SuperFastPython.com Cheat Sheet for Python threading

Why threading?

Use threads for tasks that perform blocking IO, such as read/write files or socket connections.

Create, Config, Use Thread Objects

Import
```
from threading import *
```

Create, run target function
```
thread = Thread(target=task)
```

Config thread name
```
thread = Thread(name='MyThread')
```

Config daemon thread (background thread)
```
thread = Thread(daemon=True)
```

Extend thread
```
class CustomThread(Thread):
    def run():
        # ...
```

Start thread (non-blocking)
```
thread.start()
```

Join thread, wait to finish (blocking)
```
thread.join()
```

Join thread with timeout
```
thread.join(timeout=5)
```

Check if thread is running (not finished)
```
if thread.is_alive():
    # ...
```

Check if daemon (background)
```
if thread.daemon:
    # ...
```

Access or change thread name
```
thread.name
```

Access thread native identifier
```
thread.native_id
```

Locks and Events

Locks protect critical section, events are safe flags.

Mutex lock
```
lock = Lock()
lock.acquire()
# ...
lock.release()
```

Mutex lock, context manager
```
lock = Lock()
with lock:
    # ...
```

Reentrant mutex lock, protect critical section
```
lock = RLock()
with lock:
    with lock:
        # ...
```

Semaphore, set num positions
```
semaphore = Semaphore(10)
semaphore.acquire()
# ...
semaphore.release()
```

Semaphore, context manager
```
semaphore = Semaphore(10)
with semaphore:
    # ...
```

Create event, then set event
```
event = Event()
event.set()
```

Check if event is set
```
if event.is_set():
    # ...
```

Wait for event to be set (blocking)
```
event.wait()
```

Wait for event with timeout
```
if event.wait(timeout=0.5):
    # ...
```

Condition Variables and Barriers

Conditions for wait/notify, barriers for syncing.

Condition variable
```
condition = Condition()
condition.acquire()
# ...
condition.release()
```

Wait on condition to be notified (blocking)
```
with condition:
    condition.wait()
```

Wait on condition for expression (blocking)
```
with condition:
    condition.wait_for(check)
```

Notify any single thread waiting on condition
```
with condition:
    condition.notify(n=1)
```

Notify all threads waiting on condition
```
with condition:
    condition.notify_all()
```

Barrier, set number of parties
```
barrier = Barrier(5)
```

Arrive and wait at barrier (blocking)
```
barrier.wait()
```

Arrive and wait at barrier with timeout
```
barrier.wait(timeout=0.5)
```

Timer Thread

Wait some time then execute the target function.

Create timer thread
```
tmr = Timer(5, task, args=(a1,a2))
```

Start timer thread
```
thread.start()
```

Cancel timer thread
```
thread.cancel()
```