Python for Web

Python

- Language features
  - Good readability
  - Very expressive
  - General purpose language
  - Object-oriented support
  - Dynamic typing
  - Strong typing
  - Interpreted
    - JIT compilation versions also exist
  - Portable

Language Overview

Control Flow Statements

- **if**
  - one or more **elif** parts, as short for “else if”
  - an optional else part

- **for**
  - iterates over a sequence (e.g. list or string)
  - **range** function can be used for iterating over arithmetic sequences
  - **break** and **continue** statements, similar to C
  - an optional **else** statement may be used

Functions

- **def**
  - used for defining a function with name, parameters and the function body
  - body must start at next line and must be indented
  - function acts as a user-defined type, which can be assigned another alias, similar to Javascript
  - arguments are passed by value, where value is an object reference
  - default values can be given to arguments
- Lambda expressions
  - small anonymous functions can be created with **lambda** keyword

Data Structures

- List
  - arrays, stacks and queues combined into one
  - square brackets (i.e. []) are used for definition
- Set
  - an unordered collection with no duplicates
  - curly braces (i.e. {}) are used for definition
- Dictionary
  - associative arrays or hash maps
  - collection of key-value pairs
  - curly braces as well as dict() constructor can be used for creating dictionaries

Module

- A file (.py) containing Python definitions and statements
- Module can import other modules using **from** and **import** statements
- Modules can be grouped into packages
  - package is simply a folder containing modules (python files) and may be part of a larger package
  - __init__.py files are required to make a folder be treated as package
    - may be empty or carry initialization statements

Classes

- Groups data and functions together
- Objects can be instantiated using parenthesis () operator with a class name
- Functions can be bound to class or object
  - Functions bound to objects are also called methods
  - First argument of the method is always a reference to the object, usually referred as **self**
  - __init__ method can be used as constructor
- Inheritance
  - A class can derive from a base class, mentioned in parenthesis
  - All methods are virtual by default and support polymorphism, when overridden in derived class

Server-side Web Programming

Options

- CGI Programming
  - Python programs can be integrated with a web server (e.g. Apache) using CGI
  - Different modules provide support for request/response handling and other necessary operations
- Customized Frameworks (e.g. Django)
  - Provides necessary server-side programming support using known architectures and patterns
  - Typically have built-in development servers
  - Integrated with production-ready web servers (e.g. Apache) using WSGI standard

CGI

- Provides an interface between a webserver and a user program
- Standard supported by all webservers
- Provides a mechanism
    - to handle request
    - to generate response in HTTP
- Makes available the environment variables to underlying program

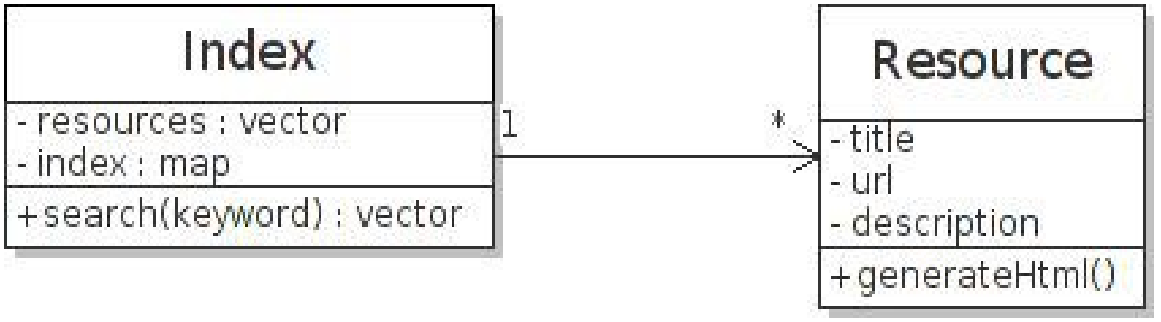
Apache httpd.conf snapshot

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Order deny,allow
                deny from none
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog /var/log/apache2/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog /var/log/apache2/access.log combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
```

Hello World Example

```
print( "Content-type:text/html\n\n" )

print( “””
<html>
	<body>
		Hello World
	</body>
</html> “””)
```

Search Engine Example: Python implementation




```
class Index:

    resources = []
    index = defaultdict(list)

    def __init__(self):
        ''' creating resources '''
        self.resources.append(Resource('Google',...))
        self.resources.append(Resource('Yahoo',...))

        ''' setting up index'''
        self.index['search engine'].append
                                        (self.resources[0])
        …
```

```
    def search(self,keyword):
        return self.index[keyword]
```

```
class Resource:

    title = ''
    url = ''
    description = ''

    def __init__(self,title,url,description):
        self.title = title
        self.url = url
        self.description = description
```

```
    def generateHtml(self):
        html = '<p><a href="">' + self.url + '">' +
            self.title +
                '</a><br/>' + self.description

        return html
```

Search Engine Example: Python implementation

```
print( 'Content-type:text/html\n\n')

print( '\
<html> \n\
	<body> \n' )

if os.environ["QUERY_STRING"] == None or os.environ["QUERY_STRING"] == '':
	'''printing form'''
	print( '\
		<form method="GET" action="main.py"> \n\
			<input type="text" name="q" /> \n\
			<input type="submit" value="Search" /> \n\
		</form> \n' )

else:
	'''printing results'''
	form = cgi.FieldStorage()
	query = form.getvalue("q")

	index = Index()

	for resource in index.search(query):
		print( resource.generateHtml() )

print( '\
	</body> \n\
</html>' )
```