CRIBLE D'ÉRATOSTHÈNE

Algorigramme (version naïve)

```
                    │
                    ▼
            ┌───────────────┐
            │   Entrer n    │
            └───────┬───────┘
                    │
                    ▼
        ┌───────────────────────┐
        │ L ←  [0, 1, 2, 3, ..., n] │
        └───────────┬───────────┘
                    │
                    ▼
            ┌───────────────┐
            │  L[1] ←── 0   │
            └───────┬───────┘
                    │
                    ▼
            ┌───────────────┐
            │   i ←── 2     │
            └───────┬───────┘
                    │
                    ▼
              ◇ i ≤ n ? ◇ ──── Non ────┐
                    │                   │
                   Oui                  ▼
                    │            ┌──────────────┐
                    ▼            │  Renvoyer L  │
            ┌───────────────┐    └──────┬───────┘
            │    k = i      │           │
            └───────┬───────┘           ▼
                    │
                    ▼
            ◇ k * i ≤ n ? ◇ ──── Non ──┐
                    │                   │
                   Oui                  │
                    │                   │
                    ▼                   │
            ┌───────────────┐           │
            │ L[k*i] ←── 0  │           │
            └───────┬───────┘           │
                    │                   │
                    ▼                   ▼
            ┌───────────────┐   ┌──────────────┐
            │ k ←── k + 1   │   │ i ←── i + 1  │
            └───────────────┘   └──────────────┘
```

1
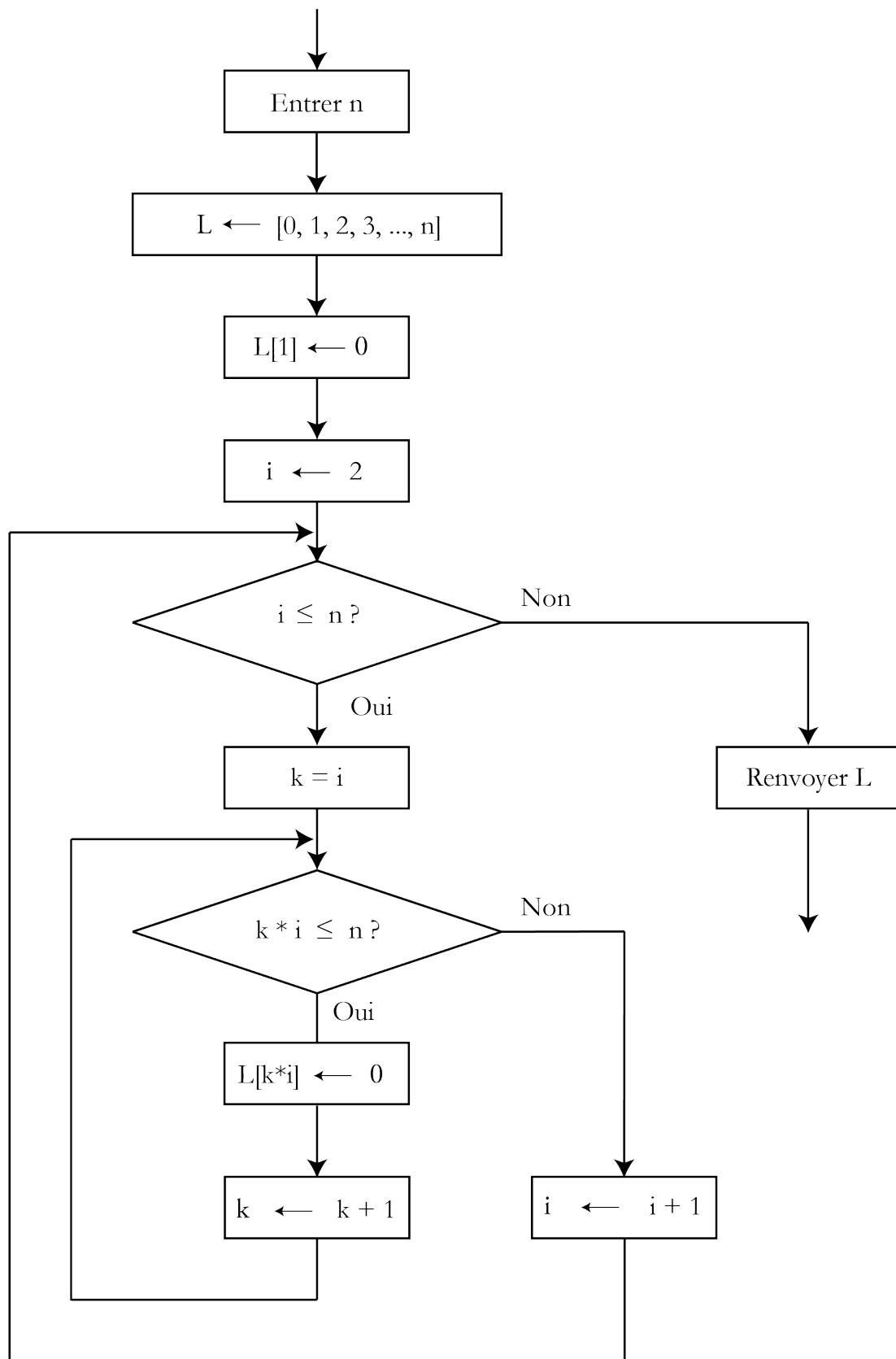
Code Python traduisant l'algorigramme ci-dessus

```python
"""
Implémentation du crible d'Ératosthène
"""
#Author : P. JANC

def primes_era(n):
    L = [numbers for numbers in range(n+1)]
    L[1] = 0
    i = 2
    while i <=n:
        k = i
        while k*i <= n:
            L[k*i] = 0
            k+=1
        i+=1
    return L
```

Code Python amélioré

```python
def primes_era_optimized(n):
    L = list(range(n+1))
    L[1] = 0
    i = 2
    while i**2 <=n:
        if L[i] >= 1:
            k = i
            while k*i <=n:
                L[k*i] = 0
                k+=1
        i+=1
    return L
```