Python for Big Data

Why study Python?

Python is a powerful, flexible, open-source language that is easy to learn, easy to use, and has powerful libraries for data manipulation and analysis. Its simple syntax is very accessible to programming novices, and will look familiar to anyone with experience in Matlab, C/C++, Java, or Visual Basic. Python has a unique combination of being both a capable general-purpose programming language as well as being easy to use for analytical and quantitative computing.

For over a decade, Python has been used in scientific computing and highly quantitative domains such as finance, oil and gas, physics, and signal processing. It has been used to improve Space Shuttle mission design, process images from the Hubble Space Telescope, and was instrumental in orchestrating the physics experiments which led to the discovery of the Higgs Boson (the so-called “God particle”).

At the same time, Python has been used to build massively scalable web applications like YouTube, and has powered much of Google’s internal infrastructure. Companies like Disney, Sony Dreamworks, and Lucasfilm ILM rely heavily on Python to coordinate massive clusters of computer graphics servers to produce the imagery for blockbuster movies. According to the TIOBE index, Python is one of the most popular programming languages in the world, ranking higher than Perl, Ruby, and JavaScript by a wide margin.

Python can be as powerful and successful for big data and business data analytics as it has been for science, engineering, and scalable computing. Python is easy for analysts to learn and use, but powerful enough to tackle even the most difficult problems in virtually any domain. It integrates well with existing IT infrastructure, and is very platform independent. Among modern languages, the agility and productivity of Python-based solutions are legendary. Companies of all sizes and in all areas — from the biggest investment banks to the smallest social/mobile web app startups — are using Python to run their business and manage their data.

Course Objectives

This course is designed to cover key concepts for writing Python code, emphasizing the design of functional and efficient code. It will set students down the road to mastering the intricacies of Python. After completing the course, students should be able to read, understand, modify, and create complex functions to perform a variety of tasks.
## Program Outline

<table>
<thead>
<tr>
<th>Day</th>
<th>TOPICS</th>
<th>ASSIGNMENTS</th>
<th>DURATION</th>
</tr>
</thead>
</table>
| **Day 1** Introduction to Python | • Installing Python  
  • Python development workflow, Python REPL and running Python files directly.  
  • Introduction to basic data types, control structures and conditions.  
  • Functions, error handling and exceptions  
  • Command line programs | • Simple programs involving basic python language constructs | 8 hours  |
| **Day 2** More Python, useful Python libraries, etc. | • Python collections such as tuples, lists, dictionaries and sets.  
  • List comprehensions, decorators  
  • Modules and libraries  
  • Object-oriented programming in Python  
  • Useful Python libraries | • Command-line utility for counting unique words in a given set of files | 8 hours  |
| **Day 3** Working with different data formats: CSV, JSON, XML | • Working with files  
  • Handling large files  
  • Python libraries for reading and writing CSV, JSON, XML and other data formats.  
  • Serializing Python data structures | • Processing data files in a given format, normalizing the data and making it available in a different data format. E.g. XML to CSV | 8 hours  |
| **Day 4** Getting data from APIs and databases | • Introduction to ‘requests’ library  
  • Connecting to and downloading data from a given API  
  • Reading web pages using Python  
  • Working with databases using Python | • Accessing Twitter API to explore the social web | 8 hours  |
| **Day 5** Cleaning messy data, dealing with missing values, scraping the web, regular expressions | • Finding and extracting raw data  
  • Tidy data and how to make data tidy, reshaping and transforming data  
  • Practical implementation through a range of Python libraries: Textual data, dates, etc  
  • Fetching web pages, extracting data from them and making the data available in various formats.  
  • Processing data in parallel. | • Web scraping | 8 hours  |