Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

Abstraindo Mecanismos de Concorrência

Abstraindo Mecanismos de Concorrência

Qual a diferença, em questão de código, entre criar um programa que utiliza concorrência através de threads ou processos?

Abstraindo Mecanismos de Concorrência

Qual a diferença, em questão de código, entre criar um programa que utiliza concorrência através de threads ou processos?

```python
import threading
import time


def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)



if __name__ == '__main__':
    ex = threading.Thread(target=processa)

    ex.start()
    ex.join()

```

```python
import multiprocessing
import time


def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)



if __name__ == '__main__':
    ex = multiprocessing.Process(target=processa)

    ex.start()
    ex.join()

```

Abstraindo Mecanismos de Concorrência

Qual a diferença, em questão de código, entre criar um programa que utiliza concorrência através de threads ou processos?

```python
import threading
import time


def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)



if __name__ == '__main__':
    ex = threading.Thread(target=processa)

    ex.start()
    ex.join()
```

1

2

```python
import multiprocessing
import time


def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)



if __name__ == '__main__':
    ex = multiprocessing.Process(target=processa)

    ex.start()
    ex.join()
```

1

2

Note que em questão de código, as diferenças são apenas duas.

Abstraindo Mecanismos de Concorrência

Através dos mecanismos de abstração da API de concorrências, podemos alternar entre duas implementações de concorrência diferentes (Threads e Processos) com apenas uma linha.

Abstraindo Mecanismos de Concorrência

- Através dos mecanismos de abstração da API de concorrências, podemos alternar entre duas implementações de concorrência diferentes (Threads e Processos) com apenas uma linha.

```python
import time

# from concurrent.futures.thread import ThreadPoolExecutor as Executor
from concurrent.futures.process import ProcessPoolExecutor as Executor



def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)



if __name__ == '__main__':
    with Executor() as executor:
        future = executor.submit(processa)

```

1

Abstraindo Mecanismos de Concorrência

- Através dos mecanismos de abstração da API de concorrências, podemos alternar entre duas implementações de concorrência diferentes (Threads e Processos) com apenas uma linha.

```python
import time

# from concurrent.futures.thread import ThreadPoolExecutor as Executor
from concurrent.futures.process import ProcessPoolExecutor as Executor



def processa():
    print('[', end='', flush=True)
    for _ in range(1,11):
        print('#', end='', flush=True)
        time.sleep(1)
    print(']', end='', flush=True)


if __name__ == '__main__':
    with Executor() as executor:
        future = executor.submit(processa)

```

1

Vamos ao código implementar isso…

Geek University

Evolua seu lado geek!

www.geekuniversity.com.br