Document Generated: 04/02/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Beginner

Course Duration: 5 Days

Introduction to Full Stack Web Development with Python and Django (TTPS4860)

About This Course:

Geared for web developers new to Python, Introduction to Full Stack Web Development with Python and Django is a five-day hands-on course that teaches students how to develop Web applications using the Django framework. As an

integral part of our Full Stack Web Developer Boot Camp series, this course teaches you the basics of creating basic applications using the MVC (model-view-controller) design pattern, as well as more advanced topics such as administration, session management, authentication, and automated testing. This comprehensive, practical course provides an in-depth exploration of working with the programming language, not an academic overview of syntax and grammar. Students will immediately be able to use Python to complete tasks in the real world.

Course Objectives:

- Develop full-stack web sites based on content stored in an RDMS
- Use python data types appropriately
- Define data models
- Understand the architecture of a Django-based web site
- Create Django templates for easy-to-modify views
- Map views to URLs
- Take advantage of the built-in Admin interface
- Provide HTML form processing

Audience:

- This introductory-level Python course is geared for experienced web developers new to Python who want to use Python and Django for full stack web development projects.

Prerequisites:

- This introductory-level Python course is geared for experienced web developers new to Python who want to use Python and Django for full stack web development projects.

Course Outline:

The Python Environment

- Starting Python
- Using the interpreter
- Running a Python script
- Getting help
- Editors and IDEs

Variables and values

- Using variables
- Built in functions
- String data
- Numeric data
- Converting data types

Basic input and output

- Writing to the screen
- String formatting
- Command line parameters
- Reading from the keyboard

Flow Control

- About flow control
- Conditional expressions (if)
- Relational and Boolean operators
- while loops
- Exiting from loops

Array-like Types

- About array-like types
- Lists and list methods
- Tuples
- Indexing and slicing
- Iterating through a sequence
- Sequence functions, keywords, and operators
- Basic sorting
- List comprehensions

Working with Files

- File I/O overview
- Opening files
- Reading/writing files

Dictionaries and Sets

- About dictionaries
- Creating and using dictionaries
- Getting values
- About sets
- Creating and using sets

Functions

- Defining functions
- Returning values
- Arguments and arameters
- Variable Scope
- Sorting with functions

Errors and Logging

- Exception overview
- Using try/catch/else/finally
- Handling multiple exceptions
- Logging setup
- Basic logging

Modules and Packages

- Creating Modules
- The import statement
- Module search path
- Creating packages

Introduction to Classes

- About OO programming
- Defining classes
- Constructors
- Properties
- Instance methods and data
- Inheritance

Django Architecture

- Django overview
- Minimal Django layout
- Built in flexibility

Getting Started

- Sites and apps
- Shared configuration
- Executing manage.py
- Starting the project
- Generating app files
- App configuration
- Database setup
- The development server

Using cookiecutter

- What is cookiecutter
- Creating projects with cookiecutter

Creating models

- Defining models
- Adding data
- Related objects
- SQL Migration
- Simple model access

The Admin interface

- Setting up the admin user
- Using the admin interface
- Editing data

Querying Models

- QuerySets

- Field lookups
- Chaining filters
- Slicing QuerySets
- Related fields
- Q objects

Views

- What is a view?
- HttpResponse
- URL route configuration
- Shortcut: get_object_or_404()

Templates

- About templates
- Variable lookups
- The url tag
- Shortcut: render()

Forms

- Forms overview
- GET and POST
- The Form class
- Processing the form
- Widgets
- Validation
- Forms in templates

Unit Testing

- Why create tests?
- When to create tests
- Using Django's test framework
- Using the test client
- Running tests
- Checking code coverage