lecture 8a program design

600152 people data and the web
This mini project provides a very high level description of how a search engine works.

Constructing a search engine can be divided into two parts: indexing and querying. In indexing, a program builds an index, which maps a set of keywords to the web pages where the keywords appears. Querying allows a user to send keyword(s) of interest to a query program that would use the index to return a list of web pages in which the keyword(s) exist.

Your task in this project can also be divided into two stages:
1. building the index
2. building a web-based query interface and its query handler.

http://docs.google.com/Doc?docid=0AaApdsmRE7lrZGZkNTdtNWRfNDc3M3F0dDNjOQ
We want to be able to search for keyword in title and body separately.
indexing xhtml pages
planning for your program

for every web page:
  load the web page as an xml file
  (thank god for xhtml!)
  get the text content of all elements under <body>
  break down the text into list of words
for every word:
  normalise each word
  add the word into the index

pickle the index
querying/searching xhtml pages

querying program

query_title = "two"
query_body = "rabbit"

HTML form

Web Browser

Search results: - page2.html

index_title.pickle

index_body.pickle
visual prototype

HTML Form

Enter your query:

- [ ] in title.
- [ ] in body.

[Submit Query] [Reset]

HTML Response

Search results for 'paragraph' in title and 'rabbit' in body:

- [page2.html]
- [page3.html]
version 0

• input and output specification
  • indexing:
    • input: pagelist.xml & a set of HTMLs
    • output: two indexes
  • querying:
    • input: body and title keywords, indexes
    • output: results found in index
version 0

• start small
  • z_make_index.py:
    test indexing for a single web page

• unit testing
  • z_query_test.py:
    emulate user input, test several cases
unit testing

the verification and validation of an individual unit of code, which is the smallest testable parts of a computer program.

the objective of unit testing is:
• to isolate and test each part of the code
• to support several test cases
• to find code bugs early in the development
version 1

- **refactoring** of version 0

- remove code duplication and merge into function
  - `index_xxxx()`

- progressive enhancement
  - slice and normalise
refactoring

the process of improving structural properties of a computer program without modifying its functional behaviour, for example:

• enhancing code readability
• removing duplicate code (don’t repeat yourself)
• divide and conquer, breaking down a large function in smaller pieces

version 2

- modularity
  - creating python module m_xxxxx.py

- encapsulation
  - m_index.py

- information hiding
  - m_text_utils.py

modularity

the decomposition of a program into a number of smaller parts (modules) that can be independently created and may be reused in various contexts. The main objective is to improve reusability. For example: Python modules

encapsulation

the wrapping up of data and functions into a single entity. An encapsulation is often an abstraction of a real world object or a collection of logically related data and functionalities. Encapsulation is closely related to information hiding.

Encapsulation is an important aspect of Object-Oriented Programming

information hiding

the hiding of the internal logic and data structures within a piece of code behind a specific component (achieved using modularity and encapsulation)

the objective of information hiding is:
• to hide the underlying complexity
• to achieve transparency, the internal work of the component can be changed without impacting the users of the component
version 3 (querying)

• separation of concerns
  • query_form.html
  • query_result.html

• query_handler.py is a merge of query_result.html and z_query_test.py
separation of concerns

the process of breaking a computer program into high level features that have minimum overlap. Working on a specific concern means "focusing one's attention upon some aspect" (Dijkstra, 1974). For example, XHTML and CSS: separation between semantic and presentation.

http://www.cs.utexas.edu/~EWD/ewd04xx/EWD447.PDF
version 4 (querying)

- further refactoring (index_load and index_get)
- make use of the earlier m_index.py