

```
py > mult_threading.py >
mult_threading.py x
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('cube:', n*n*n)

arr = [2,3,8,9]

t = time.time()
calc_square(arr)
calc_cube(arr)
```

PFForm11.pdf Show all X

```
py > mult_threading.py >
mult_threading.py x
        print('square:', n*n)

def calc_cube(numbers):
    print("calculate cube of numbers")
    for n in numbers:
        time.sleep(0.2)
        print('cube:', n*n*n)

arr = [2,3,8,9]

t = time.time()
calc_square(arr)
calc_cube(arr)

print("done in : ", time.time()-t)
print("Hah... I am done with all my work now!")
```

PFForm11.pdf Show all X

```
py > mult_threading.py
mult_threading.py x
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('cube:',n*n*n)

arr = [2,3,8,9]

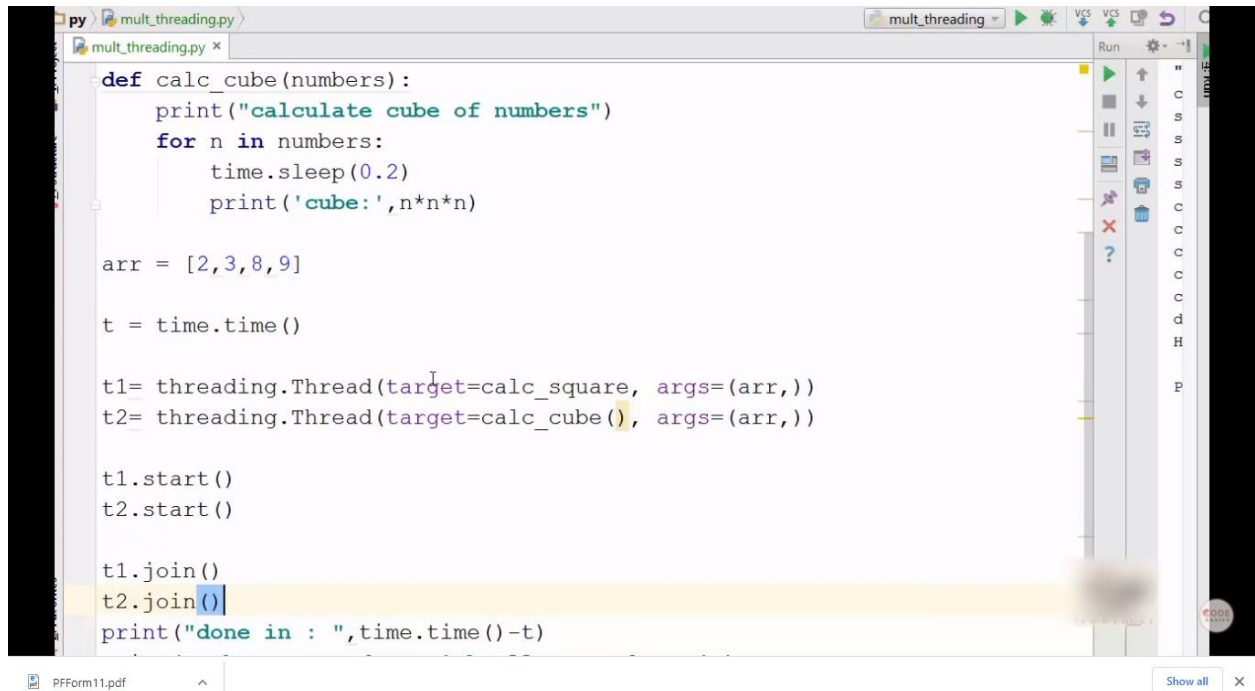
t = time.time()
calc_square(arr)
calc_cube(arr)
```

Run mult\_threading

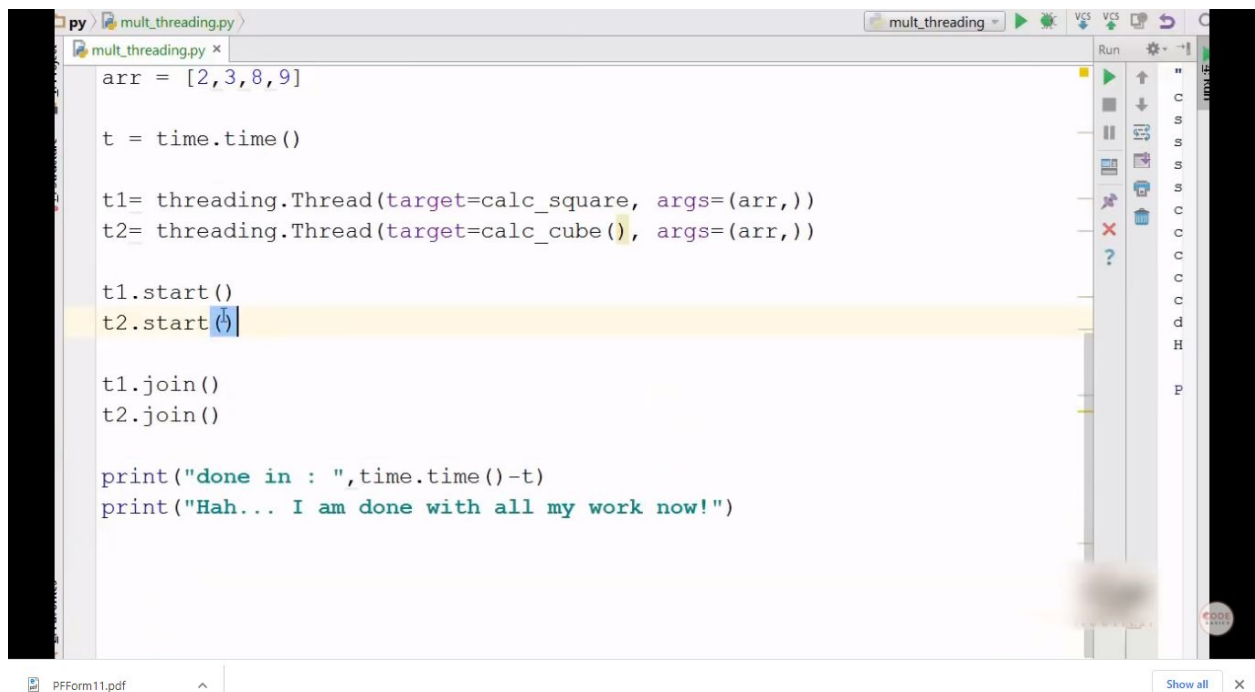
"C:\Program File  
calculate square  
square: 4  
square: 9  
square: 64  
square: 81  
calculate cube o  
cube: 8  
cube: 27  
cube: 512  
cube: 729  
done in : 1.602  
Hah... I am done  
Process finished

PFForm11.pdf

Show all

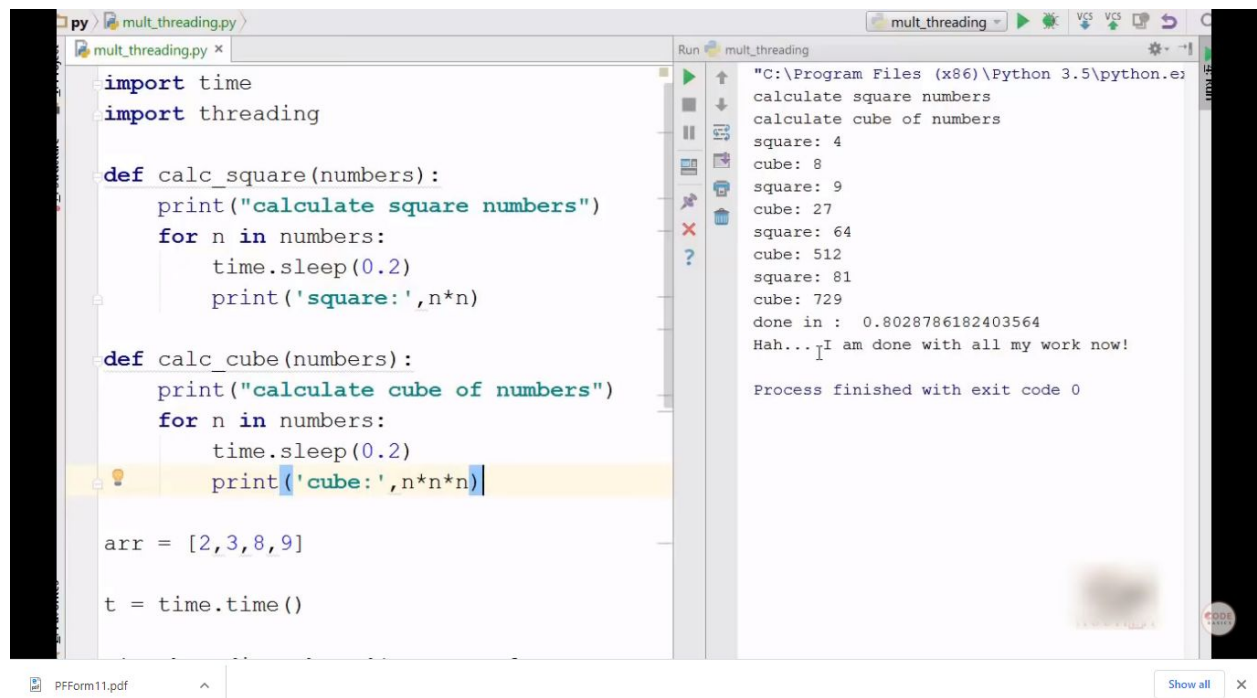


```
def calc_cube(numbers):  
    print("calculate cube of numbers")  
    for n in numbers:  
        time.sleep(0.2)  
        print('cube:',n*n*n)  
  
arr = [2,3,8,9]  
  
t = time.time()  
  
t1= threading.Thread(target=calc_square, args=(arr,))  
t2= threading.Thread(target=calc_cube(), args=(arr,))  
  
t1.start()  
t2.start()  
  
t1.join()  
t2.join()  
print("done in : ",time.time()-t)
```



```
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)  
print("Hah... I am done with all my work now!")
```

Output : parallel run program



The image shows a Python IDE window with a file named `mult_threading.py`. The code defines two functions, `calc_square` and `calc_cube`, which calculate the square and cube of numbers in a list, respectively. Both functions use `time.sleep(0.2)` to simulate a delay. The main code creates a list `arr = [2, 3, 8, 9]` and starts a timer. The output window shows the execution results, including the calculated squares and cubes, the total time taken, and a message indicating the process finished with exit code 0.

```
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('cube:', n*n*n)

arr = [2, 3, 8, 9]

t = time.time()
```

Output:

```
"C:\Program Files (x86)\Python 3.5\python.exe"
calculate square numbers
calculate cube of numbers
square: 4
cube: 8
square: 9
cube: 27
square: 64
cube: 512
square: 81
cube: 729
done in : 0.8028786182403564
Hah...I am done with all my work now!

Process finished with exit code 0
```