I/O-bound (multithreading)

```
import time
from concurrent.futures import ThreadPoolExecutor

def io_task(n):
    time.sleep(1)  # імітуємо очікування (наприклад, завантаження з інтернету)
    return n

def run_sequential():
    start = time.perf_counter()
    for i in range(10):
        io_task(i)
    print("Послідовно:", round(time.perf_counter() - start, 2), "с")

def run_threads():
    start = time.perf_counter()
    with ThreadPoolExecutor(max_workers=10) as ex:
        list(ex.map(io_task, range(10)))
    print("Потоки:", round(time.perf_counter() - start, 2), "с")

if __name__ == "__main__":
    run_sequential()
    run_threads()
```

```
Послідовно: 10.01 с
Потоки: 1.01 с
```

CPU-bound (multiprocessing)

```
from concurrent.futures import ProcessPoolExecutor
import math, time

def is_prime(x: int) -> bool:
    if x < 2: return False
    if x % 2 == 0: return x == 2
    r = int(math.sqrt(x))
    for d in range(3, r + 1, 2):
        if x % d == 0:
            return False
    return True

def count_primes(n: int) -> int:
    return sum(1 for i in range(n) if is_prime(i))

def run_sequential():
    start = time.perf_counter()
    results = [count_primes(200_000) for _ in range(4)]
    print("Послідовно:", round(time.perf_counter() - start, 2), "с",
          "| сума:", sum(results))

def run_processes():
    start = time.perf_counter()
    with ProcessPoolExecutor() as ex:
        results = list(ex.map(count_primes, [200_000]*4))
    print("Процеси:", round(time.perf_counter() - start, 2), "с",
          "| сума:", sum(results))

if __name__ == "__main__":
    run_sequential()
    run_processes()
```

```
Послідовно: 1.22 с | сума: 71936
Процеси: 0.8 с | сума: 71936
```

Почніть кодувати або генерувати код за допомогою ШІ.