Advanced Python

Syllabus

- **Python Review**
  - A brief review of Python basics including syntax, core datatypes, file I/O, functions, error handling, and classes.

- **Classes and Objects**
  - An introduction to creating and using user-defined objects in Python.
  - Describes how to use the class statement to create new objects and presents details on various special methods that can be defined to customize object behavior.
  - Commonly used object oriented programming techniques are also presented.

- **Advanced I/O Handling**
  - An in-depth examination of the Python I/O system including text handling, binary data handling, and different I/O models such as blocking, non-blocking, and event-driven I/O.

- **Idiomatic Data Handling**
  - An introduction to various tools and techniques for effective data processing.
  - Discusses different options for creating data structures and gives an inside look at how the built-in datatypes are put together along with their performance and memory usage properties.
  - Students will learn how to apply list, set, and dictionary comprehensions to various problems in data handling.

- **Inside the Python Object Model**
  - A detailed tour of how the Python object system is implemented.
  - Topics include the definition of objects, object representation, attribute binding, inheritance, descriptors, properties, slots, private attributes, static methods, and class methods.
• Cover important details concerning Python memory management and garbage collection.

• Testing, Debugging and Logging
  o Coverage of how to test and debug Python programs with a focus on three major topics.
  o First, testing Python programs with the doctest and unittest modules is described.
  o Next, the Python Debugger and profiler are presented.
  o Finally, the logging package is described.

• Packages and Distribution
  o How to organize Python code into packages and how to distribute packages to users and programmers.
  o Covers the underlying mechanics of how packages are put together and the distutils module for creating distributions.
  o Also covers more advanced packaging options such as Distribute and setup tools.

• Iterators, Generators, Coroutines
  o The section starts with a description of the iteration protocol and moves on to practical use of generators and coroutines.
  o A major focus on this section is on the use of generators and coroutines to set up processing pipelines, much like pipes in Unix programming.
  o You will see how generators and coroutines can lead to very elegant programming abstractions for processing data and how such programs can be used to process huge datafiles and streaming I/O.

• Functional Programming
  o Advanced details of how to program with functions in Python.
  o Discusses more advanced features of functions including variadic parameters, nested functions, closures, lazy evaluation, anonymous functions (lambda), and function attributes.

• Metaprogramming
  o Loosely defined, metaprogramming refers to programs that are able to manipulate their own program structure (functions, classes, etc.) or the structure of other programs as data.
  o This section introduces and covers practical examples of Python's metaprogramming features including function decorators, class decorators, metaclasses, and context managers.
A major emphasis of this section is to understand how advanced programming frameworks utilize these features to provide a richer programming environment for their end users.

- **Extension Programming**
  - How to create C and C++ extensions to Python. Covers the absolute basics of the Python C API followed by some details on using the ctypes library and Swig code generator.
  - A major focus of this section is on how to organize extension code so that it can more seamlessly integrate with the Python environment.
  - Topics include memory management, data handling, encapsulation, and common pitfalls.

- **Concurrent Programming with Threads**
  - An introduction to programming with Python threads.
  - Starts with the basics of using the threading library and dives into a variety of more advanced topics including a survey of how and when to use the different thread synchronization primitives, queues, deadlock avoidance, and thread debugging.
  - Also includes detailed information on how the Python interpreter executes programs and properties of the global interpreter lock (GIL).

- **Multiprocessing**
  - An introduction to the multiprocessing library that allows programs to distribute their work across independent processes or machines.
  - Covers processes, queues, pipes, connections and process pools.

- **Python 3**
  - A look at new features added to Python 3 and migration strategies.
  - Topics include new data processing features such a set and dictionary comprehensions, function annotations, and metaclasses.
  - In addition, details of the 2to3 conversion tool are presented.