

Python Training

A basic overview

Database Connectivity

- SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language.
- Python has support for sqlite3 by default
- Support for :
 - Cursors
 - Exception handling e.g. OperationalError, IntegrityError etc
- Demo
 - Connecting to a database
 - CREATE
 - INSERT
 - SELECT
 - DELETE
 - DROP

Regular Expressions

- Regular expressions are a powerful language for matching text patterns.
- The Python "re" module provides regular expression support.
 - `import re`
- Basic Patterns
 - `a, X, 9, <` -- ordinary characters just match themselves exactly.. `(a period)` -- matches any single character except newline `'\n'`
 - `\w` -- (lowercase w) matches a "word" character: a letter or digit or underbar `[a-zA-Z0-9_]`. Note that although "word" is the mnemonic for this, it only matches a single word char, not a whole word. `\W` (upper case W) matches any non-word character.
 - `\b` -- boundary between word and non-word
 - `\s` -- (lowercase s) matches a single whitespace character -- space, newline, return, tab, form `[\n\r\t\f]`. `\S` (upper case S) matches any non-whitespace character.
 - `\t, \n, \r` -- tab, newline, return
 - `\d` -- decimal digit `[0-9]` (some older regex utilities do not support but `\d`, but they all support `\w` and `\s`)
 - `^` = start, `$` = end -- match the start or end of the string
 - `\` -- inhibit the "specialness" of a character. So, for example, use `\.` to match a period or `\\` to match a slash. If you are unsure if a character has special meaning, such as '@', you can put a slash in front of it, `\@`, to make sure it is treated just as a character.

Regular Expressions

■ Available functions in re module

<code>re.search(pattern, string, flags=0)</code>	Search pattern in entire string and return a MatchObject of first match
<code>re.match(pattern, string, flags=0)</code>	Search pattern only at start of string and return a MatchObject if found
<code>re.findall(pattern, string, flags=0)</code>	Search pattern in entire string and return a list of all matches
<code>re.finditer(pattern, string, flags=0)</code>	Search pattern in entire string and return a list of all matches as MatchObjects.
<code>re.compile(pattern, flags=0)</code>	Compile a regular expression pattern into a regular expression object, which can be used for matching using its <code>match()</code> and <code>search()</code> methods
<code>re.sub(pattern, repl, string, count=0, flags=0)</code>	Return the string obtained by replacing the occurrences of pattern in string by the replacement <code>repl</code> .

Regular Expressions

■ Regular-expression Modifiers - Option Flags

Modifier	Description
re.I	Performs case-insensitive matching.
re.L	Interprets words according to the current locale. This interpretation affects the alphabetic group (<code>\w</code> and <code>\W</code>), as well as word boundary behavior (<code>\b</code> and <code>\B</code>).
re.M	Makes <code>\$</code> match the end of a line (not just the end of the string) and makes <code>^</code> match the start of any line (not just the start of the string).
re.S	Makes a period (dot) match any character, including a newline.
re.U	Interprets letters according to the Unicode character set. This flag affects the behavior of <code>\w</code> , <code>\W</code> , <code>\b</code> , <code>\B</code> .
re.X	Permits "cuter" regular expression syntax. It ignores whitespace (except inside a set <code>[]</code> or when escaped by a backslash) and treats unescaped <code>#</code> as a comment marker.