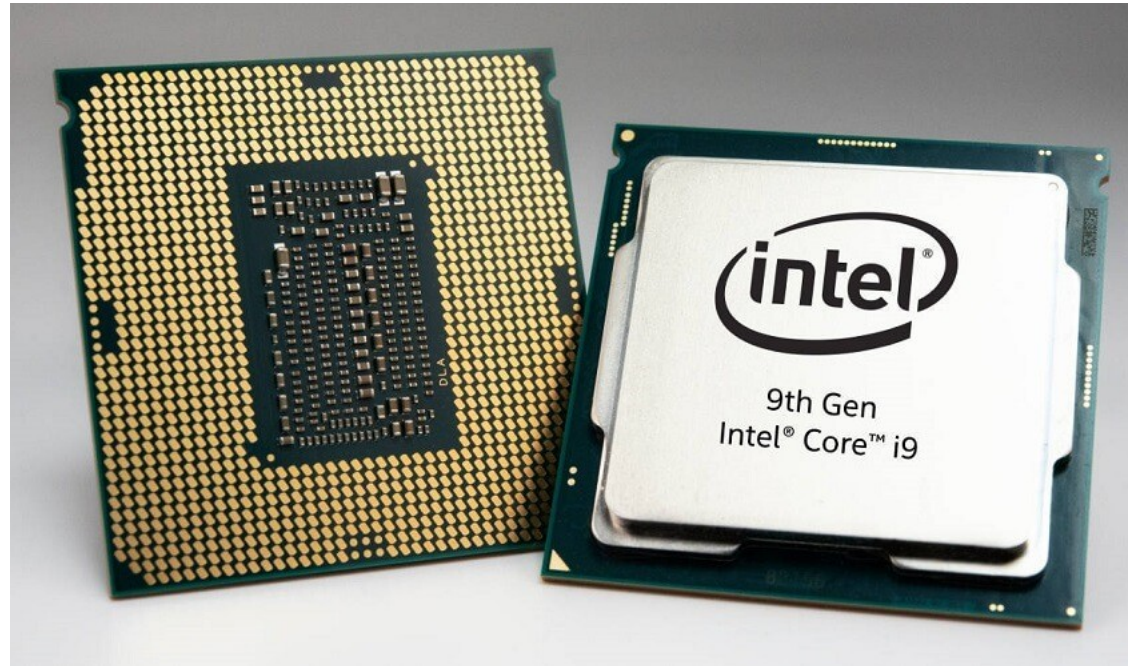


# Développer des procédures en temps réel

Module 155



# Programme simple :

```
# fonction qui attend x seconde
def delai(sec):
    print(f'On attend {sec} seconde')
    time.sleep(sec)
    print('Fin attente')

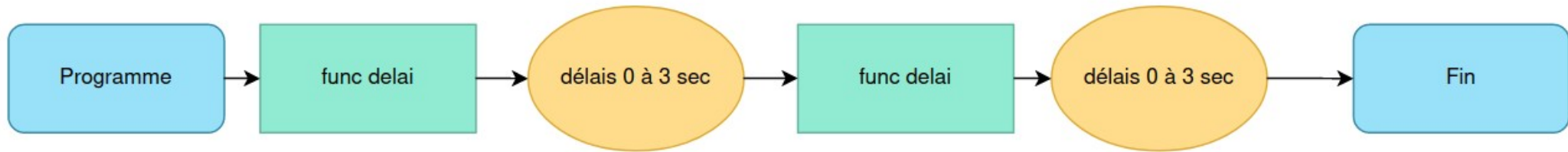
# timer du début
debut = time.perf_counter()
#on appelle 4 fois le délais
for i in range(4):
    delai(1)
# timer de fin
fin = time.perf_counter()
# Affichage
print(f"durée totale {(fin-debut)} seconde")
```

# Programme simple :


```
On attend 0 seconde  
Fin attente  
On attend 1 seconde  
Fin attente  
On attend 2 seconde  
Fin attente  
On attend 3 seconde  
Fin attente  
durée totale 6.00540254400039 seconde
```

# Programme simple :

Ici tous le code s'exécute de façon séquentiel. Ce qui ressemble à ça :



Temps



# Multithread :

Exécuter le code exempleThread sur Moodle plusieurs fois et expliquer ce qui se passe.

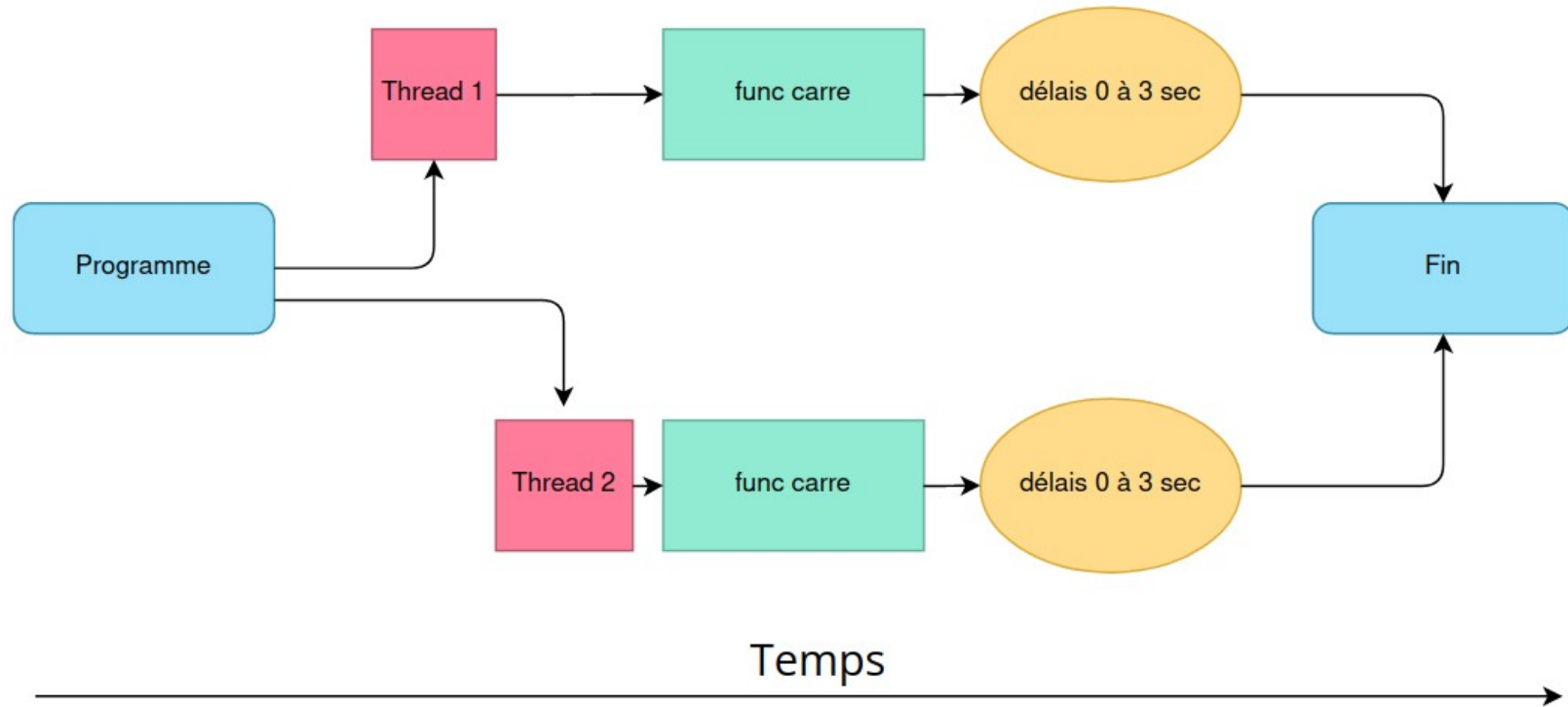
```
from threading import Thread
import time
from random import *

def carre(n):
    time.sleep(randint(0,3))
    print(f"Le carré: {n * n}")

# création de thread
t1 = Thread(target=carre, args=(1,))
t2 = Thread(target=carre, args=(2,))
# démarrer le thread t1
t1.start()
# démarrer le thread t2
t2.start()
# attendre que t1 soit exécuté
t1.join()
# attendre que t2 soit exécuté
t2.join()
```

# Multithread :

Comme on peut le voir le thread 1 ou le thread 2 peuvent terminer avant l'autre thread, cela va dépendre du temps aléatoire de délais de chaque thread.





# Questions :

