**Parallel processing**

# Serial processing

import time

def calc\_square(numbers):

print("Calculate square numbers")

for n in numbers:

time.sleep(0.2)

print('square : ', n\*n)

def calc\_cube(numbers):

print("Calculate cube of numbers")

for n in numbers:

time.sleep(0.2)

print('square : ', n\*n\*n)

arr = [2,3,8,9]

t = time.time()

calc\_square(arr)

calc\_cube(arr)

print("done in", time.time()-t)

**Output**

Calculate square numbers

square : 4

square : 9

square : 64

square : 81

Calculate cube of numbers

square : 8

square : 27

square : 512

square : 729

done in 1.6503331661224365

# multithreading

import time

import threading

def calc\_square(numbers):

print("Calculate square numbers")

for n in numbers:

time.sleep(0.2)

print('square : ', n\*n)

def calc\_cube(numbers):

print("Calculate cube of numbers")

for n in numbers:

time.sleep(0.2)

print('square : ', n\*n\*n)

arr = [2,3,8,9]

t = time.time()

t1 = threading.Thread(target=calc\_square, args=(arr,))

t2 = threading.Thread(target=calc\_cube, args=(arr,))

t1.start()

t2.start()

t1.join()

t2.join()

print("done in", time.time()-t)

**Output**

Calculate square numbersCalculate cube of numbers

square : 4

square : 8

square : square : 9

27

square : square : 64

512

square : square : 81

729

done in 0.8298640251159668

# multiprocessing

import time

import multiprocessing

def calc\_square(numbers):

#print("Calculate square numbers")

for n in numbers:

##time.sleep(0.2)

print('square : '+ str(n\*n))

def calc\_cube(numbers):

#print("Calculate square numbers")

for n in numbers:

##time.sleep(0.2)

print('cube : '+ str(n\*n\*n))

if \_\_name\_\_ == "\_\_main\_\_":

arr = [2,3,8,9]

p1 = multiprocessing.Process(target=calc\_square, args=(arr,))

p2 = multiprocessing.Process(target=calc\_cube, args=(arr,))

p1.start()

p2.start()

p1.join()

p2.join()

print("Done")

import time

import multiprocessing

def calc\_square(numbers):

for n in numbers:

print('square ' + str(n\*n))

def calc\_cube(numbers):

for n in numbers:

print('cube ' + str(n\*n\*n))

if \_\_name\_\_ == "\_\_main\_\_":

arr = [2,3,8]

p1 = multiprocessing.Process(target=calc\_square, args=(arr,))

p2 = multiprocessing.Process(target=calc\_cube, args=(arr,))

p1.start()

p2.start()

p1.join()

p2.join()

print("Done!")

import time

import multiprocessing

square\_result = []

def calc\_square(numbers):

global square\_result

for n in numbers:

print('square ' + str(n\*n))

square\_result.append(n\*n)

if \_\_name\_\_ == "\_\_main\_\_":

arr = [2,3,8,9]

p1 = multiprocessing.Process(target=calc\_square, args=(arr,))

p1.start()

p1.join()

print('result'+str(square\_result))

print("Done!")

Output

square 4

square 9

square 64

square 81

result[]

Done!

import time

import multiprocessing

square\_result = []

def calc\_square(numbers):

global square\_result

for n in numbers:

print('square ' + str(n\*n))

square\_result.append(n\*n)

print('within a process result'+str(square\_result))

if \_\_name\_\_ == "\_\_main\_\_":

arr = [2,3,8,9]

p1 = multiprocessing.Process(target=calc\_square, args=(arr,))

p1.start()

p1.join()

print('result'+str(square\_result))

print("Done!")

Output

square 4

square 9

square 64

square 81

within a process result[4, 9, 64, 81]

result[]

Done!

def f(n):

return n\*n

if \_\_name\_\_ == "\_\_main\_\_":

array = [1,2,3,4,5]

result = []

for n in array:

result.append(f(n))

print(result)

Output

[1, 4, 9, 16, 25]

from multiprocessing import Pool

def f(n):

return n\*n

if \_\_name\_\_ == "\_\_main\_\_":

array = [1,2,3,4,5]

p = Pool()

result = p.map(f, array)

print(result)

Output

[1, 4, 9, 16, 25]

from multiprocessing import Pool

import time

def f(n):

sum =0

for x in range(1000):

sum += x\*x

return sum

if \_\_name\_\_ == "\_\_main\_\_":

t1= time.time()

p = Pool()

result = p.map(f, range(10000))

p.close()

p.join()

print("Pool took :", time.time()-t1)

t2 = time.time()

result = []

for x in range(10000):

result.append(f(x))

print("Serial processing took:", time.time()-t2)

Output

Pool took : 1.7104525566101074

Serial processing took: 7.472449064254761