

```
1
2
3 Python Parallel {
4
5 [Grupo: 6]
6
7
8
9     < Julian Felipe Tolosa - 2170107 >
10    < Jhan Eduardo Rojas Niño - 2182690 >
11    < Juan Camilo Pertuz - 2160032 >
12 }
13
14
```

Tabla de 'Contenido' {

01 ¿Qué es?

< ¿Qué es Python Parallel? >

02 Características

03 ¿Por qué es útil?

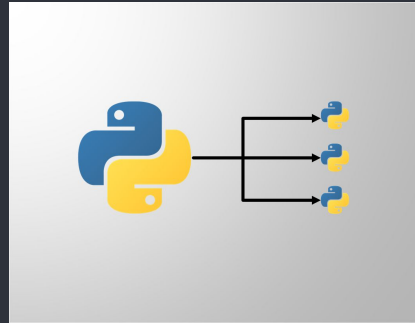
< ¿Por qué es útil la
paralelización en Python? >

04 Ejemplo sencillo y
ejercicio aplicado

}

01 {

[¿Qué es?]



}

1 ¿Qué es? {
2
3

4 'Python Parallel es un módulo de Python que
5 proporciona un mecanismo para la ejecución
6 paralela de código de Python en SMP y clústeres.'

7 <p Es ligero, fácil de instalar e integrar con
8 otro software de Python.

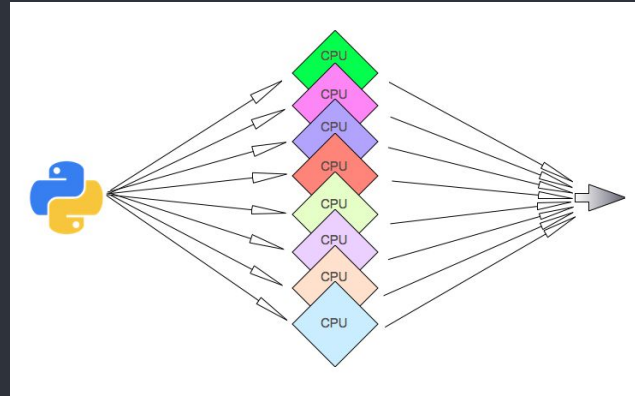
9 Parallel Python es un módulo de código abierto
10 y multiplataforma escrito en Python puro >

11
12
13 }
14

</p>



02 {

[Características]



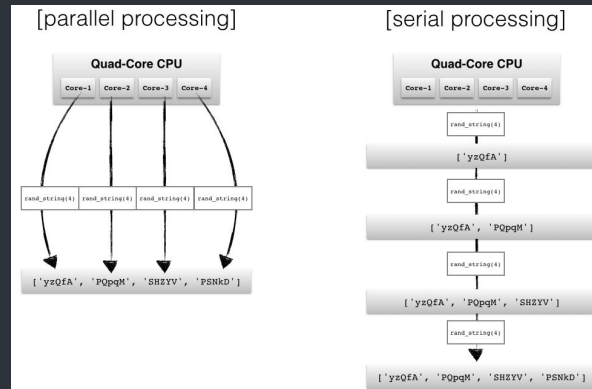
}

Características < /1 > {

- 
 - < Ejecución paralela de código Python en SMP y clústeres >
 - < Técnica de paralelización basada en trabajos fácil de entender e implementar >
 - < Asignación dinámica de procesadores >
 - < Baja sobrecarga para trabajos posteriores con la misma función >
- 
 - < Equilibrio de carga dinámico >
 - < Tolerancia a fallas >
 - < Portabilidad e interoperabilidad multiplataforma >
 - < Portabilidad e interoperabilidad entre arquitecturas >

03 {

[¿Por qué es útil la
paralelización en Python?]



¿Por qué es importante 'la paralelización en Python?' {

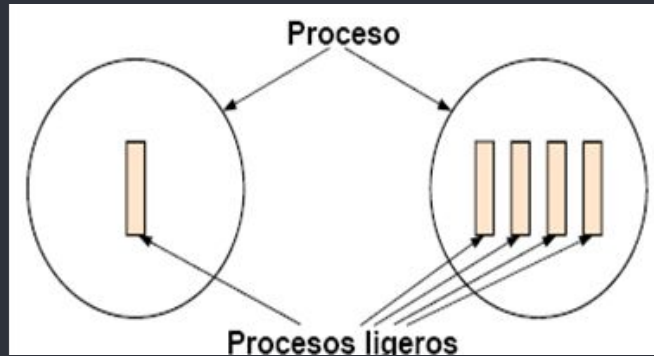
<p Tiene sentido aprovechar la paralelización en Python para programas que usan mucho la CPU para el trabajo computacional. Tales cargas de trabajo en el mundo real incluyen transformaciones aritméticas en grandes conjuntos de números, matemáticas vectoriales, procesamiento de imágenes, sonido y video, y predicción de resultados utilizando modelos de aprendizaje automático >

</p>

}

04 {

[Ejemplo sencillo Python Parallel]



}

```
from multiprocessing import Process

def numbers(start_num):
    for i in range(5):
        print(start_num+i, end=' ')

if __name__ == '__main__':
    p1 = Process(target=numbers, args=(1,))
    p2 = Process(target=numbers, args=(10,))
    p1.start()
    p2.start()
    # wait for the processes to finish
    p1.join()
    p2.join()

# output:
# 1 2 3 4 5 10 11 12 13 14
```

Ejercicio aplicado { Ecuación de Calor



}