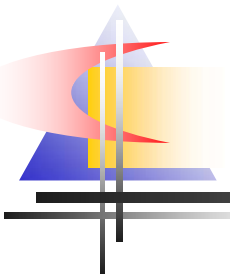
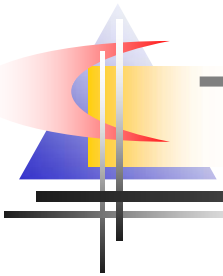


Threads in Python 2.3

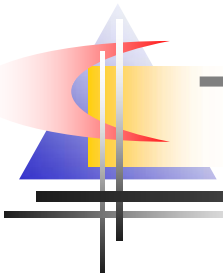
Alex Martelli

with ideas from: Just van Rossum, Guido van Rossum, ...



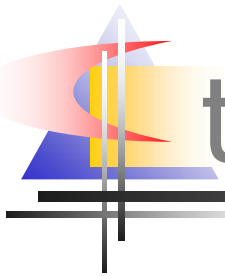
Threads in general

- don't use them!
  - 90% of the time you think you need multiple threads, you're better off with alternatives:
    - async (event-driven) programming
    - multiple *processes*
- but, if you must... (10%)...



Threads in Python

- `Queue`-based architecture
  - 90% of the 10% (9% net) you're best off with `Queue`s
- `RLock`, `Condition`, `Lock`, `Semaphore`, `Event`
  - cover 90% of the rest (0.9% net)
- ...what about the other 0.1%?

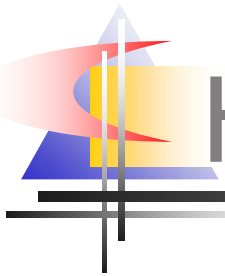


thread.interrupt_main()

- it's in 2.3 since Beta 2
- it comes from ex-IDLEfork (now IDLE)
- it's limited:
  - can't interrupt blocking system calls
  - could cause deadlocks (less likely now!)
- ...but it can still be useful (for 90% of the 0.1%...!-)

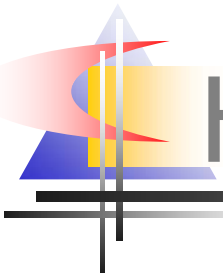
deadlocks are less likely...

```
lock.acquire()
#interrupts now masked here...!
try:
    ...
finally:
    lock.release()
```



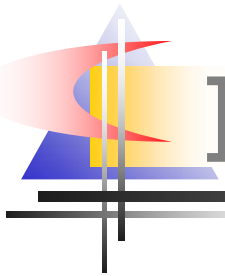
How is interrupt_main useful

- fundamental known use case:
  - make new "monitor" thread
  - run user script in main
  - and now, monitor can interrupt a "runaway" (buggy) user script
- (...in **most** cases...)



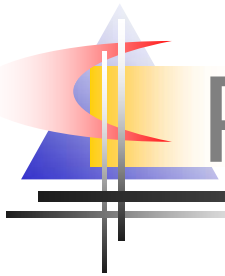
However...

- main thread may be otherwise taken
- e.g., may have to control event-loop
- so -- could we interrupt *other* threads?
- ...pretty please...?-)
  - it covers a whopping 0.009%... :-)
  - we don't want newbies messing with it
  - so: accessible as a **C-API *only*** ("ordeal")



Interrupting other threads

```
if(!PyArg_ParseTuple(args, "iO",
   &threadid, &exceptionClass))
       return 0;
count = PyThreadState_SetAsyncExc(
   threadid, exceptionClass);
if(count > 1) /*we're in trouble!*/
   PyThreadState_SetAsyncExc(
       threadid, NULL);
return Py_BuildValue("i", count);
```



PS: why not multi-process?

- when Python is the embedded scripting language, forking may be impractical
- in some Windows versions making a new process takes "forever" and inter-process control can be hairy and buggy (while threads do work fine there)