1. Q — What is GIL. How does it impact concurrency in Python? What kinds of applications does it impact more than others?

A — GIL stands for Global Interpreter Lock. Implementations of Python like CPython has GIL built into it, some others don't have it. GIL is a mutex that prevents multiple native threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe.(Thread-safe implies the code functions correctly during simultaneous execution by multiple threads.) GIL increases speed of single threaded programs and allows integration of C libraries that are not thread safe. But it doesn't allow python programs to take gull advantage of a multiprocessor environment. Works best with multithreaded programs which are i/o bound. Works worst with multithreaded applications which are bound by CPU. To overcome GIL's performance bottleneck in multiprocessor systems, we can use multiple instances of CPython interpreter process rather than going for multiple threads.

2. Q — How do you iterate over a list and pull element indices at the same time?

A — Enumerate function Example: cities = ["london", "new york", "delhi", "moscow"] for i, city in enumerate(cities): print i, city

3. Q — So what is CPython

A — CPython is the original Python implementation. It is the implementation you download from Python.org. People call it CPython to distinguish it from other, later, Python implementations, and to distinguish the implementation of the language engine from the Python programming language itself.

The latter part is where your confusion comes from; you need to keep Python-the-language separate from whatever runs the Python code.

CPython happens to be implemented in C. That is just an implementation detail really. CPython compiles your python code into bytecode (transparently) and interprets that bytecode in a evaluation loop.

CPython is also the first to implement new features; Python-the-language development uses CPython as the base, other implementations follow.

What about Jython, etc.

Jython, IronPython and PyPy are the current 'other' implementations of the Python programming language; these are implemented in Java, C# and RPython (a subset of Python), respectively. Jython compiles your Python code to Java bytecode, so your Python code can

run on the JVM. IronPython lets you run Python on the Microsoft CLR. And PyPy, being implemented in (a subset of) Python, lets you run Python code faster than CPython.

Actually compiling to C

So CPython does not translate your Python code to C by itself. It instead runs a interpreter loop. There is a project that does translate Python-ish code to C, and that is called Cython. Cython adds a few extensions to the Python language, and lets you compile your code to C extensions, code that plugs into the CPython interpreter.