IPS Instituto Politécnico de Setúbal Escola Superior de Tecnologia de Setúbal	COMPUTAÇÃO PARALELA E DISTRIBUÍDA 2022/2023
	Laboratório 03: Utilização de *Threads*

Objetivos do laboratório

Pretende-se que o aluno:

1. Perceba as vantagens de se utilizar programação **multi-threading** em Python
2. Consiga programar com threads em Python
3. Seja capaz de decidir quando é vantajoso utilizar threads nas suas aplicações

A imagem seguinte apresenta os estados na vida de uma thread.

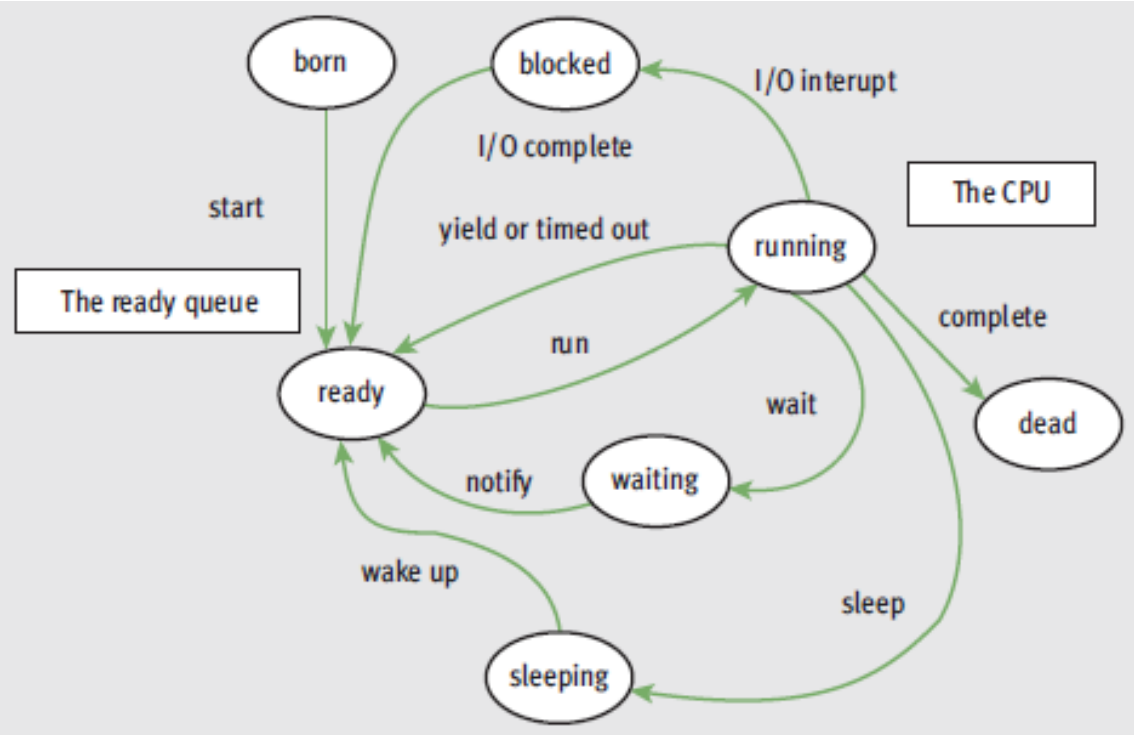

Figura 1 – Estados na vida de uma thread - (página 396 do livro Fundamentals of Python)

Nível 1

Considere o código a seguir que simula várias ligações a um servidor, primeiro em modo sequencial e depois utilizando multi-threading. Escreva o código apresentado e responda as questões colocadas.

```
import threading
import time


def conecta():  # simula uma ligação remota a um servidor
    print("\tligado")
    time.sleep(2)
    print("\tdesligado")


if __name__ == '__main__':
    numero_de_ligacoes = 2
    inicio = time.time()
    print("Início do processamento sequencial")
    for i in range(numero_de_ligacoes):
        print(f'{i+1}ª ligação')
        conecta()
    fim = time.time()
    print(f'Tempo gasto para realizar {numero_de_ligacoes}ª ligações sequenciais: {fim-inicio}s')

    print("Início do processamento multi-threading")
    # multi-threaded
    threads = []
    inicio = time.time()
    for i in range(numero_de_ligacoes):
        t = threading.Thread(target=conecta)
        threads.append(t)
        t.start()
    for i in range(numero_de_ligacoes):
        threads[i].join()
    fim = time.time()
    print(f'Tempo gasto para realizar {numero_de_ligacoes} ligações multi-threading: {fim - inicio}s')
```

- Explique a diferença no desempenho entre o processamento sequencial e **multi-threading**. Relacione com a instrução **sleep**.
- Altere o número de ligações para 20 e analise de um modo critico os resultados obtidos

Nível 2

Escreva o código apresentado e responda as questões colocadas.

```
if __name__ == '__main__':
    n = 10000
    with open('outBig.bin', 'wb') as file:
        for i in range(n):
            file.write(i.to_bytes(4, byteorder='big', signed=False))
    with open('outLittle.bin', 'wb') as file:
        for i in range(n):
            file.write(i.to_bytes(4, byteorder='little', signed=False))
```

- Diga o que faz a instrução ***i.to_bytes*** e explique a diferença entre ***byteorder='big'*** e *byteorder='little'*
- Descreva o que faz o programa
- Abra os ficheiros **outBig.bin** e **outLittle.bin** e verifique a forma como estão representados os inteiros. Aceda ao site HexEd.it para abrir os ficheiros e, no menu configurações, altere o número de bytes por linha para 4. Verifique como está representado o número 1024 em cada um dos ficheiros.

Nível 3

Escreva o código apresentado e responda as questões colocadas.

```
import time

inicio = time.time()
for k in range(10):
    fileName = "fich"+str(k+1)+".dat"
    with open(fileName, 'wb') as file:
        for i in range((k+1)*1000000):
            file.write(i.to_bytes(4, byteorder='big', signed=False))
print(f"Tempo escrita = {time.time()-inicio}")
```

- Diga o que faz o programa.

Nível 4

Escreva o código apresentado e responda as questões colocadas.

```
import shutil
import time
import threading

if __name__ == '__main__':
    # Versao sequencial
    print("Cópia sequencial de ficheiros")
    inicio = time.time()
    for k in range(10):
        fileName1 = "fich" + str(k + 1) + ".dat"
        fileName2 = "outFich" + str(k + 1) + ".dat"
        shutil.copy(fileName1, fileName2)
    print(f"\tTempo utilizado na cópia sequencial de ficheiros = {time.time()-inicio}")
    # Versao com threads
    print("Cópia multi-threading de ficheiros")
    inicio = time.time()
    threads = []
    for k in range(10):
        fileName1 = "fich" + str(k + 1) + ".dat"
        fileName2 = "outFich" + str(k + 1) + ".dat"
        t = threading.Thread(target=shutil.copy, args=[fileName1, fileName2])
        threads.append(t)
        t.start()
    for i in range(10):
        threads[i].join()
    print(f"\tTempo utilizado na cópia multi-threading de ficheiros = {time.time() - inicio}")
```

- Diga o que faz o programa.
- Explique a diferença de desempenho observadas

Nível 5

Escreva um pequeno texto que relacione a execução de programas **CPU-Bound** e **I/O-Bound**, quando executados em thread única ou **multi-threading**, considerando a performance e estados das threads apresentados na Figura 1 – Estados na vida de uma thread - (página 396 do livro Fundamentals of Python).