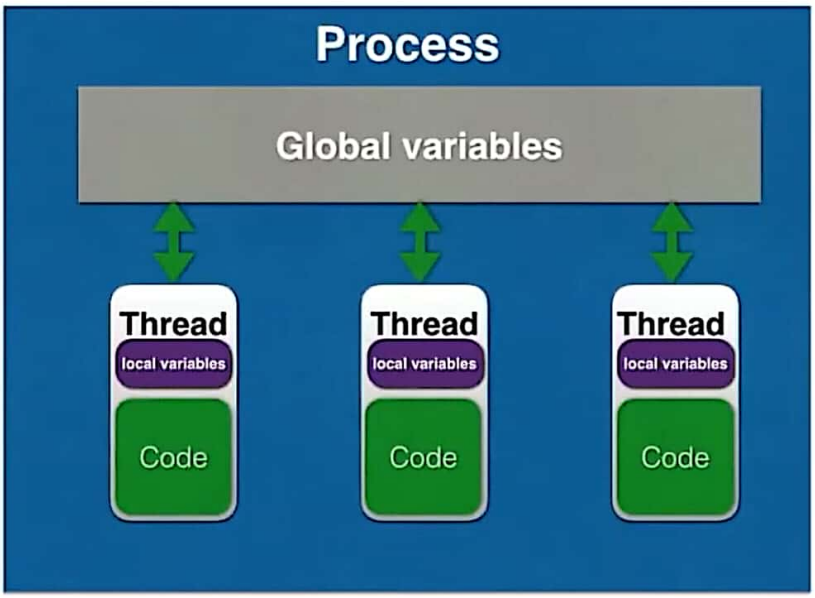
Return to call 32:03

# What is a Thread?

- Individual and separate unit of execution that is part of a process. multiple threads can work together to accomplish a common goal.
- Threads allow the program to run tasks in parallel
- Video Game example
  - one thread for graphics
  - one thread for user interaction
  - one thread for networking

Viren Patel's screen



There are two different kind of threads

- kernel thread
- user thread

# ADVANTAGES OF THREADING

- Multithreaded programs can run faster on computer systems with multiple CPUs.
- A program can remain responsive to input. This is true both on single and on multiple CPUs.
- Allows to do something else while one thread is waiting for an I/O task to complete.
- Some programs are easy to express using concurrency.
- Threads of a process can share the memory of global variables.

# THREADS ISSUES

- Scheduling
- Resource Sharing
- Synchronization

# Threads in Python

- In Python, a thread is an object
  - hold data,
  - run with methods,
  - stored in data structure
  - passed as parameters to methods
- A thread can also be executed as a process
- During its lifetime, a thread can be in various states

# Threads in Python

- modules which support the usage of threads in Python.
  - thread
  - threading
- The thread module has been considered as "deprecated"
  - renamed to "_thread" for backwards incompatibilities in Python3
    one thread for user interaction
- The module "thread" treats a thread as a function, while the module "threading" is implemented in an object oriented way

helloWorld 〉 thread_example.py 〉

hello

thread_example.py

```python
import _thread
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 3:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time())

_thread.start_new_thread(print_epoch, ("thread 1", 1))
_thread.start_new_thread(print_epoch, ("thread 2", 3))
```

Run hello

```
/Users/codebind/PycharmProjects/helloWorl

Process finished with exit code 0
```

helloWorld 〉 thread_example.py 〉

hello

thread_example.py

```python
import _thread
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 3:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time())

_thread.start_new_thread(print_epoch, ("thread 1", 1))
_thread.start_new_thread(print_epoch, ("thread 2", 3))

input()
```

Run

```
/Users/codebind/PycharmProjects/helloWorl
thread 1 --------- 1543984347.4233978
thread 1 --------- 1543984348.426662
thread 2 --------- 1543984349.4225652
thread 1 --------- 1543984349.430916
```

helloWorld 〉 threading_example.py 〉

Project

threading_example.py

- helloWorld ~/PycharmProje
  - venv
  - thread_example.py
  - threading_example.py
- External Libraries

```python
import threading
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time())

if __name__ == "__main__":
    t1 = threading.Thread(target=print_epoch, args=("Thread 1", 1))
    t2 = threading.Thread(target=print_epoch, args=("Thread 2", 2))

    t1.start()
    t2.start()

    t1.join()
    t2.join()

    print("Done")
```

helloWorld 〉 threading_example.py 〉

threading_example

Project

helloWorld ~/PycharmProj

venv

thread_example.py

threading_example.py

External Libraries

threading_example.py

```python
import threading
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time()

if __name__ == "__main__":
    t1 = threading.Thread(target=print_epoch, args=(
    t2 = threading.Thread(target=print_epoch, args=(

    t1.start()
    t2.start()

    t1.join()
    t2.join()

    print("Done")
```

Run threading_example

```
/Users/codebind/PycharmProjects/helloWorld/ven
Thread 1 --------- 1544186553.275574
Thread 2 --------- 1544186554.275077
Thread 1 --------- 1544186554.310875
Thread 1 --------- 1544186555.315556
Thread 2 --------- 1544186556.278785
Thread 1 --------- 1544186556.318468
Thread 1 --------- 1544186557.322315
Thread 2 --------- 1544186558.280946
Thread 2 --------- 1544186560.285197
Thread 2 --------- 1544186562.285345
Done

Process finished with exit code 0
```

helloWorld 〉 threading_example.py 〉

threading_example

Project

helloWorld ~/PycharmProje

venv

thread_example.py

threading_example.py

External Libraries

threading_example.py

```python
import threading
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------

def print_cube(num):
    print("Cube = {}".format(num*num*n

def print_sqare(num):
    print("Sqare = {}".format(num*num)

if __name__ == "__main__":
    t1 = threading.Thread(target=print
    t2 = threading.Thread(target=print

    t1.start()
    t2.start()

    t1.join()
    t2.join()

    print("Done")
```

Run threading_example

```
/Users/codebind/PycharmProjects/helloWorld/venv/bin/pyt
Done
Exception in thread Thread-2:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3
    self.run()
  File "/Library/Frameworks/Python.framework/Versions/3
    self._target(*self._args, **self._kwargs)
TypeError: print_sqare() argument after * must be an it
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3
    self.run()
  File "/Library/Frameworks/Python.framework/Versions/3
    self._target(*self._args, **self._kwargs)
TypeError: print_cube() argument after * must be an ite


Process finished with exit code 0
```

0:09:53

0:01:27

helloWorld 〉 threading_example.py 〉

threading_example

Project

threading_example.py

- helloWorld ~/PycharmProje
  - venv
  - thread_example.py
  - threading_example.py
- External Libraries

```python
import threading
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time())

def print_cube(num):
    print("Cube = {}".format(num*num*num))

def print_sqare(num):
    print("Sqare = {}".format(num*num))

if __name__ == "__main__":
    t1 = threading.Thread(target=print_cube, args=(2,))
    t2 = threading.Thread(target=print_sqare, args=(2,))

    t1.start()
    t2.start()

    t1.join()
    t2.join()

    print("Done")
```

Run threading_example

```
/Users/codebind/PycharmP
Done
Exception in thread Thre
Traceback (most recent c
  File "/Library/Framewo
    self.run()
  File "/Library/Framewo
    self._target(*self._
TypeError: print_sqare(
Exception in thread Thre
Traceback (most recent c
  File "/Library/Framewo
    self.run()
  File "/Library/Framewo
    self._target(*self._
TypeError: print_cube()


Process finished with ex
```

helloWorld 〉 threading_example.py 〉

threading_example

Project

helloWorld ~/PycharmProje

venv

thread_example.py

threading_example.py

External Libraries

threading_example.py

```python
import threading
import time

def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time())

def print_cube(num):
    print("Cube = {}".format(num*num*num))

def print_sqare(num):
    print("Sqare = {}".format(num*num))

if __name__ == "__main__":
    t1 = threading.Thread(target=print_cube, args=(2,))
    t2 = threading.Thread(target=print_sqare, args=(2,))

    t1.start()
    t2.start()

    t1.join()
    t2.join()

    print("Done")
```

Run threading_example

```
/Users/codebind/PycharmP
Cube = 8
Sqare = 4
Done

Process finished with ex
```

0:10:56

0:00:24

helloWorld > threading_subclass.py

Project

- helloWorld ~/PycharmProj
  - venv
  - thread_example.py
  - threading_example.py
  - threading_subclass.py
- External Libraries

threading_subclass.py

```python
import threading
import time


def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time(

class MyThread(threading.Thread):
    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        print("start thread")
        print_epoch(self.name, self.delay)
        print("end thread")


if __name__ == "__main__":

    t1 = MyThread("Thread-1", 1)
    t2 = MyThread("Thread-2", 2)

    t1.start()
    t2.start()

    t1.join()
    t2.join()
```

Run threading_subclass

```
/Users/codebind/PycharmProjects/helloWorld
start thread
start thread
Thread-1 --------- 1546355437.835844
Thread-2 --------- 1546355438.83686
Thread-1 --------- 1546355438.8649652
Thread-1 --------- 1546355439.866988
Thread-2 --------- 1546355440.840229
Thread-1 --------- 1546355440.871066
Thread-1 --------- 1546355441.872913
end thread
Thread-2 --------- 1546355442.843808
Thread-2 --------- 1546355444.845789
Thread-2 --------- 1546355446.848715
end thread
Done

Process finished with exit code 0
```

SUBSCRIBE

helloWorld > threading_subclass.py

Project | threading_subclass.py | Run threading_subclass

- helloWorld ~/PycharmProj
  - venv
  - thread_example.py
  - threading_example.py
  - threading_subclass.py
- External Libraries

```python
import threading
import time


def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time(

class MyThread(threading.Thread):
    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        print("start thread")
        print_epoch(self.name, self.delay)
        print("end thread")


if __name__ == "__main__":

    t1 = MyThread("Thread-1", 1)
    t2 = MyThread("Thread-2", 2)

    t1.start()
    t2.start()

    t1.join()
    t2.join()
```

```
/Users/codebind/PycharmProjects/helloWorld
start thread
start thread
Thread-1 --------- 1546355437.835844
Thread-2 --------- 1546355438.83686
Thread-1 --------- 1546355438.8649652
Thread-1 --------- 1546355439.866988
Thread-2 --------- 1546355440.840229
Thread-1 --------- 1546355440.871066
Thread-1 --------- 1546355441.872913
end thread
Thread-2 --------- 1546355442.843808
Thread-2 --------- 1546355444.845789
Thread-2 --------- 1546355446.848715
end thread
Done

Process finished with exit code 0
```

helloWorld > threading_subclass.py

Project

- helloWorld ~/PycharmProj
  - venv
  - thread_example.py
  - threading_example.py
  - threading_subclass.py
- External Libraries

threading_subclass.py

```python
import threading
import time


def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time(

class MyThread(threading.Thread):
    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        print("start thread")
        print_epoch(self.name, self.delay)
        print("end thread")


if __name__ == "__main__":

    t1 = MyThread("Thread-1", 1)
    t2 = MyThread("Thread-2", 2)

    t1.start()
    t2.start()

    t1.join()
    t2.join()
```

Run threading_subclass

```
/Users/codebind/PycharmProjects/helloWorld
start thread
start thread
Thread-1 --------- 1546355437.835844
Thread-2 --------- 1546355438.83686
Thread-1 --------- 1546355438.8649652
Thread-1 --------- 1546355439.866988
Thread-2 --------- 1546355440.840229
Thread-1 --------- 1546355440.871066
Thread-1 --------- 1546355441.872913
end thread
Thread-2 --------- 1546355442.843808
Thread-2 --------- 1546355444.845789
Thread-2 --------- 1546355446.848715
end thread
Done

Process finished with exit code 0
```

helloWorld > threading_subclass.py

Project

helloWorld ~/PycharmProj

venv

thread_example.py

threading_example.py

threading_subclass.py

External Libraries

threading_subclass.py

```python
import threading
import time


def print_epoch(nameOfThread, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time(

class MyThread(threading.Thread):
    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        print("start thread:", self.name)
        print_epoch(self.name, self.delay)
        print("end thread:", self.name)


if __name__ == "__main__":

    t1 = MyThread("Thread-1", 1)
    t2 = MyThread("Thread-2", 2)

    t1.start()
    t2.start()

    t1.join()
    t2.join()
```

threading_subclass

Run threading_subclass

```
/Users/codebind/PycharmProjects/helloWorld
start thread: Thread-1
start thread: Thread-2
Thread-1 --------- 1546355519.5812638
Thread-2 --------- 1546355520.579198
Thread-1 --------- 1546355520.585679
Thread-1 --------- 1546355521.590199
Thread-2 --------- 1546355522.582044
Thread-1 --------- 1546355522.593638
Thread-1 --------- 1546355523.5972962
end thread: Thread-1
Thread-2 --------- 1546355524.58689
Thread-2 --------- 1546355526.588277
Thread-2 --------- 1546355528.5921369
end thread: Thread-2
Done

Process finished with exit code 0
```

SUBSCRIBE

helloWorld > threading_subclass.py

Project

- helloWorld ~/PycharmProj
  - venv
  - thread_example.py
  - threading_example.py
  - threading_subclass.py
- External Libraries

threading_subclass.py

```
7       while count < 5:
8           time.sleep(delay)
9           count += 1
10          print(nameOfThread, "------
11
12  class MyThread(threading.Thread):
13      def __init__(self, name, delay):
14          threading.Thread.__init__(s
15          self.name = name
16          self.delay = delay
17
18      def run(self):
19          print("start thread:", self
20          print_epoch(self.name, self
21          print("end thread:", self.n
22
23
24  if __name__ == "__main__":
25
26      t1 = MyThread("Thread-1", 1)
27      t2 = MyThread("Thread-2", 2)
28
29      t1.start()
30      t2.start()
31
32      print(threading.activeCount())
33      print(threading.currentThread()
34      print(threading.enumerate())
35
36      t1.join()
37      t2.join()
38
39      print("Done")
```

Run threading_subclass

```
/Users/codebind/PycharmProjects/helloWorld/venv/bin/python
start thread: Thread-1
start thread: Thread-2
3
<_MainThread(MainThread, started 140734891062720)>
[<_MainThread(MainThread, started 140734891062720)>, <MyTh
Thread-1 --------- 1546355633.611926
Thread-2 --------- 1546355634.612434
Thread-1 --------- 1546355634.616672
Thread-1 --------- 1546355635.6188421
Thread-2 --------- 1546355636.616951
Thread-1 --------- 1546355636.62199
Thread-1 --------- 1546355637.625323
end thread: Thread-1
Thread-2 --------- 1546355638.6209228
Thread-2 --------- 1546355640.621652
Thread-2 --------- 1546355642.624728
end thread: Thread-2
Done

Process finished with exit code 0
```

helloWorld 〉 threading_subclass.py 〉

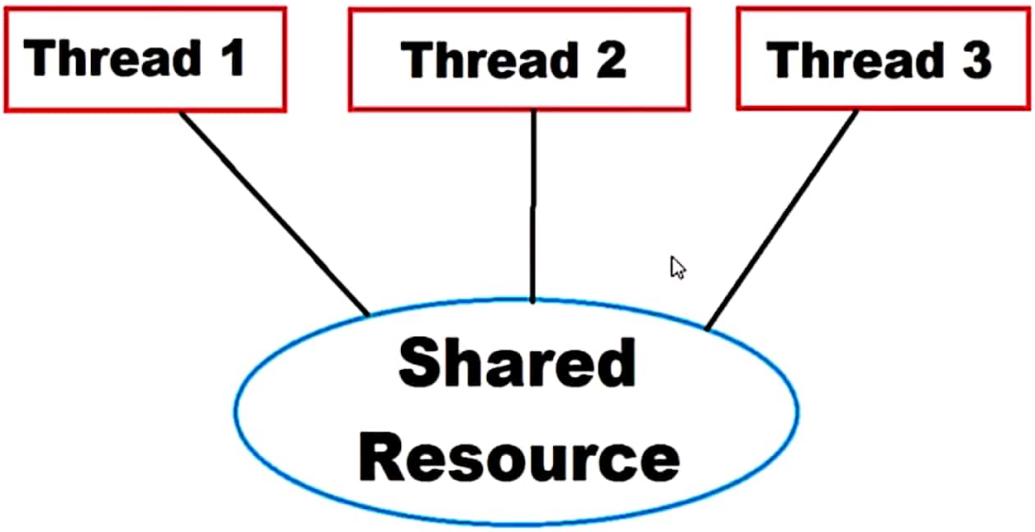
Project | threading_subclass.py

- helloWorld ~/PycharmProj
  - venv
  - thread_example.py
  - threading_example.py
  - threading_subclass.py
- External Libraries

```python
    while count < 5:
        time.sleep(delay)
        count += 1
        print(nameOfThread, "---------", time.time()

class MyThread(threading.Thread):
    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        print("start thread:", self.name)
        print_epoch(self.name, self.delay)
        print("end thread:", self.name)


if __name__ == "__main__":

    t1 = MyThread("Thread-1", 1)
    t2 = MyThread("Thread-2", 2)

    t1.start()
    t2.start()

    print(t1.getName())
    print(t2.getName())
    print(threading.activeCount())
    print(threading.currentThread())
    print(threading.enumerate())

    t1.join()
    t2.join()
```

Run threading_subclass

```
/Users/codebind/PycharmProjects/helloWor
start thread: Thread-1
start thread: Thread-2
Thread-1
Thread-2
3
<_MainThread(MainThread, started 1407349
[<_MainThread(MainThread, started 140734
Thread-1 --------- 1546355771.672677
Thread-2 --------- 1546355772.6714182
Thread-1 --------- 1546355772.676539
Thread-1 --------- 1546355773.678914
Thread-2 --------- 1546355774.675305
Thread-1 --------- 1546355774.6819499
Thread-1 --------- 1546355775.6848378
end thread: Thread-1
Thread-2 --------- 1546355776.678873
Thread-2 --------- 1546355778.68047
Thread-2 --------- 1546355780.683106
end thread: Thread-2
Done

Process finished with exit code 0
```



ThreadSamples [C:\Users\ProgrammingKnowledge\PycharmProjects\ThreadSamples] - ...\threads_synchronization.py [ThreadSamples] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ThreadSamples > threads_synchronization.py

threads_synchronization.py

```python
def thread_task():
    global x
    for i in range(10):
        x += 1

def main_task():
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

    t1.start()
    t2.start()
    t1.join()
    t2.join()


if __name__ == "__main__":
    main_task()
    print("{0}".format(x))
```

Python Console Terminal 6: TODO

4 chars 20:9 CRLF ÷ UTF-8 ÷

ThreadSamples [C:\Users\ProgrammingKnowledge\PycharmProjects\ThreadSamples] - ...\threads_synchronization.py [ThreadSamples] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ThreadSamples > threads_synchronization.py

threads_synchronization.py

```python
import threading
x = 0

def thread_task():
    global x
    for i in range(10):
        x += 1

def main_task():
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

    t1.start()
    t2.start()
    t1.join()
    t2.join()


if __name__ == "__main__":
    main_task()
    print("{0}".format(x))

```

main_task()

Python Console Terminal 6: TODO

14:10 CRLF ÷ UTF-8 ÷

ThreadSamples [C:\Users\ProgrammingKnowledge\PycharmProjects\ThreadSamples] - ...\threads_synchronization.py [ThreadSamples] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ThreadSamples 〉 threads_synchronization.py 〉

threads_synchronization.py

```python
import threading
x = 0

def thread_task():
    global x
    for i in range(10):
        x += 1

def main_task():
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

    t1.start()
    t2.start()
    t1.join()
    t2.join()


if __name__ == "__main__":
    main_task()
    print("{0}".format(x))
```

if __name__ == "__main__"

Run: threads_synchronization

```
C:\Users\ProgrammingKnowled
20

Process finished with exit
```

Python Console Terminal 6: TODO

Toggle using soft wraps in current editor

2 chars 2:3 CRLF ÷ UTF-8 ÷

ThreadSamples [C:\Users\ProgrammingKnowledge\PycharmProjects\ThreadSamples] - ...\threads_synchronization.py [ThreadSamples] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ThreadSamples > threads_synchronization.py

threads_synchronization.py

```python
import threading
x = 0

def thread_task():
    global x
    for i in range(100000):
        x += 1

def main_task():
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)

    t1.start()
    t2.start()
    t1.join()
    t2.join()


if __name__ == "__main__":
    main_task()
    print("{0}".format(x))
```

thread_task() > for i in range(100000)

Run: threads_synchronization

```
C:\Users\ProgrammingKnowled
141938

Process finished with exit
```

Python Console Terminal 6: TODO

6:26 CRLF ÷ UTF-8 ÷

ThreadSamples [C:\Users\ProgrammingKnowledge\PycharmProjects\ThreadSamples] - ...\threads_synchronization.py [ThreadSamples] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

ThreadSamples > threads_synchronization.py

threads_synchronization.py

```python
import threading
x = 0

def thread_task(lock):
    global x
    for i in range(100000):
        lock.acquire()
        x += 1
        lock.release()

def main_task():
    lock = threading.Lock()

    t1 = threading.Thread(target=thread_task, args=(lock,))
    t2 = threading.Thread(target=thread_task, args=(lock,))

    t1.start()
    t2.start()
    t1.join()
    t2.join()
```

thread_task() > for i in range(100000)

Run: threads_synchronization

```
C:\Users\ProgrammingKnowledge\Py
200000

Process finished with exit code
```

Python Console Terminal 6: TODO

6 chars 6:26 CRLF ÷ UTF-8 ÷

py > mult_threading.py

mult_threading.py ×

```python
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)
```

py > mult_threading.py

mult_threading.py ×

```
        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!")
```

py > mult_threading.py

mult_threading.py

```python
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 Files (x86)\Python 3.5\python.exe" C:
calculate square numbers
square: 4
square: 9
square: 64
square: 81
calculate cube of numbers
cube: 8
cube: 27
cube: 512
cube: 729
done in :  1.6022827625274658
Hah... I am done with all my work now!

Process finished with exit code 0
```

Every process has its own address space (virtual memory). Thus program variables are not shared between two processes. You need to use interprocess communication (IPC) techniques if you want to share data between two processes

py - [C:\Code\py] - ...\Multiprocessing\multi_proc.py - PyCharm Community Edition 5.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py 〉 Multiprocessing 〉 multi_proc.py 〉

multi_proc.py ×

```python
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!")
```

Run multi_proc

```
"C:\Program Files
square 4
square 9
square 64
square 81
within a process:
result [4]
Done!

Process finished w
```

py - [C:\Code\py] - ...\Multiprocessing\queue_pipe_multiprocessing.py - PyCharm Community Edition 3.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py 〉 Multiprocessing 〉 queue_pipe_multiprocessing.py 〉 queue_pipe_multiprocessing

queue_pipe_multiprocessing.py ×

```python
import multiprocessing

def calc_square(numbers, result):
    for idx, n in enumerate(numbers):
        result[idx] = n*n

if __name__ == "__main__":
    numbers = [2,3,5]
    result = multiprocessing.Array('i',3)
    p = multiprocessing.Process(target=calc_square, args=(numbers, result)

    p.start()
    p.join()

    print(result[:])
```

Run queue_pipe_multiprocessing

```
"C:\Program Files (x
[4, 9, 25]

Process finished wit
```

py - [C:\Code\py] - ...\Multiprocessing\queue_pipe_multiprocessing.py - PyCharm Community Edition 3.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py 〉 Multiprocessing 〉 queue_pipe_multiprocessing.py 〉 queue_pipe_multiprocessing

queue_pipe_multiprocessing.py ×

```python
import multiprocessing

def calc_square(numbers, result, v):
    v.value = 5.67
    for idx, n in enumerate(numbers):
        result[idx] = n*n

if __name__ == "__main__":
    numbers = [2,3,5]
    result = multiprocessing.Array('i',3)
    v = multiprocessing.Value('d', 0.0)
    p = multiprocessing.Process(target=calc_square, args=(numbers, result, v))

    p.start()
    p.join()

    print(v.value)
```

Run queue_...

```
"C:\Progr
5.67

Process f
```

## Multiprocessing Queue

```
import multiprocessing
q = multiprocessing.Queue()
```

- Lives in shared memory
- Used to share data between processes

## Queue Module

```
import queue
q = queue.Queue()
```

- Lives in in-process memory
- Used to share data between threads

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_queue_pipe.py - PyCharm Community Edition 5.0.3

5:06:09 PM 09/18/2016

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py 〉 Multiprocessing 〉 multiprocessing_queue_pipe.py 〉

multiprocessing_queue_pipe

multiprocessing_queue_pipe.py × | multiprocessing_value_array.py ×

```python
import multiprocessing

def calc_square(numbers, q):
    for n in numbers:
        q.put(n*n)

if __name__ == "__main__":
    numbers = [2,3,5]
    q = multiprocessing.Queue()
    p = multiprocessing.Process(target=calc_square, args=(numbe

    p.start()
    p.join()

    while q.empty() is False:
        print(q.get())
```

Run multiprocessing_queue_pipe

```
"C:\Program Files (x86)\Pythor
4
9
25

Process finished with exit coc
```

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_lock.py - PyCharm Community Edition 5.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py > Multiprocessing > multiprocessing_lock.py

multiprocessing_lock

multiprocessing_lock.py

```python
import time
import multiprocessing

def deposit(balance):
    for i in range(100):
        time.sleep(0.01)
        balance.value = balance.value + 1

def withdraw(balance):
    for i in range(100):
        time.sleep(0.01)
        balance.value = balance.value - 1

if __name__ == '__main__':
    balance = multiprocessing.Value('i', 200)
    d = multiprocessing.Process(target=deposit, args=(balance,))
    w = multiprocessing.Process(target=withdraw, args=(balance,))
    d.start()
    w.start()
    d.join()
    w.join()
    print(balance.value)
```

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_lock.py - PyCharm Community Edition 3.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py 〉 Multiprocessing 〉 multiprocessing_lock.py 〉 multiprocessing_lock

multiprocessing_lock.py ×

```python
import time
import multiprocessing

def deposit(balance, lock):
    for i in range(100):
        time.sleep(0.01)
        lock.acquire()
        balance.value = balance.value + 1
        lock.release()

def withdraw(balance, lock):
    for i in range(100):
        time.sleep(0.01)
        lock.acquire()
        balance.value = balance.value - 1
        lock.release()

if __name__ == '__main__':
    balance = multiprocessing.Value('i', 200)
    lock = multiprocessing.Lock()
    d = multiprocessing.Process(target=deposit, args=(balance,lock))
    w = multiprocessing.Process(target=withdraw, args=(balance,lock))
```

```python
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)
```

```
"C:\Program Files (x86)\Python 3.5\python.exe" C:/Code/py/Multipro
[1, 4, 9, 16, 25]

Process finished with exit code 0
```

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_pool.py - PyCharm Community Edition 5.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py > Multiprocessing > multiprocessing_pool.py

multiprocessing_pool.py ×



```python
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(100000))
    p.close()
    p.join()

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

    t2 = time.time()
    result = []
    for x in range(100000):
        result.append(f(x))

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

```
"C:\Program Files (x86)\Python 3.5\python.exe" C:/Code/py/Mult
```

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_pool.py - PyCharm Community Edition 5.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py > Multiprocessing > multiprocessing_pool.py

multiprocessing_pool

multiprocessing_pool.py

```python
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(100000))
    p.close()
    p.join()

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

    t2 = time.time()
    result = []
    for x in range(100000):
        result.append(f(x))

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

Run multiprocessing_pool

```
"C:\Program Files (x86)\Python 3.5\python.exe" C:/Code/py/Mult
Pool took: 2.8357343673706055
Serial processing took:  9.5448899269104

Process finished with exit code 0
```

py - [C:\Code\py] - ...\Multiprocessing\multiprocessing_pool.py - PyCharm Community Edition 5.0.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

py > Multiprocessing > multiprocessing_pool.py

multiprocessing_pool.py

```python
from multiprocessing import Pool


def f(n):
    return n*n

if __name__ == "__main__":
    p = Pool(processes=3)
    result = p.map(f,[1,2,3,4,5])
    for n in result:
        print(n)
```

Run multiprocessing_pool

```
"C:\Program F
1
4
9
16
25

Process finis
```