```
# Esempio di soluzione del classico prolema dei cinque filosofi a
# cena. La soluzione è quella descritta nel Tanenbaum (AST), a parte
# alcuni cambi docuti alla piattaforma (Linux), alcune aggiunte di
# parti mancanti (es: think, eat e il main!), la parte di "logging"
# per cercare di capire se e come funziona il tutto e.
#
# Inoltre, in questo caso, il linguaggio è Python e non C. Ho comunque
# mantenuto il più possibile la struttura e i nomi della versione AST,
# inclusi i nomi 'down' e 'up' dati alle funzioni che 'inscatolano' i
# relativi metodi Python.  Alla fine il risultato è in generale
# abbastanza 'pythonico' nella sostanza, e la leggibilità è "di un
# altro pianeta".

import threading
import time
import random

N  = 5
def LEFT  (i): return (i-1) % N
def RIGHT (i): return (i+1) % N
THINKING = 0
HUNGRY   = 1
EATING   = 2

mutex = threading.Semaphore(1)
s = [threading.Semaphore(1) for n in range(N)]
state = [THINKING] * N

def down(sem): sem.acquire()
def up(sem):   sem.release()

def sleep_between(min, max):

    time.sleep(min)
    time.sleep(random.random()*(max-min))

def philosopher(i):

    for o in range(10):
        think(i)
        take_forks(i)
        eat(i)
        put_forks(i)

def think(i):

    print("phil %d THINKING" % i)
    sleep_between(0.5, 2.0)

def eat(i):

    print("phil %d EATING" % i)
    print_eaters()
    sleep_between(0.2, 1.0)

def test(i):

    if (state[i]     == HUNGRY and
        state[LEFT(i)]  != EATING and
        state[RIGHT(i)] != EATING):

        state[i] = EATING
        up(s[i])

def take_forks(i):

    down(mutex)
    state[i] = HUNGRY
```

```
    print("phil %d HUNGRY" % i)
    test(i)
    up(mutex)
    down(s[i])

def put_forks(i):

    down(mutex)
    state[i] = THINKING
    test(LEFT(i))
    test(RIGHT(i))
    up(mutex)

def print_eaters():

    state_names = "THE"

    down(mutex);

    ss = [state_names[state[i]] for i in range(N)]
    print("states: %s" % " ".join(ss))
    ss = [str(i)
        for i in range(N)
        if state[i] == EATING]
    c  = len(ss)
    if c > 2:
        print("ERROR: more than one phil eating!")
    if c > 0:
        print("eaters: %s" % " ".join(ss))
    up(mutex);

def main():

    print("MAIN: starting with %d philosophers" % N)

    up(mutex)

    tt = [threading.Thread(target=philosopher, args=(i,))
          for i in range(N)]
    for t in tt:
        t.start()
    for t in tt:
        t.join()

if __name__ == '__main__':

    main()
```