FACULTY OF COMPUTER SCIENCE
Master of Computer Application (Sem-IV)
In Effect from Academic Year 2017-18

Subject Code : 1CS2010405	**Subject Title: ADVANCED PYTHON**
Pre-requisite :	Basic Python Programming

Course Objective:

The objectives of the course are to:

- To be able to understand the various regular expressions available in Python programming language and apply them
- To understand the advanced concepts of text processing, database programming, multithreading and extension
- To understand the concept of Web application development
- To be able to use extension for creating applications
- To understand python based web application framework like Django

Teaching Scheme (Hours per week)				Evaluation Scheme (Marks)				
				Theory		Practical		
Lecture	Tutorial	Practical	Credit	University Assessment	Continuous Assessment	University Assessment	Continuous Assessment	Total
4	-	3	7	60	40	30	20	150

Subject Contents			
Sr. No	**Topic**	**Total Hours**	**Weight (%)**
1	**Regular Expressions and Text Processing** **Regular Expressions:** Special Symbols and Characters, Regexes and Python, A Longer Regex example (like Data Generators, matching a string etc.) **Text Processing:** Comma Sepearated values,JavaScript Object Notation (JSON),Python and XML **Case Study:** Create Regular expressions (Custom), Process telephone numbers, Generate log data, HTML Generators, Tweet Scrub, Amazone ScreenScrapper, Mailmerge	09	20
2	**Advanced Python Programming** **Multithreded Programming :** Threads and Pythong, Thread and threading module, Single thread and Multithreaded execution, Multithreading example. **Database Programming:** Databases and Python, The Python DB-API, Python and ORMs, Non-Relational Databases **Module Extension:** Extending Python by writing extensions **Case Study:** Create Library/Module for Language Processing	12	25
3	**Web Development** **Web Clients and Servers;** Python web Client tools, Web (HTTP) servers and Related Modules **Web Application Programming:** Helping web servers processing client data, Building CGI applications (Creating form page, Generating Result Page, Fully interactive web sites) Advanced CGI (like Multi part form submission, File upload, Cookies), Introduction to WSGI, Real world Web development	12	25
4	**Python and Data Analytics** Understand the problem By Understanding the Data Predictive Model Building: Balancing Performance, Complexity, and the Big Data	10	20
5	**Web Framework : Django**	05	10

Course Outcome:

At the end of this course, the student would be able understand advanced programming concept of Python programming like Text processing, web application development, multithreading and machine learning.

List of References:

1. Wesley J Chun, Core Python Applications Programming, 3rd Edition.Pearson
2. Michael Bowles, Machine Leaning in Python, Esssential techniques for predictive analysis, Wiley
3. Mark Pilgrim, Dive into Python: Python Novice to pro(source: http://diveintopython.org/.)
4. Alex Martelli, Python Cookbook, O'REILLY
5. Luke Sneeringer, Professional Python, WROX
6. Laura Cassell, Python Projects, WROX

Web Resources

1. http://docs.python.org/library/csv
2. http://docs.python.org/library/json
3. http://docs.python.org/library/ext
4. http://en.wikibooks.org/wiki/Python_Programming
5. http://learnpythonthehardway.org/
6. http://jason.org
7. Nosql-database.org
8. www.mongodb.org/

Practical List

1. Create Regular Expressions that
    a. Recongnize following strings bit, but, bat, hit, hat or hut
    b. Match any pair of words separated by a single space, that is, first and last names.
    c. Match any word and single letter separated by a comma and single space, as in last name, first initial.
    d. Match simple Web domain names that begin with www. and end with a ".com" suffix; for example, www.yahoo.com. Extra Credit: If your regex also supports other high-level domain names, such as .edu, .net, etc. (for example, www.foothill.edu).
    e. Match a street address according to your local format (keep your regex general enough to match any number of street words, including the type designation). For example, American street addresses use the format: 1180 Bordeaux Drive. Make your regex flexible enough to support multi-word street names such as: 3120 De la Cruz Boulevard.
2. Create Regular Expressions That:
    a. Extract the complete timestamps from each line.
    b. Extract the complete e-mail address from each line.
    c. Extract only the months from the timestamps.
    d. Extract only the years from the timestamps.
    e. Extract only the time (HH:MM:SS) from the timestamps.
3. Create utility script to process telephone numbers such that
    a. Area codes (the first set of three-digits and the accompanying hyphen) are optional, that is, your regex should match both 800-555-1212 as well as just 555-1212.
    b. Either parenthesized or hyphenated area codes are supported, not to mention optional; make your regex match 800-555-1212, 555-1212, and also (800) 555-1212.
4. Create utility scripts when processing online data:
    a. HTML Generation. Given a list of links (and optional short description), whether user- provided on command-line, via input from another script, or from a database, generate a Web page (.html) that includes all links as hypertext anchors, which upon viewing in a Web browser, allows users to click those links and visit the corresponding site. If the short description is provided, use that as the hypertext instead of the URL.
    b. Tweet Scrub. Sometimes all you want to see is the plain text of a tweet as posted to the Twitter service by users. Create a function that takes a tweet and an optional "meta" flag defaulted False, and then returns a string of the scrubbed tweet, removing all the extraneous information, such as

an "RT" notation for "retweet", a leading ., and all "#hashtags". If the meta flag is True, then also return a dict containing the metadata. This can include a key "RT," whose value is a tuple of strings of users who retweeted the message, and/or a key "hashtags" with a tuple of the hashtags. If the values don't exist (empty tuples), then don't even bother creating a key-value entry for them.

c. Amazon Screenscraper. Create a script that helps you to keep track of your favorite books and how they're doing on Amazon (or any other online bookseller that tracks book. rankings). For example, the Amazon link for any book is of the format, http://amazon.com/dp/ISBN (for example, http://amazon.com/dp/0132678209). You can then change the domain name to check out the equivalent rankings on Amazon sites in other countries, such as Germany (.de), France (.fr), Japan (.jp), China (.cn), and the UK (.co.uk). Use regular expressions or a markup parser, such as BeautifulSoup, lxml, or html5lib to parse the ranking, and then let the user pass in a commandline argument that specifies whether the output should be in plain text, perhaps for inclusion in an e-mail body, or formatted in HTML for Web consumption.

5. Process Result.txt file having data in the format (Format: EnrollmaentNumber, Name, Sub1_marks, Sub2_Marks, Sub3_marks,sub4_Marks) and generate analysis in following format
   a. Print Marksheet (Design your own format)
   b. Generate Student Summary (Enrollment Number, Name, Marks Obtained, Pass/Fail)
   c. Grade wise Summary
6. Process large mailbox.txt file having all email messages. Using regular expressions recognize all email addresses and URLs and save them into links.html. Use threads to segregate the conversion process.
7. Create a set of threads to count howmany lines are there in a set of text files.
8. Create online registration Web page for Youth festival. Apply Database CRUD operations.
9. Create an extension or module library in Python to implement Calculator.
10. Take any C code you have created, make it an extension module.
11. Create Web Database Application "Address Book" with options to add, modify, view and delete entry option