Multi-Threading: The Unknown Truth of Python

- **My Bio:** Senior Python Developer at Xoriant Solutions, Mumbai with 9+ years of experience in Embedded, Mobile and Web Applications. Have years of implementation experience on various Python Modules, Libraries and Tools.

- **Proposal Type (Category):** About experiences and usage of Python and Python-based tools and libraries for research or teaching.

- **Technical Level:** Intermediate

- **Pre-Requisites:** Knowledge of Python Language (2.7 and above), Understanding of Threads and Cores.

- **Abstract:**

Performance is one of the most important aspects of any application.
But **"How to achieve it"** is an **"Answer"** we look for. This is where **"Multi-Threading"** comes into picture.
Like any other language, Python also supports Multi-Threading **"but"** before you consider this feature to achieve improved performance of your application **"Think Twice",** yes you read it right **"Think Thrice"** and **WHY** is what I will be explaining in my Proposal/Paper/Presentation.

Outline:

1. What will you learn from this Proposal:
    - The **"Concurrency Concept"** and its relation to Multi-Threading.
    - The **"GIL" – Global Interpreter Lock** concept and the mystery behind it.
    - How GIL limits thread performance.
    - **"Where"** and **"Why"** not to use Multi-Threading – The hidden truth of Python Multi-Threading.
    - What is an alternative to it? – A brief overview of "Multi-Processing"

2. Why do you need to know this:
    - Will help you in : **"Decision making"** , **"Time Saving"** , **"Low Project Cost"** , **"Project Performance"** and **"HOW"** - Next time when you consider Multi-Threading as an option for improving system performance , you know beforehand exactly **why / why not** to use it.

3. The Case Study - **CPU Bound and I/O bound task**
    - How CPU bound task effects performance
    - How I/O bound task improves performance

4. The Real Life Project Implementation:
   CPU Bound and I/O bound task (one for each) - I will show you how in our project we improved application performance by over 30-40% (Approximately)
    - A sample implementation using "threading" module

- Time comparison using 1 and multiple threads.

Details:

- Multithreaded Application to download the huge historical data files (csv format in GB's) from a website, read the files, do some slicing and dicing on the data and dump in the database.
- **Analysis Processing Time:**
  - When the multithreaded code had only download functionality implemented (I/O Task):
    - **Single Thread : 15 Seconds**
    - **3 Threads : 9 Seconds**
  - When the multithreaded code had data processing/formatting functionality (CPU Bound Task) along with download functionality implemented (I/O Task):
    - **Single Thread : 15 Seconds**
    - **3 Threads : 30 Seconds**

5. The before and after situation:
   - How the same code takes more time when made multi-threaded.

6. The "**Common Mistakes"** People make and how it can be avoided.
   - Explanation to the common mistakes made in problem identification while making an application Multi-Threaded and how to avoid it.

7. Q&A