PYTHON & WEB DEVELOPMENT

Python is a beautiful language. It's easy to learn and fun, and its syntax (the rules) is clear and concise. Python is a popular choice for beginners, yet still powerful enough to back some of the world's most popular products and applications from companies like NASA, Google, IBM, Cisco, Microsoft, Industrial Light & Magic among others.

One area where Python shines is web development. Python offers many frameworks from which to choose from including bottle.py, Flask, CherryPy, Pyramid, Django and web2py. These frameworks have been used to power some of the world's most popular sites such as Spotify, Mozilla, Reddit, the Washington Post and Yelp. The tutorials and articles in this section cover techniques used in the development of Python Web applications and focus on how to program real-world solutions to problems that ordinary people actually want to solve.

WEB DEVELOPER IN PYTHON

Python is not used in a web browser. The language executed in browsers such as Chrome, Firefox and Internet Explorer is JavaScript. Projects such as pyjs can compile from Python to JavaScript. However, most Python developers write their web applications using a combination of Python and JavaScript.

- Desining Website, Admin Panels
- Write Algorithms to communicate Database and File Server
- Create Databases SQL and NO-SQL
- Make use of web frameworks to develop web application
- Testing and validation

PYTHON PROGRAMMING BENEFITS

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

- Elegant syntax and dynamic typing
- interpreted nature
- rapid application development in many areas on most platforms
- huge standard library
- high level data structure

COURSE CONTENTS – PYTHON

INTRODUCTION TO PYTHON PROGRAMMING

- Why Python?
- Program structure in Python
- Interactive Shell
- Executable or script files.
- User Interface or IDE

STATEMENTS AND SYNTAX IN PYTHON

- Assignments, Expressions and prints
- If tests and Syntax Rules
- While and For Loops
- Iterations and Comprehensions

FUNCTIONS IN PYTHON

- Function definition and call
- Function Scope
- Arguments
- Function Objects
- Anonymous Function

MODULES AND PACKAGES

- Module Creations and Usage
- Module Search Path
- Module Vs. Script
- Package Creation and Importing

FILE HANDLING

- Opening a file
- Directories in Python
- file Object Attributes
- Other File tools

MEMORY MANAGEMENT AND GARBAGE COLLECTIONS

- Object creation and deletion
- Object properties
- Numbers
- Strings

CONTAINERS

- List
- Tuple

  Set, Frozen Set
- Dictionary
- Other Core Types

OBJECT ORIENTED PROGRAMMING

- Classes and instances
- Classes method calls
- Inheritance and Compositions
- Static and Class Methods
- Bound and Unbound Methods
- Operator Overloading
- Polymorphism

EXCEPTION HANDLING

- What is Exception
- Assertions in Python
- Default Exception Handler
- Catching Exceptions
- Raise an exception
- User defined exception

REGEX WITH ANALYTIC RESULTS

- Regular Expressions Syntax
- Module re
- methods of Regular Expressions
- Match
- Compile
- Sub
- search
- regex operators
- string manipulation

ANGULAR 8

INTRODUCTION TO ANGULAR FRAMEWORK

- Introduction to Angular Framework, History & Overview
- Environment Setup
- Angular CLI, Installing Angular CLI
- NPM commands & package.json
- Bootstrapping Angular App, Components, AppModule
- Project Setup, Editor Environments
- First Angular App & Directory Structure
- Angular Fundamentals, Building Blocks
- MetaData

ESSENTIALS OF ANGULAR

- Component Basics
- Setting up the templates
- Creating Components using CLI
- Nesting Components
- Data Binding - Property & Event Binding, String Interpolation, Style binding
- Two-way data binding
- Input Properties, Output Properties, Passing Event Data

TEMPLATES, STYLES & DIRECTIVES

- Template, Styles, View Encapsulation, adding bootstrap to angular app
- Built-in Directives
- Creating Attribute Directive
- Using Renderer to build attribute directive
- Host Listener to listen to Host Events
- Using Host Binding to bind to Host Properties
- Building Structural Directives

PIPES, SERVICES & DEPENDENCY INJECTION

- Parametrized Pipes
- Chaining Multiple Pipes
- Creating a Custom Pipe
- Creating a Filter Pipe
- Pure and Impure Pipes (or: How to "fix" the Filter Pipe)
- Understanding the "async" Pipe
- Services
- Dependency Injections
- Creating Data Service
- Understanding Hierarchical Injector
- Services for Cross Component Communication
- Injection Tokens

HTTP REQUESTS

- App & Backend Setup
- Sending Requests (Example: POST Request)
- Adjusting Request Headers
- Sending GET Requests
- Sending a PUT Request
- Transform Responses Easily with Observable Operators (map())
- Using the Returned Data
- Catching Http Errors
- Using the "async" Pipe with Http Requests

OBSERVABLES & RXJS OPERATORS

- Basics of Observables & Promises
- Analysing a Built-in Angular Observable
- Building & Using a First Simple Observable
- Building & Using a Custom Observable from Scratch
- Understanding Observable Operators
- Using Subjects to pass and listen to data

D JANGO

INTRODUCTION TO DJANGO

- Django components
- How to install and Configure Django components

HOW TO CREATE DJANGO VIEWS

- About View Functions
- Using Django's HttpResponse Class
- Understanding HttpRequest Objects
- Using QueryDict Objects

CONFIGURING URLS

- About URLconf
- Regular Expressions
- Expression Examples
- Simple URLConf Examples
- Using Multiple URLConf's
- Passing URL Arguments

DJANGO TEMPLATES

- Template Fundamentals
- Creating Template Objects
- Loading Template Files
- Filling in Template Content (Context Objects)
- Template Tags
- Template Filters
- More on For Loops
- Template Inheritance
- Easy Rendering of Templates
- RequestContext Processors
- Global Context Processors

DJANGO FORMS

- Form classes
- Validation
- Authentication
- Advanced Forms processing techniques

DJANGO REST API

- Django REST framework
- Django-piston

UNIT TESTING WITH DJANGO

- Using Python’s unittest2 library
- Test
- Test Databases
- Doctests
- Debugging

DATABASE MODELS

- About Database Models
- Configuring Django for Database Access
- Understanding Django Apps
- Understanding Model Fields & Options
- Table Naming Conventions
- Creating A Django Model
- Adding the App to Your Project
- Specifying Field Lookups
- Lookup Types
- Slicing QuerySets
- Specifying Ordering in QuerySets
- Common QuerySet Methods
- Using Q Objects
- Creating Forms from Models

ADMIN INTERFACE

- Enabling the Admin Interface
- Creating an Admin User

ACCESS CONTROL WITH SESSIONS AND USERS

- Cookies & Django
- The Django Session Framework
- Sessions in Views
- Installing Django User Authentication
- Using Authentication in Views
- Login and Logout
- Building your Own Login/Logout Views
- Adding & Deactivating Users