# bugs.python.org

- 2007-2013, issue1215, documentation doesn't say that you can't handle C segfaults from python
    - A: module faulthandler可以报告同步错误（自己测试时未成功）
- 20170603, issue30560, Py_SetFatalErrorAbortFunc: Allow embedding program to handle fatal errors
- 20180621, issue33930, Segfault with deep recursion into object().**dir**

# Stackoverflow

## Survey

- Key words and results
    - python performance crash, 151 results,
    - python+performance+crash, 147 results,

## List

### 从C++/C调用Python的问题

- 相关文档：
    - Python C API Reference Manual（python版本3.7.2)
    - Extending and Embedding the Python Interpreter（only describes the general principles of extension writing)
- 问题分类
    - reference count(可能导致内存泄漏)
        - Python C API, High reference count for new Object, 20121025

          new reference：调用者不用这个引用的时候必须显示的调用 `Py_DECREF()` 或者 `Py_XDECREF()` 来释放这个引用。

          borrow reference：函数的调用者只管用这个引用，不用关心它的引用计数，用完了也不用显示调用 `Py_DECREF()` 或者 `Py_XDECREF()` 来释放这个引用。

          steal reference：表示函数内部只会使用这个引用，不会调用 `Py_INCREF` 来增加这个引用的引用计数。
        - Python-C api reference count exceptions, 20120207
    - Threads management

The Python interpreter is not fully thread-safe.

- Segfault when trying to call a Python function from C , 20110223

  management of state of GIL(Global Interpreter Lock)

- Calling python method from C++ (or C) callback , 20140108
- How to call python in parallel from C , 20160329

Python的多线程在多核CPU上，只对于IO密集型计算产生正面效果；而当有至少有一个CPU密集型线程存在，那么多线程效率会由于GIL而大幅下降。为了避开GIL的这种缺陷，目前的方案有：

- 用multiprocess替代Thread
- Reworking the GIL

  将切换颗粒度从基于opcode计数改成基于时间片计数 避免最近一次释放GIL锁的线程再次被立即调度 新增线程优先级功能（高优先级线程可以迫使其他线程释放所持有的GIL锁）

- python --> cython --> c/c++
  - Call python code from c via cython , 20140323

## Python调用C/C++的问题

- 当前比较流行的库：

  使用者对以下库的讨论：

  - ctypes

    Releted issue：How can I use a DLL file from Python? , 20160404

    pros：

    - The resulting code would be pure Python. (neutral)
    - The result does not depend on compiler you are using at all.
    - Release the GIL when calling the native functions.
    - Nice to do simple things and to quickly get something running.

    cons:

    - ctypes won't know about `#define` constants and stuff in the library you're using, only the functions, so you'll have to **redefine** those constants in your own code.
    - **Speed limit**. At some point, you will almost certainly find that you have to call into your C library a lot, either in a loop or in a longer series of interdependent calls, and you would like to speed that up.You can't do that with ctypes. Or, when you need callback functions and you find that your Python callback code becomes a bottleneck, you'd like to speed it up and/or move it down into C as well. Again, you cannot do that with ctypes.
  - Cython

    pros:

        - Expose the relevant parts from the C library to Python.
        - Free to make the wrapping and calling code as thin or thick as you want.
        - Quick speedups of python code (loops and integer comparisons are two areas where cython particularly shines).
        - Great at wrapping numpy.

      cons:

        - Limited by the infrastructure.
        - Bad performance in loops.
    - SWIG

      pros:

        - With the new -builtin option, its performance is improved.
        - You can generate bindings for many scripting languages.

      cons:

        - Type conversions is complicated.
    - Boost Python

      pros:

        - A very complete library. It allows you to do almost everything that is possible with the C-API, but in C++.

      cons:

        - Python libraries created with B.P tend to become obese.
        - Takes a **lot** of time to compile them.
    - CFFI

        - Handles the problem in a fascinating, clean way (as opposed to *ctypes*).
        - Doesn't require to write non Python code (as in *SWIG, Cython*, ...)

- 问题分类：
    - How to catch a fatal error?
        - Py_SetFatalErrorAbortFunc: Allow embedding program to handle fatal errors（python 3.6) (unsolved), 20170603
        - Crash in Python native extension module: Can interpreter fail gracefully? , 20131104

          Q: crash in native code (e.g. segmentation fault) => interpreter crash

          A: The faulthandler module (version 3.3)can print tracebacks on system-level errors.

          Trying to handle the errors from Python code apparently doesn't work.
    - 在做extension时所考虑的因素
        - 人工代码量，修改时对源代码的改动
        - performance: 在不同特点的代码下(bindings / calls into C library / loops）的开销
        - 生成代码的体量
        - 是否便于使用，平台兼容性

## Performance

- 相关文档
  - CrashingPython
    - Bogus Input
    - Exhausting Resources
    - Weakref Interaction
    - Dangerous Modules
    - Multithreading --> segmentation fault
- 问题分类：
  - Python memory profilers , 20170808
    - Heapy(open source, only works for python2, do not support archs) (bad official docs, a better tutorial is this)

      pros:

      - Find out from where objects are referenced and get statistics about that.
      - Works fine in a multithreaded app.

      cons:

      - Doesn't include memory allocated in python extensions.
    - objgraph (python2, python3)
      - GUI
      - Excellent for tracking down memory leaks.
    - Muppy(focus on *memory leaks*)
      - Enables the tracking of memory usage during runtime and the identification of objects which are leaking.
      - Locates the source of not released objects.
      - Gives the memory usage on **all** the running processes (cannot detect memory leak with least memory usage capture).
    - tracemalloc(python3)
      - Provides detailed statistics about which code is allocating the most memory.
      - Memory not allocated by the python interpreter **is not seen** by tracemalloc, such as malloc(this could be switched to `PyMalloc` to have it traced.)
  - How to parallelize python programs
    - Parallelize Python program , 20100413
      - multiprocessing
      - subprocess
      - threading
      - twisted
      - distributed processing

## Python with opencv

- 问题分类
  - cv版本的不同带来的区别

    Performance comparison of OpenCV-Python interfaces, cv and cv2, Use python modules to speed up cv2's performance , 20120220
- pyopencv: a completely new and different approach in wrapping OpenCV from traditional swig-based and ctypes-based approaches. It is intended to be a successor of ctypes-opencvand to provide Python bindings for OpenCV 2.x.

## Python with ROS

- 相关文档：
  - rospy
- 问题分类：
  - Logging with Python, ROS, and C++ ,  how to integrate various loggers into one module? , 20180709
  - Using ROS message classes in another python script outside of ROS , 20160905

## Python with SLAM

- 相关文档
  - 一起做RGBD SLAM
- 问题分类
  - active computer vision
  - 长时间/大规模