00A0 $\exists$ $\forall$ $\subseteq$ 2713x 27FA$\Longleftrightarrow$ $\surd$ $\sqrt[3]{}$ $\oplus$ $\otimes$

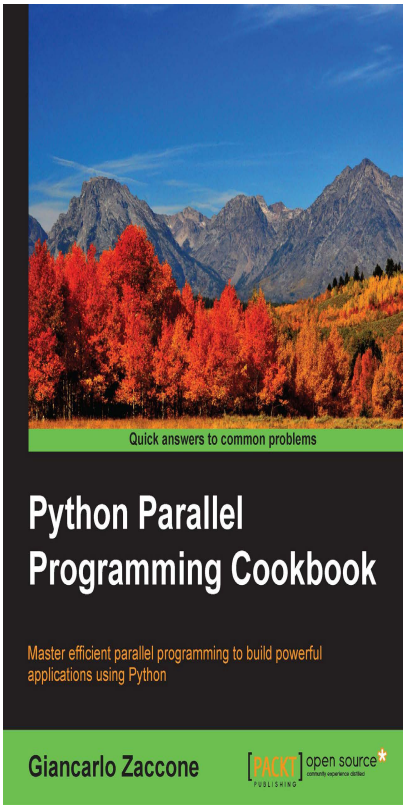
python-parallel-programming-cookbook-cn Documentation

1.0

laixintao

2020 11 03

Contents:


Quick answers to common problems
Python Parallel Programming Cookbook
Master efficient parallel programming to build powerful applications using Python
Giancarlo Zaccone
PACKT open source
PUBLISHING

CHAPTER 1

Python

1.1

F

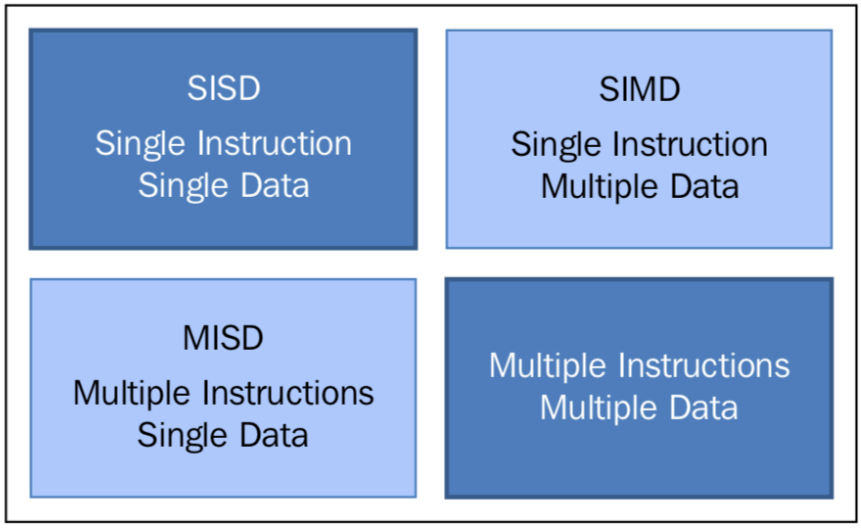
Python Python Python

GPU

1.2

- (SISD)
- (SIMD)
- (MISD)
- (MIMD)

“ ”:



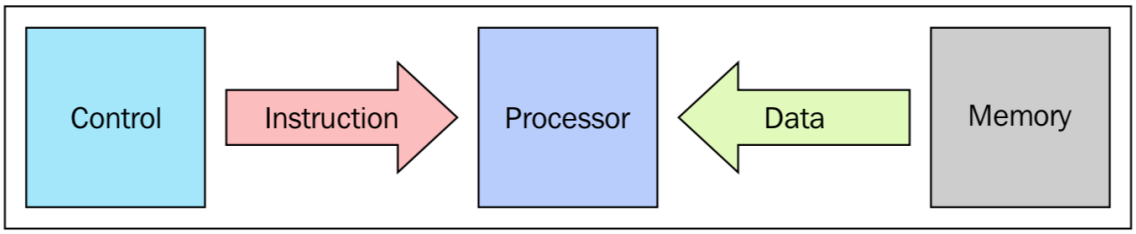
1.2.1 SISD

“ CPU ” SISD

“CPU ” CPU

- **Fetch**: CPU
- **Decode**: CPU
- **Execute**:

Execute CPU 1



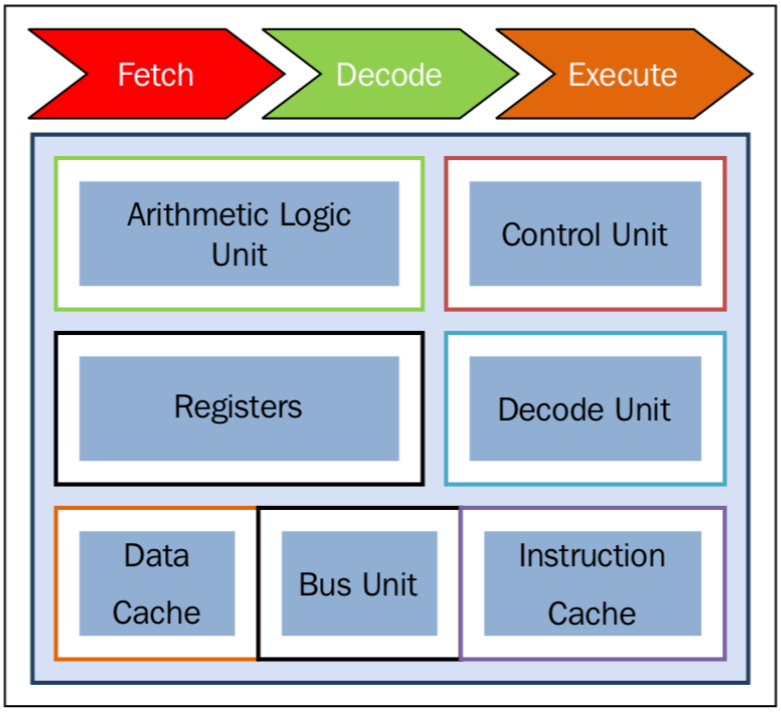
The SISD architecture schema

CPU SISD

.

-
- CPU /
- I/O

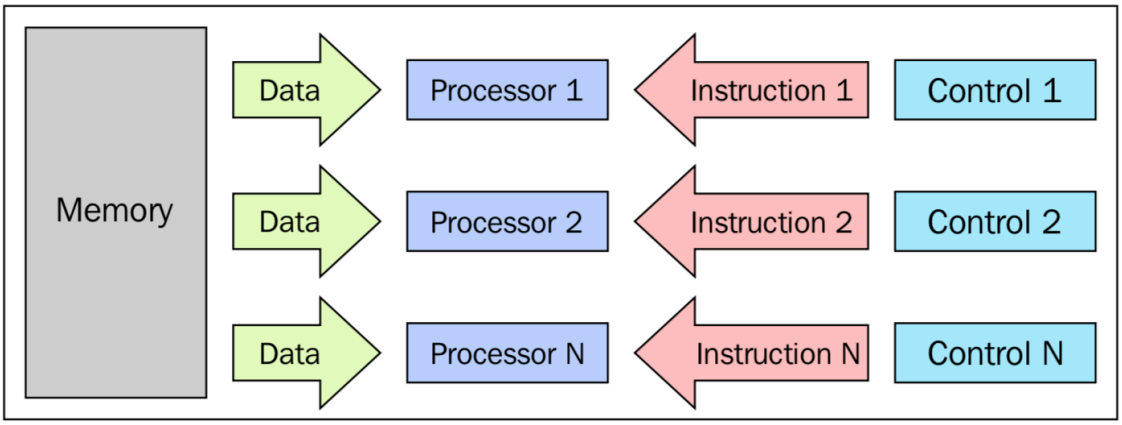
SISD CPU Fetch Decode Execute



CPU's components in the fetch-decode-execute phase

1.2.2 MISD

n CPU MISD



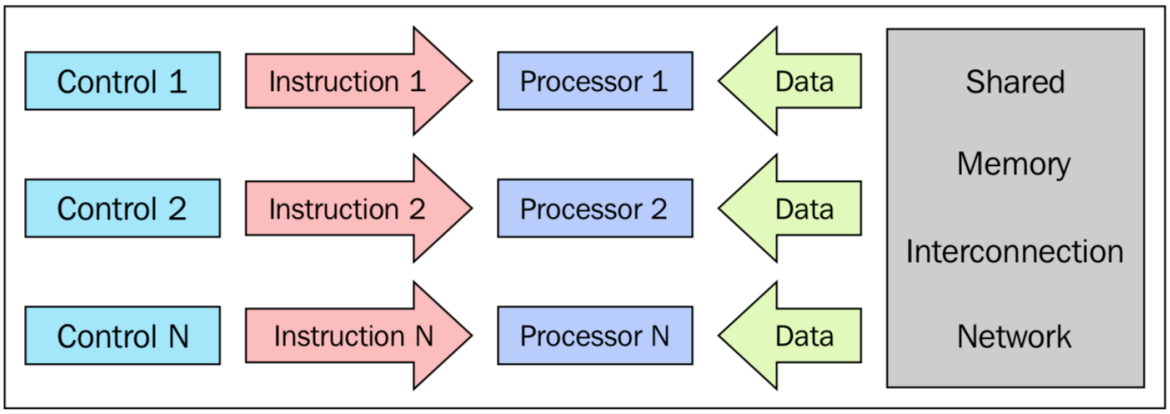
The MISD architecture scheme

1.2.3 SIMD

SIMD n SIMD MISD SIMD
Machine 1985 Thinking Machine) MPP NASA-1983 . GPU Python GPU SIMD

1.2.4 MIMD

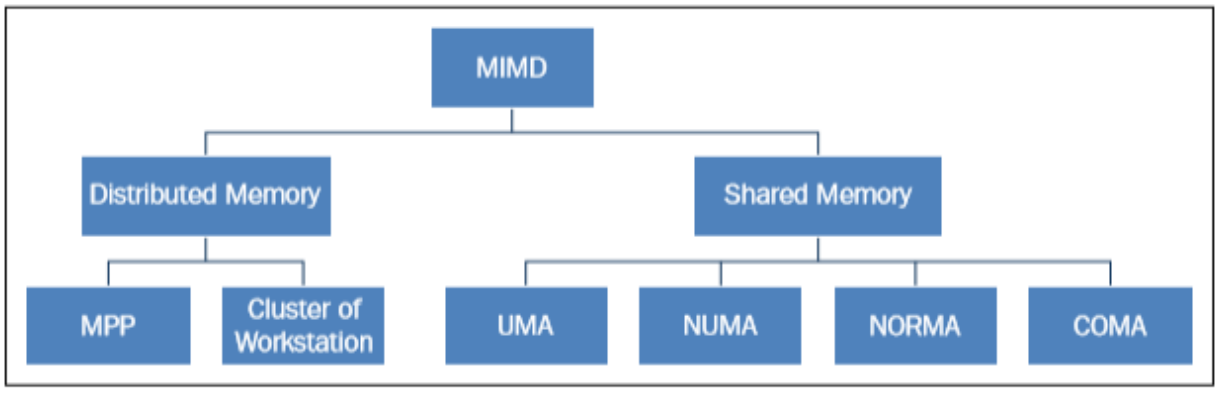
n n n MIMD SIMD MIMD



The MIMD architecture scheme

1.3

I/O

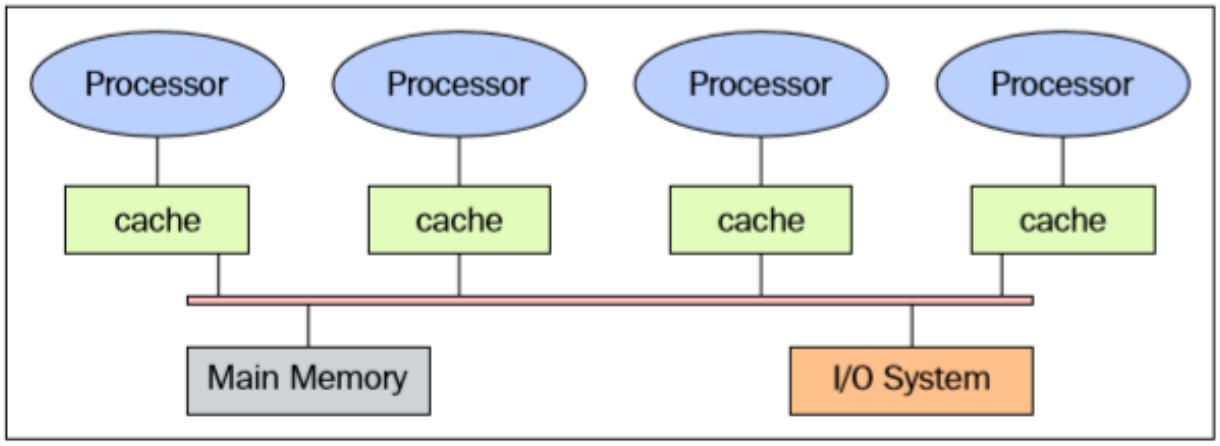


The memory organization in MIMD architecture

MIMD
load R0,i i R0 i
R0 i R0 R0 i

1.3.1

Cache Cache



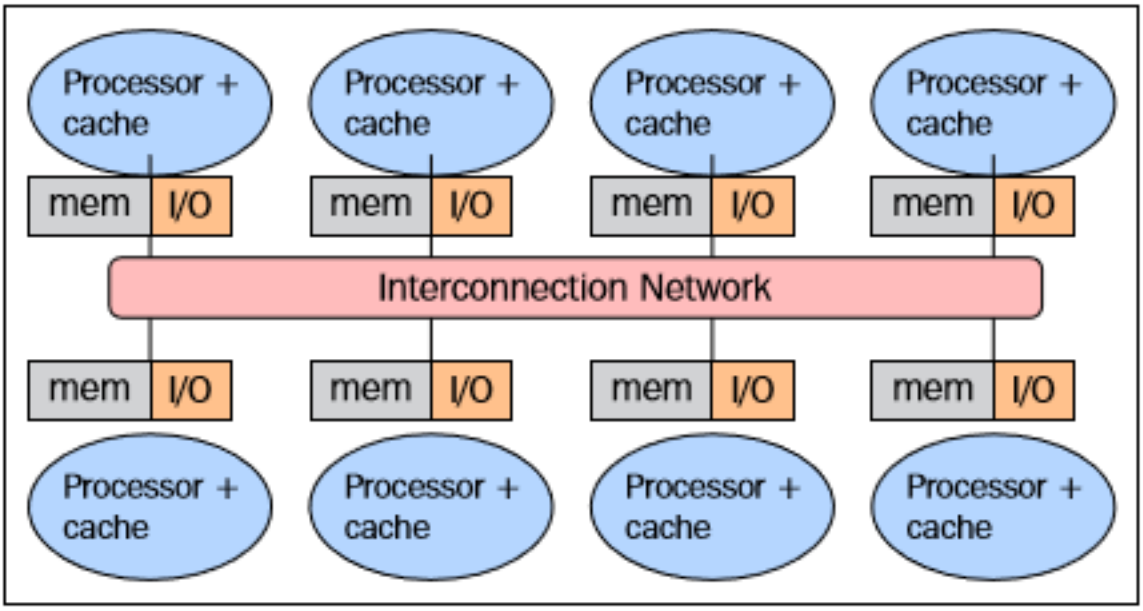
The shared memory architecture schema

-
-
-
-

- (Uniform memory access (UMA)) (symmetric multiprocessor (SMP))
- (Non-uniform memory access (NUMA)) (Distributed Shared Memory Systems (DSM))
- (No remote memory access (NORMA))
- Cache (Cache only memory access (COMA)) Cache NUMA Cache COMA Cache

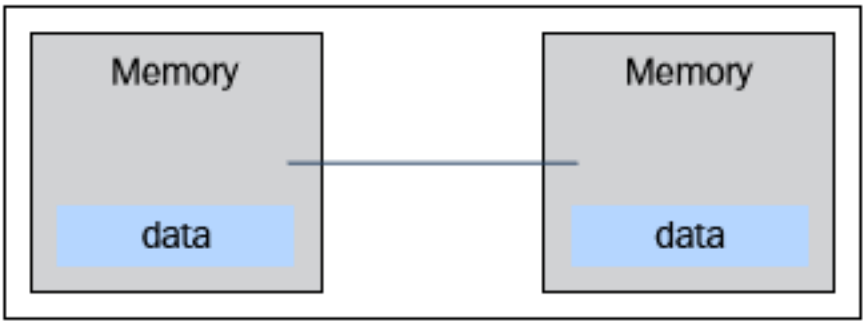
1.3.2

“ ”



The distributed memory architecture scheme

Cache



Basic message passing

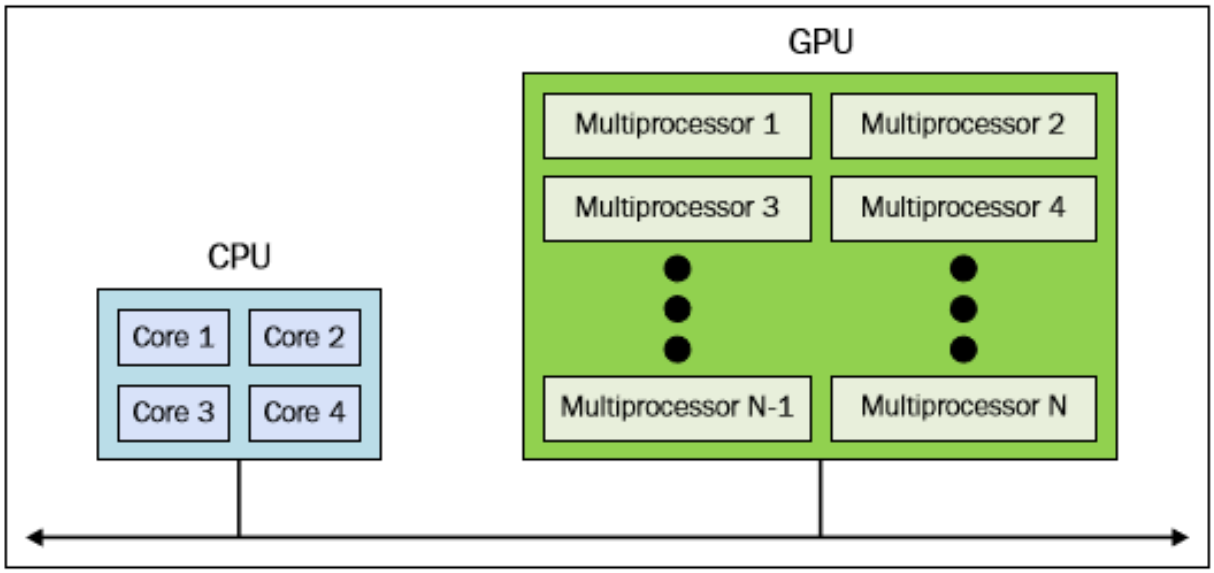
-
- ( )
- —— CPU
- CPU

(Massively parallel processing (MPP))

MPP ( ) :Earth Simulator Blue Gene ASCI White ASCI Red ASCI Purple Red Storm

- (The fail-over cluster)
- (The load balancing cluster)
- (The high-performance cluster) :

GPU GPU GPU CPU CPU CPU GP



The heterogeneous architecture scheme

1.4

-
-
- /
-

Python

1.4.1

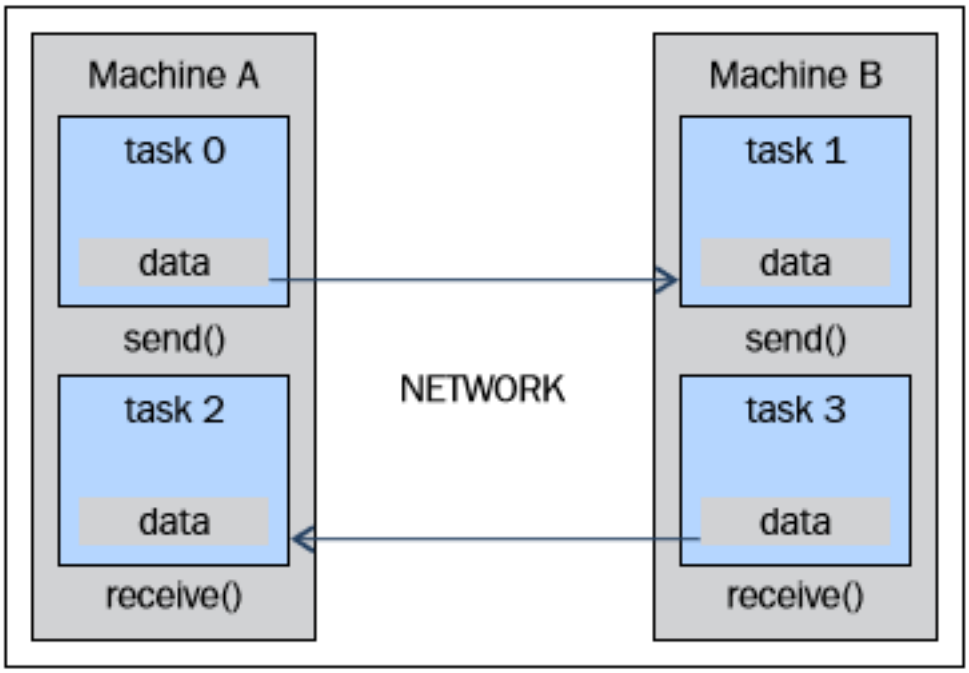
, ;

1.4.2

CPU POSIX
Intel (Hyper-threading) I/O

1.4.3

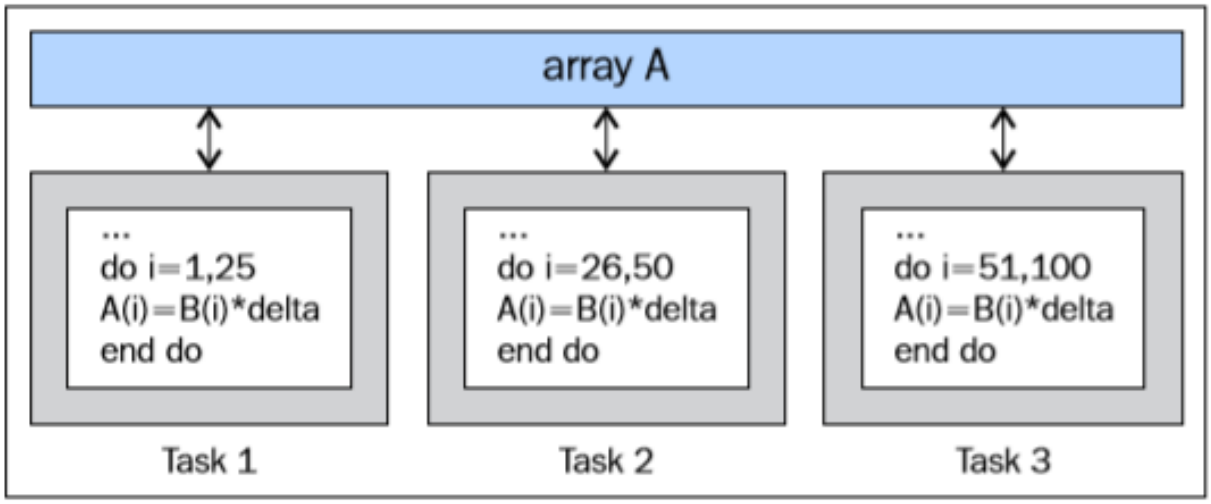
20 80 90 ——
MPI (the Message Passing Interface,) MPI



The message passing paradigm model

1.4.4

GPU



The data parallel paradigm model

1.5

- (Task decomposition)
- (Task assignment)
- (Agglomeration)
- (Mapping)

1.5.1

- (Domain decomposition)
- (Functional decomposition)

1.5.2

1.5.3

(GPU)

(TCP)

1.5.4

-
-

NP —— ( )

1.5.5

( )

1.5.6 / (Manager/worker)

1.5.7 / (Hierarchical manager/worker)

/ ( )

1.5.8 (Decentralize)

1.6

/

(Ahmdal's law) (Gustafson's law)

1.6.1

T_S p T_P $S = \frac{T_S}{T_P}$ $S = p$
T_S T_S

- $S = p$

- $S < p$
- $S > p$

1.6.2

p p

E $E = \frac{S}{P} = \frac{T_S}{pT_p}$ $E = 1$ E 1

- $E = 1$
- $E < 1$
- $E \ll 1$

1.6.3

( )

1.6.4 (Ahmdal's law)

$S = \frac{1}{1-p}$ $1 - p$ 90% 10% 9

1.6.5 (Gustafson's law)

-
-

$S(P) = P - \alpha(P - 1)$ P S α

( )

1.7 Python

Python

-
-
- Python C/C++ 10
- Exception
-
-

Python Python C/C++ Python C/C++
: https://www.python.org/doc/essays/omg-darpa-mcc-position/

1.7.1

Python https://www.python.org/downloads/

NotePad TextEdit Python Integrated Development Environment, IDE

Python IDE IDEL https://docs.python.org/3/library/idle.html PyCharm https://www.jetbrains.com/pycharm/ Sublime Textd https://www.sublimetext.com/

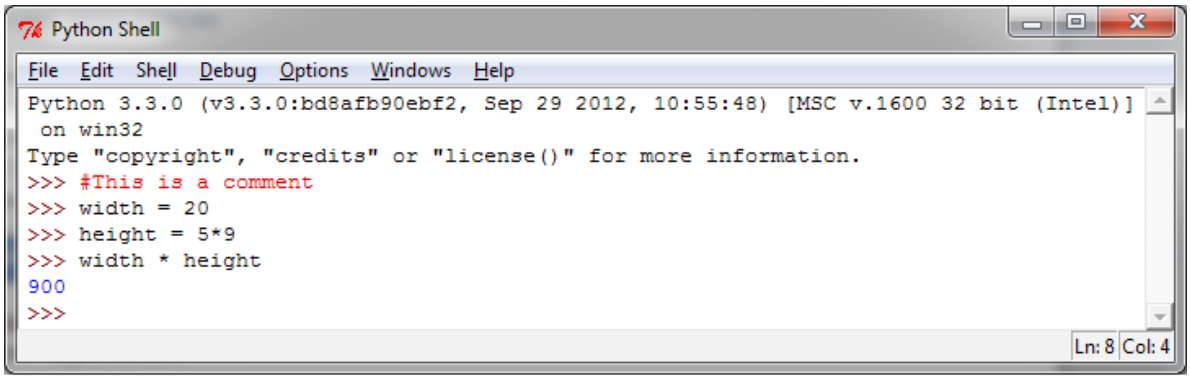
1.7.2 ...

Python >>> Python

-

```
>>> # This is a comment
>>> width = 20
>>> height = 5*9
>>> width * height
900
```

Python :



- (abs(a) = 5

```
>>> a=1.5+0.5j
>>> a.real
1.5
>>> a.imag
0.5
>>> abs(a)  # sqrt(a.real**2 + a.imag**2)
1.5811388300841898
```

-

```
>>> word = 'Help' + 'A' >>> word
'HelpA'
>>> word[4]
'A'
>>> word[0:2]
'He'
>>> word[-1]  #
'A'
```

- list

```
>>> a = ['spam', 'eggs', 100, 1234] >>> a[0]
'spam'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['eggs', 100]
>>> len(a)
4
```

- while

```
# Fibonacci series:
>>> while b < 10:
...     print b
...     a, b = b, a+b
...
1
1
2
3
5
8
```

- if input()

```
>>>x = int(input("Please enter an integer here: "))
Please enter an integer here:
```

if

```
>>>if x < 0:
...      print ('the number is negative')
...elif x == 0:
...      print ('the number is zero')
...elif x == 1:
...      print ('the number is one')
...else:
...      print ('More')
...
```

- for :

```
>>> # Measure some strings:
... a = ['cat', 'window', 'defenestrate'] >>> for x in a:
... print (x, len(x))
...
cat 3
window 6
defenestrate 12
```

-

```
>>> def fib(n):  #  n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while b < n:
...         print(b),
...         a, b = b, a+b
>>> # Now call the function we just defined:
... fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

-

```
>>> import math
>>> math.sin(1)
0.8414709848078965
>>> from math import *
>>> log(1)
0.0
```

-

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

1.8 Python

Python C C++ Python Python Python C CPython Python Pyt
Python

1.9

“ ”

-
- CPU ——
-

Python Python

1.10 Python

Python

1.10.1

Python

Python https://www.python.org/

1.10.2 ...

- called_Process.py
- calling_Process.py

Python IDE(3.3.0)

called_Process :

```
print("Hello Python Parallel Cookbook!!")
closeInput = raw_input("Press ENTER to exit")
print"Closing calledProcess"
```

calling_Process

```
## The following modules must be imported
import os
import sys

## this is the code to execute
```

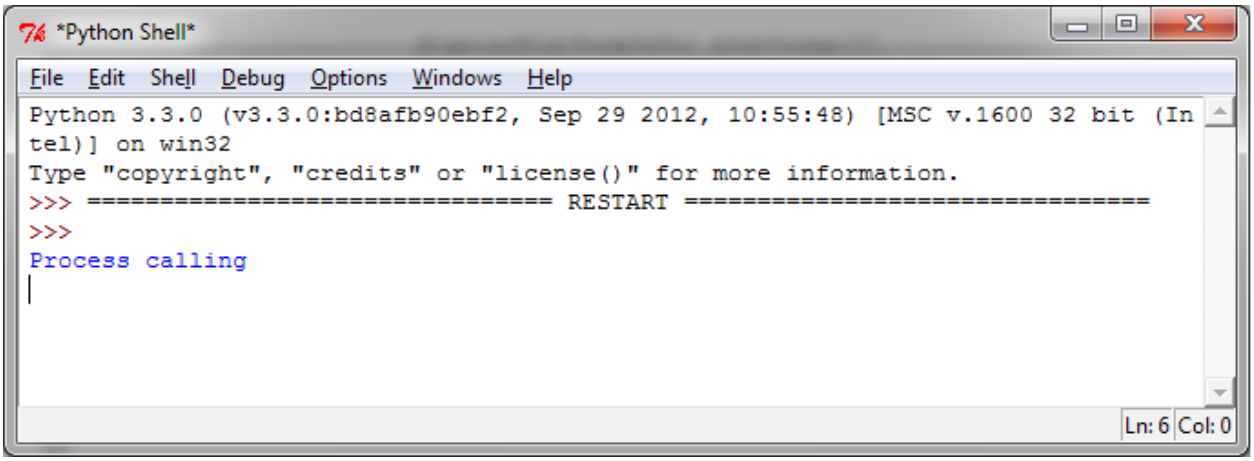
( )

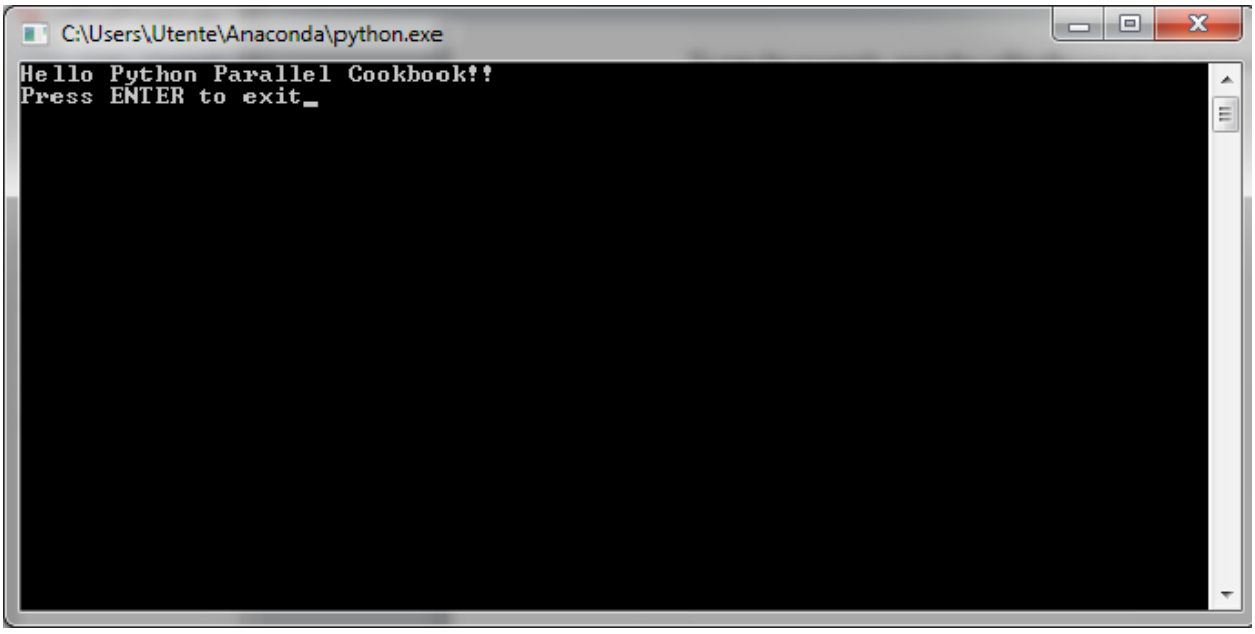
( )

```
program = "python"
print("Process calling")
arguments = ["called_Process.py"]

## we call the called_Process.py script
os.execvp(program, (program,) + tuple(arguments))
print("Good Bye!!")
```

Python IDE calling_Process F5.

Python shell





Enter

1.10.3 ...

execvp ”Good Bye” called_Process called_Process`` ``input() multiprocessing

1.11 Python

Python Python CPython Global Interpreter Lock GIL Python Python GIL

Python

1.11.1 ...

helloPythonWithThreads.py

```
# To use threads you need import Thread using the following code:
from threading import Thread
# Also we use the sleep function to make the thread "sleep"
from time import sleep

# To create a thread in Python you'll want to make your class work as a thread.
# For this, you should subclass your class from the Thread class
class CookBook(Thread):
    def __init__(self):
        Thread.__init__(self)
        self.message = "Hello Parallel Python CookBook!!\n"

    # this method prints only the message
    def print_message(self):
        print(self.message)

    # The run method prints ten times the message
    def run(self):
        print("Thread Starting\n")
        x = 0
        while (x < 10):
            self.print_message()
            sleep(2)
            x += 1
        print("Thread Ended\n")

# start the main process
print("Process Started")

# create an instance of the HelloWorld class
hello_Python = CookBook()

# print the message...starting the thread
hello_Python.start()
```

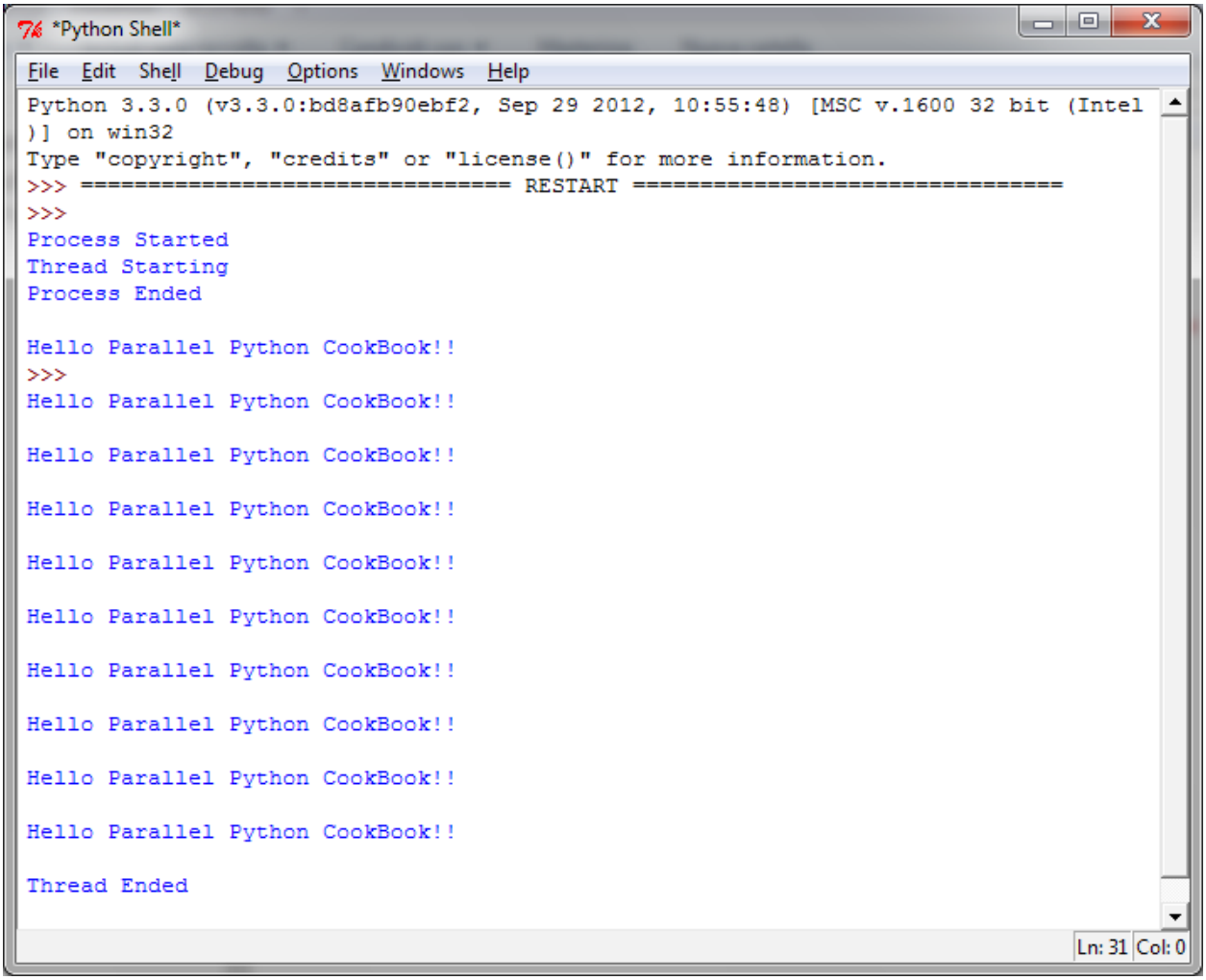
( )

( )

```
# end the main process
print("Process Ended")
```

Python IDE `helloPythonWithThreads.py` F5.

Python shell



1.11.2

CHAPTER 2

2.1

3 ready,running,blocked ——

2.2 Python

Python `threading`

-
- Lock
- RLock
-
-
-

Python 3.3 Python 2.7

2.3

Thread `start()` Python threading `Thread()`

```
class threading.Thread(group=None,
                       target=None,
                       name=None,
                       args=(),
                       kwargs={})
```

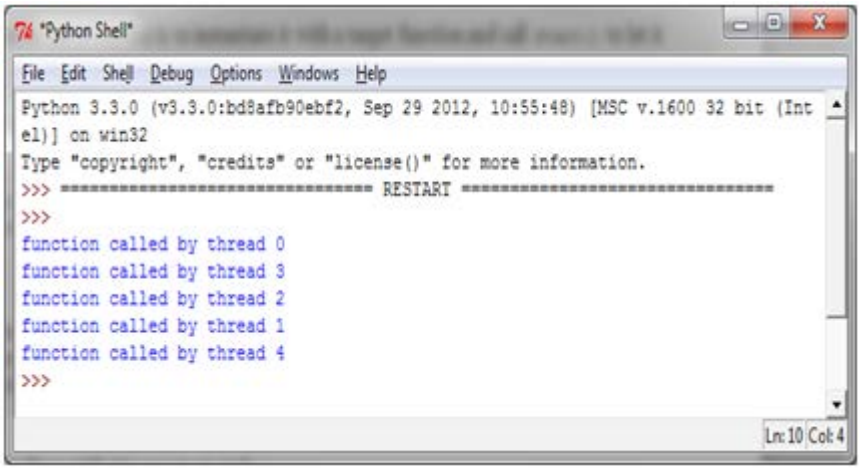
- `group`: `None`
- `target`:
- `name`: `Thread-N`
- `args`: `target` tuple
- `kwargs`: dict

‘target‘ ‘arg‘ ‘kwarg‘

2.3.1 ...

threading

```
import threading

def function(i):
    print ("function called by thread %i\n" % i)
    return

threads = []

for i in range(5):
    t = threading.Thread(target=function , args=(i, ))
    threads.append(t)
    t.start()
    t.join()
```



stdout

t.join() t t t for t 01234

2.3.2

threading python

```
import threading
```

function Thread

```
t = threading.Thread(target=function , args=(i, ))
```

start() join() t

```
t.start()
t.join()
```

2.4

Thread

2.4.1 ...

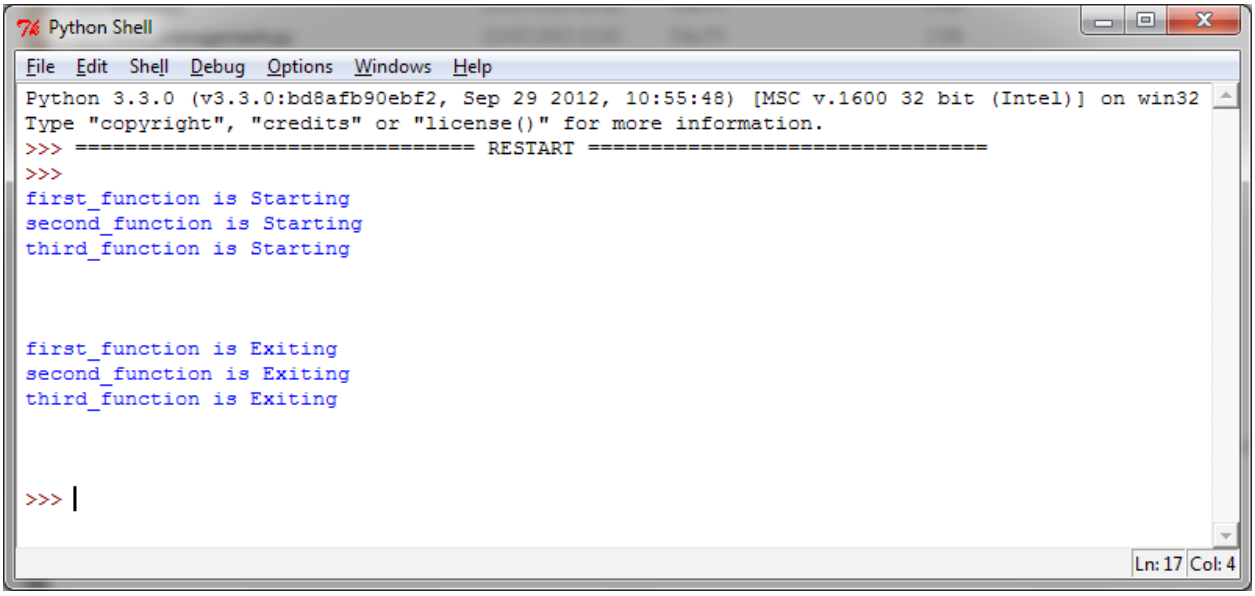
time 2s

```
import threading
import time

def first_function():
    print(threading.currentThread().getName() + str(' is Starting '))
    time.sleep(2)
    print (threading.currentThread().getName() + str(' is Exiting '))
    return
```

( )

( )

```
def second_function():
    print(threading.currentThread().getName() + str(' is Starting '))
    time.sleep(2)
    print (threading.currentThread().getName() + str(' is Exiting '))
    return

def third_function():
    print(threading.currentThread().getName() + str(' is Starting '))
    time.sleep(2)
    print(threading.currentThread().getName() + str(' is Exiting '))
    return

if __name__ == "__main__":
    t1 = threading.Thread(name='first_function', target=first_function)
    t2 = threading.Thread(name='second_function', target=second_function)
    t3 = threading.Thread(name='third_function', target=third_function)
    t1.start()
    t2.start()
    t3.start()
```



2.4.2

name

```
t1 = threading.Thread(name='first_function', target=first_function)
t2 = threading.Thread(name='second_function', target=second_function)
t3 = threading.Thread(target=third_function)
```

3 Thread-1 is Starting Thread-1 is Exiting

start() join()

```
t1.start()
t2.start()
t3.start()
t1.join()
t2.join()
t3.join()
```

2.5

threading 3

- Thread
- __init__(self [,args])
- run(self, [,args])

Thread start() run()

2.5.1 ...

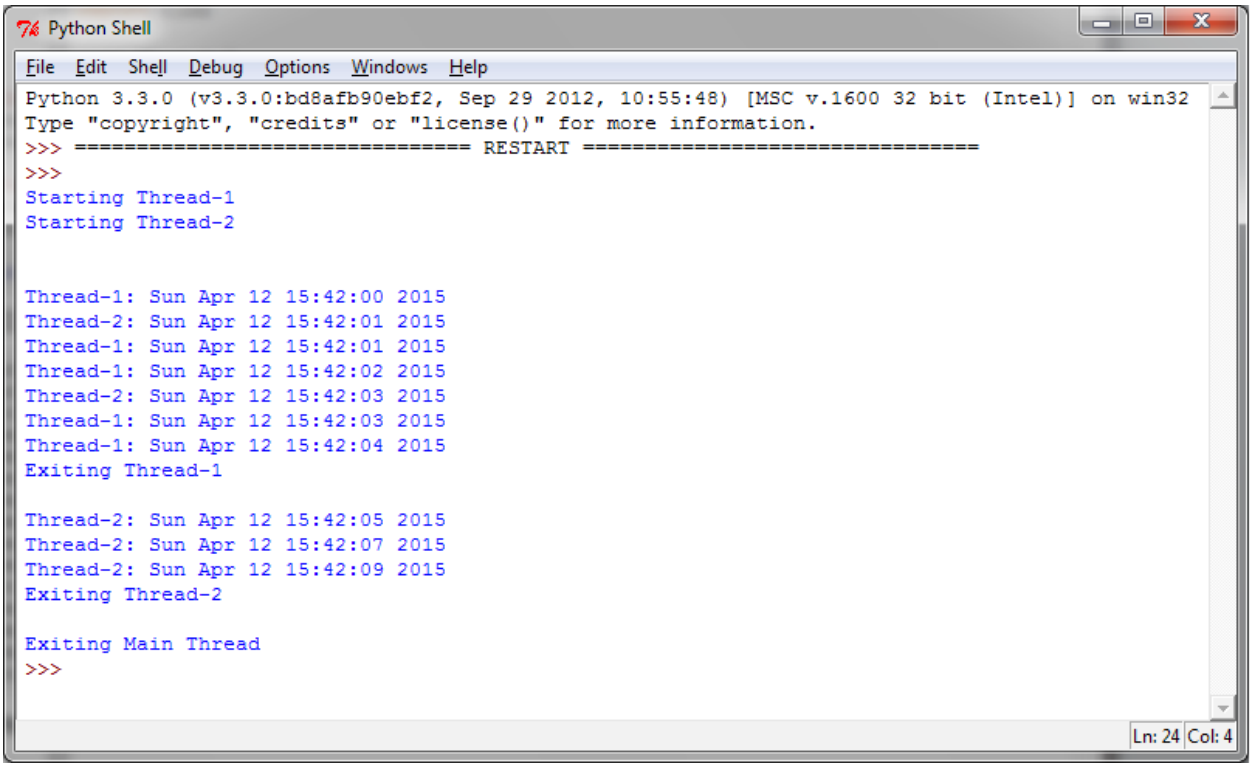
myThread

```
import threading
import time

exitFlag = 0

class myThread (threading.Thread):
    def __init__(self, threadID, name, counter):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.counter = counter

    def run(self):
        print("Starting " + self.name)
        print_time(self.name, self.counter, 5)
        print("Exiting " + self.name)

def print_time(threadName, delay, counter):
    while counter:
        if exitFlag:
            #      thread  Python3    thread _thread
            # import _thread
            # _thread.exit()
            thread.exit()
        time.sleep(delay)
        print("%s: %s" % (threadName, time.ctime(time.time())))
        counter -= 1

# Create new threads
```

( )

( )

```
thread1 = myThread(1, "Thread-1", 1)
thread2 = myThread(2, "Thread-2", 2)

# Start new Threads
thread1.start()
thread2.start()

#
#
thread1.join()
thread2.join()
print("Exiting Main Thread")
```



2.5.2

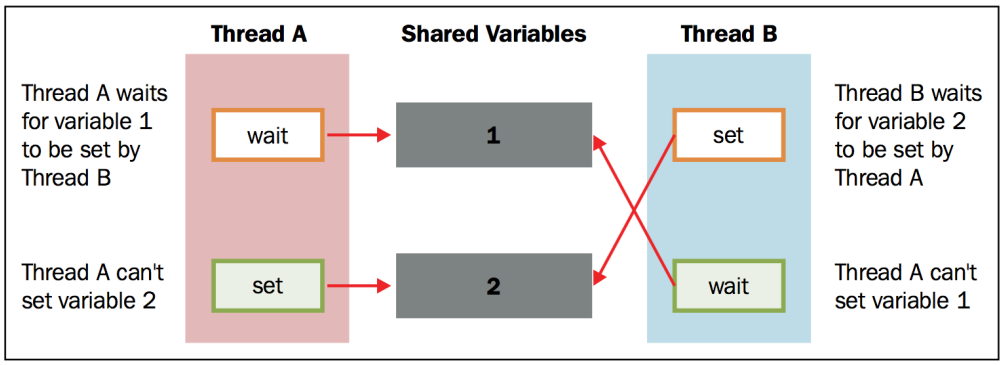
threading Thread run() myThread start()
Thread.__init__ start() run()

join()

2.6 Lock

bug

work



Deadlock

A B) 1 2. A 1 B 2. A 2 B 1

Python lock()

2.6.1 ...

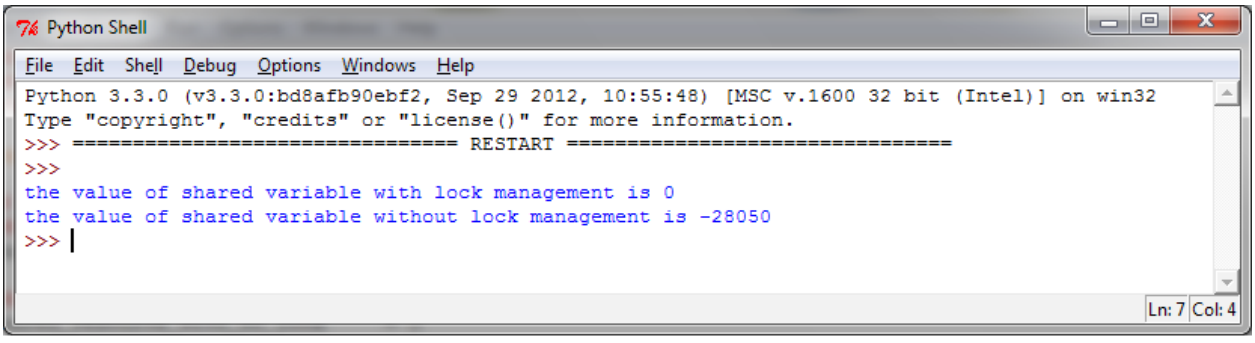
lock() increment() decrement() 1 1. 0

```
# -*- coding: utf-8 -*-

import threading

shared_resource_with_lock = 0
shared_resource_with_no_lock = 0
COUNT = 100000
shared_resource_lock = threading.Lock()

#
def increment_with_lock():
    global shared_resource_with_lock
    for i in range(COUNT):
        shared_resource_lock.acquire()
        shared_resource_with_lock += 1
        shared_resource_lock.release()

def decrement_with_lock():
    global shared_resource_with_lock
    for i in range(COUNT):
        shared_resource_lock.acquire()
```

( )

( )

```
        shared_resource_with_lock -= 1
        shared_resource_lock.release()

#
def increment_without_lock():
    global shared_resource_with_no_lock
    for i in range(COUNT):
        shared_resource_with_no_lock += 1

def decrement_without_lock():
    global shared_resource_with_no_lock
    for i in range(COUNT):
        shared_resource_with_no_lock -= 1

if __name__ == "__main__":
    t1 = threading.Thread(target=increment_with_lock)
    t2 = threading.Thread(target=decrement_with_lock)
    t3 = threading.Thread(target=increment_without_lock)
    t4 = threading.Thread(target=decrement_without_lock)
    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t1.join()
    t2.join()
    t3.join()
    t4.join()
    print ("the value of shared variable with lock management is %s" % shared_resource_
↪with_lock)
    print ("the value of shared variable with race condition is %s" % shared_resource_
↪with_no_lock)
```



2.6.2

```
t1 = threading.Thread(target=increment_with_lock)
t2 = threading.Thread(target=decrement_with_lock)
```

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

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

`increment_with_lock()` `decrement_with_lock()` lock `acquire()` `release()`:

```
shared_resource_lock.acquire()
shared_resource_with_lock -= 1
shared_resource_lock.release()
```

- locked unlocked
- `acquire()` `release()`

- unlocked `acquire()` locked
- locked `acquire()` block `release()`
- unlocked `release()` `RuntimError`
- locked `release()` unlocked

2.6.3

debug

2.7 RLock

`RLock()` `Lock()` `RLock()` `acquire()` `release()` `RLock()`

RLock RLock “Reentrant Lock” “ ” Lock 1. A B A 2. acquire 3. acquire release release RLock unlocked

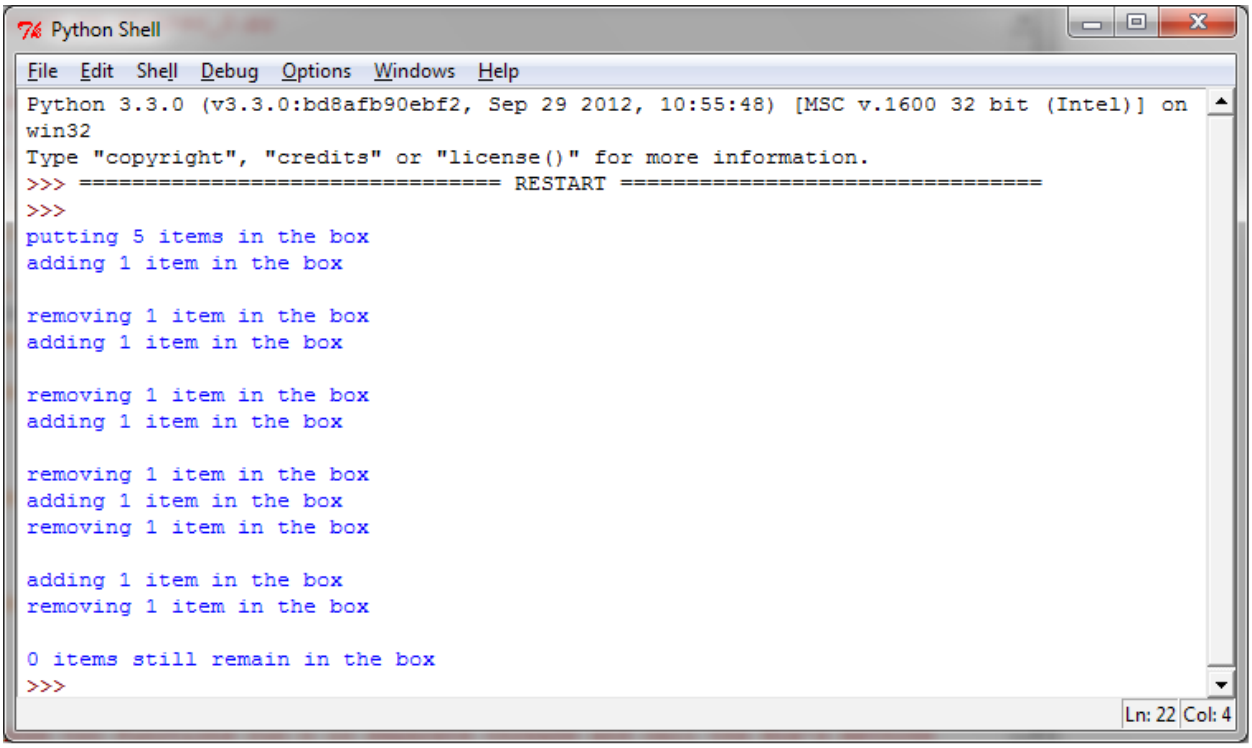
2.7.1 ...

`Box` `add()` `remove()` `execute()` `execute()` `Rlock()`

```
import threading
import time

class Box(object):
    lock = threading.RLock()

    def __init__(self):
        self.total_items = 0

    def execute(self, n):
        Box.lock.acquire()
        self.total_items += n
        Box.lock.release()

    def add(self):
        Box.lock.acquire()
        self.execute(1)
        Box.lock.release()

    def remove(self):
        Box.lock.acquire()
        self.execute(-1)
        Box.lock.release()

## These two functions run n in separate
## threads and call the Box's methods
def adder(box, items):
    while items > 0:
        print("adding 1 item in the box")
        box.add()
        time.sleep(1)
        items -= 1

def remover(box, items):
    while items > 0:
        print("removing 1 item in the box")
        box.remove()
        time.sleep(1)
        items -= 1

## the main program build some
## threads and make sure it works
if __name__ == "__main__":
    items = 5
    print("putting %s items in the box " % items)
    box = Box()
    t1 = threading.Thread(target=adder, args=(box, items))
    t2 = threading.Thread(target=remover, args=(box, items))
    t1.start()
    t2.start()

    t1.join()
```

( )

```
    t2.join()
    print("%s items still remain in the box " % box.total_items)
```



2.7.2

t1 t2 adder() remover() item 0 RLock() Box

```
class Box(object):
    lock = threading.RLock()
```

adder() remover() Box items Box add() remove() lock() RLock() acquire() release()

```
Box.lock.acquire()
# ...do something
Box.lock.release()
```

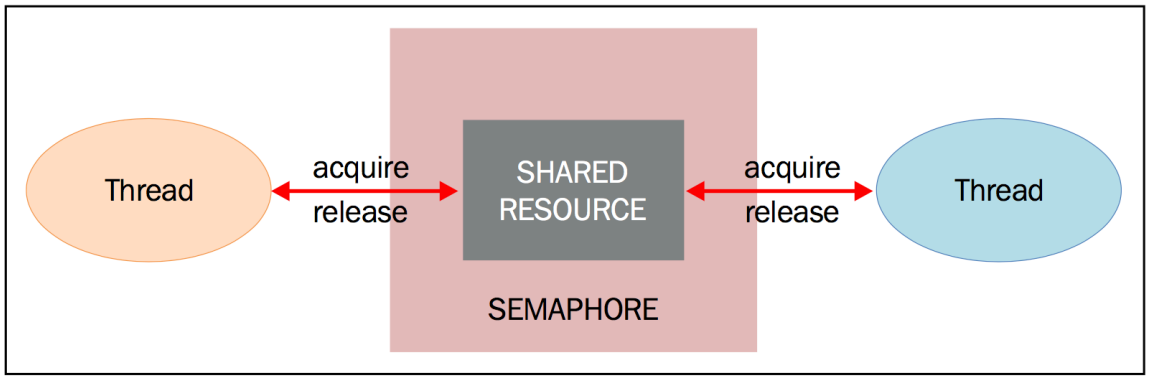
2.8

E.Dijkstra

threading acquire() release()

- acquire() ,
- release()

.



Thread synchronization with semaphores

1 A 1 0 B B 0 -1 A

2.8.1

producer() consumer() item producer() item consumer() item item consumer() item producer()

2.8.2 ...

- item

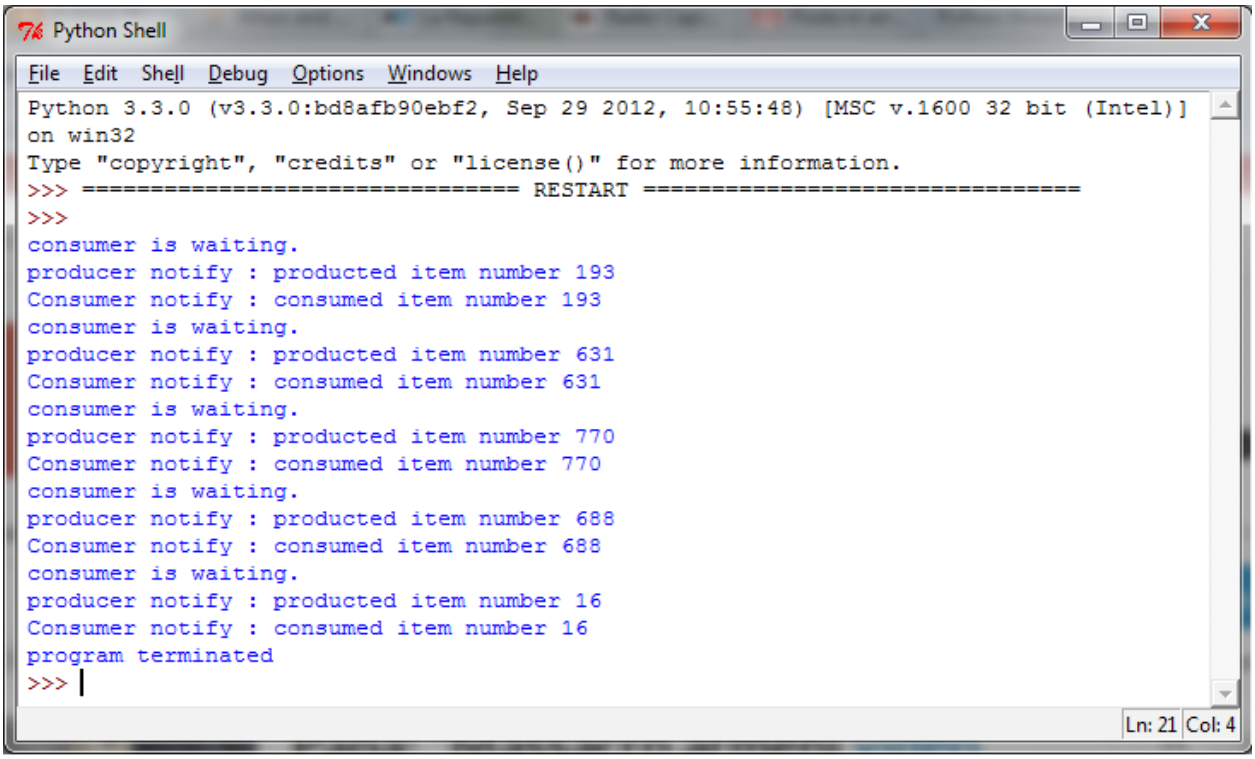
```
# -*- coding: utf-8 -*-

"""Using a Semaphore to synchronize threads"""
import threading
import time
import random

# The optional argument gives the initial value for the internal
# counter;
# it defaults to 1.
# If the value given is less than 0, ValueError is raised.
semaphore = threading.Semaphore(0)

def consumer():
        print("consumer is waiting.")
        # Acquire a semaphore
        semaphore.acquire()
        # The consumer have access to the shared resource
        print("Consumer notify : consumed item number %s " % item)
```

( )

( )

```
def producer():
        global item
        time.sleep(10)
        # create a random item
        item = random.randint(0, 1000)
        print("producer notify : produced item number %s" % item)
         # Release a semaphore, incrementing the internal counter by one.
        # When it is zero on entry and another thread is waiting for it
        # to become larger than zero again, wake up that thread.
        semaphore.release()

if __name__ == '__main__':
        for i in range (0,5) :
                t1 = threading.Thread(target=producer)
                t2 = threading.Thread(target=consumer)
                t1.start()
                t2.start()
                t1.join()
                t2.join()
        print("program terminated")
```

5



2.8.3

0

```
semaphore = threading.Semaphore(0)
```

lock producer() item

```
semaphore.release()
```

release() consumer()

```
semaphore.acquire()
```

0 acquire() 0 -1

```
print("Consumer notify : consumed item number %s " % item)
```

2.8.4

1

t1 s1 s2 t2 s2 s1 t1 s2 t2 s1

2.9

2.9.1

-

2.9.2 ...

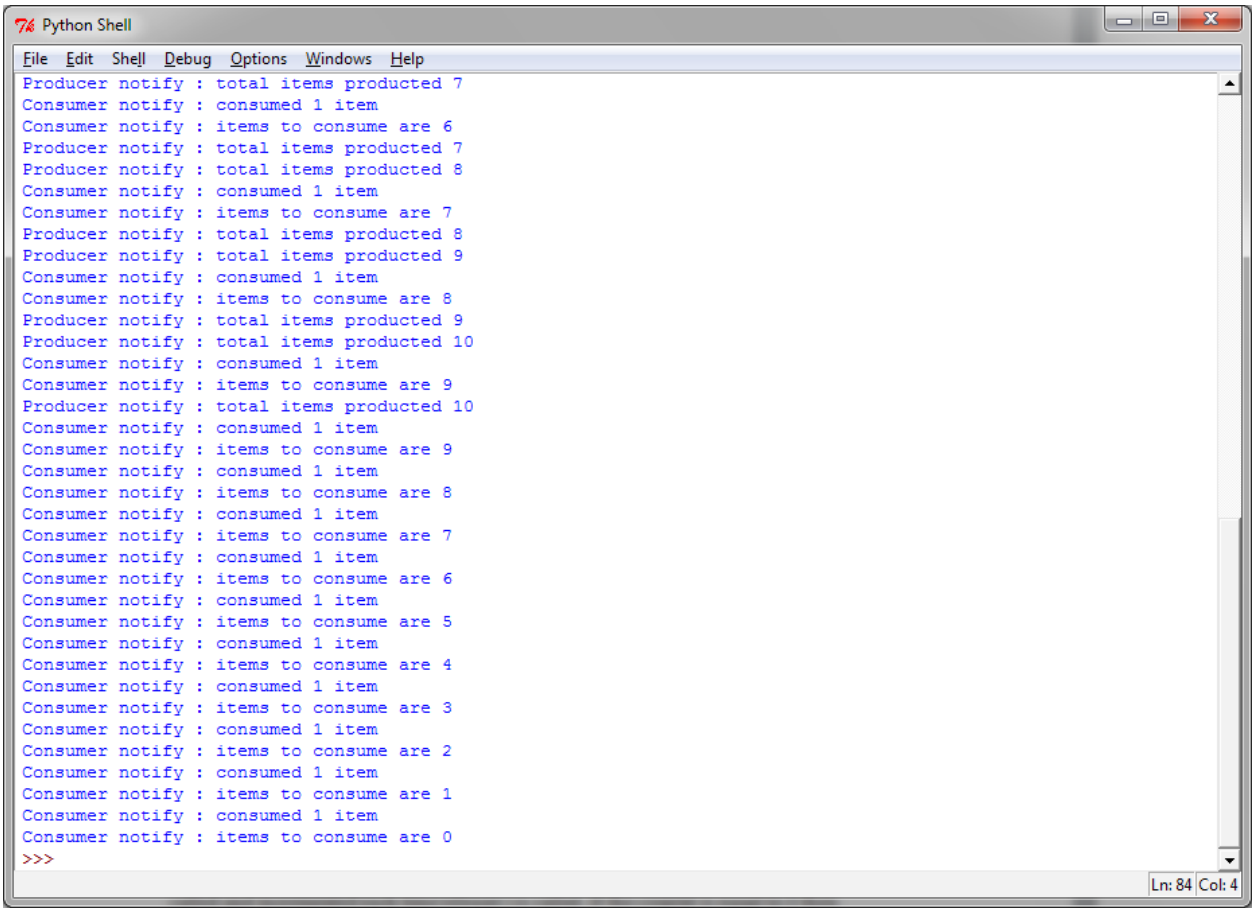
-

```
from threading import Thread, Condition
import time

items = []
condition = Condition()

class consumer(Thread):

    def __init__(self):
        Thread.__init__(self)

    def consume(self):
        global condition
        global items
        condition.acquire()
        if len(items) == 0:
```

( )

( )

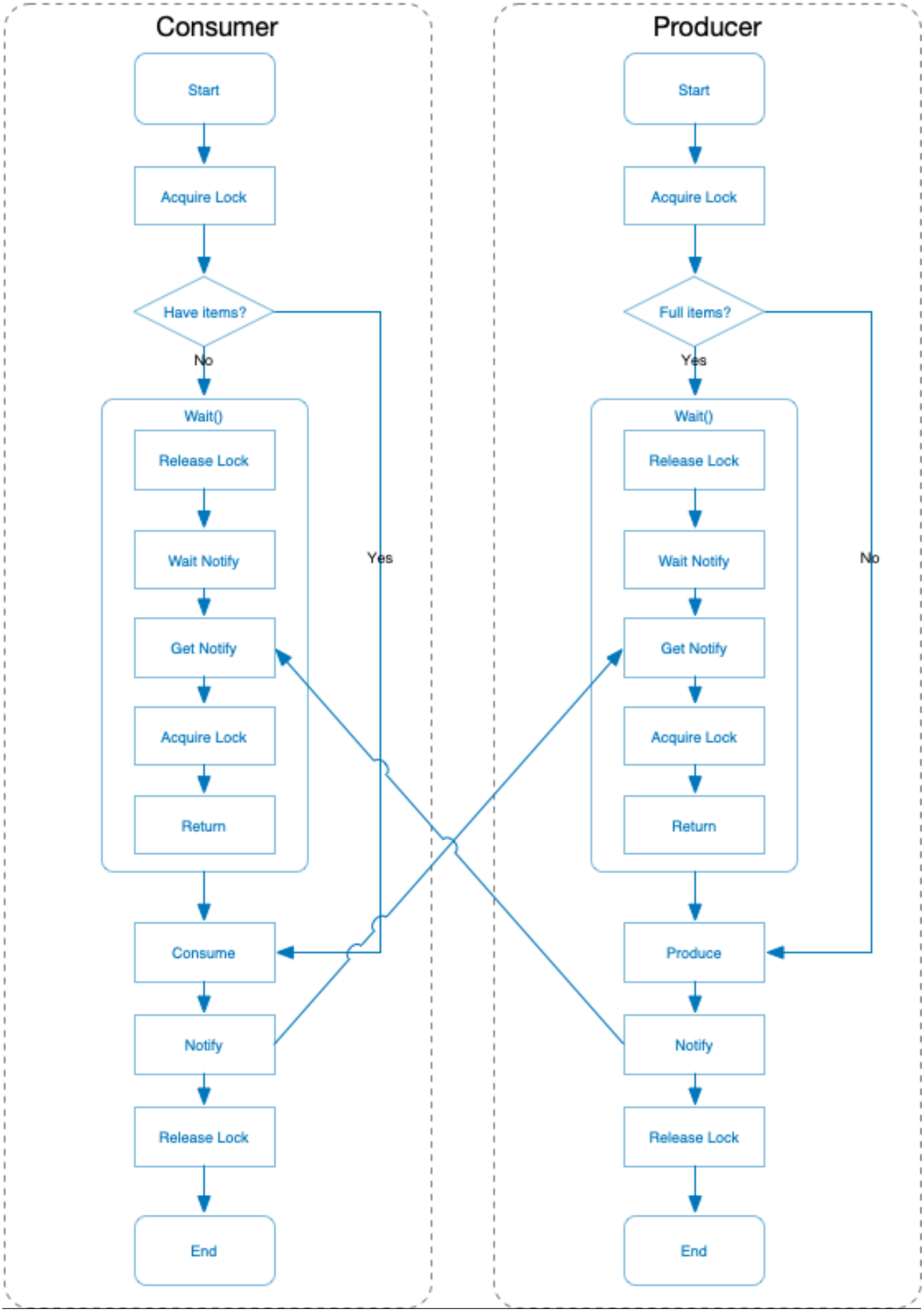
```
            condition.wait()
            print("Consumer notify : no item to consume")
        items.pop()
        print("Consumer notify : consumed 1 item")
        print("Consumer notify : items to consume are " + str(len(items)))

        condition.notify()
        condition.release()

    def run(self):
        for i in range(0, 20):
            time.sleep(2)
            self.consume()

class producer(Thread):

    def __init__(self):
        Thread.__init__(self)

    def produce(self):
        global condition
        global items
        condition.acquire()
        if len(items) == 10:
            condition.wait()
            print("Producer notify : items producted are " + str(len(items)))
            print("Producer notify : stop the production!!")
        items.append(1)
        print("Producer notify : total items producted " + str(len(items)))
        condition.notify()
        condition.release()

    def run(self):
        for i in range(0, 20):
            time.sleep(1)
            self.produce()

if __name__ == "__main__":
    producer = producer()
    consumer = consumer()
    producer.start()
    consumer.start()
    producer.join()
    consumer.join()
```



2.9.3

(condition.acquire() .wait() .wait() .notify() . notify() .notify() .release()

wait()


Consumer
Start
Acquire Lock
Have items?
No
Wait()
Release Lock
Wait Notify
Get Notify
Acquire Lock
Return
Consume
Notify
Release Lock
End
Yes
Producer
Start
Acquire Lock
Full items?
Yes
Wait()
Release Lock
Wait Notify
Get Notify
Acquire Lock
Return
Produce
Notify
Release Lock
End
No

https://docs.python.org/3/library/threading.html)

items[]

```
condition.acquire()
```

list 0

```
if len(items) == 0:
    condition.wait()
```

pop item

```
items.pop()
```

```
condition.notify()
condition.release()
```

10 item

```
condition.acquire()
if len(items) == 10:
    condition.wait()
```

1 item

```
condition.notify()
condition.release()
```

2.9.4

Python _Condition RLock() RLock acquire() release()

```
class _Condition(_Verbose):
    def __init__(self, lock=None, verbose=None):
        _Verbose.__init__(self, verbose)
        if lock is None:
            lock = RLock()
        self.__lock = lock
```

(3 ABC 10 Condition

```
# -*- coding: utf-8 -*-

"""
Three threads print A B C in order.
"""


from threading import Thread, Condition

condition = Condition()
```

( )

( )

```
current = "A"


class ThreadA(Thread):
    def run(self):
        global current
        for _ in range(10):
            with condition:
                while current != "A":
                    condition.wait()
                print("A")
                current = "B"
                condition.notify_all()


class ThreadB(Thread):
    def run(self):
        global current
        for _ in range(10):
            with condition:
                while current != "B":
                    condition.wait()
                print("B")
                current = "C"
                condition.notify_all()


class ThreadC(Thread):
    def run(self):
        global current
        for _ in range(10):
            with condition:
                while current != "C":
                    condition.wait()
                print("C")
                current = "A"
                condition.notify_all()


a = ThreadA()
b = ThreadB()
c = ThreadC()

a.start()
b.start()
c.start()

a.join()
b.join()
c.join()
```

“B” current = 'B' B C “C”

2.10

set() true clear() false wait() true

2.10.1 ...

-

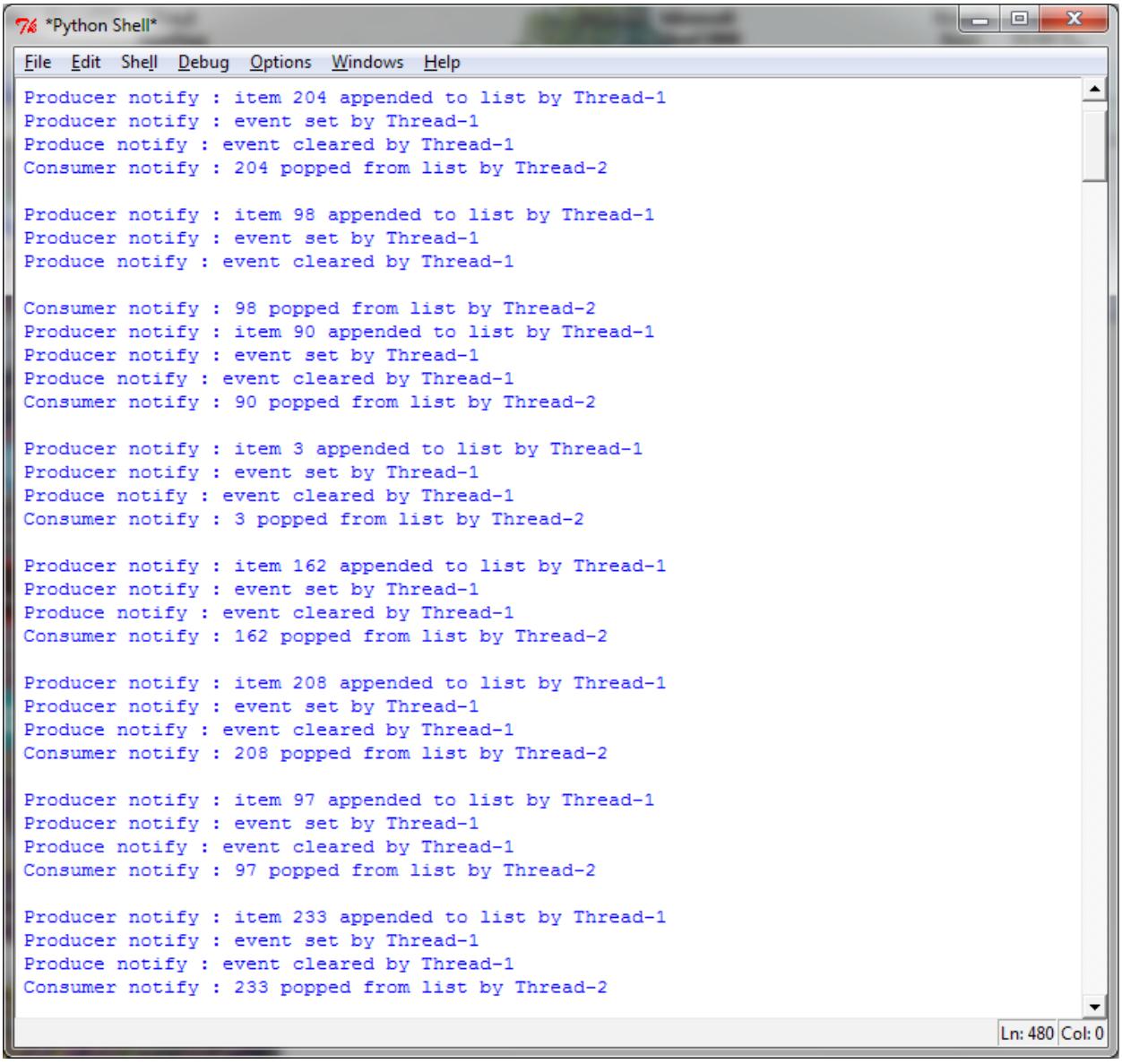
```
# -*- coding: utf-8 -*-

import time
from threading import Thread, Event
import random
items = []
event = Event()

class consumer(Thread):
    def __init__(self, items, event):
        Thread.__init__(self)
        self.items = items
        self.event = event

    def run(self):
        while True:
            time.sleep(2)
            self.event.wait()
            item = self.items.pop()
            print('Consumer notify : %d popped from list by %s' % (item, self.name))

class producer(Thread):
    def __init__(self, items, event):
        Thread.__init__(self)
        self.items = items
        self.event = event

    def run(self):
        global item
        for i in range(100):
            time.sleep(2)
            item = random.randint(0, 256)
            self.items.append(item)
            print('Producer notify : item N° %d appended to list by %s' % (item, self.
↪name))
            print('Producer notify : event set by %s' % self.name)
            self.event.set()
            print('Produce notify : event cleared by %s '% self.name)
            self.event.clear()

if __name__ == '__main__':
```

( )

( )

```
    t1 = producer(items, event)
    t2 = consumer(items, event)
    t1.start()
    t2.start()
    t1.join()
    t2.join()
```

t1 list event wait() list



2.10.2

producer item list Event list

```
class consumer(Thread):
    def __init__(self, items, event):
        Thread.__init__(self)
        self.items = items
        self.event = event
```

run item producer item list

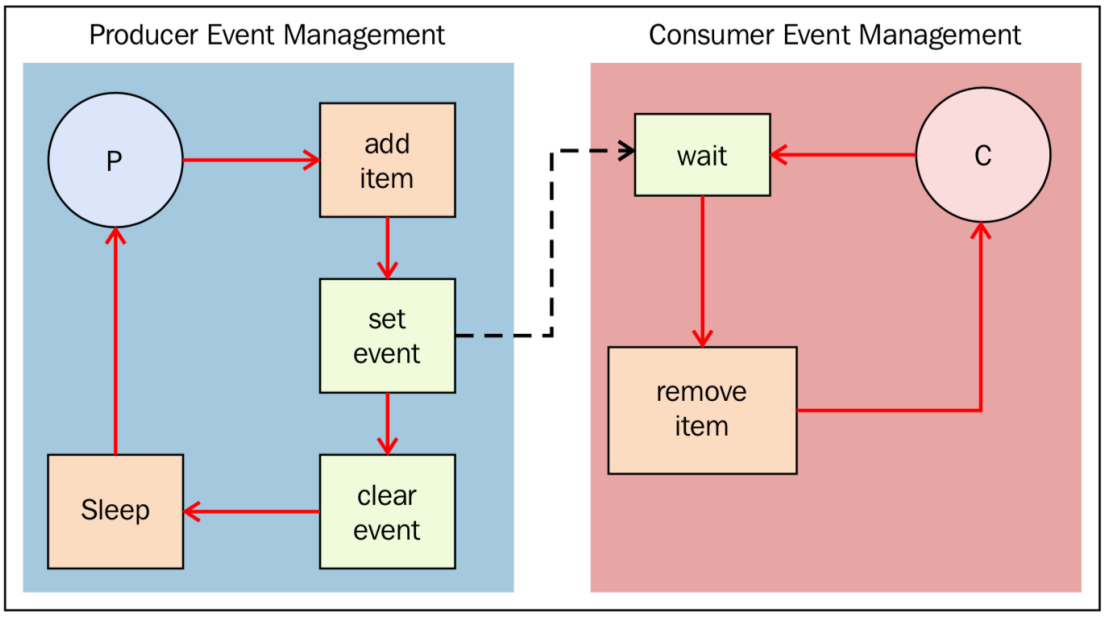
```
self.event.set()
```

```
self.event.clear()
```

consumer item list Event() item

```
def run(self):
    while True:
        time.sleep(2)
        self.event.wait()
        item = self.items.pop()
        print('Consumer notify : %d popped from list by %s' % (item, self.name))
```

producer consumer



Thread synchronization with event objects

2.11 with

Python 2.5 with with with with
“ ” threading acquire() release()

with

- Lock
- RLock
- Condition
- Semaphore

2.11.1

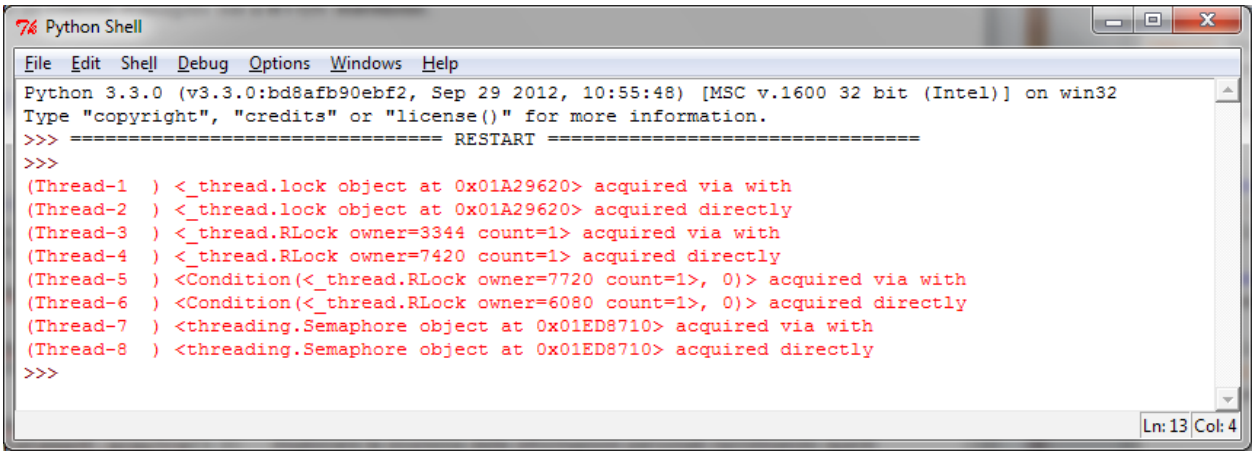
with

2.11.2 ...

with with

```
import threading
import logging
logging.basicConfig(level=logging.DEBUG, format='(%(threadName)-10s) %(message)s',)

def threading_with(statement):
    with statement:
        logging.debug('%s acquired via with' % statement)

def threading_not_with(statement):
    statement.acquire()
    try:
        logging.debug('%s acquired directly' % statement )
    finally:
        statement.release()

if __name__ == '__main__':
    # let's create a test battery
    lock = threading.Lock()
    rlock = threading.RLock()
    condition = threading.Condition()
    mutex = threading.Semaphore(1)
    threading_synchronization_list = [lock, rlock, condition, mutex]
    # in the for cycle we call the threading_with e threading_no_with function
    for statement in threading_synchronization_list :
       t1 = threading.Thread(target=threading_with, args=(statement,))
       t2 = threading.Thread(target=threading_not_with, args=(statement,))
       t1.start()
       t2.start()
       t1.join()
       t2.join()
```

with



2.11.3

list threading_synchronization_list

```
lock = threading.Lock()
rlock = threading.RLock()
condition = threading.Condition()
mutex = threading.Semaphore(1)
threading_synchronization_list = [lock, rlock, condition, mutex]
```

for

```
for statement in threading_synchronization_list :
   t1 = threading.Thread(target=threading_with, args=(statement,))
   t2 = threading.Thread(target=threading_not_with, args=(statement,))
```

threading_with with

```
def threading_with(statement):
    with statement:
        logging.debug('%s acquired via with' % statement)
```

2.11.4

Python logging

```
logging.basicConfig(level=logging.DEBUG, format='(%(threadName)-10s) %(message)s',)
```

% (threadName) logging

Python with https://www.kawabangga.com/posts/2010

2.12 queue

Python threading queue

Queue

- `put()`: queue item
- `get()`: queue item item
- `task_done()`: item
- `join()`: item

2.12.1 ...

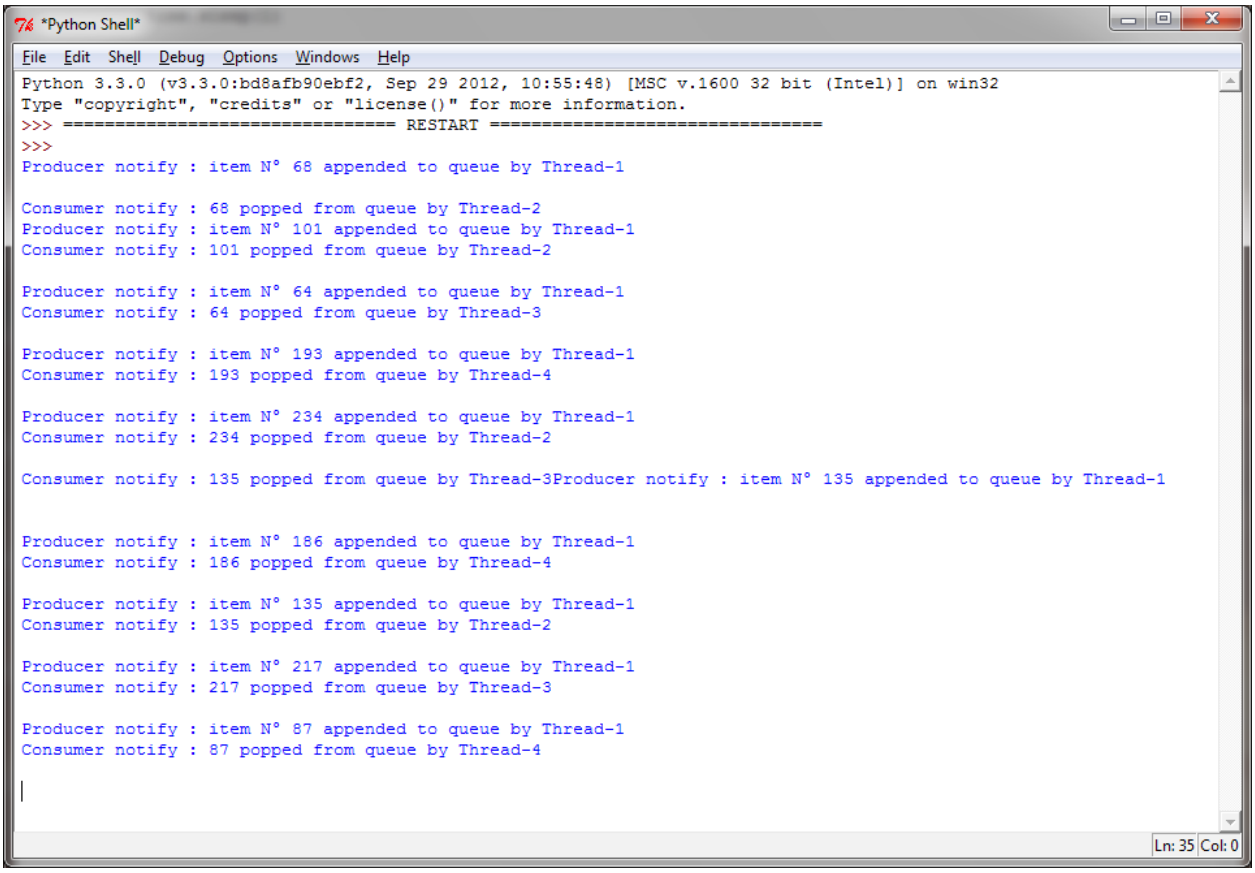
threading queue

```
from threading import Thread, Event
from queue import Queue
import time
import random
class producer(Thread):
    def __init__(self, queue):
        Thread.__init__(self)
        self.queue = queue

    def run(self) :
        for i in range(10):
            item = random.randint(0, 256)
            self.queue.put(item)
            print('Producer notify: item N° %d appended to queue by %s' % (item, self.
↪name))
            time.sleep(1)

class consumer(Thread):
    def __init__(self, queue):
        Thread.__init__(self)
        self.queue = queue

    def run(self):
        while True:
            item = self.queue.get()
            print('Consumer notify : %d popped from queue by %s' % (item, self.name))
            self.queue.task_done()

if __name__ == '__main__':
    queue = Queue()
    t1 = producer(queue)
    t2 = consumer(queue)
    t3 = consumer(queue)
    t4 = consumer(queue)
    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t1.join()
    t2.join()
```

( )

( )

```
    t3.join()
    t4.join()
```



2.12.2

list

```
class producer(Thread):
    def __init__(self, queue):
        Thread.__init__(self)
        self.queue = queue
```

producer for

```
def run(self) :
    for i in range(10):
        item = random.randint(0, 256)
        self.queue.put(item)
        print('Producer notify: item N° %d appended to queue by %s' % (item, self.name))
        time.sleep(1)
```

Queue.put(item [,block[, timeout]]) queue Queue

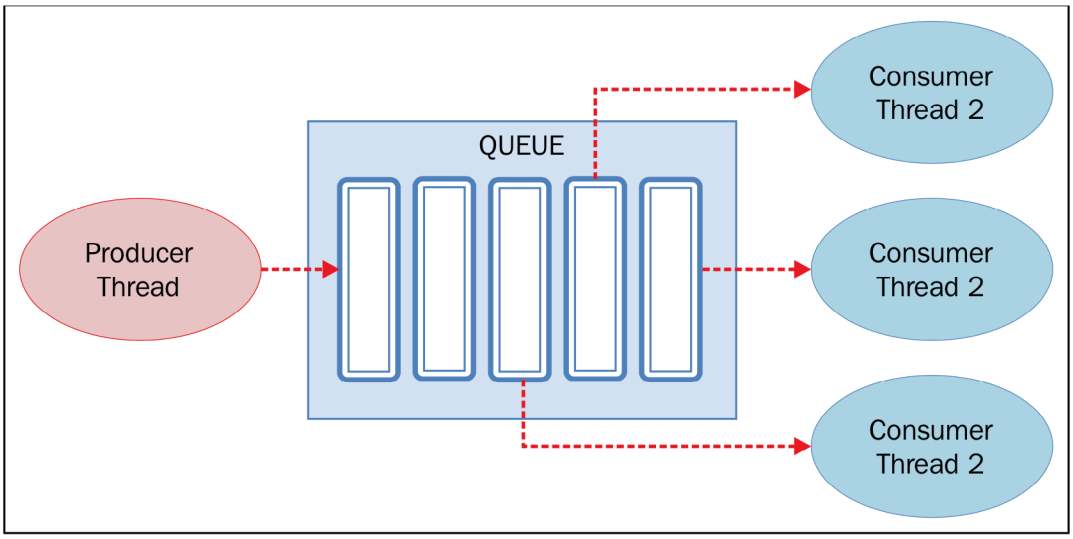
- block True timeout None timeout
- block False timeout put() wait()

task_done()

Queue.get([block[, timeout]]) queue

t t1, t2, t3

```
if __name__ == '__main__':
    queue = Queue()
    t1 = producer(queue)
    t2 = consumer(queue)
    t3 = consumer(queue)
    t4 = consumer(queue)
    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t1.join()
    t2.join()
    t3.join()
    t4.join()
```



Thread synchronization with the queue module

2.13

GIL GIL CPython GIL GIL GIL Python G
— Python GIL CPython GIL GIL

2.13.1 ...

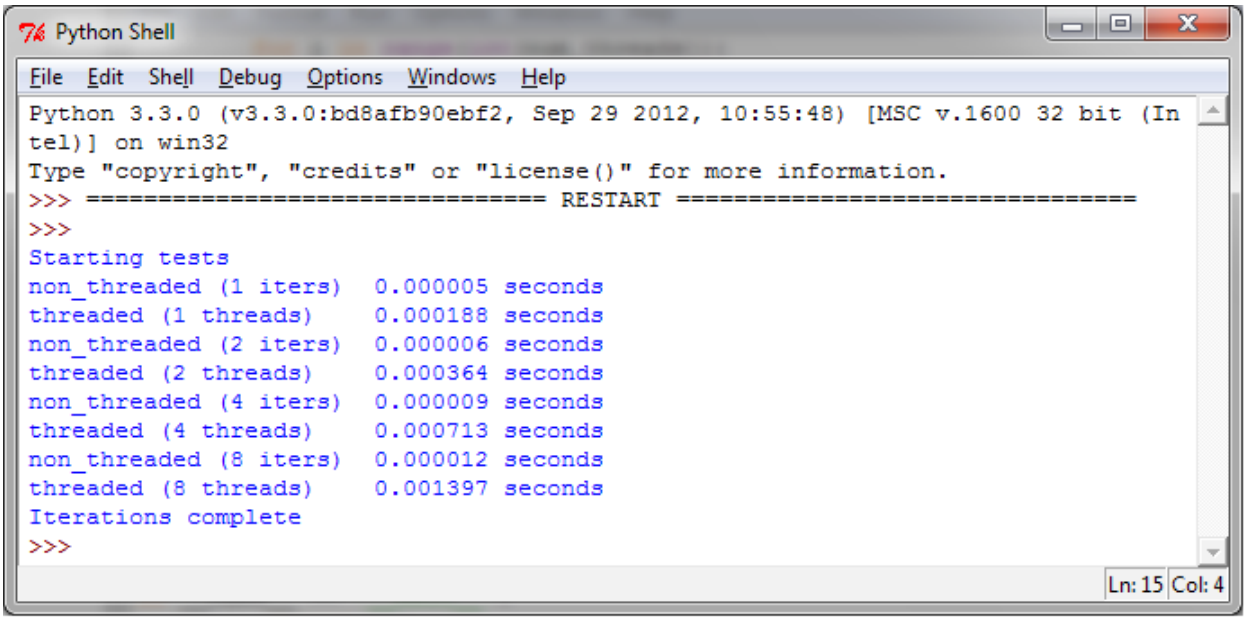
100 for non_threaded threaded
1 2 3 4 8 Python timer

```
from threading import Thread

class threads_object(Thread):
    def run(self):
        function_to_run()

class nothreads_object(object):
    def run(self):
        function_to_run()

def non_threaded(num_iter):
    funcs = []
    for i in range(int(num_iter)):
        funcs.append(nothreads_object())
    for i in funcs:
        i.run()

def threaded(num_threads):
    funcs = []
    for i in range(int(num_threads)):
        funcs.append(threads_object())
    for i in funcs:
        i.start()
    for i in funcs:
        i.join()

def function_to_run():
    pass

def show_results(func_name, results):
    print("%-23s %4.6f seconds" % (func_name, results))

if __name__ == "__main__":
    import sys
    from timeit import Timer
    repeat = 100
    number = 1
    num_threads = [1, 2, 4, 8]
    print('Starting tests')
    for i in num_threads:
        t = Timer("non_threaded(%s)" % i, "from __main__ import non_threaded")
        best_result = min(t.repeat(repeat=repeat, number=number))
        show_results("non_threaded (%s iters)" % i, best_result)
        t = Timer("threaded(%s)" % i, "from __main__ import threaded")
        best_result = min(t.repeat(repeat=repeat, number=number))
        show_results("threaded (%s threads)" % i, best_result)
        print('Iterations complete')
```

2.13.2

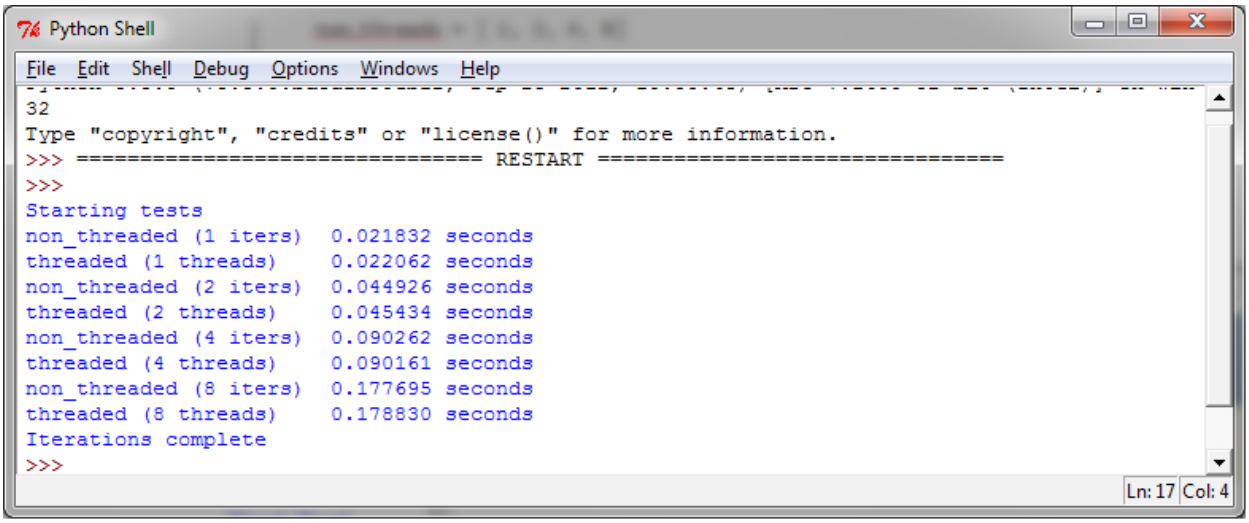
(three 3 function `function_to_run()`
Core 2 Duo CPU – 2.33Ghz

```
def function_to_run():
    pass
```



4 0.0007143 8 0.001397

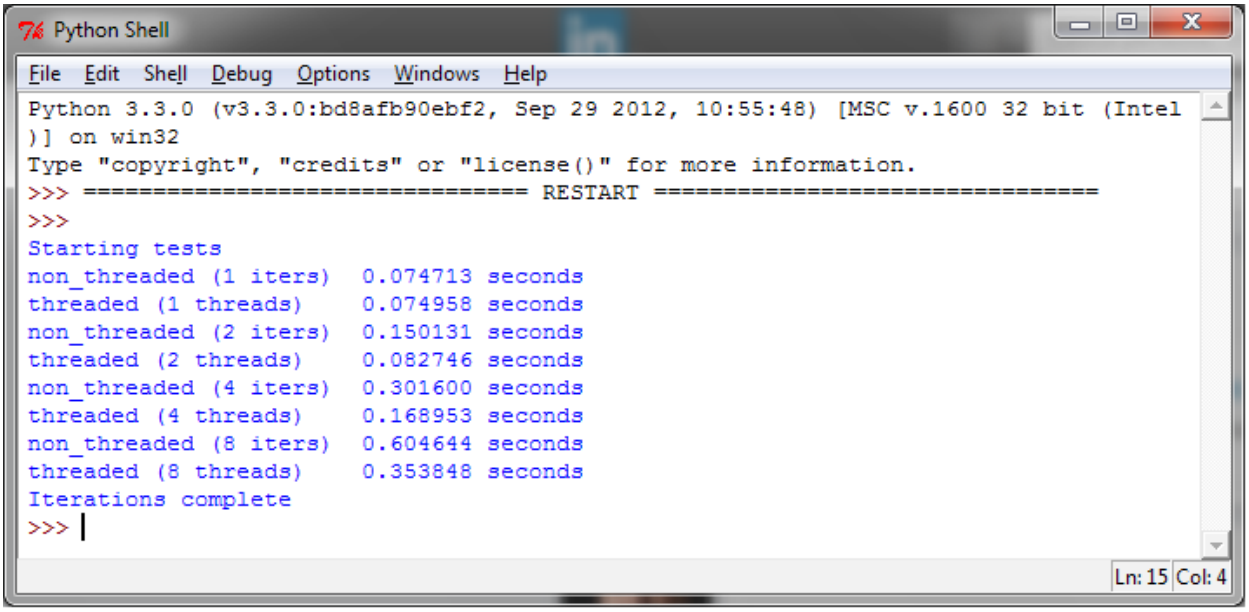
```
def function_to_run():
    a, b = 0, 1
    for i in range(10000):
        a, b = b, a + b
```



GIL GIL

1kb 1000

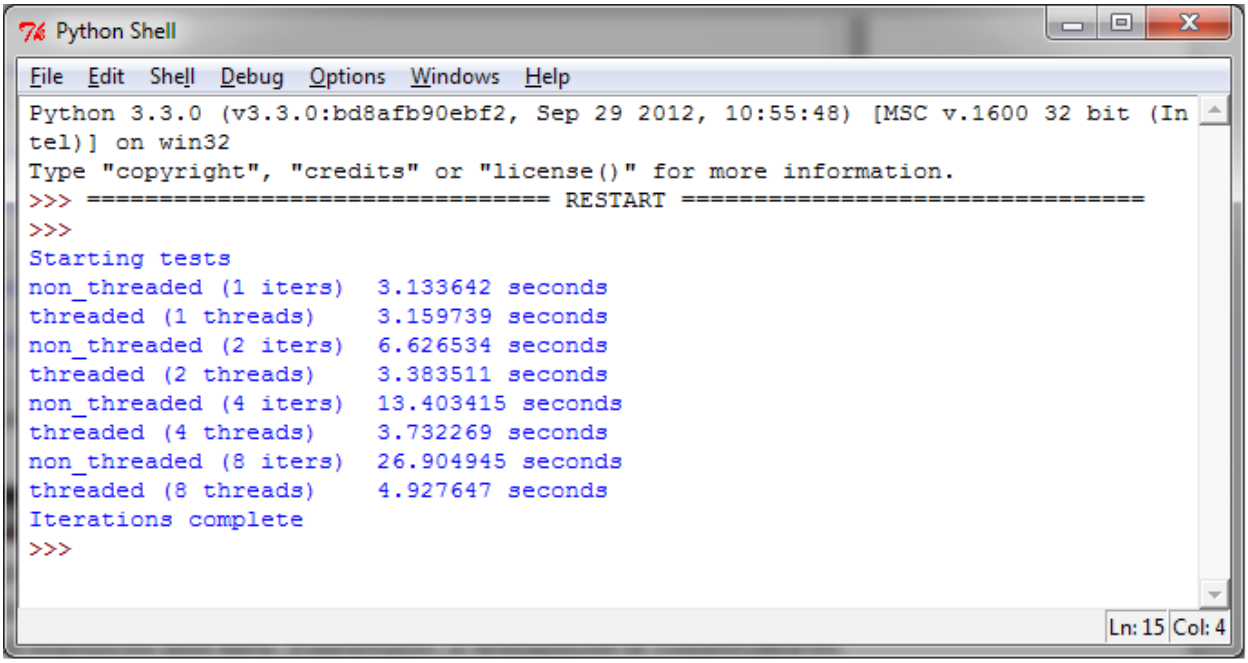
```
def function_to_run():
    fh=open("C:\\CookBookFileExamples\\test.dat","rb")
    size = 1024
    for i in range(1000):
        fh.read(size)
```



urllib.request Python URL socket C

https://www.packpub.com 1k

```
def function_to_run():
    import urllib.request
    for i in range(10):
        with urllib.request.urlopen("https://www.packtpub.com/")as f:
            f.read(1024)
```



I/O GIL I/O GIL

2.13.3

worker GIL Python CP

CHAPTER 3

mpi4py Mac OS

3.1

Python multiprocessing mpi4py

multiprocessing Python

mpi4py shared nothing

send() receive

Python multiprocessing main https://docs.python.org/3.3/library/multiprocessing.html

__main__ IDLE IDLE

```
python multiprocessing example.py
```

multiprocessing_example.py Python3.3 Python2.7

Python

```
>>> from multiprocessing import Pool
>>> p = Pool(5)
>>> def f(x):
...     return x*x
...
>>> p.map(f, [1,2,3])
Process PoolWorker-1:
Process PoolWorker-2:
Process PoolWorker-3:
Traceback (most recent call last):
```

( )

( )

```
AttributeError: 'module' object has no attribute 'f'
AttributeError: 'module' object has no attribute 'f'
AttributeError: 'module' object has no attribute 'f'
```

Python

```
In [1]: from multiprocessing import Pool

In [2]: p = Pool(5)

In [4]: import func

In [5]: p.map(func.f, [1,2,3])
Out[5]: [1, 4, 9]
```

3.2

" " spawn Python multiprocessing

1.
2. start()
3. join()

3.2.1 ...

5 foo(i) i id

```
# -*- coding: utf-8 -*-

import multiprocessing

def foo(i):
    print ('called function in process: %s' %i)
    return

if __name__ == '__main__':
    Process_jobs = []
    for i in range(5):
        p = multiprocessing.Process(target=foo, args=(i,))
        Process_jobs.append(p)
        p.start()
        p.join()
```

spawn_a_process.py

```
python spawn_a_process.py
```

```
$ python process_2.py
called function in process: 0
called function in process: 1
called function in process: 2
called function in process: 3
called function in process: 4
```

3.2.2

multiprocessing

```
import multiprocessing
```

```
p = multiprocessing.Process(target=foo, args=(i,))
```

start()

```
p.start()
```

foo() join()

join() idle

3.2.3

__main__

```
import multiprocessing
import target_function
if __name__ == '__main__':
    Process_jobs = []
    for i in range(5):
        p = multiprocessing.Process(target=target_function.function,args=(i,))
        Process_jobs.append(p)
        p.start()
        p.join()
```

target_function.py

```
def function(i):
    print('called function in process: %s' %i)
    return
```

3.3

debug

3.3.1 ...

foo()

```
#
import multiprocessing
import time

def foo():
    name = multiprocessing.current_process().name
    print("Starting %s \n" % name)
    time.sleep(3)
    print("Exiting %s \n" % name)

if __name__ == '__main__':
    process_with_name = multiprocessing.Process(name='foo_process', target=foo)
    process_with_name.daemon = True  #
    process_with_default_name = multiprocessing.Process(target=foo)
    process_with_name.start()
    process_with_default_name.start()
```

:

```
python naming_process.py
```

```
$ python naming_process.py
Starting foo_process
Starting Process-2
Exiting foo_process
Exiting Process-2
```

3.3.2

name

```
process_with_name = multiprocessing.Process(name='foo_process', target=foo)
```

foo_function

```
name = multiprocessing.current_process().name
```

3.4

Python multiprocessing

3.4.1 ...

```
import multiprocessing
import time

def foo():
    name = multiprocessing.current_process().name
    print("Starting %s " % name)
    time.sleep(3)
    print("Exiting %s " % name)

if __name__ == '__main__':
    background_process = multiprocessing.Process(name='background_process', target=foo)
    background_process.daemon = True
    NO_background_process = multiprocessing.Process(name='NO_background_process',␣
↪target=foo)
    NO_background_process.daemon = False
    background_process.start()
    NO_background_process.start()
```

```
python background_process.py
```

```
$ python background_process.py
Starting NO_background_process
Exiting NO_background_process
```

3.4.2

daemon True

```
background_process.daemon = True
```

3.4.3

Unix daemons or services

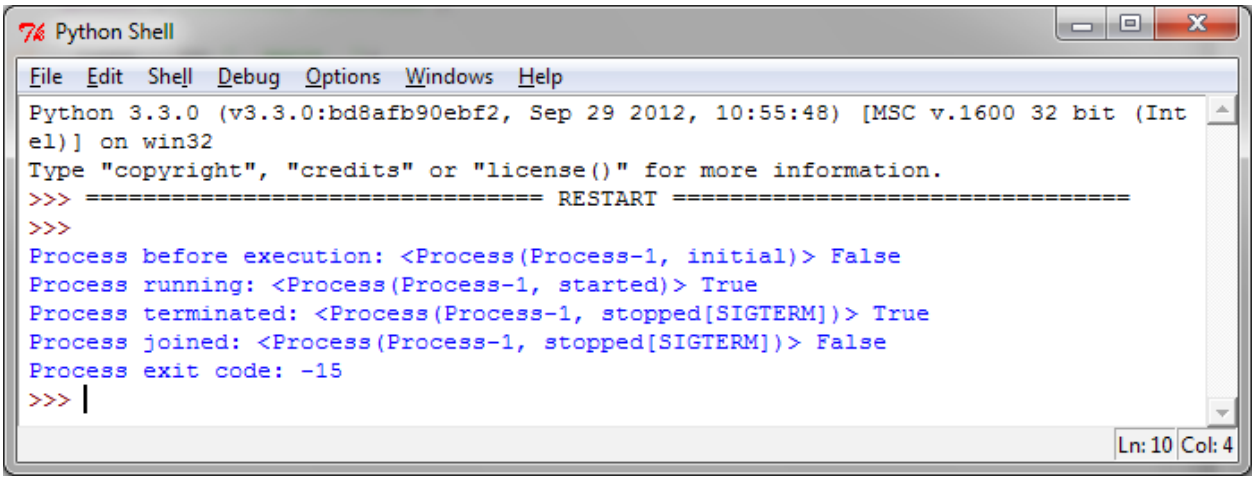
3.5

terminate() is_alive()

3.5.1 ...

foo() terminate()

```
#
import multiprocessing
import time

def foo():
        print('Starting function')
        time.sleep(0.1)
        print('Finished function')

if __name__ == '__main__':
        p = multiprocessing.Process(target=foo)
        print('Process before execution:', p, p.is_alive())
        p.start()
        print('Process running:', p, p.is_alive())
        p.terminate()
        print('Process terminated:', p, p.is_alive())
        p.join()
        print('Process joined:', p, p.is_alive())
        print('Process exit code:', p.exitcode)
```



3.5.2

is_alive() terminate()

ExitCode status code ExitCode

- == 0:
- > 0:
- < 0: -1 * ExitCode

ExitCode -15 15

3.6

- Process
- __init__(self [,args])
- run(self, [.args]) Process

Porcess start() run()

3.6.1 ...

```
# -*- coding: utf-8 -*-
#
import multiprocessing

class MyProcess(multiprocessing.Process):
        def run(self):
                print ('called run method in process: %s' % self.name)
                return

if __name__ == '__main__':
        jobs = []
        for i in range(5):
                p = MyProcess()
                jobs.append(p)
                p.start()
                p.join()
```

```
python subclass_process.py
```

```
$ python subclass.py
called run method in process: MyProcess-1
called run method in process: MyProcess-2
called run method in process: MyProcess-3
called run method in process: MyProcess-4
called run method in process: MyProcess-5
```

3.6.2

Process run()

```
class MyProcess(multiprocessing.Process):
    def run(self):
        print ('called run method in process: %s' % self.name)
        return
```

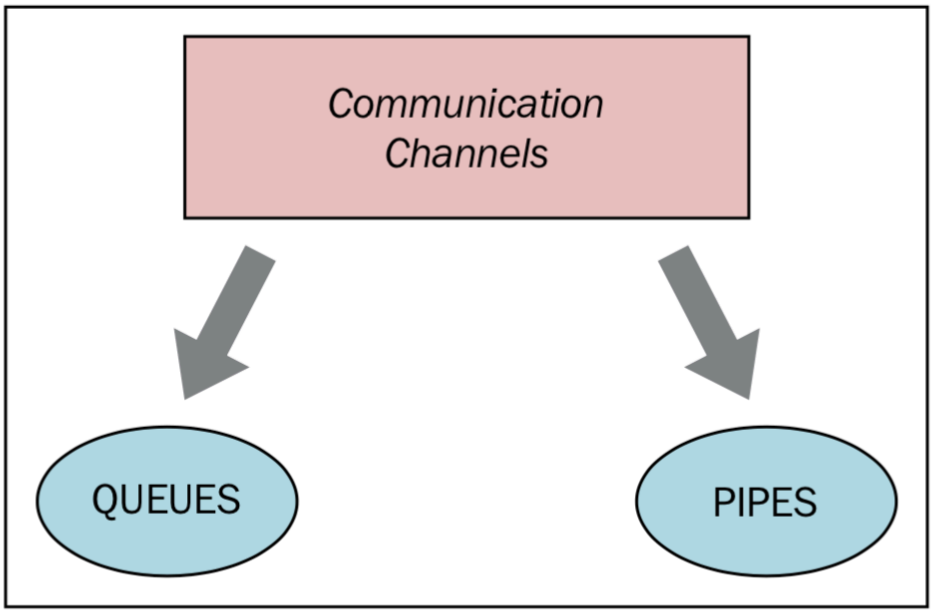
MyProcess() start()

```
p = MyProcess()
p.start()
```

join()

3.7

Multiprocessing Communication Channel (queue) pipe



Communication channels in the multiprocessing module

3.7.1

Queue Python pickable

3.7.2 ...

- Producer item Consumer

```
import multiprocessing
import random
import time

class Producer(multiprocessing.Process):
```

( )

( )

```
    def __init__(self, queue):
        multiprocessing.Process.__init__(self)
        self.queue = queue

    def run(self):
        for i in range(10):
            item = random.randint(0, 256)
            self.queue.put(item)
            print("Process Producer : item %d appended to queue %s" % (item, self.name))
            time.sleep(1)
            print("The size of queue is %s" % self.queue.qsize())

class Consumer(multiprocessing.Process):
    def __init__(self, queue):
        multiprocessing.Process.__init__(self)
        self.queue = queue

    def run(self):
        while True:
            if self.queue.empty():
                print("the queue is empty")
                break
            else:
                time.sleep(2)
                item = self.queue.get()
                print('Process Consumer : item %d popped from by %s \n' % (item, self.
↪name))
                time.sleep(1)

if __name__ == '__main__':
    queue = multiprocessing.Queue()
    process_producer = Producer(queue)
    process_consumer = Consumer(queue)
    process_producer.start()
    process_consumer.start()
    process_producer.join()
    process_consumer.join()
```

macOS High Sierra NotImplementedError self._sem._semlock._get_value()

```
C:\Python CookBook\Chapter 3 - Process Based Parallelism\Example Codes
Chapter 3>python using_queue.py
Process Producer : item 69 appended to queue producer-1
The size of queue is 1
Process Producer : item 168 appended to queue producer-1
The size of queue is 2
Process Consumer : item 69 popped from by consumer-2
Process Producer : item 235 appended to queue producer-1
The size of queue is 2
Process Producer : item 152 appended to queue producer-1
The size of queue is 3
Process Producer : item 213 appended to queue producer-1
```

( )

( )

```
Process Consumer : item 168 popped from by consumer-2
The size of queue is 3
Process Producer : item 35 appended to queue producer-1
The size of queue is 4
Process Producer : item 218 appended to queue producer-1
The size of queue is 5
Process Producer : item 175 appended to queue producer-1
Process Consumer : item 235 popped from by consumer-2
The size of queue is 5
Process Producer : item 140 appended to queue producer-1
The size of queue is 6
Process Producer : item 241 appended to queue producer-1
The size of queue is 7
Process Consumer : item 152 popped from by consumer-2
Process Consumer : item 213 popped from by consumer-2
Process Consumer : item 35 popped from by consumer-2
Process Consumer : item 218 popped from by consumer-2
Process Consumer : item 175 popped from by consumer-2
Process Consumer : item 140 popped from by consumer-2
Process Consumer : item 241 popped from by consumer-2
the queue is empty
```

3.7.3 ...

multiprocessing Queue

```
if __name__ == '__main__':
    queue = multiprocessing.Queue()
```

Queue

```
process_producer = Producer(queue)
process_consumer = Consumer(queue)
```

put() 10 item

```
for i in range(10):
    item = random.randint(0, 256)
    self.queue.put(item)
```

get() item break while

```
def run(self):
    while True:
        if self.queue.empty():
            print("the queue is empty")
            break
        else:
            time.sleep(2)
            item = self.queue.get()
            print('Process Consumer : item %d popped from by %s \n' % (item, self.name))
            time.sleep(1)
```

3.7.4

JoinaleQueue

- task_done(): get() item
- join(): item

Microndgt task_done

join put task_done join

3.7.5

Communication Channel

-
- send/receive

3.7.6 ...

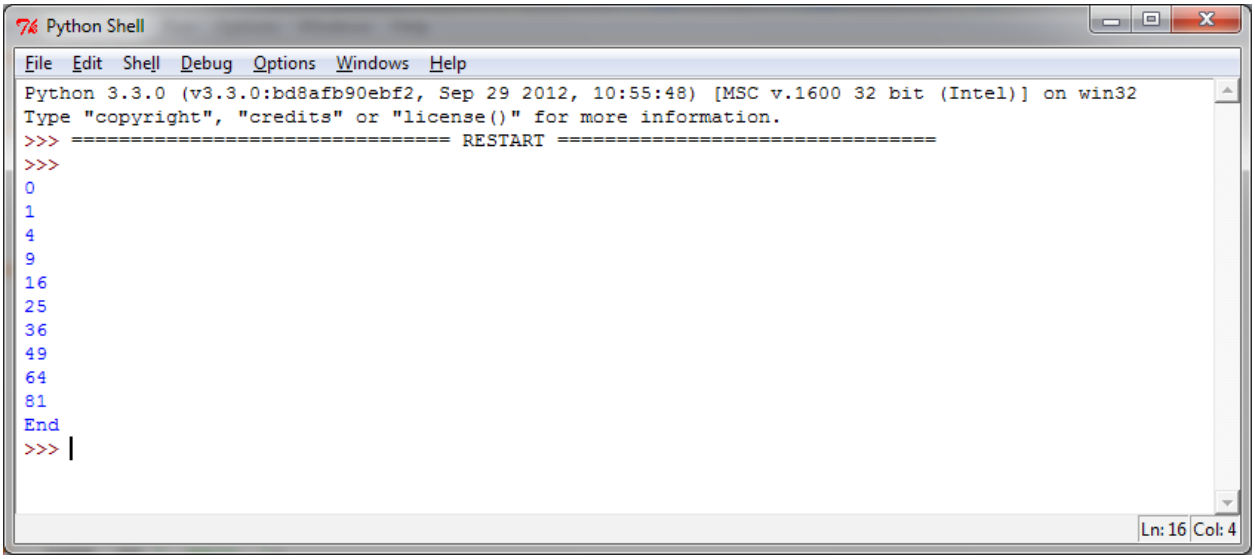
0 9

```
import multiprocessing

def create_items(pipe):
    output_pipe, _ = pipe
    for item in range(10):
        output_pipe.send(item)
    output_pipe.close()

def multiply_items(pipe_1, pipe_2):
    close, input_pipe = pipe_1
    close.close()
    output_pipe, _ = pipe_2
    try:
        while True:
            item = input_pipe.recv()
            output_pipe.send(item * item)
    except EOFError:
        output_pipe.close()

if __name__== '__main__':
    #
    pipe_1 = multiprocessing.Pipe(True)
    process_pipe_1 = multiprocessing.Process(target=create_items, args=(pipe_1,))
    process_pipe_1.start()
    #
    pipe_2 = multiprocessing.Pipe(True)
    process_pipe_2 = multiprocessing.Process(target=multiply_items, args=(pipe_1, pipe_2,
↪))
    process_pipe_2.start()
```

( )

( )

```
    pipe_1[0].close()
    pipe_2[0].close()
    try:
        while True:
            print(pipe_2[1].recv())
    except EOFError:
        print("End")
```



3.7.7

Pipe() out_pipe 0-9 create_items() :

```
def create_items(pipe):
    output_pipe, _ = pipe
    for item in range(10):
        output_pipe.send(item)
    output_pipe.close()
```

```
process_pipe_2 = multiprocessing.Process(target=multiply_items, args=(pipe_1, pipe_2,))
```

:

```
try:
    while True:
        print(pipe_2[1].recv())
except EOFError:
    print("End")
```

3.8

- **Lock**: locked unlocked Lock `acquire()` `release()`
- **Event**: `Event` `set()` `clear()`
- **Condition**: `wait()` `notify_all()`
- **Semaphore**:
- **Rlock**: `Threading`
- **Barrier**: Barrier

3.8.1 ...

`barrier()` 4 1 2 barrier 3 4

```
import multiprocessing
from multiprocessing import Barrier, Lock, Process
from time import time
from datetime import datetime

def test_with_barrier(synchronizer, serializer):
    name = multiprocessing.current_process().name
    synchronizer.wait()
    now = time()
    with serializer:
        print("process %s ----> %s" % (name, datetime.fromtimestamp(now)))

def test_without_barrier():
    name = multiprocessing.current_process().name
    now = time()
    print("process %s ----> %s" % (name, datetime.fromtimestamp(now)))

if __name__ == '__main__':
    synchronizer = Barrier(2)
    serializer = Lock()
    Process(name='p1 - test_with_barrier', target=test_with_barrier, args=(synchronizer,
↪serializer)).start()
    Process(name='p2 - test_with_barrier', target=test_with_barrier, args=(synchronizer,
↪serializer)).start()
    Process(name='p3 - test_without_barrier', target=test_without_barrier).start()
    Process(name='p4 - test_without_barrier', target=test_without_barrier).start()
```

1 2

```
$ python process_barrier.py
process p1 - test_with_barrier ----> 2015-05-09 11:11:33.291229
process p2 - test_with_barrier ----> 2015-05-09 11:11:33.291229
process p3 - test_without_barrier ----> 2015-05-09 11:11:33.310230
process p4 - test_without_barrier ----> 2015-05-09 11:11:33.333231
```

10 with_barrier 1 2 without_barrier 3 4

3.8.2

barrier barrier

```
if __name__ == '__main__':
    synchronizer = Barrier(2)
    serializer = Lock()
    Process(name='p1 - test_with_barrier', target=test_with_barrier, args=(synchronizer,
↪serializer)).start()
    Process(name='p2 - test_with_barrier', target=test_with_barrier, args=(synchronizer,
↪serializer)).start()
    Process(name='p3 - test_without_barrier', target=test_without_barrier).start()
    Process(name='p4 - test_without_barrier', target=test_without_barrier).start()
```

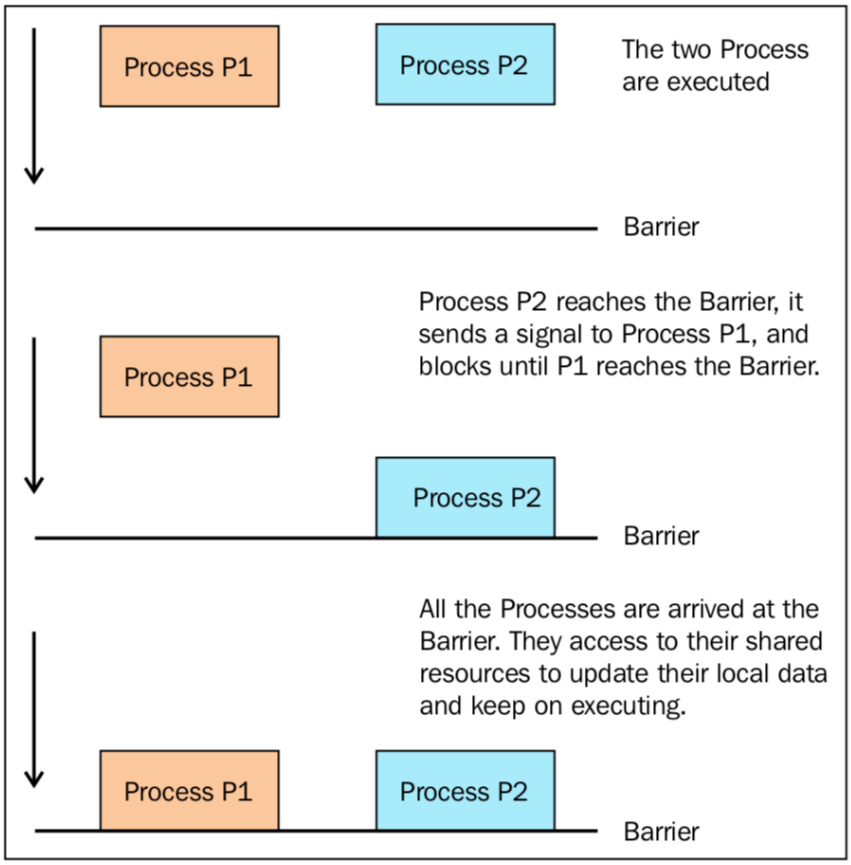
test_with_barrier barrier wait()

```
def test_with_barrier(synchronizer, serializer):
    name = multiprocessing.current_process().name
    synchronizer.wait()
```

wait()

```
now = time()
with serializer:
    print("process %s ----> %s" % (name, datetime.fromtimestamp(now)))
```

barrier



Process management with a barrier

3.9

Python (Manager) Python

-
-

3.9.1 ...

1. n taskWorkers worker index
2. worker stdout :

```
import multiprocessing

def worker(dictionary, key, item):
```

( )

( )

```
   dictionary[key] = item
   print("key = %d value = %d" % (key, item))

if __name__ == '__main__':
    mgr = multiprocessing.Manager()
    dictionary = mgr.dict()
    jobs = [multiprocessing.Process(target=worker, args=(dictionary, i, i*2)) for i␣
↪in range(10)]
    for j in jobs:
        j.start()
    for j in jobs:
        j.join()
    print('Results:', dictionary)
```

: print

```
$ python manager.py
key = 0 value = 0
key = 3 value = 6
key = 2 value = 4
key = 1 value = 2
key = 4 value = 8
key = 5 value = 10
key = 8 value = 16
key = 6 value = 12
key = 7 value = 14
key = 9 value = 18
Results: {0: 0, 3: 6, 2: 4, 1: 2, 4: 8, 5: 10, 8: 16, 6: 12, 7: 14, 9: 18}
```

3.9.2

manager

```
mgr = multiprocessing.Manager()
```

dictionary

```
dictionary = mgr.dict()
```

```
jobs = [multiprocessing.Process(target=worker, args=(dictionary, i, i*2)) for i in␣
↪range(10)]
for j in jobs:
    j.start()
```

taskWorker item

```
def worker(dictionary, key, item):
   dictionary[key] = item
```

```
for j in jobs:
    j.join()
print('Results:', dictionary)
```

3.10

Pool Pool

- apply():
- apply_async(): apply() result
- map(): map()
- map_async(): map() result callable result result

3.10.1 ...

4 map()

```
import multiprocessing

def function_square(data):
    result = data*data
    return result

if __name__ == '__main__':
    inputs = list(range(100))
    pool = multiprocessing.Pool(processes=4)
    pool_outputs = pool.map(function_square, inputs)
    pool.close()
    pool.join()
    print ('Pool    :', pool_outputs)
```

```
$ python poll.py
('Pool    :', [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256,␣
↪289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024, 1089,␣
↪1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209,␣
↪2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249, 3364, 3481, 3600, 3721,␣
↪3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625,␣
↪5776, 5929, 6084, 6241, 6400, 6561, 6724, 6889, 7056, 7225, 7396, 7569, 7744, 7921,␣
↪8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801])
```

3.10.2 ...

multiprocessing.Pool function_square 4

```
pool = multiprocessing.Pool(processes=4)
```

pool.map

```
pool_outputs = pool.map(function_square, inputs)
```

input 0 100 list

```
inputs = list(range(100))
```

pool_outputs

```
print ('Pool    :', pool_outputs)
```

pool.map() Python map() pool.map()

3.11 Python mpi4py

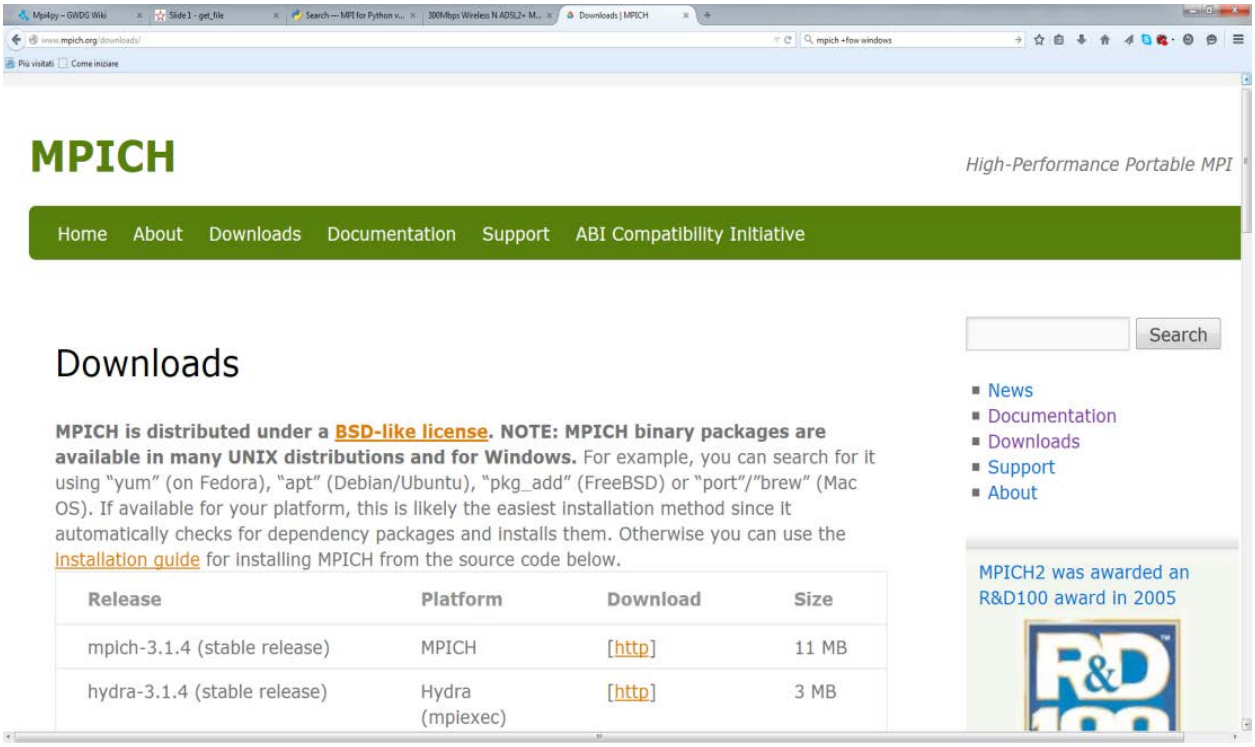
Python MPI mpi4py MPI-1/2 C++ MPI-2 MPI C Python MPI

-
-
-

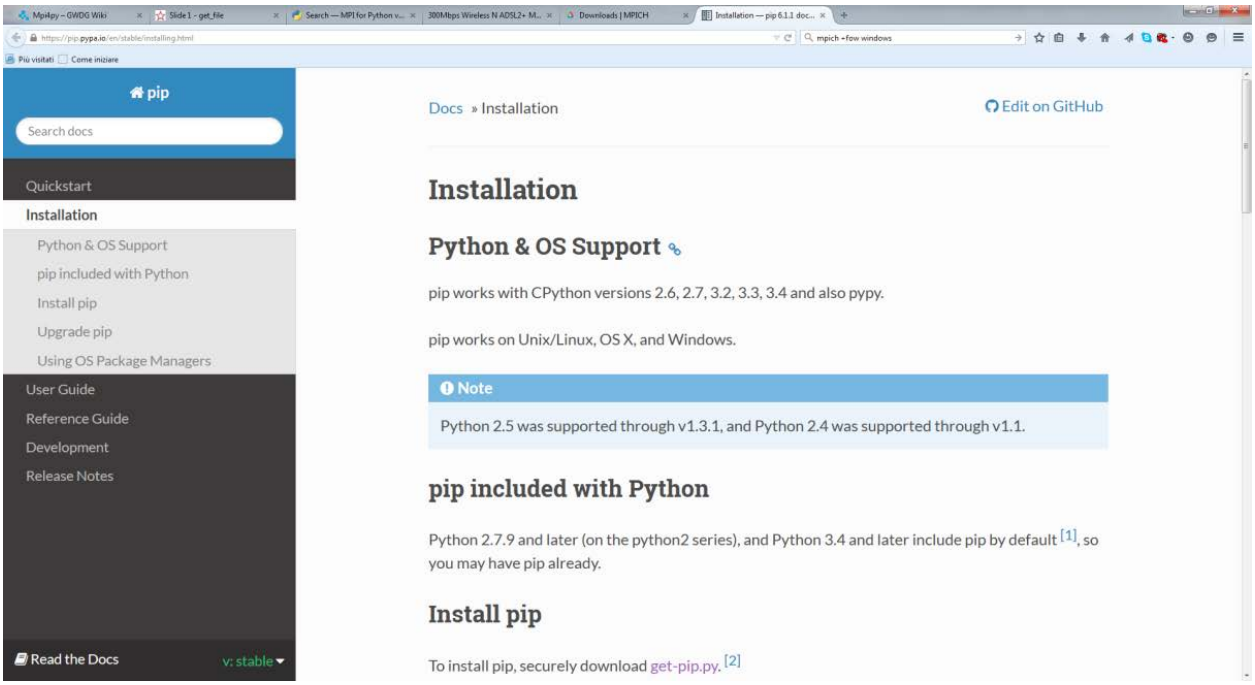
3.11.1

Windows mpi4py http://mpi4py.scipy.org/docs/usrman/install.html :

1. MPI (http://www.mpich.org/downloads/) mpich



2. “ ”
3. `msiexec /i mpich_installation_file.msi` MPICH2
4. “ ”
5. `wmpiconfig` windows
6. `C:\Program Files\MPICH2\bin`
7. `smpd- status` `smpd` `smpd running on $hostname$`
8. `$MPICHROOT\examples` `mpiexec -n 4 cpi` `cpi.exe`
9. https://pip.pypa.io/en/stable/installing.html Python `pip` Python `pip.exe`



10. `mpi4py`

C:> pip install mpi4py

3.11.2 ...

"Hello world" MPI

```
# hello.py
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
print("hello world from process ", rank)
```

```
C:> mpiexec  -n 5 python  helloWorld_MPI.py
```

```
('hello world from process ', 1)
('hello world from process ', 0)
('hello world from process ', 2)
('hello world from process ', 3)
('hello world from process ', 4)
```

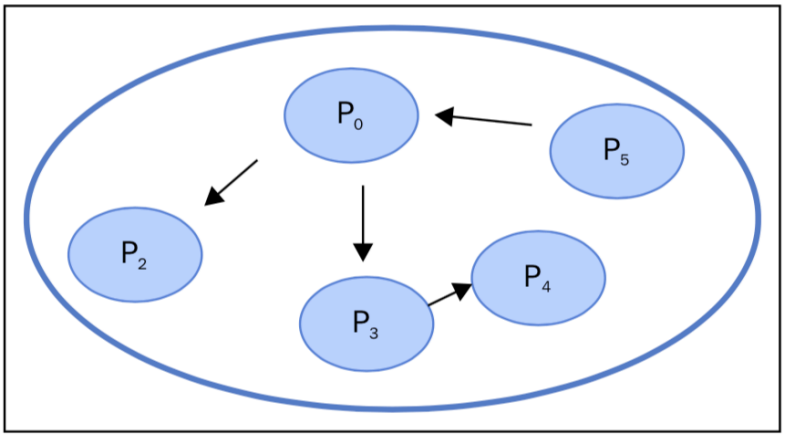
3.11.3

MPI rank p rank 0 p-1 MPI rank

```
rank = comm.Get_rank()
```

rank comm Communicator

```
comm = MPI.COMM_WORLD
```



An example of communication between processes in MPI.COMM_WORLD

3.11.4

MPI

3.12

MPI

Python mpi4py

- Comm.Send(data, process_destination):
- Comm.Recv(process_source):

Comm

```
comm = MKPI.COMM_WORLD
```

3.12.1 ...

comm.send comm.recv

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.rank
print("my rank is : " , rank)
```

( )

( )

```
if rank == 0:
    data = 10000000
    destination_process = 4
    comm.send(data,dest=destination_process)
    print("sending data % s " % data + "to process % d" % destination_process)

if rank == 1:
    destination_process = 8
    data = "hello"
    comm.send(data,dest=destination_process)
    print("sending data % s :" % data + "to process % d" % destination_process)

if rank == 4:
   data = comm.recv(source = 0)
   print("data received is = % s" % data)

if rank == 8:
   data1 = comm.recv(source = 1)
   print("data1 received is = % s" % data1)
```

```
$ mpiexec -n 9 python pointToPointCommunication.py
```

```
('my rank is : ', 5)
('my rank is : ', 1)
sending data hello :to process 8
('my rank is : ', 3)
('my rank is : ', 0)
sending data 10000000 to process 4
('my rank is : ', 2)
('my rank is : ', 7)
('my rank is : ', 4)
data received is = 10000000
('my rank is : ', 8)
data1 received is = hello
('my rank is : ', 6)
```

3.12.2

9 comm 9

```
comm = MPI.COMM_WORLD
```

rand

```
rank = comm.rand
```

rank 0 rank 4

```
if rank==0:
    data= 10000000
    destination_process = 4
    comm.send(data,dest=destination_process)
```

rank 4 rank comm.recv

```
...
if rank == 4:
    data = comm.recv(source=0)
```

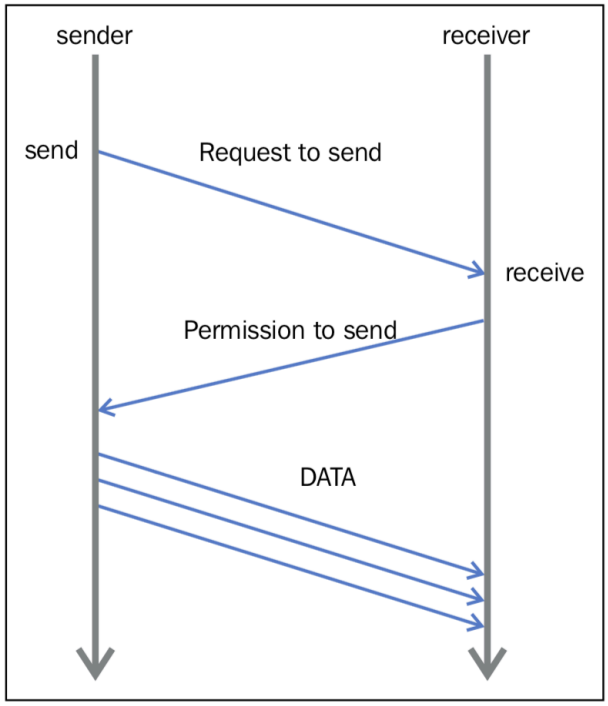
rank 1 rank 8 String

```
if rank==1:
       destination_process = 8
       data= "hello"
       comm.send(data,dest=destination_process)
```

rank

```
if rank==8:
       data1=comm.recv(source=1)
```

mpi4py :



The send/receive transmission protocol

/

3.12.3

comm.send() comm.recv() MPI

- buffer
-

buffer buffer / /

3.13

mpi4py

3.13.1 ...

Python rank 1 rank 5

```
from mpi4py import MPI
comm=MPI.COMM_WORLD
rank = comm.rank
print("my rank is : " , rank)

if rank==1:
    data_send= "a"
    destination_process = 5
    source_process = 5
    data_received=comm.recv(source=source_process)
    comm.send(data_send,dest=destination_process)
    print("sending data %s " %data_send + "to process %d" %destination_process)
    print("data received is = %s" %data_received)

if rank==5:
    data_send= "b"
    destination_process = 1
    source_process = 1
    data_received=comm.recv(source=source_process)
    comm.send(data_send,dest=destination_process)
    print("sending data %s :" %data_send + "to process %d" %destination_process)
    print("data received is = %s" %data_received)
```

3.13.2

```
$ mpiexec -n 9 python deadLockProblems.py
('my rank is : ', 8)
('my rank is : ', 3)
('my rank is : ', 2)
('my rank is : ', 7)
('my rank is : ', 0)
```

( )

( )

```
('my rank is : ', 4)
('my rank is : ', 6)
```

MPI comm.recv() comm.send() comm.send() MPI comm.recv()

```
if rank==1:
    data_send= "a"
    destination_process = 5
    source_process = 5
    comm.send(data_send,dest=destination_process)
    data_received=comm.recv(source=source_process)
if rank==5:
    data_send= "b"
    destination_process = 1
    source_process = 1
    data_received=comm.recv(source=source_process)
    comm.send(data_send,dest=destination_process)
```

buffer comm.send() buffer buffer buffer comm.send()

```
if rank==1:
    data_send= "a"
    destination_process = 5
    source_process = 5
    comm.send(data_send,dest=destination_process)
    data_received=comm.recv(source=source_process)
if rank==5:
    data_send= "b"
    destination_process = 1
    source_process = 1
    comm.send(data_send,dest=destination_process)
    data_received=comm.recv(source=source_process)
```

```
$ mpiexec -n 9 python deadLockProblems.py
('my rank is : ', 7)
('my rank is : ', 0)
('my rank is : ', 8)
('my rank is : ', 1)
sending data a to process 5
data received is = b
('my rank is : ', 5)
sending data b :to process 1
data received is = a
('my rank is : ', 2)
('my rank is : ', 3)
('my rank is : ', 4)
('my rank is : ', 6)
```

3.13.3

Sendrecv

```
Sendrecv(self, sendbuf, int dest=0, int sendtag=0, recvbuf=None, int source=0, int␣
↪recvtag=0, Status status=None)
```

comm.send() MPI comm.recv() MPI

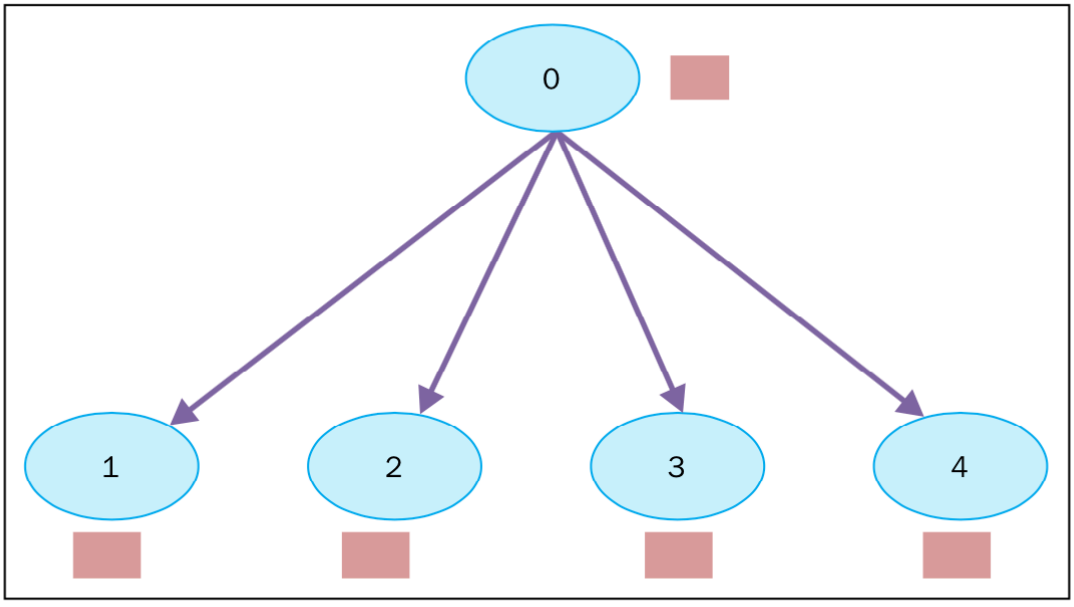
```
if rank==1:
    data_send= "a"
    destination_process = 5
    source_process = 5
    data_received=comm.sendrecv(data_send,dest=destination_process,source =source_
↪process)
if rank==5:
    data_send= "b"
    destination_process = 1
    source_process = 1
    data_received=comm.sendrecv(data_send,dest=destination_process, source=source_
↪process)
```

3.14 broadcast

)

0 1 2 3 4 5 6

MPI



Broadcasting data from process 0 to processes 1, 2, 3, and 4

2 —— mpi4py

```
buf = comm.bcast(data_to_share, rank_of_root_process)
```

root comm root comm

3.14.1 ...

root rank 0 variable_to_share

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
if rank == 0:
    variable_to_share = 100
else:
    variable_to_share = None
variable_to_share = comm.bcast(variable_to_share, root=0)
print("process = %d" %rank + " variable shared  = %d " %variable_to_share)
```

10

```
C:\>mpiexec -n 10 python broadcast.py
process = 0 variable shared  = 100
process = 8 variable shared  = 100
process = 2 variable shared  = 100
process = 3 variable shared  = 100
process = 4 variable shared  = 100
process = 5 variable shared  = 100
process = 9 variable shared  = 100
process = 6 variable shared  = 100
process = 1 variable shared  = 100
process = 7 variable shared  = 100
```

3.14.2

rank 0 root variable_to_share 100. :

```
if rank == 0:
    variable_to_share = 100
```

```
variable_to_share = comm.bcast(variable_to_share, root=0)
```

10 variable_to_share print

```
print("process = %d" %rank + " variable shared  = %d " %variable_to_share)
```

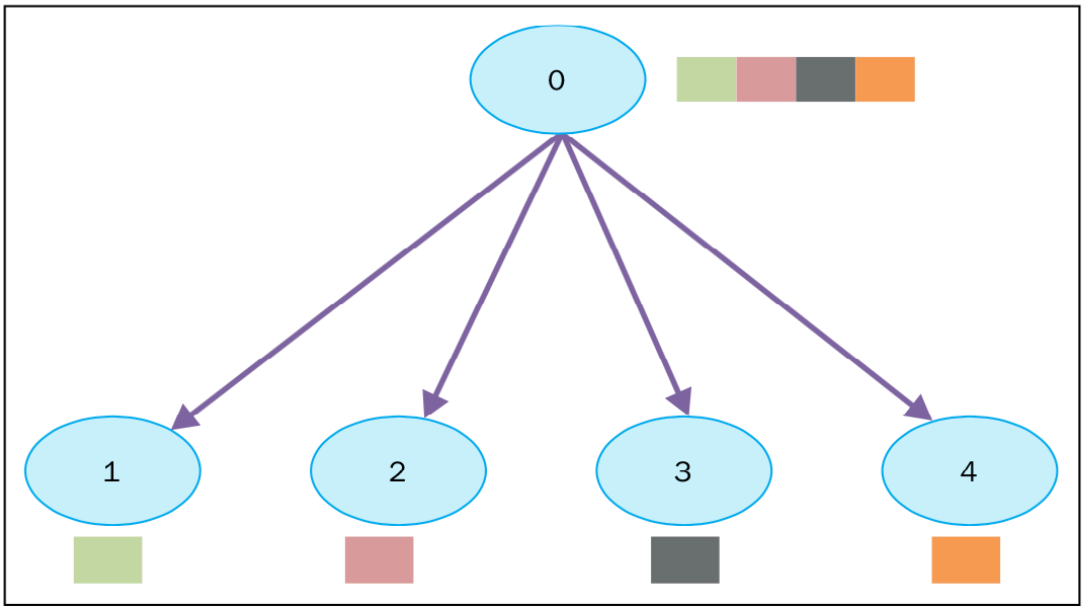
3.14.3

mpi4py

1.
2.
   -
   -
   -
3.

3.15 scatter

scatter comm.bcast comm.scatter scatter



Scattering data from process 0 to processes 1, 2, 3, 4

comm.scatter array rank 0 1 mpi4py

```
recvbuf  = comm.scatter(sendbuf, rank_of_root_process)
```

3.15.1 ...

scatter

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
if rank == 0:
    array_to_share = [1, 2, 3, 4 ,5 ,6 ,7, 8 ,9 ,10]
```

( )

( )

```
else:
    array_to_share = None
recvbuf = comm.scatter(array_to_share, root=0)
print("process = %d" %rank + " recvbuf = %d " %recvbuf)
```

```
C:\>mpiexec -n 10 python scatter.py
process = 0 variable shared  = 1
process = 4 variable shared  = 5
process = 6 variable shared  = 7
process = 2 variable shared  = 3
process = 5 variable shared  = 6
process = 3 variable shared  = 4
process = 7 variable shared  = 8
process = 1 variable shared  = 2
process = 8 variable shared  = 9
process = 9 variable shared  = 10
```

3.15.2

rank 0 array_to_share

```
array_to_share = [1, 2, 3, 4 ,5 ,6 ,7, 8 ,9 ,10]
```

recvbuf i comm.scatter i

```
recvbuf = comm.scatter(array_to_share, root=0)
```

comm.scatter

```
    C:\> mpiexec -n 3 python scatter.py
    Traceback (most recent call last):
       File "scatter.py", line 13, in <module>
         recvbuf = comm.scatter(array_to_share, root=0)
       File "Comm.pyx", line 874, in mpi4py.MPI.Comm.scatter␣
↪(c:\users\utente\appdata\local\temp\pip-build-h14iaj\mpi4py\src\mpi4py.MPI.c:73400)
       File "pickled.pxi", line 658, in mpi4py.MPI.PyMPI_scatter␣
↪(c:\users\utente\appdata\local\temp\pip-build-h14iaj\mpi4py\src\mpi4py.MPI.c:34035)
File "pickled.pxi", line 129, in mpi4py.MPI._p_Pickle.dumpv␣
↪(c:\users\utente\appdata\local\temp\pip-build-h14iaj\mpi4py\src\mpi4py.MPI.c:28325)
       ValueError: expecting 3 items, got 10 mpiexec aborting job...
       job aborted:
       rank: node: exit code[: error message]
       0: Utente-PC: 123: mpiexec aborting job
       1: Utente-PC: 123
       2: Utente-PC: 123
```

3.15.3

mpi4py

- `comm.scatter(sendbuf, recvbuf, root=0)`: communicator
- `comm.scatterv(sendbuf, recvbuf, root=0)`:

`sendbuf` `recvbuf` list `comm.send`

```
buf = [data, data_size, data_type]
```

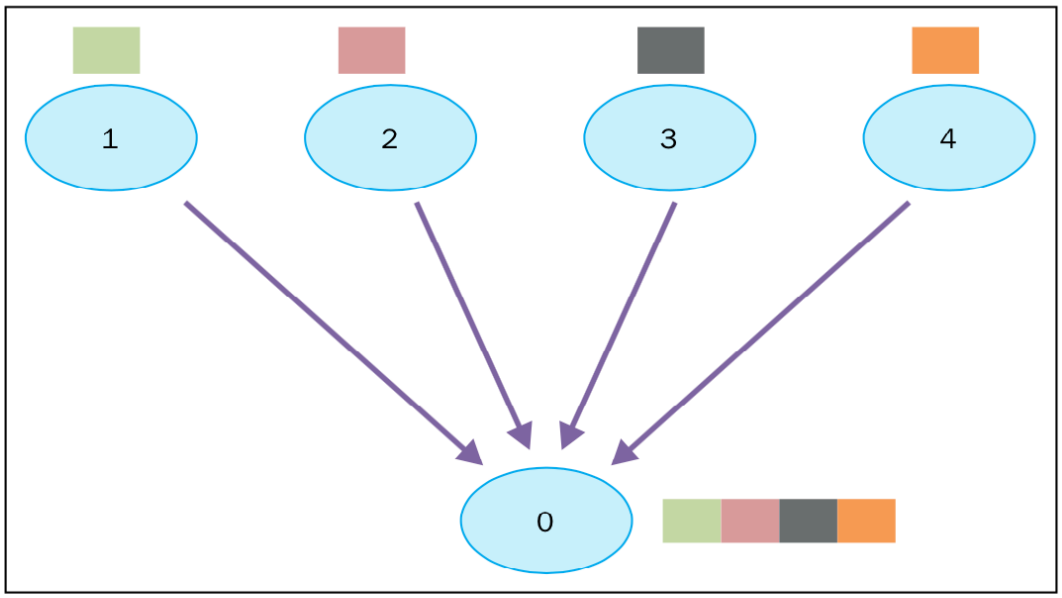
`data` buffer-like size `data_size` `data_type`

3.16 gather

`gather` `scatter` root `mpi4py` `gather`

```
recvbuf = comm.gather(sendbuf, rank_of_root_process)
```

`sendbuf` `rank_of_root_process`



Gathering data from processes 1, 2, 3, 4

3.16.1 ...

root rank 0

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
data = (rank+1)**2
data = comm.gather(data, root=0)
if rank == 0:
```

( )

( )

```
    print ("rank = %s " %rank + "...receiving data to other process")
    for i in range(1, size):
        data[i] = (i+1)**2
        value = data[i]
        print(" process %s receiving %s from process %s" % (rank , value , i))
```

5

```
C:\>mpiexec -n 5 python gather.py
rank = 0 ...receiving data to other process
process 0 receiving 4 from process 1
process 0 receiving 9 from process 2
process 0 receiving 16 from process 3
process 0 receiving 25 from process 4
```

root

3.16.2

n

```
data = (rank+1)**2
```

rank 0 array

```
if rank == 0:
    print ("rank = %s " %rank + "...receiving data to other process")
    for i in range(1, size):
        data[i] = (i+1)**2
        value = data[i]
        print(" process %s receiving %s from process %s" % (rank , value , i))
```

:

```
data = (rank+1)**2
```

3.16.3

mpi4py

- **gathering to one task**: comm.Gather, comm.Gatherv, comm.gather
- **gathering to all tasks**: comm.Allgather, comm.Allgatherv, comm.allgather

3.17 Alltoall

Alltoall scatter gather mpi4py Alltoall

- comm.Alltoall(sendbuf, recvbuf) :
- comm.Alltoallv(sendbuf, recvbuf) :

- comm.Alltoallw(sendbuf, recvbuf) :

3.17.1 ...

mpi4py comm.Alltoall

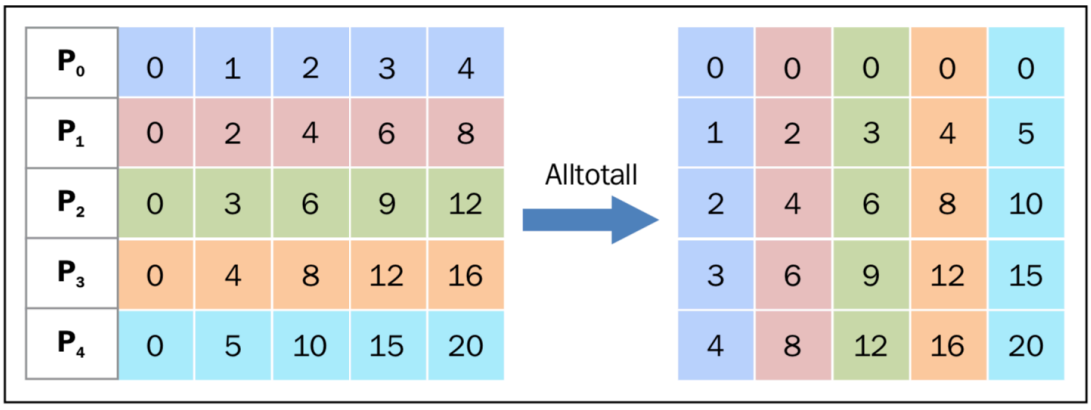
```
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
a_size = 1

senddata = (rank+1)*numpy.arange(size,dtype=int)
recvdata = numpy.empty(size*a_size,dtype=int)

comm.Alltoall(senddata,recvdata)
print(" process %s sending %s receiving %s" % (rank , senddata , recvdata))
```

5

```
C:\>mpiexec -n 5 python alltoall.py
process 0 sending [0 1 2 3 4] receiving [0 0 0 0 0]
process 1 sending [0 2 4 6 8] receiving [1 2 3 4 5]
process 2 sending [0 3 6 9 12] receiving [2 4 6 8 10]
process 3 sending [0 4 8 12 16] receiving [3 6 9 12 15]
process 4 sending [0 5 10 15 20] receiving [4 8 12 16 20]
```

3.17.2 ...

comm.alltoall task j sendbuf i task i recvbuf j



The Alltoall collective communication

- **P0** [0 1 2 3 4] **0** **1** **P1** **2** **P2** **3** **P3** **4** **P4**
- ……

3.17.3

All-to-all Join

3.18

comm.gather comm.reduce root

mpi4py

```
comm.Reduce(sendbuf, recvbuf, rank_of_root_process, op = type_of_reduction_operation)
```

op comm.gather mpi4py

- MPI.MAX :
- MPI.MIN :
- MPI.SUM :
- MPI.PROD :
- MPI.LAND :
- MPI.MAXLOC :
- MPI.MINLOC :

3.18.1 ...

MPI.SUM 3 numpy

```
import numpy
import numpy as np
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.size
rank = comm.rank
array_size = 3
recvdata = numpy.zeros(array_size, dtype=numpy.int)
senddata = (rank+1)*numpy.arange(size,dtype=numpy.int)
print("process %s sending %s " % (rank , senddata))
comm.Reduce(senddata, recvdata, root=0, op=MPI.SUM)
print('on task', rank, 'after Reduce:    data = ', recvdata)
```

3

```
C:\>mpiexec -n 3 python reduction2.py
 process 2 sending [0 3 6]
on task 2 after Reduce:    data =  [0 0 0]
 process 1 sending [0 2 4]
```

( )

( )

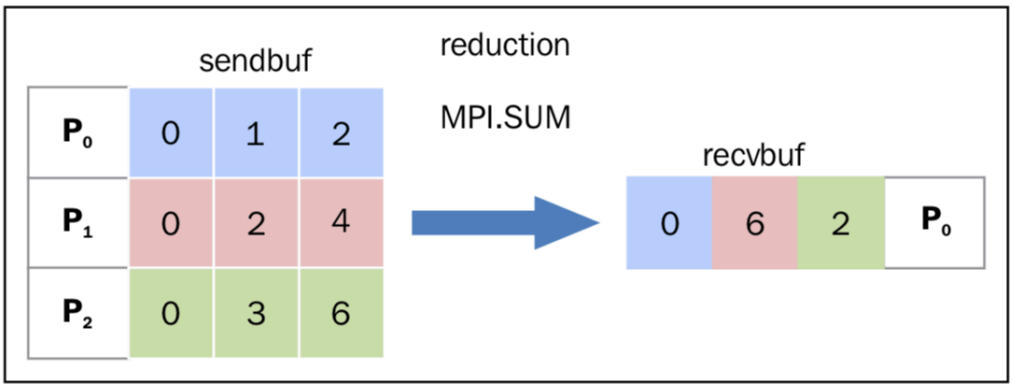
```
on task 1 after Reduce:    data =  [0 0 0]
 process 0 sending [0 1 2]
on task 0 after Reduce:    data =  [ 0  6 12]
```

3.18.2

comm.Reduce recvbuf rank 0, recvdata

```
comm.Reduce(senddata, recvdata, root=0, op=MPI.SUM)
```

op = MPI.SUM



The reduction collective communication

- **P0** [0 1 2]
- **P1** [0 2 4]
- **P2** [0 3 6]

task i **P0** i **P0** [0 6 12]

“ ”

3.19

MPI MPI_COMM_WORLD n 0 - n-1 rank MPI

MPI

```
comm.Create_cart((number_of_rows,number_of_columns))
```

number_of_rows number_of_columns

3.19.1 ...

M x N

```
from mpi4py import MPI
import numpy as np
UP = 0
DOWN = 1
LEFT = 2
RIGHT = 3
neighbour_processes = [0,0,0,0]

if __name__ == "__main__":
    comm = MPI.COMM_WORLD
    rank = comm.rank
    size = comm.size
    grid_rows = int(np.floor(np.sqrt(comm.size)))
    grid_column = comm.size // grid_rows
    if grid_rows*grid_column > size:
        grid_column -= 1
    if grid_rows*grid_column > size:
        grid_rows -= 1
    if (rank == 0) :
        print("Building a %d x %d grid topology:" % (grid_rows, grid_column) )
    cartesian_communicator = comm.Create_cart( (grid_rows, grid_column), periods=(True,␣
↪True), reorder=True)
    my_mpi_row, my_mpi_col = cartesian_communicator.Get_coords( cartesian_communicator.
↪rank )
    neighbour_processes[UP], neighbour_processes[DOWN] = cartesian_communicator.Shift(0,␣
↪1)
    neighbour_processes[LEFT], neighbour_processes[RIGHT] =  cartesian_communicator.
↪Shift(1, 1)
    print ("Process = %s row = %s column = %s ----> neighbour_processes[UP] = %s␣
↪neighbour_processes[DOWN] = %s neighbour_processes[LEFT] =%s neighbour_
↪processes[RIGHT]=%s" % (
    rank, my_mpi_row, my_mpi_col,neighbour_processes[UP],
    neighbour_processes[DOWN], neighbour_processes[LEFT],
    neighbour_processes[RIGHT]))
```

```
C:\>mpiexec -n 4 python virtualTopology.py
Building a 2 x 2 grid topology:
Process = 0 row = 0 column = 0 ---->
neighbour_processes[UP] = -1
neighbour_processes[DOWN] = 2
neighbour_processes[LEFT] =-1
neighbour_processes[RIGHT]=1
Process = 1 row = 0 column = 1 ---->
neighbour_processes[UP] = -1
neighbour_processes[DOWN] = 3
neighbour_processes[LEFT] =0
neighbour_processes[RIGHT]=-1
```

( )

( )

```
Process = 2 row = 1 column = 0 ---->
neighbour_processes[UP] = 0
neighbour_processes[DOWN] = -1
neighbour_processes[LEFT] =-1
neighbour_processes[RIGHT]=3
Process = 3 row = 1 column = 1 ---->
neighbour_processes[UP] = 1
neighbour_processes[DOWN] = -1
neighbour_processes[LEFT] =2
neighbour_processes[RIGHT]=-1
```

neighbour_processes = -1 neighbour_processes

3.19.2

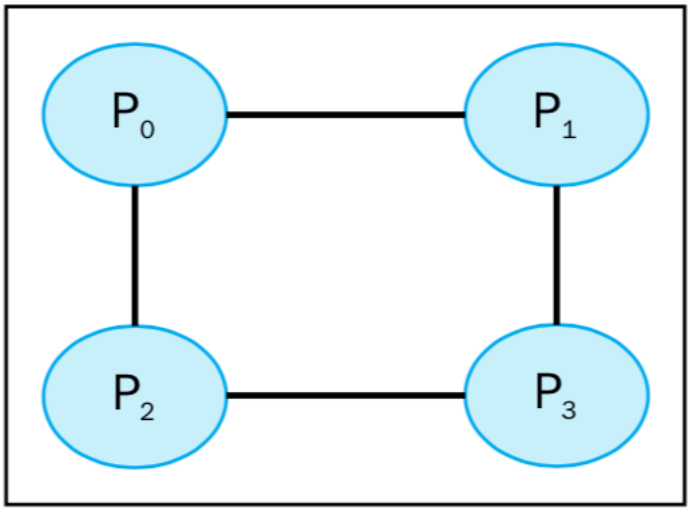
2x2 4

```
grid_rows = int(np.floor(np.sqrt(comm.size)))
grid_column = comm.size // grid_rows
if grid_rows*grid_column > size:
    grid_column -= 1
if grid_rows*grid_column > size:
    grid_rows -= 1
```

```
cartesian_communicator = comm.Create_cart( (grid_rows, grid_column), periods=(True,␣
↪True), reorder=True)
```

Get_coords() :

```
my_mpi_row, my_mpi_col = cartesian_communicator.Get_coords( cartesian_communicator.rank )
```



The virtual mesh 2x2 topology

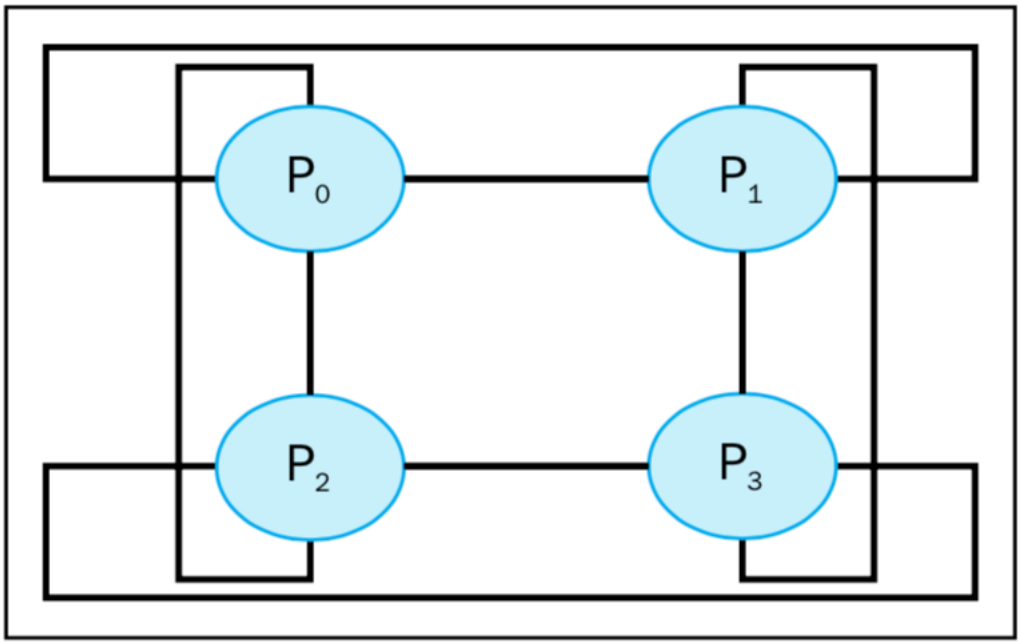
3.19.3

M x N

```
cartesian_communicator = comm.Create_cart( (grid_rows, grid_column), periods=(True,␣
↪True), reorder=True)
```

```
C:\>mpiexec -n 4 python VirtualTopology.py
Building a 2 x 2 grid topology:
Process = 0 row = 0 column = 0 ---->
neighbour_processes[UP] = 2
neighbour_processes[DOWN] = 2
neighbour_processes[LEFT] =1
neighbour_processes[RIGHT]=1
Process = 1 row = 0 column = 1 ---->
neighbour_processes[UP] = 3
neighbour_processes[DOWN] = 3
neighbour_processes[LEFT] =0
neighbour_processes[RIGHT]=0
Process = 2 row = 1 column = 0 ---->
neighbour_processes[UP] = 0
neighbour_processes[DOWN] = 0
neighbour_processes[LEFT] =3 neighbour_processes[RIGHT]=3
Process = 3 row = 1 column = 1 ---->
neighbour_processes[UP] = 1
neighbour_processes[DOWN] = 1
neighbour_processes[LEFT] =2
neighbour_processes[RIGHT]=2
```

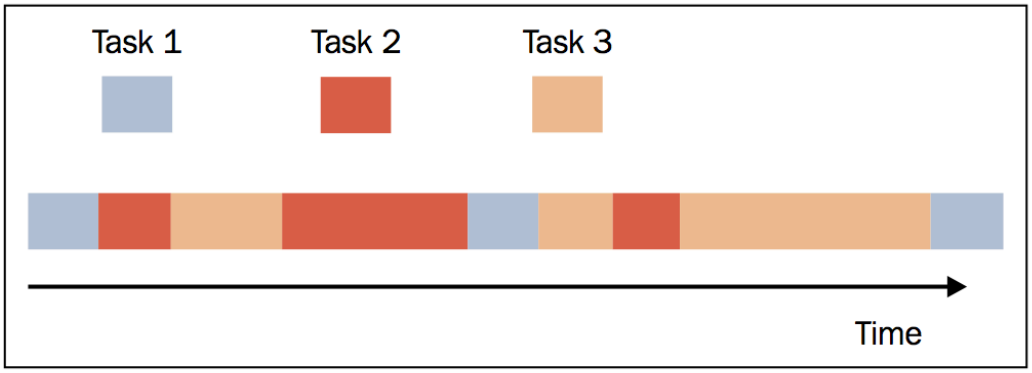
:



The virtual toroidal 2x2 topology

CHAPTER 4

4.1



Asynchronous programming model

4.2 Python `concurrent.futures`

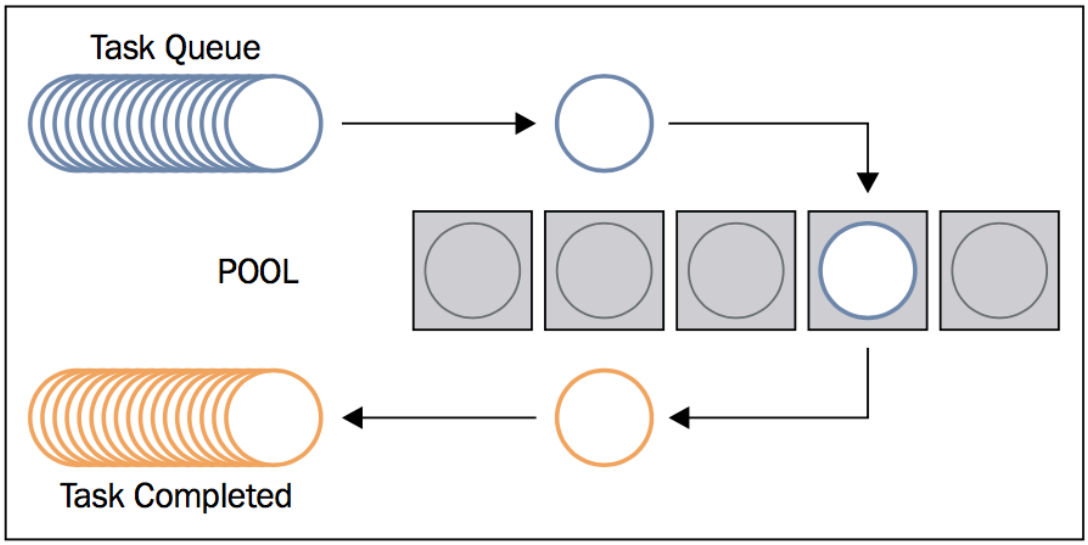
Python3.2 `concurrent.futures` /

- `concurrent.futures.Executor`:
- `submit(function, argument)`: `argument`
- `map(function, argument)`: `argument`
- `shutdown(Wait=True)`:
- `concurrent.futures.Future`: Future submit functions executor

Executor `ExecutorPools` “ ” launcher executor

4.2.1

/ executor



Pooling management

4.2.2

`current.Futures` `Executor`

- `concurrent.futures.ThreadPoolExecutor(max_workers)`
- `concurrent.futures.ProcessPoolExecutor(max_workers)`

`max_workers` worker

4.2.3 ...

list `number_list` 1 10 list 1+2+3...+10000000

-

- 5 worker
- 5 worker

2 #16 @Microndgt :

```
import concurrent.futures
import time
number_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def evaluate_item(x):
        #
        result_item = count(x)
        #
        return result_item

def  count(number) :
        for i in range(0, 10000000):
                i=i+1
        return i * number

if __name__ == "__main__":
        #
        start_time = time.time()
        for item in number_list:
                print(evaluate_item(item))
        print("Sequential execution in " + str(time.time() - start_time), "seconds")
        #
        start_time_1 = time.time()
        with concurrent.futures.ThreadPoolExecutor(max_workers=5) as executor:
                futures = [executor.submit(evaluate_item, item) for item in number_list]
                for future in concurrent.futures.as_completed(futures):
                        print(future.result())
        print ("Thread pool execution in " + str(time.time() - start_time_1), "seconds")
        #
        start_time_2 = time.time()
        with concurrent.futures.ProcessPoolExecutor(max_workers=5) as executor:
                futures = [executor.submit(evaluate_item, item) for item in number_list]
                for future in concurrent.futures.as_completed(futures):
                        print(future.result())
        print ("Process pool execution in " + str(time.time() - start_time_2), "seconds")
```

:

```
$ python3 pool.py
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
```

( )

( )

```
Sequential execution in 7.936585903167725 seconds
10000000
30000000
40000000
20000000
50000000
70000000
90000000
100000000
80000000
60000000
Thread pool execution in 7.633088827133179 seconds
40000000
50000000
10000000
30000000
20000000
70000000
90000000
60000000
80000000
100000000
Process pool execution in 4.787093639373779 seconds
```

4.2.4

list 10 1 10000000 number_list :

```
def evaluate_item(x):
    #
    result_item = count(x)
    #
    print ("item " + str(x) + " result " + str(result_item))

def  count(number) :
    for i in range(0, 10000000):
        i=i+1
    return i * number
```

:

```
if __name__ == "__main__":
    #
    start_time = time.clock()
    for item in number_list:
        evaluate_item(item)
    print("Sequential execution in " + str(time.clock() - start_time), "seconds")
```

futures.ThreadPoolExecutor :

```
with concurrent.futures.ThreadPoolExecutor(max_workers=5) as executor:
    for item in number_list:
        executor.submit(evaluate_item,  item)
print ("Thread pool execution in " + str(time.clock() - start_time_1), "seconds")
```

ThreadPoolExecutor 5

:

```
print ("Thread pool execution in " + str(time.clock() - start_time_1), "seconds")
```

ProcessPoolExecutor :

```
with concurrent.futures.ProcessPoolExecutor(max_workers=5) as executor:
    for item in number_list:
        executor.submit(evaluate_item,  item)
```

ThreadPoolExecutor ProcessPoolExecutor executor ThreadPoolExecutor
ProcessPoolExecutor GIL

4.2.5

4.3 Asyncio

Python Asyncio

- : Asyncio
- : IO
- **Futures**: Future concurrent.futures
- **Tasks**: Asyncio

4.3.1

:

```
while (1) {
    events = getEvents();
    for (e in events)
        processEvent(e);
}
```

while

4.3.2

Asyncio

- loop = get_event_loop():
- loop.call_later(time_delay, callback, argument): time_delay callback
- loop.call_soon(callback, argument): callback, call_soon() callback
- loop.time(): float
- asyncio.set_event_loop():
- asyncio.new_event_loop():
- loop.run_forever(): stop()

4.3.3 ...

Asyncio

```
import asyncio
import datetime
import time

def function_1(end_time, loop):
    print ("function_1 called")
    if (loop.time() + 1.0) < end_time:
        loop.call_later(1, function_2, end_time, loop)
    else:
        loop.stop()

def function_2(end_time, loop):
    print ("function_2 called ")
    if (loop.time() + 1.0) < end_time:
        loop.call_later(1, function_3, end_time, loop)
    else:
        loop.stop()

def function_3(end_time, loop):
    print ("function_3 called")
    if (loop.time() + 1.0) < end_time:
        loop.call_later(1, function_1, end_time, loop)
    else:
        loop.stop()

def function_4(end_time, loop):
    print ("function_5 called")
    if (loop.time() + 1.0) < end_time:
        loop.call_later(1, function_4, end_time, loop)
    else:
        loop.stop()

loop = asyncio.get_event_loop()
```

( )

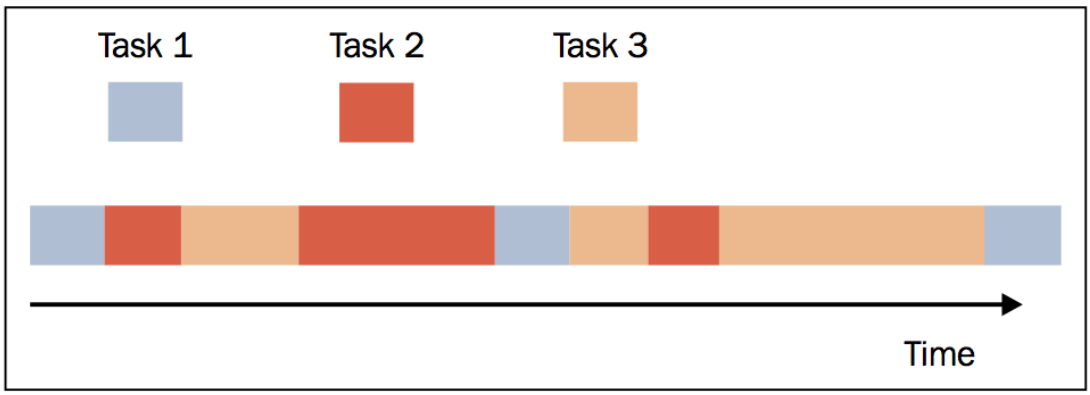
( )

```
end_loop = loop.time() + 9.0
loop.call_soon(function_1, end_loop, loop)
# loop.call_soon(function_4, end_loop, loop)
loop.run_forever()
loop.close()
```

:

```
python3 event.py
function_1 called
function_2 called
function_3 called
function_1 called
function_2 called
function_3 called
function_1 called
function_2 called
function_3 called
```

4.3.4



Task execution in the example

:

```
loop = asyncio.get_event_loop()
```

call_soon function_1()

```
end_loop = loop.time() + 9.0
loop.call_soon(function_1, end_loop, loop)
```

function_1() :

```
def function_1(end_time, loop):
    print ("function_1 called")
    if (loop.time() + 1.0) < end_time:
        loop.call_later(1, function_2, end_time, loop)
    else:
        loop.stop()
```

- end_time: function_1() call_later function_2()
- loop: get_event_loop()

function_1()

```
print ("function_1 called")
```

loop.time() +1s call_later 1 function_2()

```
if (loop.time() + 1.0) < end_time:
    loop.call_later(1, function_2, end_time, loop)
else:
    loop.stop()
```

function_2() function_3()

```
loop.run_forever()
loop.close()
```

4.4 Asyncio

- yield
-

yield “yield ”

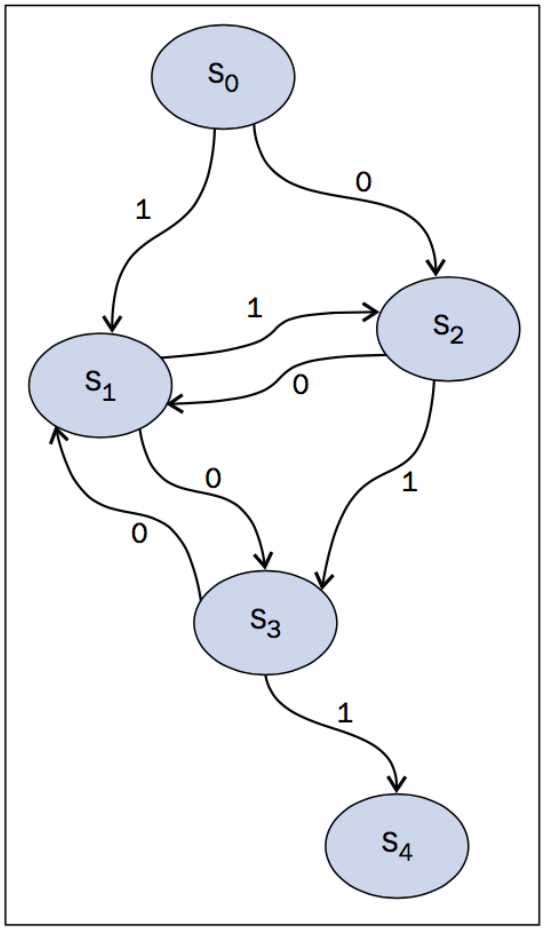
4.4.1

Asyncio

```
import asyncio

@asyncio.coroutine
def coroutine_function(function_arguments):
    # DO_SOMETHING
```

4.4.2 ...

Asyncio (finite state machine or automaton, FSA)



Finite state machine

S1, **S2**, **S3**, **S4** , **0** **1** 1 **S0** **S1** 0 **S0** **S2** .Python **S0** **S4**

```
# Asyncio Finite State Machine
import asyncio
import time
from random import randint


@asyncio.coroutine
def StartState():
    print("Start State called \n")
    input_value = randint(0, 1)
    time.sleep(1)
    if (input_value == 0):
```

( )

```
        result = yield from State2(input_value)
    else:
        result = yield from State1(input_value)
    print("Resume of the Transition : \nStart State calling " + result)

@asyncio.coroutine
def State1(transition_value):
    outputValue =  str("State 1 with transition value = %s \n" % transition_value)
    input_value = randint(0, 1)
    time.sleep(1)
    print("...Evaluating...")
    if input_value == 0:
        result = yield from State3(input_value)
    else :
        result = yield from State2(input_value)
    result = "State 1 calling " + result
    return outputValue + str(result)

@asyncio.coroutine
def State2(transition_value):
    outputValue =  str("State 2 with transition value = %s \n" % transition_value)
    input_value = randint(0, 1)
    time.sleep(1)
    print("...Evaluating...")
    if (input_value == 0):
        result = yield from State1(input_value)
    else :
        result = yield from State3(input_value)
    result = "State 2 calling " + result
    return outputValue + str(result)

@asyncio.coroutine
def State3(transition_value):
    outputValue = str("State 3 with transition value = %s \n" % transition_value)
    input_value = randint(0, 1)
    time.sleep(1)
    print("...Evaluating...")
    if (input_value == 0):
        result = yield from State1(input_value)
    else :
        result = yield from EndState(input_value)
    result = "State 3 calling " + result
    return outputValue + str(result)

@asyncio.coroutine
def EndState(transition_value):
    outputValue = str("End State with transition value = %s \n" % transition_value)
    print("...Stop Computation...")
    return outputValue

if __name__ == "__main__":
    print("Finite State Machine simulation with Asyncio Coroutine")
```

( )

```
    loop = asyncio.get_event_loop()
    loop.run_until_complete(StartState())
```

.

```
$ python3 coroutines.py
Finite State Machine simulation with Asyncio Coroutine
Start State called

...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Stop Computation...
Resume of the Transition :
Start State calling State 2 with transition value = 0
State 2 calling State 1 with transition value = 0
State 1 calling State 2 with transition value = 1
State 2 calling State 1 with transition value = 0
State 1 calling State 2 with transition value = 1
State 2 calling State 3 with transition value = 1
State 3 calling End State with transition value = 1

$ python3 coroutines.py
Finite State Machine simulation with Asyncio Coroutine
Start State called

...Evaluating...
...Evaluating...
...Stop Computation...
Resume of the Transition :
Start State calling State 2 with transition value = 0
State 2 calling State 3 with transition value = 1
State 3 calling End State with transition value = 1

$ python3 coroutines.py
Finite State Machine simulation with Asyncio Coroutine
Start State called

...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Evaluating...
...Stop Computation...
Resume of the Transition :
Start State calling State 1 with transition value = 1
State 1 calling State 2 with transition value = 1
```

( )

( )

```
State 2 calling State 1 with transition value = 0
State 1 calling State 3 with transition value = 0
State 3 calling State 1 with transition value = 0
State 1 calling State 2 with transition value = 1
State 2 calling State 3 with transition value = 1
State 3 calling End State with transition value = 1
```

4.4.3

```
@asyncio.coroutine
```

S0

```
@asyncio.coroutine
def StartState():
    print("Start State called \n")
    input_value = randint(0, 1)
    time.sleep(1)
    if (input_value == 0):
        result = yield from State2(input_value)
    else:
        result = yield from State1(input_value)
    print("Resume of the Transition : \nStart State calling " + result)
```

random randint(0, 1) input_value 1 0

```
input_value = randint(0, 1)
```

input_value yield from

```
if (input_value == 0):
    result = yield from State2(input_value)
else:
    result = yield from State1(input_value)
```

result string

```
if __name__ == "__main__":
    print("Finite State Machine simulation with Asyncio Coroutine")
    loop = asyncio.get_event_loop()
    loop.run_until_complete(StartState())
```

4.5 Asyncio

Asyncio asyncio.Task() ,

4.5.1

Asyncio asyncio.Task(coroutine) future yield future future future exception future

```
"""
Asyncio using Asyncio.Task to execute three math function in parallel
"""
import asyncio
@asyncio.coroutine
def factorial(number):
    f = 1
    for i in range(2, number + 1):
        print("Asyncio.Task: Compute factorial(%s)" % (i))
        yield from asyncio.sleep(1)
        f *= i
    print("Asyncio.Task - factorial(%s) = %s" % (number, f))

@asyncio.coroutine
def fibonacci(number):
    a, b = 0, 1
    for i in range(number):
        print("Asyncio.Task: Compute fibonacci (%s)" % (i))
        yield from asyncio.sleep(1)
        a, b = b, a + b
    print("Asyncio.Task - fibonacci(%s) = %s" % (number, a))

@asyncio.coroutine
def binomialCoeff(n, k):
    result = 1
    for i in range(1, k+1):
        result = result * (n-i+1) / i
        print("Asyncio.Task: Compute binomialCoeff (%s)" % (i))
        yield from asyncio.sleep(1)
    print("Asyncio.Task - binomialCoeff(%s , %s) = %s" % (n, k, result))

if __name__ == "__main__":
    tasks = [asyncio.Task(factorial(10)),
             asyncio.Task(fibonacci(10)),
             asyncio.Task(binomialCoeff(20, 10))]
    loop = asyncio.get_event_loop()
    loop.run_until_complete(asyncio.wait(tasks))
    loop.close()
```

4.5.2 ...

Asyncio.Task()

```
python3 task.py
Asyncio.Task: Compute factorial(2)
```

( )

( )

```
Asyncio.Task: Compute fibonacci (0)
Asyncio.Task: Compute binomialCoeff (1)
Asyncio.Task: Compute factorial(3)
Asyncio.Task: Compute fibonacci (1)
Asyncio.Task: Compute binomialCoeff (2)
Asyncio.Task: Compute factorial(4)
Asyncio.Task: Compute fibonacci (2)
Asyncio.Task: Compute binomialCoeff (3)
Asyncio.Task: Compute factorial(5)
Asyncio.Task: Compute fibonacci (3)
Asyncio.Task: Compute binomialCoeff (4)
Asyncio.Task: Compute factorial(6)
Asyncio.Task: Compute fibonacci (4)
Asyncio.Task: Compute binomialCoeff (5)
Asyncio.Task: Compute factorial(7)
Asyncio.Task: Compute fibonacci (5)
Asyncio.Task: Compute binomialCoeff (6)
Asyncio.Task: Compute factorial(8)
Asyncio.Task: Compute fibonacci (6)
Asyncio.Task: Compute binomialCoeff (7)
Asyncio.Task: Compute factorial(9)
Asyncio.Task: Compute fibonacci (7)
Asyncio.Task: Compute binomialCoeff (8)
Asyncio.Task: Compute factorial(10)
Asyncio.Task: Compute fibonacci (8)
Asyncio.Task: Compute binomialCoeff (9)
Asyncio.Task - factorial(10) = 3628800
Asyncio.Task: Compute fibonacci (9)
Asyncio.Task: Compute binomialCoeff (10)
Asyncio.Task - fibonacci(10) = 55
Asyncio.Task - binomialCoeff(20 , 10) = 184756.0
```

4.5.3

factorial, fibonacci binomialCoeff asyncio.coroutine

```
@asyncio.coroutine
def factorial(number):
    do Something

@asyncio.coroutine
def fibonacci(number):
    do Something

@asyncio.coroutine
def binomialCoeff(n, k):
    do Something
```

task list

```
if __name__ == "__main__":
    tasks = [asyncio.Task(factorial(10)),
             asyncio.Task(fibonacci(10)),
             asyncio.Task(binomialCoeff(20, 10))]
```

```
loop = asyncio.get_event_loop()
```

```
loop.run_until_complete(asyncio.wait(tasks))
```

asyncio.wait(tasks)

```
loop.close()
```

4.6 Asyncio Futures

Asyncio Future concurrent.futures.Futures Asyncio asyncio.Futures Exception

4.6.1

Asyncio Future

```
import asyncio
future = asyncio.Future()
```

- cancel(): future
- result(): future
- exception(): future Exception
- add_done_callback(fn): future
- remove_done_callback(fn): “call whten done” callback
- set_result(result): future result
- set_exception(exception): future Exception

4.6.2 ...

Future first_coroutine n second_coroutine n

```
# -*- coding: utf-8 -*-

"""
Asyncio.Futures -  Chapter 4 Asynchronous Programming
"""
import asyncio
import sys

@asyncio.coroutine
def first_coroutine(future, N):
    """n  """
    count = 0
    for i in range(1, N + 1):
        count = count + i
    yield from asyncio.sleep(3)
    future.set_result("first coroutine (sum of N integers) result = " + str(count))

@asyncio.coroutine
def second_coroutine(future, N):
    count = 1
    for i in range(2, N + 1):
        count *= i
    yield from asyncio.sleep(4)
    future.set_result("second coroutine (factorial) result = " + str(count))

def got_result(future):
   print(future.result())

if __name__ == "__main__":
   N1 = int(sys.argv[1])
   N2 = int(sys.argv[2])
   loop = asyncio.get_event_loop()
   future1 = asyncio.Future()
   future2 = asyncio.Future()
   tasks = [
       first_coroutine(future1, N1),
       second_coroutine(future2, N2)]
   future1.add_done_callback(got_result)
   future2.add_done_callback(got_result)
   loop.run_until_complete(asyncio.wait(tasks))
   loop.close()
```

```
$ python asy.py 1 1
first coroutine (sum of N integers) result = 1
second coroutine (factorial) result = 1
$ python asy.py 2 2
first coroutine (sum of N integers) result = 3
second coroutine (factorial) result = 2
$ python asy.py 3 3
first coroutine (sum of N integers) result = 6
second coroutine (factorial) result = 6
```

( )

( )

```
$ python asy.py 4 4
first coroutine (sum of N integers) result = 10
second coroutine (factorial) result = 24
```

4.6.3

future

```
if __name__ == "__main__":
   ...
   future1 = asyncio.Future()
   future2 = asyncio.Future()
```

tasks future

```
tasks = [
    first_coroutine(future1, N1),
    second_coroutine(future2, N2)]
```

future

```
def got_result(future):
   print(future.result())
```

future 3s 4s

```
yield from asyncio.sleep(4)
```

future `future.set_result()`

4.6.4

2 1

```
$ python asy.py 3 3
second coroutine (factorial) result = 6
first coroutine (sum of N integers) result = 6
$ python asy.py 4 4
second coroutine (factorial) result = 24
first coroutine (sum of N integers) result = 10
```

CHAPTER 5

Python

5.1

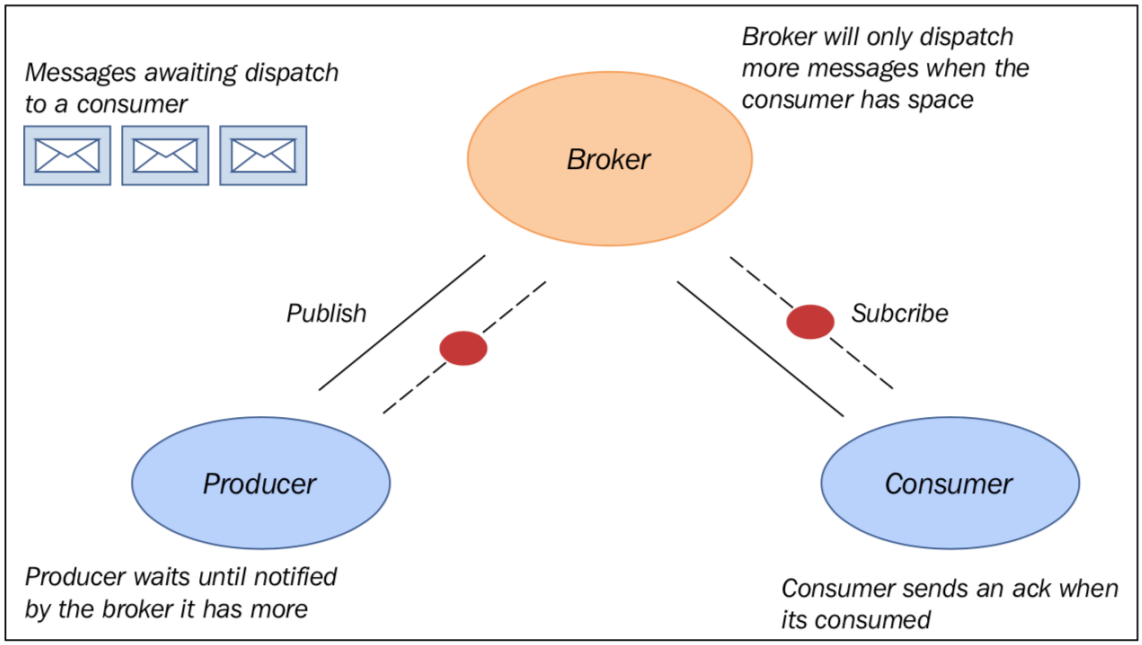
/ /
Celery SCOOP Pyro4 RPyC PyCSP Disco Python MapReduce

5.2 Celery

Celery Python feature

Celery

- Celery
- (Message Broker) Celery worker /



The message broker architecture

Celery RebbitMQ Redis

5.2.1 ...

Celery *pip*

```
pip install celery
```

RabbitMQ **AMQP** RabbitMQ Erlang **Open Telecom Platform(OPT)** RabbitMQ

1. Erlang http://www.erlang.org/download.html
2. RabbitMQ http://www.rabbitmq.com/download.html

RabbitMQ

Flower http://flower.readthedocs.org/ web Celery

```
pip install -U flower
```

Celery :

```
celery --version
```

( ):

```
4.2.1 (windowlicker)
```

Celery

```
Usage: celery <command> [options]
```

```
usage: celery <command> [options]

Show help screen and exit.

positional arguments:
  args

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

Global Options:
  -A APP, --app APP
  -b BROKER, --broker BROKER
  --result-backend RESULT_BACKEND
  --loader LOADER
  --config CONFIG
  --workdir WORKDIR
  --no-color, -C
  --quiet, -q

---- -- - - ---- Commands- -------------- --- ------------

+ Main:
|    celery worker
|    celery events
|    celery beat
|    celery shell
|    celery multi
|    celery amqp

+ Remote Control:
|    celery status

|    celery inspect --help
|    celery inspect active
|    celery inspect active_queues
|    celery inspect clock
|    celery inspect conf [include_defaults=False]
|    celery inspect memdump [n_samples=10]
|    celery inspect memsample
|    celery inspect objgraph [object_type=Request] [num=200 [max_depth=10]]
|    celery inspect ping
|    celery inspect query_task [id1 [id2 [... [idN]]]]
|    celery inspect registered [attr1 [attr2 [... [attrN]]]]
|    celery inspect report
|    celery inspect reserved
|    celery inspect revoked
|    celery inspect scheduled
```

( )

```
|    celery inspect stats

|    celery control --help
|    celery control add_consumer <queue> [exchange [type [routing_key]]]
|    celery control autoscale [max [min]]
|    celery control cancel_consumer <queue>
|    celery control disable_events
|    celery control election
|    celery control enable_events
|    celery control heartbeat
|    celery control pool_grow [N=1]
|    celery control pool_restart
|    celery control pool_shrink [N=1]
|    celery control rate_limit <task_name> <rate_limit (e.g., 5/s | 5/m | 5/h)>
|    celery control revoke [id1 [id2 [... [idN]]]]
|    celery control shutdown
|    celery control terminate <signal> [id1 [id2 [... [idN]]]]
|    celery control time_limit <task_name> <soft_secs> [hard_secs]

+ Utils:
|    celery purge
|    celery list
|    celery call
|    celery result
|    celery migrate
|    celery graph
|    celery upgrade

+ Debugging:
|    celery report
|    celery logtool
---- -- - - --------- -- - -------------- --- ------------

Type 'celery <command> --help' for help using a specific command.
```

5.2.2

Celery http://www.celeryproject.org/

5.3 Celery

Celery Celery

- `apply_async(args[, kwargs[, ...]])`:
- `delay(*args, **kwargs)`:

delay

```
task.delay(arg1, arg2, kwarg1='x', kwarg2='y')
```

apply_async

```
task.apply_async (args=[arg1, arg2] kwargs={'kwarg1': 'x','kwarg2': 'y'})
```

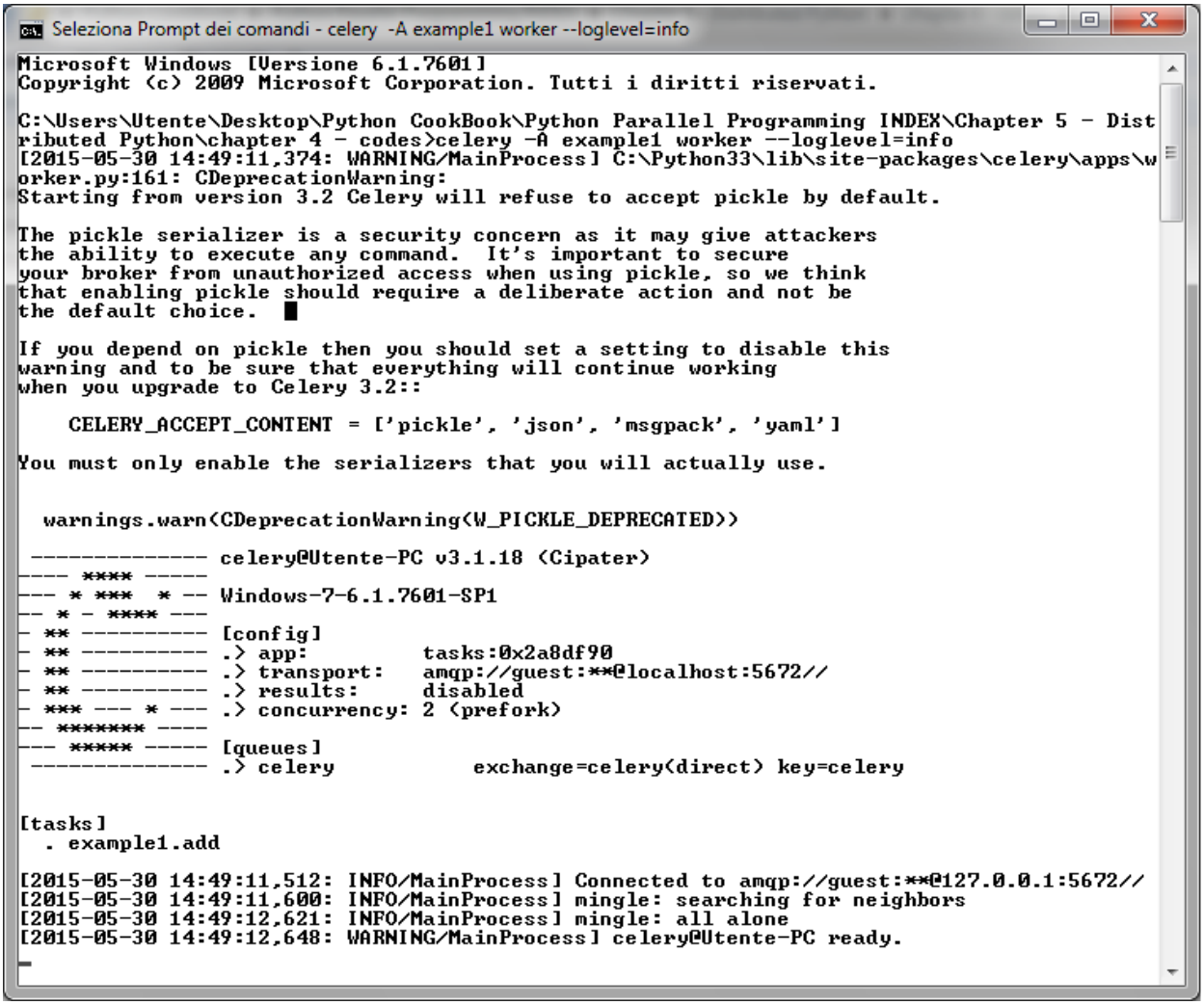
5.3.1 ...

```
# addTask.py: Executing a simple task
from celery import Celery
app = Celery('addTask', broker='amqp://guest@localhost//')
@app.task
def add(x, y):
    return x + y
```

```
# addTask_main.py : RUN the AddTask example with

import addTask
if __name__ == '__main__':
    result = addTask.add.delay(5,5)
```

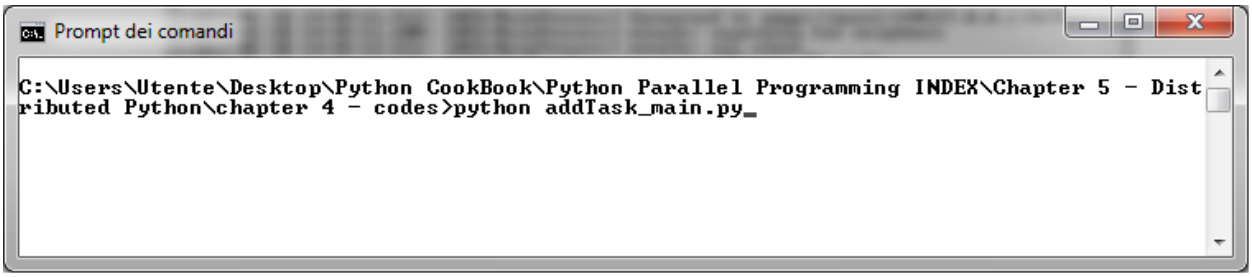
RabbitMQ Celery

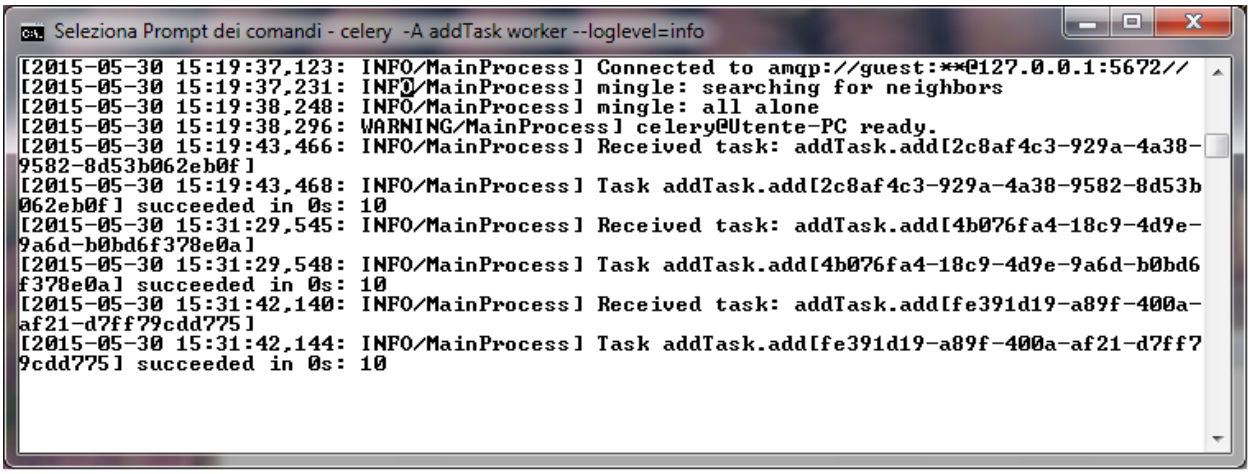
```
celery -A addTask worker --loglevel=info
```



pickle pickle (Python) pickle CELERY_ACCEPT_CONTENT http://celery.readthedocs.org/en/latest/configuration.html

addTask_main.py





10

5.3.2

addTask.py Celery RabbitMQ

```
from celery import Celery
app = Celery('addTask', broker='amqp://guest@localhost//')
```

Celery module addTask.py) broker(RabbitMQ). @app.task

Celery worker

```
@app.task
def add(x, y):
    return x + y
```

AddTask_main.py delay()

```
if __name__ == '__main__':
    result = addTask.add.delay(5,5)
```

apply_async() apply_async()

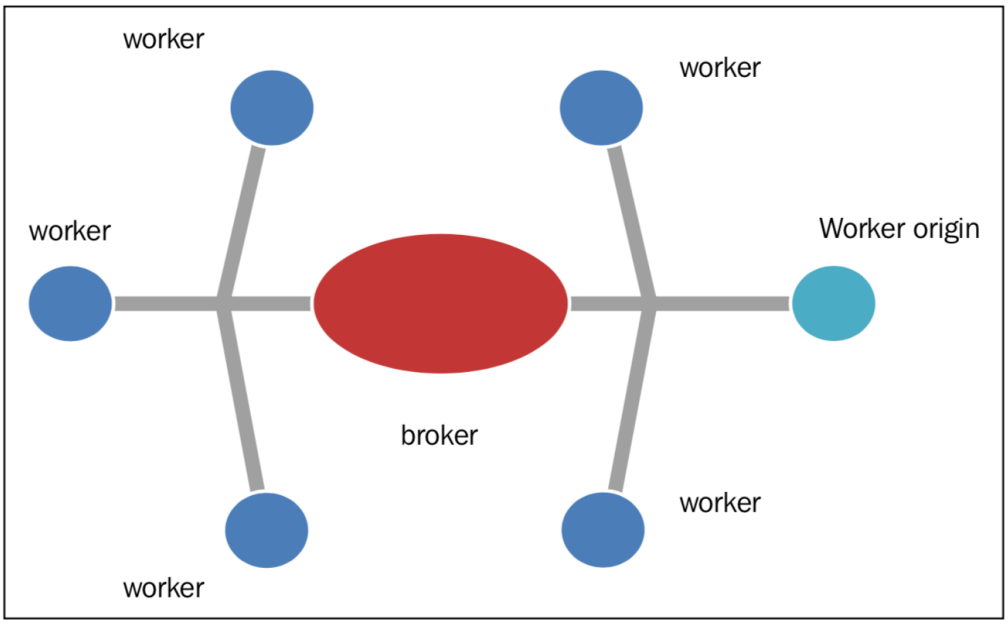
5.3.3

RabbitMQ Celery amqp://scheme

5.4 SCOOP

Scalable Concurrent Operations in Python (SCOOP) Python Python Futures
Q Futures SCOOP

Futures SCOOP Broker



The SCOOP architecture

Broker Broker Futures Broker Broker I/O CPU

5.4.1

SCOOP https://github.com/soravux/scoop/

- Python >= 2.6 or >= 3.2
- Distribute >= 0.6.2 or setuptools >= 0.7
- Greenlet >= 0.3.4
- pyzmq >= 13.1.0 and libzmq >= 3.2.0
- SSH for remote execution

SCOOP Linux, Mac, Windows Disco SSH SCOOP http://scoop.readthedocs.org/en/0.7/install.html

Window SCOOP pip :

```
pip install SCOOP
```

SCOOP

```
Python setup.py install
```

5.4.2 ...

SCOOP SCOOP π

```
import math
from random import random
from scoop import futures
from time import time


def evaluate_points_in_circle(attempts):
  points_fallen_in_unit_disk = 0
     for i in range (0,attempts) :
          x = random()
          y = random()
          radius = math.sqrt(x*x + y*y)
          #the test is ok if the point fall in the unit circle
          if radius < 1 :
              #if ok the number of points in a disk is increased
              points_fallen_in_unit_disk = \
                      points_fallen_in_unit_disk + 1
      return points_fallen_in_unit_disk

def pi_calculus_with_Montecarlo_Method(workers, attempts):
    print("number of workers %i - number of attempts %i" % (workers,attempts))
    bt = time()
    #in this point we call scoop.futures.map function
    #the evaluate_number_of_points_in_unit_circle \
    #function is executed in an asynchronously way
    #and several call this function can be made concurrently
    evaluate_task = \
                futures.map(evaluate_points_in_circle,
                       [attempts] * workers)
    taskresult= sum(evaluate_task)
    print ("%i points fallen in a unit disk after " \
           %(taskresult/attempts))
    piValue = (4. * taskresult/ float(workers * attempts))
    computationalTime = time() - bt
    print("value of pi = " + str(piValue))
    print ("error percentage = " + \
           str((((abs(piValue - math.pi)) * 100) / math.pi)))
    print("total time: " + str(computationalTime))

if __name__ == "__main__":
  for i in range (1,4):
     # let's fix the numbers of workers...only two,
     # but it could be much greater
     pi_calculus_with_Montecarlo_Method(i*1000, i*1000)
     print(" ")
```

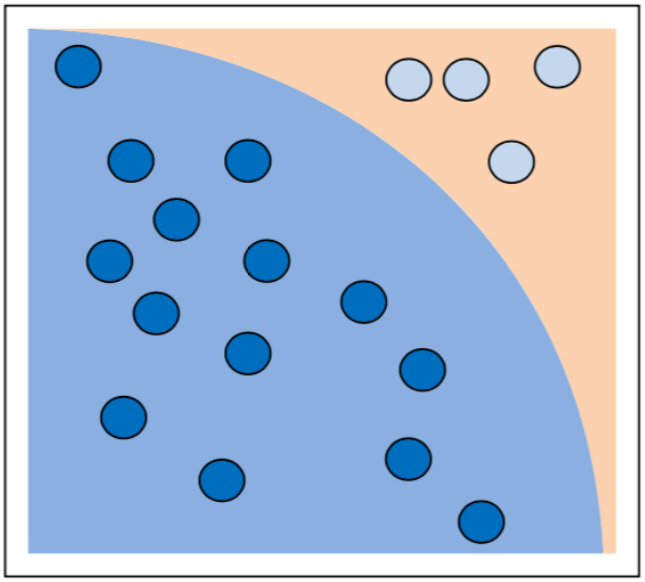
```
python -m scoop name_file.py
```

```
C:\Python CookBook\Chapter 5 - Distributed Python\chapter 5 - codes>python -m scoop pi_
↪calculus_with_montecarlo_method.py
```

( )

```
[2015-06-01 15:16:32,685] launcher  INFO    SCOOP 0.7.2 dev on win32 using Python 3.3.0␣
↪(v3.3.0:bd8afb90ebf2, Sep 29 2012, 10:55:48) [MSC v.1600 32 bit (Intel)], API: 1013
[2015-06-01 15:16:32,685] launcher  INFO Deploying 2 worker(s) over 1 host(s).
[2015-06-01 15:16:32,685] launcher  INFO Worker d--istribution:
[2015-06-01 15:16:32,686] launcher  INFO 127.0.0.1:1 + origin
Launching 2 worker(s) using an unknown shell.
number of workers 1000 - number of attempts 1000
785 points fallen in a unit disk after
value of pi = 3.140636
error percentage = 0.03045122952842962
total time: 10.258585929870605

number of workers 2000 - number of attempts 2000
1570 points fallen in a unit disk after
value of pi = 3.141976
error percentage = 0.012202295220195048
total time: 20.451170206069946

number of workers 3000 - number of attempts 3000
2356 points fallen in a unit disk after
value of pi = 3.1413777777777776
error percentage = 0.006839709526630775
total time: 32.3558509349823

[2015-06-01 15:17:36,894] launcher  (127.0.0.1:59239) INFO
process is done.
[2015-06-01 15:17:36,896] launcher  (127.0.0.1:59239) INFO
cleaning spawned subprocesses.
```

attempts worker π



Monte Carlo evaluation of π: counting points inside the circle

5.4.3 ...

π evaluate_points_in_circle() (x, y)

points_fallen_in_unit_disk 1. pi π / 4 π 4.

taskresult workers * attempts π / 4 π

```
piValue = (4. * Taskresult/ float(workers * attempts))
```

SCOOP

```
futures.map(evaluate_points_in_circle, [attempts] * workers)
```

SCOOP evaluate_points_in_circle

5.5 SCOOP map

list Python IDLE list

```
>>>items = [1,2,3,4,5,6,7,8,9,10]
>>>updated_items = []
>>>for x in items:
>>>    updated_items.append(x*2)
>>> updated_items
>>>  [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

Python

Python `map(aFunction, aSequence)` list

```
>>>items = [1,2,3,4,5,6,7,8,9,10]
>>>def multiplyFor2(x):return x*2
>>>print(list(map(multiplyFor2,items)))
>>>[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

`multiplyFor2` `map` . `items` list

lambda `map` :

```
>>>items = [1,2,3,4,5,6,7,8,9,10]
>>>print(list(map(lambda x:x*2,items)))
>>>[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

map for

5.5.1

SCOOP map

- `futures.map(func, iterables, kargs)` : `map`
- `futures.map_as_completed(func, iterables, kargs)` : yield
- `futures.scoop.futures.mapReduce(mapFunc, reductionOp, iterables, kargs)` : map reduction

5.5.2 ...

SCOOP MapReduce Python

```
"""
Compare SCOOP MapReduce with a serial implementation
"""
import operator
import time
from scoop import futures

def simulateWorkload(inputData):
    time.sleep(0.01)
    return sum(inputData)

def CompareMapReduce():
    mapScoopTime = time.time()
    res = futures.mapReduce(
        simulateWorkload,
        operator.add,
        list([a] * a for a in range(1000)),
    )
    mapScoopTime = time.time() - mapScoopTime
    print("futures.map in SCOOP executed in {0:.3f}s with result:{1}".format(
```

( )

( )

```
          mapScoopTime, res))

    mapPythonTime = time.time()
    res = sum(map(simulateWorkload, list([a] * a for a in range(1000))))
    mapPythonTime = time.time() - mapPythonTime
    print("map Python executed in: {0:.3f}s with result: {1}".format(
        mapPythonTime, res))

if __name__ == '__main__':
    CompareMapReduce()
```

```
python -m scoop map_reduce.py
> [2015-06-12 20:13:25,602] launcher  INFO    SCOOP 0.7.2 dev on win32
using Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC
v.1600 32 bit (Intel)], API: 1013
[2015-06-12 20:13:25,602] launcher  INFO Deploying 2 worker(s) over 1
host(s).
[2015-06-12 20:13:25,602] launcher  INFO Worker d--istribution:
[2015-06-12 20:13:25,602] launcher  INFO 127.0.0.1:       1 + origin
Launching 2 worker(s) using an unknown shell.
futures.map in SCOOP executed in 8.459s with result: 332833500
map Python executed in: 10.034s with result: 332833500
[2015-06-12 20:13:45,344] launcher  (127.0.0.1:2559) INFO
is done.
[2015-06-12 20:13:45,368] launcher  (127.0.0.1:2559) INFO
cleaning spawned subprocesses.
```

5.5.3

SCOOP MapReduce Python MapReduce `CompareMapReduce()`

```
mapScoopTime = tme.time()
#Run SCOOP MapReduce
mapScoopTime = time.time() - mapScoopTime

mapPythonTime = time.time()
#Run serial MapReduce
mapPythonTime = time.time() - mapPythonTime
```

```
futures.map in SCOOP executed in 8.459s with result: 332833500
map Python executed in: 10.034s with result: 332833500
```

`simulatedWordload` `time.sleep`

```
def simulateWorkload(inputData, chose=None):
    time.sleep(0.01)
    return sum(inputData)
```

SCOOP MapReduce

```
res = futures.mapReduce(
    simulateWorkload,
    operator.add,
    list([a] * a for a in range(1000)),
)
```

futures.mapReduce

- simulateWork : Futures callable
- operator.add : reduce
- list(...) : callable Future

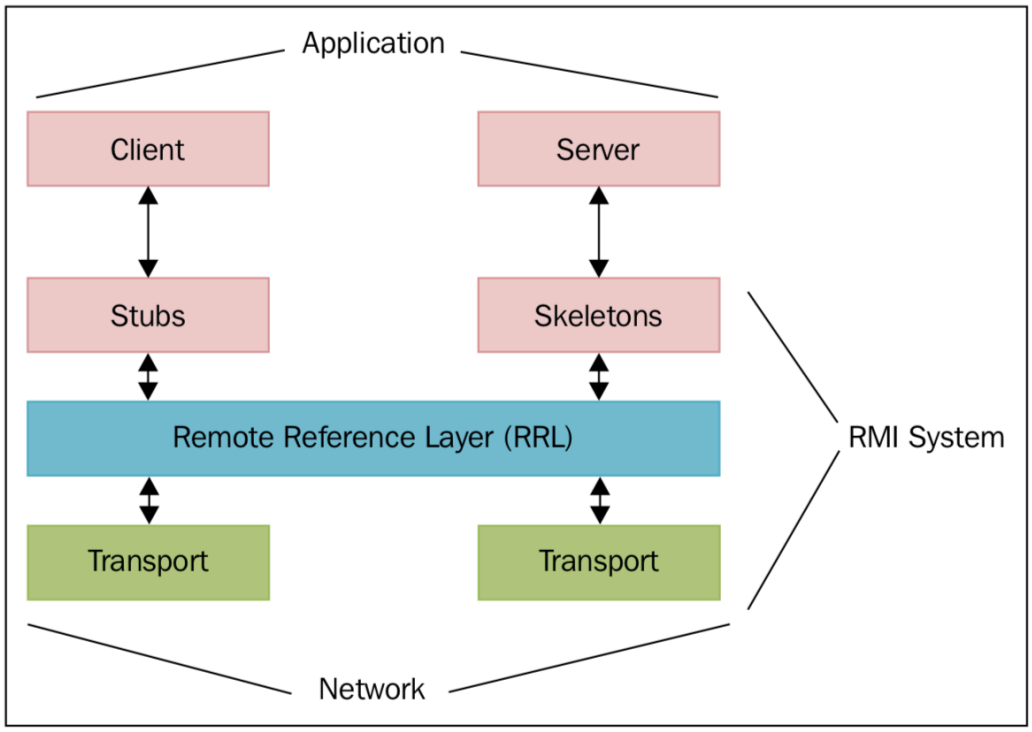
Python MapReduce

```
res = sum(map(simulateWorkload,
              list([a] * a for a in range(1000))))
```

Python map() simulateWorkload list() Reduce Python sum()

5.6 Pyro4

Python Remote Objects (Pyro4) Java Remote Method Invocation, RMI). RMI remote procedure call RPC RMI —
—



Remote Method Invocation

Pyro4 / Pyro4 Pyro4

5.6.1

pip

```
pip install pyro
```

https://github.com/irmen/Pyro4 setup.py

Python3.3 Windows

5.6.2 ...

Pyro4 server

```
import Pyro4

class Server(object):
    def welcomeMessage(self, name):
        return ("Hi welcome " + str (name))

def startServer():
    server = Server()
```

( )

( )

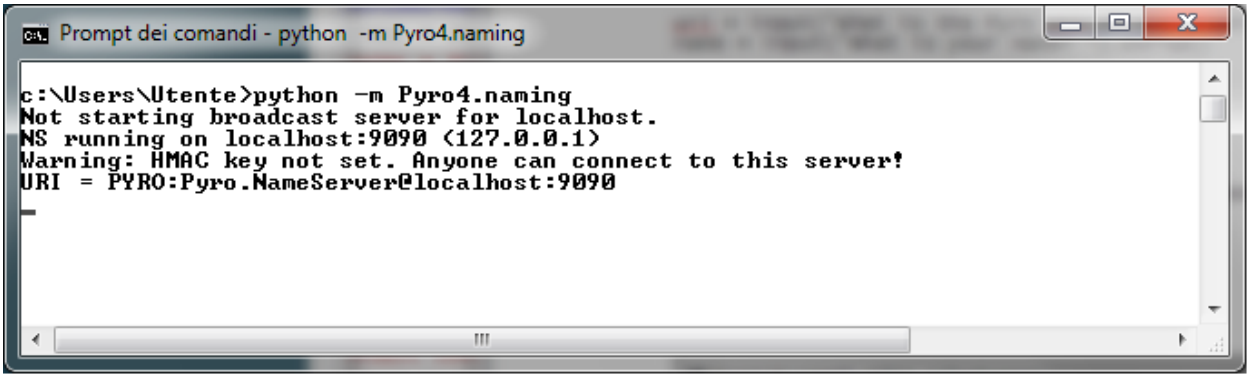
```
    daemon = Pyro4.Daemon()
    ns = Pyro4.locateNS()
    uri = daemon.register(server)
    ns.register("server", uri)
    print("Ready. Object uri =", uri)
    daemon.requestLoop()

if __name__ == "__main__":
    startServer()
```

client.py

```
import Pyro4
uri = input("What is the Pyro uri of the greeting object? ").strip()
name = input("What is your name? ").strip()
server = Pyro4.Proxy("PYRONAME:server")
print(server.welcomeMessage(name))
```

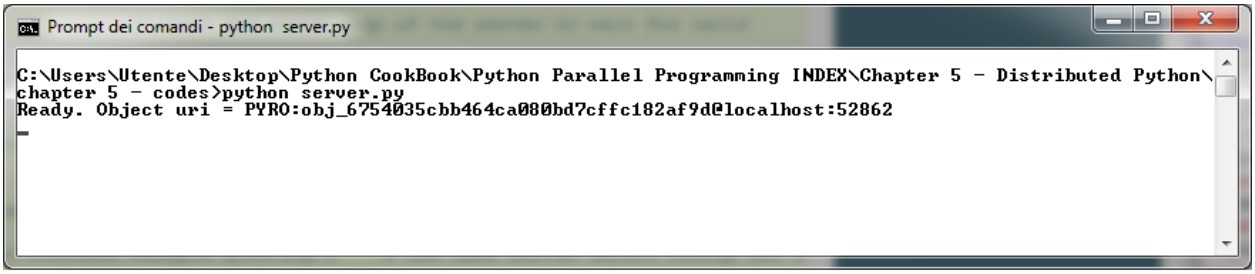
name server

```
python  -m Pyro4.naming
```



name server Server Client Server

```
python server.py
```



```
python client.py
```

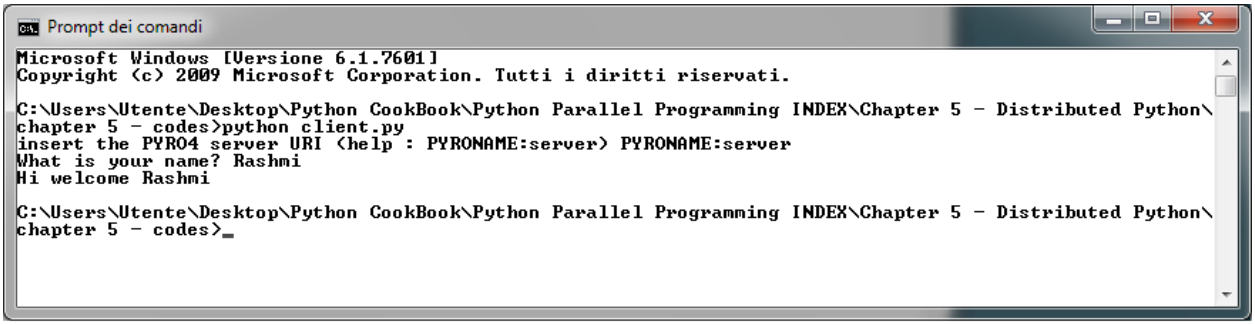
```
insert the PYRO4 server URI (help : PYRONAME:server)
```

Pyro4 PYRONAME: server

```
insert the PYRO4 server URI (help : PYRONAME:server) PYRONAME:server
```

```
What is your name? Rashmi
```

Hi welcome Rashmi .



5.6.3

Server welcomeMessage()

```
class Server(object):
    def welcomeMessage(self, name):
        return ("Hi welcome " + str (name))
```

Server

1. Server server): server = Server()
2. Pyro4 daemon = Pyro4.Daemon() . Pyro4 Server Server
3. name server name server ns = Pyro4.locateNS()
4. server Pyro4 Pyro4 uri = daemon.register(server) . URI
5. name server
6. eventloop , server

Pyro4 API welcomeMessage() Pyro4

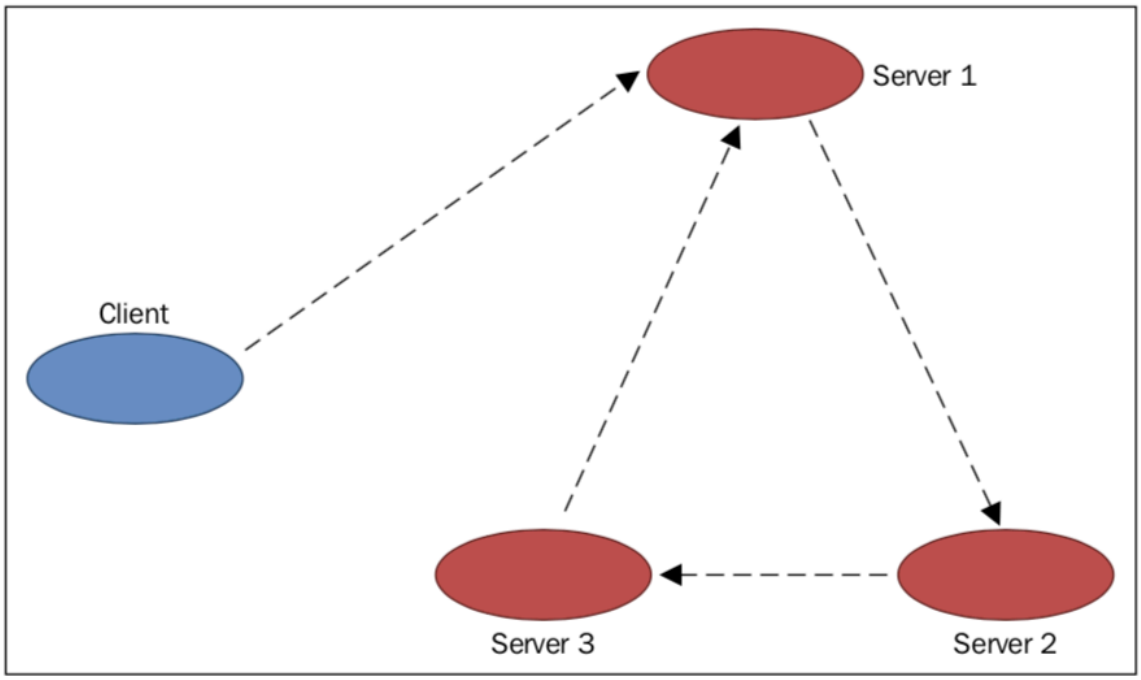
```
server = Pyro4.Proxy("PYRONAME:server")
```

server

```
print(server.welcomeMessage(name))
```

5.7 Pyro4

Pyro4



Chaining an object with Pyro4

Server Server1 Server2 Server3. Server3 Server1

5.7.1 ...

Pyro4 5 Python Client,

```
from __future__ import print_function
import Pyro4
obj = Pyro4.core.Proxy("PYRONAME:example.chain.A")
print("Result=%s" % obj.process(["hello"]))
```

Server Server `this` Server `that`

`server_1.py`

```
from __future__ import print_function
import Pyro4
import chainTopology
this = "1"
next = "2"
servername = "example.chainTopology." + this
daemon = Pyro4.core.Daemon()
obj = chainTopology.Chain(this, next)
uri = daemon.register(obj)
ns = Pyro4.naming.locateNS()
ns.register(servername, uri)
# enter the service loop.
print("server_%s started " % this)
daemon.requestLoop()
```

server_2.py

```
from __future__ import print_function
import Pyro4
import chainTopology
this = "2"
next = "3"
servername = "example.chainTopology." + this
daemon = Pyro4.core.Daemon()
obj = chain.chainTopology(this, next)
uri = daemon.register(obj)
ns = Pyro4.naming.locateNS()
ns.register(servername, uri)
# enter the service loop.
print("server_%s started " % this)
daemon.requestLoop()
```

server_3.py

```
from __future__ import print_function
import Pyro4
import chainTopology
this = "3"
next = "1"
servername = "example.chainTopology." + this
daemon = Pyro4.core.Daemon()
obj = chain.chainTopology(this, next)
uri = daemon.register(obj)
ns = Pyro4.naming.locateNS()
ns.register(servername, uri)
# enter the service loop.
print("server_%s started " % this)
daemon.requestLoop()
```

chain

```
chainTopology.py:
```

( )

( )

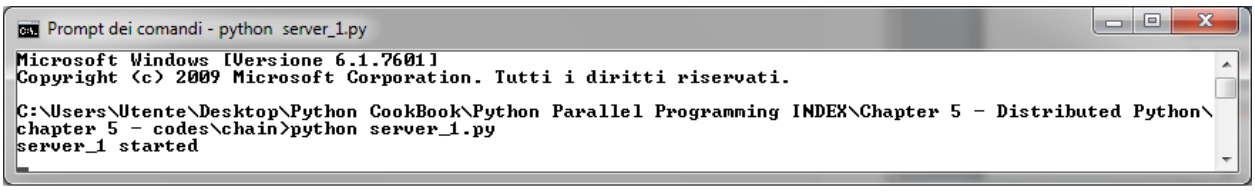
```
from __future__ import print_function
import Pyro4
class Chain(object):
    def __init__(self, name, next):
        self.name = name
        self.nextName = next
        self.next = None
    def process(self, message):
        if self.next is None:
            self.next = Pyro4.core.Proxy("PYRONAME:example.chain." + self.nextName)
        if self.name in message:
            print("Back at %s; the chain is closed!" % self.name)
            return ["complete at " + self.name]
        else:
            print("%s forwarding the message to the object %s" \
                  % (self.name, self.nextName))
            message.append(self.name)
            result = self.next.process(message)
            result.insert(0, "passed on from " + self.name)
            return result
```

Pyro4 name server

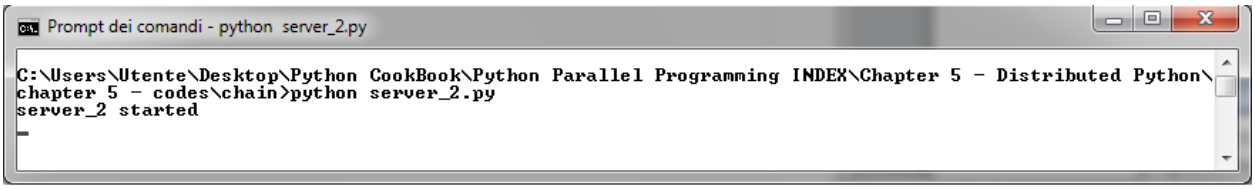
```
C:>python -m Pyro4.naming
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

server python server_name.py

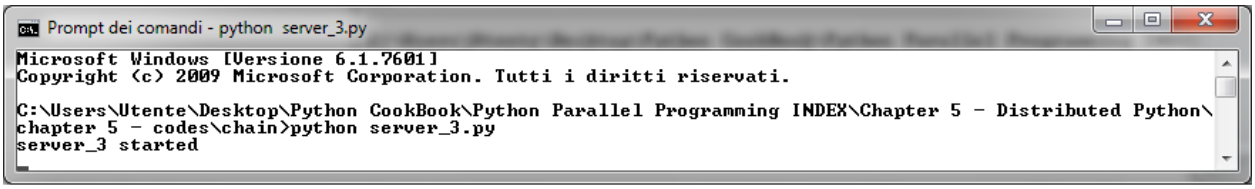
server_1



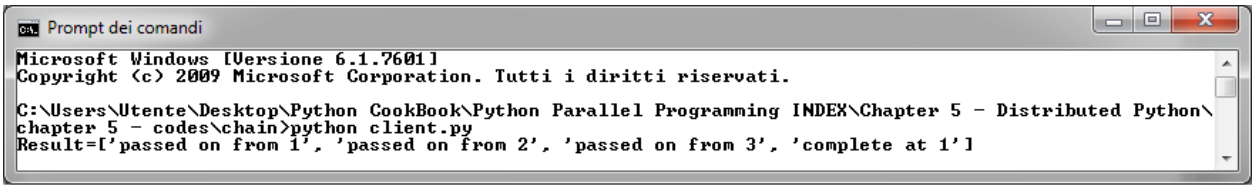
server_2 :



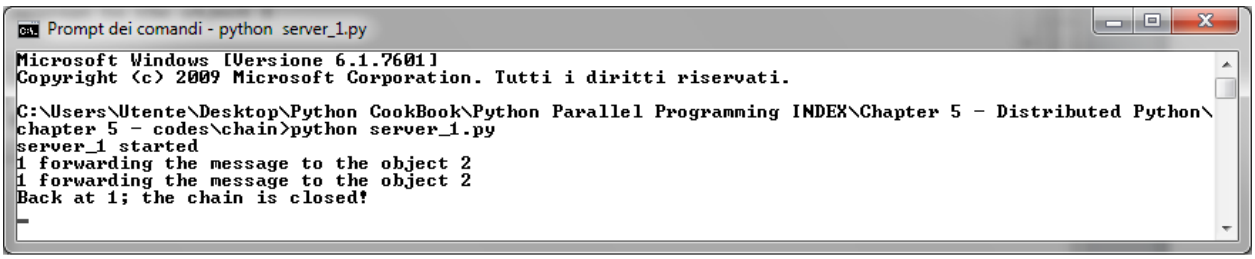
server_3



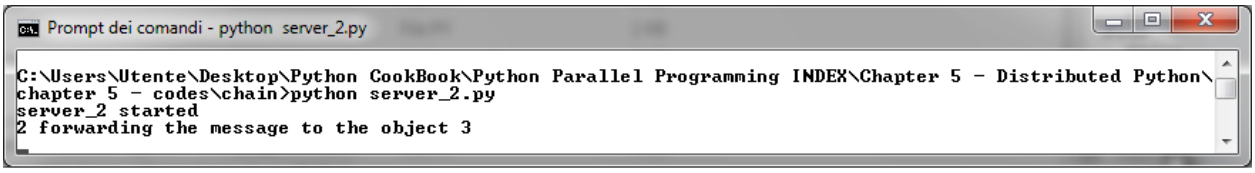
client.py



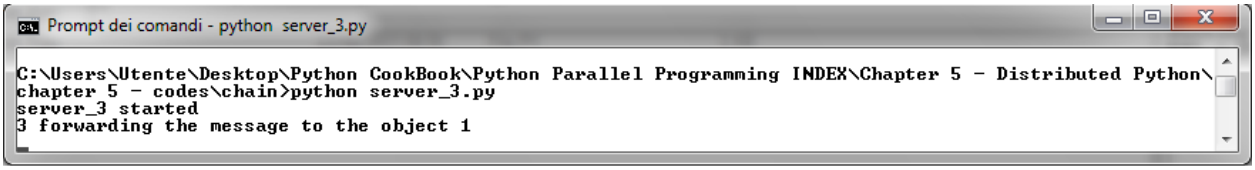
server_1 server_1



server_2



server_3



5.7.2

chainTopology.py Chain server class chainTopology.py process Pyro4.core.proxy

```
if self.next is None:
    self.next = Pyro4.core.Proxy("PYRONAME:example.chainTopology." + self.nextName)
```

server_3 server_1

```
if self.name in message:
    print("Back at %s; the chain is closed!" % self.name)
    return ["complete at " + self.name]
```

```
print("%s forwarding the message to the object %s" \
      % (self.name, self.nextName))
message.append(self.name)
result = self.next.process(message)
result.insert(0, "passed on from " + self.name)
return result
```

Server server_1

```
this = "1"
next = "2"
```

```
servername = "example.chainTopology." + this
daemon = Pyro4.core.Daemon()
obj = chain.chainTopology(this, next)
uri = daemon.register(obj)
ns = Pyro4.naming.locateNS()
ns.register(servername, uri)
# enter the service loop.
print("server_%s started " % this)
daemon.requestLoop()
```

server_1

```
obj = Pyro4.core.Proxy("PYRONAME:example.chainTopology.1")
```

5.8 Pyro4 -

Pyro4 -

- / server server

5.8.1 ...

client shop server .

Server server.py):

```
"""The Shops server"""

from __future__ import print_function
import Pyro4
import shop

ns = Pyro4.naming.locateNS()
daemon = Pyro4.core.Daemon()
uri = daemon.register(shop.Shop())
ns.register("example.shop.Shop", uri)
```

( )

( )

```
print(list(ns.list(prefix="example.shop.").keys()))
daemon.requestLoop()
```

Client (client.py):

```
from __future__ import print_function
import sys
import Pyro4

# A Shop client.
class client(object):
    def __init__(self, name, cash):
        self.name = name
        self.cash = cash

    def doShopping_deposit_cash(self, Shop):
        print("\n*** %s is doing shopping with %s:" % (self.name, Shop.name()))
        print("Log on")
        Shop.logOn(self.name)
        print("Deposit money %s" % self.cash)
        Shop.deposit(self.name, self.cash)
        print("balance=%.2f" % Shop.balance(self.name))
        print("Deposit money %s" % self.cash)
        Shop.deposit(self.name, 50)
        print("balance=%.2f" % Shop.balance(self.name))
        print("Log out")
        Shop.logOut(self.name)

    def doShopping_buying_a_book(self, Shop):
        print("\n*** %s is doing shopping with %s:" % (self.name, Shop.name()))
        print("Log on")
        Shop.logOn(self.name)
        print("Deposit money %s" % self.cash)
        Shop.deposit(self.name, self.cash)
        print("balance=%.2f" % Shop.balance(self.name))
        print("%s is buying a book for %s$" % (self.name, 37))
        Shop.buy(self.name, 37)
        print("Log out")
        Shop.logOut(self.name)


if __name__ == "__main__":
    ns = Pyro4.naming.locateNS()
    uri = ns.lookup("example.shop.Shop")
    print(uri)
    Shop = Pyro4.core.Proxy(uri)
    meeta = client("Meeta", 50)
    rashmi = client("Rashmi", 100)
    rashmi.doShopping_buying_a_book(Shop)
    meeta.doShopping_deposit_cash(Shop)
    print("")
    print("")
```

( )

( )

```
    print("")
    print("")
    print("The accounts in the %s:" % Shop.name())
    accounts = Shop.allAccounts()
    for name in accounts.keys():
        print("  %s : %.2f" % (name, accounts[name]))
```

Shop (shop.py):

```
class Account(object):
    def __init__(self):
        self._balance = 0.0

    def pay(self, price):
        self._balance -= price

    def deposit(self, cash):
        self._balance += cash

    def balance(self):
        return self._balance


class Shop(object):
    def __init__(self):
        self.accounts = {}
        self.clients = ["Meeta", "Rashmi", "John", "Ken"]

    def name(self):
        return "BuyAnythingOnline"

    def logOn(self, name):
        if name in self.clients:
            self.accounts[name] = Account()
        else:
            self.clients.append(name)
            self.accounts[name] = Account()

    def logOut(self, name):
        print("logout %s" % name)

    def deposit(self, name, amount):
        try:
            return self.accounts[name].deposit(amount)
        except KeyError:
            raise KeyError("unknown account")

    def balance(self, name):
        try:
            return self.accounts[name].balance()
        except KeyError:
            raise KeyError("unknown account")
```

( )

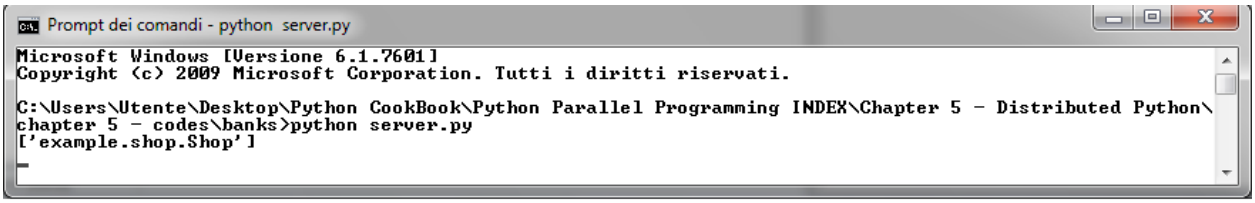
( )

```
    def allAccounts(self):
        accs = {}
        for name in self.accounts.keys():
            accs[name] = self.accounts[name].balance()
        return accs

    def buy(self, name, price):
        balance = self.accounts[name].balance()
        self.accounts[name].pay(price)
```

Pyro4 name server:

```
C:>python -m Pyro4.naming
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

python server.py Server.




```
python client.py
```

```
C:\Users\Utente\Desktop\Python CookBook\Python Parallel Programming
INDEX\Chapter 5 - Distributed Python\
 chapter 5 - codes\banks>python client.py
PYRO:obj_8c4a5b4ae7554c2c9feee5b0113902e0@localhost:59225
*** Rashmi is doing shopping with BuyAnythingOnline:
Log on
Deposit money 100
balance=100.00
Rashmi is buying a book for 37$
Log out
*** Meeta is doing shopping with BuyAnythingOnline:
Log on
Deposit money 50
balance=50.00
Deposit money 50
balance=100.00
Log out
The accounts in the BuyAnythingOnline:
```

( )

( )

```
  Meeta : 100.00
  Rashmi : 63.00
```

Meeta Rashmi .

5.8.2

Shop()

```
ns = Pyro4.naming.locateNS()
```

channel:

```
daemon = Pyro4.core.Daemon()
uri = daemon.register(shop.Shop())
ns.register("example.shop.Shop", uri)
daemon.requestLoop()
```

shop.py shop

```
class Shop(object):
    def logOn(self, name):
        if name in self.clients:
            self.accounts[name] = Account()
        else:
            self.clients.append(name)
            self.accounts[name] = Account()

    def logOut(self, name):
        print("logout %s" % name)

    def deposit(self, name, amount):
        try:
            return self.accounts[name].deposit(amount)
        except KeyError:
            raise KeyError("unknown account")

    def balance(self, name):
        try:
            return self.accounts[name].balance()
        except KeyError:
            raise KeyError("unknown account")

    def buy(self, name, price):
        balance = self.accounts[name].balance()
        self.accounts[name].pay(price)
```

Account

```
class Account(object):
    def __init__(self):
        self._balance = 0.0
```

( )

( )

```
    def pay(self, price):
        self._balance -= price
    def deposit(self, cash):
        self._balance += cash
    def balance(self):
        return self._balance
```

client.py main Rashmi Meeta :

```
meeta = client('Meeta',50)
rashmi = client('Rashmi',100)
rashmi.doShopping_buying_a_book(Shop)
meeta.doShopping_deposit_cash(Shop)
```

Rashmi

```
def doShopping_buying_a_book(self, Shop):
     Shop.logOn(self.name)
     Shop.deposit(self.name, self.cash)
     Shop.buy(self.name,37)
     Shop.logOut(self.name)
```

Meeta $100

```
def doShopping_deposit_cash(self, Shop):
    Shop.logOn(self.name)
    Shop.deposit(self.name, self.cash)
    Shop.deposit(self.name, 50)
    Shop.logOut(self.name)
```

```
print("The accounts in the %s:" % Shop.name())
accounts = Shop.allAccounts()
for name in accounts.keys():
    print("  %s : %.2f" % (name, accounts[name]))
```

5.9 PyCSP

PyCSP Python PyCSP :

-
-
- channels

Channels - -

PyCSP channel : One2One, One2Any, Any2One, Any2Any. readers writers channel

5.9.1

(python-csp

PyCSP pip

```
pip install python-csp
```

Github : https://github.com/futurecore/python-csp .

```
python setup.py install
```

Python2.7 .

5.9.2 ...

PyCSP processes channels. counter printer.

```
# -*- coding: utf-8 -*-
from pycsp.parallel import *

@process
def processCounter(cout, limit):
    for i in xrange(limit):
        cout(i)
    poison(cout)

@process
def processPrinter(cin):
    while True:
        print cin(),

A = Channel('A')

Parallel(
    processCounter(A.writer(), limit=5),
    processPrinter(A.reader())
)

shutdown()
```

Python2.7

```
Python 2.7.9 (default, Dec 10 2014, 12:28:03) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========================RESTART ============================
>>>
0 1 2 3 4
```

5.9.3

// TODO

5.9.4

CSP CSP Occam

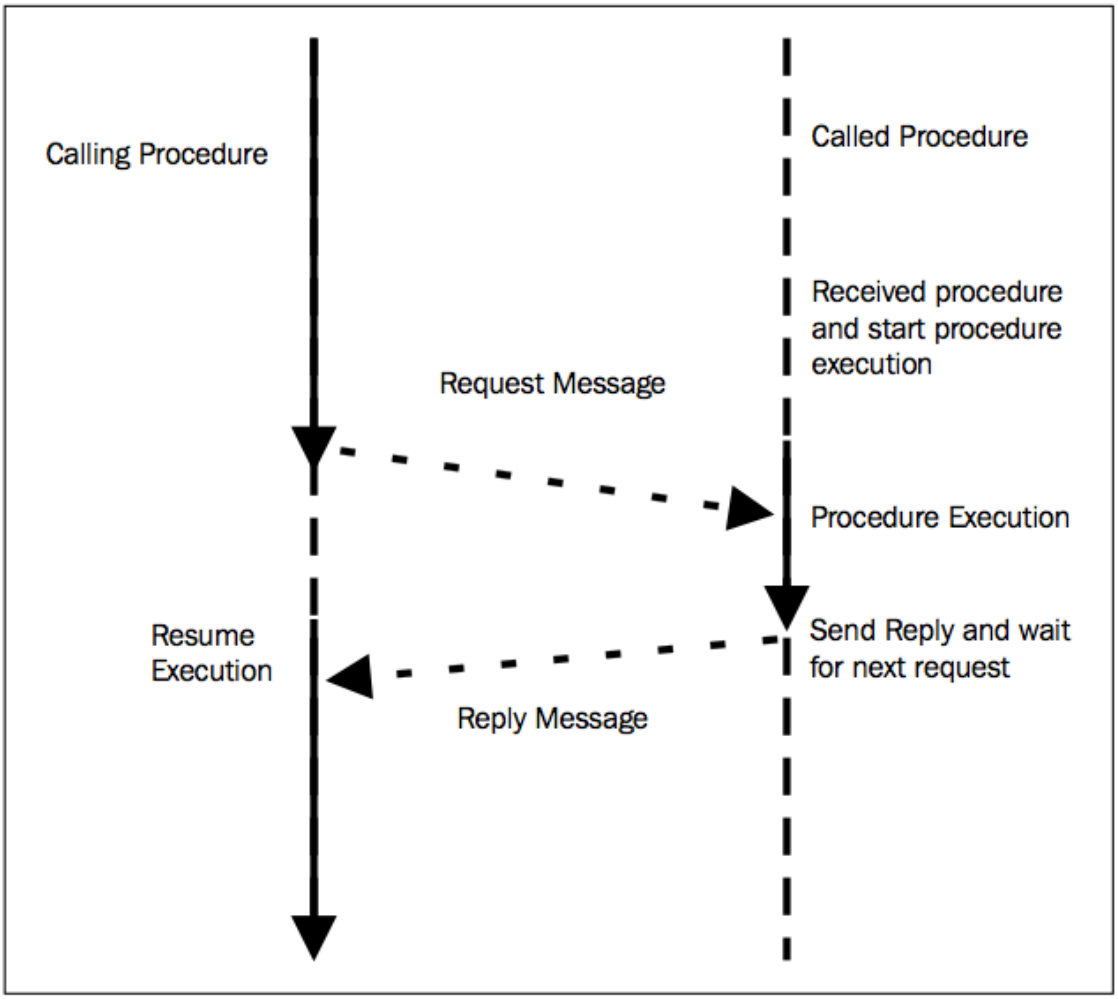
CSP http://www.usingcsp.com/cspbook.pdf .

5.10 Disco MapReduce

5.11 RPyC

Remote Python Call(RPyC) Python RPC ( ) ( )
- (stubs) RPyC

-
-
-
-



The remote procedure call model

5.11.1

pip pip install rpyc

https://github.com/tomerfiliba/rpyc (.zip) python setup.py install

localhost rpyc rpyc ../rpyc-master/bin rpyc_classic.py: python rpyc_classic.py

```
INFO:SLAVE/18812:server started on [0.0.0.0]:18812
```

5.11.2 ...

stdout:

```
import rpyc
import sys
c = rpyc.classic.connect("localhost")
c.execute("print('hi python cookbook')")
c.modules.sys.stdout = sys.stdout
c.execute("print('hi here')")
```

```
INFO:SLAVE/18812:server started on [0.0.0.0]:18812
INFO:SLAVE/18812:accepted 127.0.0.1:6279
INFO:SLAVE/18812:welcome [127.0.0.1]:6279
hi python cookbook
```

(c.modules.sys.stdout = sys.stdout print)

5.11.3

```
import rpyc
c = rpyc.classic.connect("localhost")
```

rpyc.classic.connect(host, port) host port rpyc

c.execute("print('hi python cookbook')")

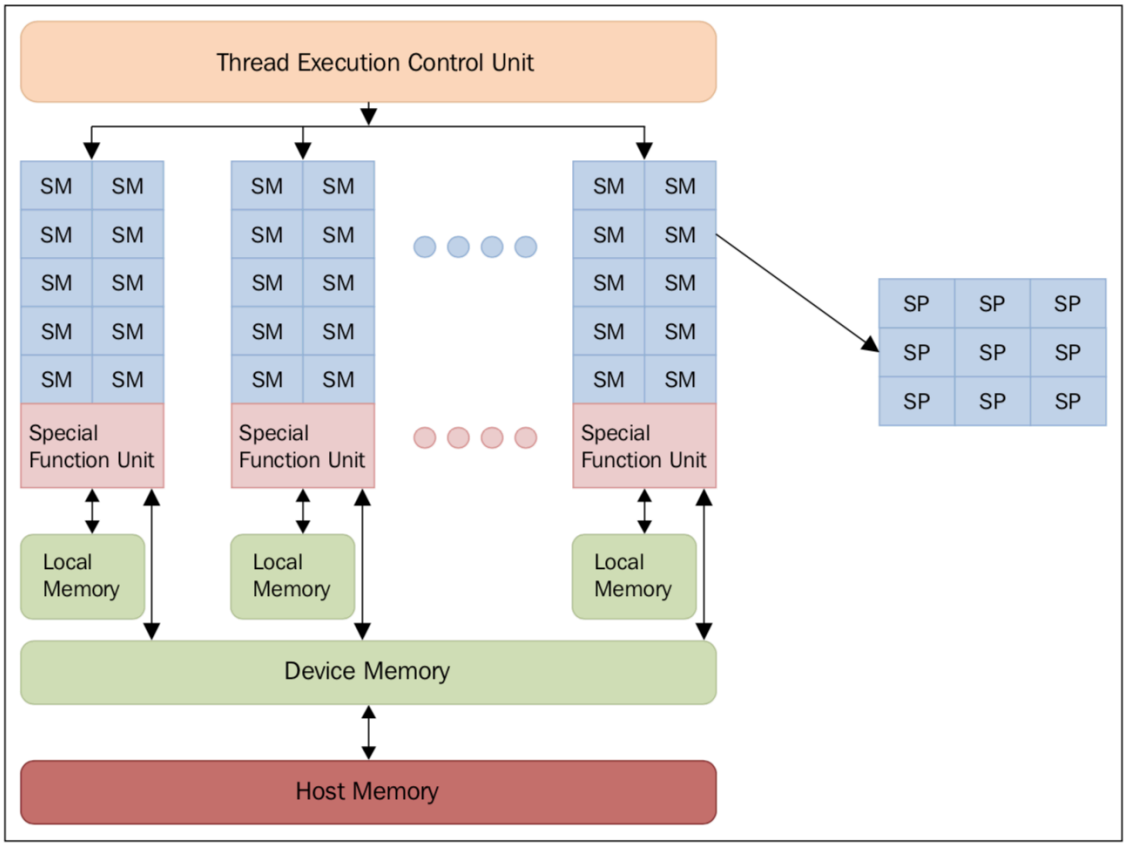
print (exec)

CHAPTER 6

Python GPU

6.1

GPU GPU GPU GPU
GPU GPU GPU
GPU **shaders** GPU
GPU Streaming Multiprocessor SM SM



The GPU architecture

SM (Stream Processors, SP) SP SP SM SP GPU SP SIMD

SM general-purpose computing on graphics processing units, GP-GPU GPU GPU GPU GPU Stream Processing),

GPU CUDA OpenCL Python

6.2 PyCUDA

6.3 PyCUDA

6.4 PyCuDA

6.5 GPUArray

6.6 PyCUDA

6.7 PyCUDA MapReduce

6.8 NumbaPro GPU

6.9 GPU

6.10 PyOpenCL

6.11 PyOpenCL

6.12 PyOpenCL

6.13 PyOpenCL GPU

CHAPTER 7

Indices and tables

- search