

## List comprehension

```
1 res = [func(i) for i in some_iter if func2(i)]
2 res = [i ** 0.5 for i in [-1, 0, 1, 2, 3] if i >= 0]
3 res = (func(i) for i in some_iter if func2(i))
4 res = {func(i) for i in some_iter if func2(i)}
```

## inline if

```
1 res = x if x >= 0 else -x
2 # res = (x >= 0 ? x : -x)
```

DRY

## Потоки

```
1  import threading
2
3  th = threading.Thread(None, func, None, args, kwargs)
4  th.daemon = True
5
6  th.start()
7
8  th.is_alive()
9  th.join(timeout)
```

## Потоки

- `threading.enumerate`
- `threading.local`

```
1      mydata = threading.local()  
2      mydata.x = 1
```

- `sys.setcheckinterval(N)`

## Потоки

- winpdb
- GIL (cache, .....)
- Не отменяемые (thread2)

```
1      class MyTask( object ):
2          def my_thread_func( self ):
3              pass
4
5          def start_thread( self ):
6              self.th = Thread( None, self.my_thread_func )
```

## Примитивы синхронизации

- `threading.Lock`
- `threading.Semaphore`
- `threading.RLock`
- `threading.Event`
- `threading.Condition`

```
1     lock = threading.Lock()
2     ...
3     # lock.acquire()
4     with lock:
5         pass
6     # lock.release()
```

```
1      cvar = threading.Condition()
2      ...
3      def th1():
4          with cvar:
5              cvar.wait()
6              ...
7
8      def th2():
9          with cvar:
10             cvar.wait()
11             ...
12
13     def th3():
14         with cvar:
15             #cvar.notify()
16             cvar.notify_all()
```

## Queue

```
1  import Queue
2
3  q = Queue.Queue(maxsize=0)
4  q.put(val, block=True, timeout=None)
5  q.get(block=True, timeout=None)
```



## concurrent - python 3.2

```
1  def worker(param_q, result_q, func):
2      while True:
3          param = param_q.get()
4          if param is None:
5              break
6          result_q.put((param, func(param)))
7
8  result_q = Queue.Queue()
9  param_q = Queue.Queue()
10 workers = []
11 worker_params = (param_q, result_q, func)
12
13 for i in range(pool_sz):
14     th = threading.Thread(None, worker,
15                             "worker-{}".format(i),
16                             worker_params)
17     th.daemon = True
18     th.start()
```

```
19         workers.append(th)
20
21     # params_q.put(...)
22     # result_q.get(...)
23
24     for i in range(pool_sz):
25         params_q.put(None)
26
27     for th in workers:
28         th.join()
```

## concurrent - python 3.2

```
1  from concurrent.futures import ThreadPoolExecutor
2  with ThreadPoolExecutor(max_workers=4) as pool:
3      res = pool.map(func, iter)
4      future = pool.submit(func, ....)
5
6      #future.cancel()
7      #future.done()
8      print future.result(timeout=None)
```

multiprocessing