Optimumpartners

Python

OPTIMUM PARTNERS

YOUR SUCCESS PARTNER

OPTIMUM
PARTNERS

Python Introduction

OPTIMUM PARTNERS

Day 1: Python Basics and Programming Fundamentals

Topics:

- Python installation and setup
- Python syntax, variables, and data types
- Type casting (implicit and explicit)
- Operators and expressions
- Basic input/output (input(), print())
- Introduction to control flow (if, else, elif)

Hands-on:

Write a program to make a tip calculator.

Exercise:

Create a program to check if a number is prime.

OPTIMUM PARTNERS

Day 2: Control Flow, Functions, and Docstrings

Topics:

- Loops (for, while)
- Loop control statements (break, continue, pass)
- Functions (definition, arguments, return values, scope)
- Lambda functions
- Docstrings (writing and accessing documentation)

Hands-on:

Write a program to generate Fibonacci series using loops and functions

Exercise:

Create a program to find the factorial of a number using recursion and add docstrings

OPTIMUM PARTNERS

Day 3: Modules and Packages

Topics:

- What are modules and packages?
- Creating and importing modules
- Standard library modules (math, random, os, sys, etc.)
- Creating packages (__init__.py)
- Using third-party packages (pip, PyPI)

Hands-on:

Create a custom module with utility functions

Install and use a third-party package (e.g., requests)

Exercise:

Write a program that uses multiple modules and packages to perform a task (e.g., calculate statistics using math and statistics modules).

Day 4: Data Structures - Lists, Tuples, and Union Operators

Topics:

- Lists (creation, indexing, slicing, methods)
- List comprehension
- Tuples (creation, immutability, use cases)
- Union operators (| for merging sets and dictionaries)

Hands-on:

Manipulate lists and tuples (e.g., sort, filter, and transform data)

Exercise:

Write a program to find the second-largest number in a list and merge two dictionaries using union operators

OPTIMUM
PARTNERS

Day 5: Data Structures - Sets, Dictionaries, and Walrus Operator

Topics:

- Sets (creation, operations, methods)
- Dictionaries (creation, methods, dictionary comprehension)
- Walrus operator (:=) and its use cases

Hands-on:

Use sets to remove duplicates and dictionaries to count word frequency

Exercise:

Create a program to merge two dictionaries and handle key conflicts using the walrus operator

OPTIMUM PARTNERS

Day 6: Strings, File Handling, and Errors/Exception Handling

Topics:

- String operations and methods
- String formatting (f-strings, format())
- File handling (reading, writing, appending)
- Errors and exception handling (try, except, finally, custom exceptions)

Hands-on:

Write a program to read a file, process its content, and write the output to another file

Exercise:

Create a program to count the frequency of each word in a text file with proper exception handling

OPTIMUM
PARTNERS

Day 7: Object-Oriented Programming (OOP) - Part 1

Topics:

- Classes and objects
- Constructors (__init__)
- Instance vs class variables
- Methods and attributes

Hands-on:

Create a class to model a real-world entity (e.g., a car or a bank account)

Exercise:

Write a program to model a library system using OOP

OPTIMUM PARTNERS

Day 8: Object-Oriented Programming (OOP) - Part 2

Topics:

- Inheritance and method overriding
- Encapsulation (private vs public members)
- Polymorphism
- Magic methods (__str__, __repr__, __add__, etc.)

Hands-on:

Implement inheritance and polymorphism in a real-world scenario

Exercise:

Create a program to model shapes (e.g., circle, rectangle) using OOP

OPTIMUM PARTNERS

Day 9: Advanced Functions, Decorators, and Context Managers

Topics:

- Closures
- Decorators (creation and use cases)
- Context managers (with statement)
- Python with statement for resource management

Hands-on:

Create custom decorators to add functionality to functions

Exercise:

Write a program to measure the execution time of a function using a decorator and use a context manager for file handling

Day 10: Iterators, Generators, and Comprehensions

Topics:

- Iterables and iterators
- Generators (yield)
- List, dictionary, and set comprehensions

Hands-on:

Create custom iterators and generators

Exercise:

Write a generator to produce an infinite sequence of prime numbers

OPTIMUM PARTNERS

Day 11: Regular Expressions, Date/Time, and Timezones

Topics:

- Regular expressions (re module)
- Pattern matching and extraction
- Working with dates and times (datetime module)
- Timezones and conversions (pytz library)

Hands-on:

Use regex to validate and extract data from strings

Write a program to handle date/time conversions

Exercise:

Write a program to validate email addresses and display the current time in different timezones.

Day 12: Database Connectivity and Virtual Environments

Topics:

- Connecting to databases (sqlite3 module)
- Executing SQL queries
- Fetching and inserting data
- Virtual environments (venv, virtualenv, pipenv, pyenv)

Hands-on:

Create a database and perform CRUD operations

Set up and activate a virtual environment

Exercise:

Write a program to manage a student database and install dependencies in a virtual environment

OPTIMUM PARTNERS

Day 13: Multithreading, Multiprocessing, and Async Programming

Topics:

- Introduction to concurrency
- Threading (threading module)
- Multiprocessing (multiprocessing module)
- Asynchronous programming (async, await)

Hands-on:

Create a multithreaded program to perform concurrent tasks

Write an asynchronous program to fetch data from multiple URLs

Exercise:

Write a program to simulate downloading multiple files concurrently using async

OPTIMUM PARTNERS

Day 14: Unit Testing in Python

Topics:

- Introduction to unit testing
- Writing test cases using pytest
- Mocking and patching (pytest mock)

Hands-on:

Write unit tests for existing Python functions and classes

Exercise:

Create a test suite for a Python module and ensure 100% coverage

OPTIMUM
PARTNERS

Day 15: Advanced Libraries - NumPy, Pandas, and Working with Images

Topics:

- Introduction to NumPy (arrays, operations)
- Introduction to Pandas (DataFrames, Series, data manipulation)
- Working with images (PIL or Pillow library)

Hands-on:

Perform data analysis using Pandas
Manipulate images (e.g., resize, crop, filter)

Exercise:

Write a program to analyze a dataset and process an image (e.g., convert to grayscale)

OPTIMUM
PARTNERS

Day 16: Working with APIs and Sending Emails

Topics:

- Making HTTP requests (requests library)
- Parsing JSON data
- Sending emails using Python (smtplib, email libraries)

Hands-on:

Fetch data from a public API (e.g., weather API)
Send an email with attachments

Exercise:

Write a program to fetch data from an API and email it as a report.

OPTIMUM PARTNERS

Day 17: Web Scraping and Automation

Topics:

- Introduction to web scraping (BeautifulSoup, Scrapy)
- Automating tasks with Python (selenium, pyautogui)

Hands-on:

Scrape data from a website and store it in a CSV file
Automate filling out a web form

Exercise:

Write a program to scrape data and save it to a database

OPTIMUM PARTNERS

Day 18: Flask Basics and Routing

Topics:

- Introduction to Flask: What is Flask? Why use Flask?
- Setting up Flask: Installation and virtual environment setup
- Flask project structure: Basic app setup (app.py)
- Routing: Creating routes, URL rules, and HTTP methods (GET, POST).
- Dynamic routing: Using variables in routes
- Debug mode: Enabling and using Flask's debugger

Hands-on:

Create a basic Flask app with a "Hello, World!" route
Add dynamic routes (e.g., /user/<username>)
Experiment with different HTTP methods (GET, POST)

Exercise:

Build a Flask app with the following routes:

- `/`: Displays a welcome message
- `/about`: Displays information about the app
- `/user/<name>`: Displays a personalized greeting

Day 19: Templates and Static Files

Topics:

- Introduction to Jinja2 templating engine
- Rendering HTML templates using `render_template()`
- Template inheritance: Creating base templates and extending them
- Passing data from routes to templates
- Serving static files (CSS, JS, images) in Flask

Hands-on:

Create a base template (`base.html`) with a navigation bar
Extend the base template for multiple pages (e.g., `index.html`, `about.html`)
Add static files (CSS) to style the pages

Exercise:

Build a Flask app with the following features:

- A homepage (`/`) with styled content
- An about page (`/about`) with a description
- Use template inheritance to avoid code duplication

OPTIMUM PARTNERS

Day 20: Forms and User Input

Topics:

- Handling forms in Flask: GET vs. POST methods
- Using `request` object to access form data
- Flask-WTF: Introduction to form handling with Flask-WTF
- Validating form inputs using Flask-WTF validators
- Displaying form errors in templates

Hands-on:

Create a simple contact form with fields for name, email, and message
Handle form submission and display the submitted data
Add validation to ensure all fields are filled

Exercise:

Build a Flask app with a registration form:

- Fields: Name, Email, Password, Confirm Password
- Validate that all fields are filled and passwords match
- Display success or error messages after submission

OPTIMUM
PARTNERS

Day 21: Databases and Advanced Features

Topics:

- Introduction to databases in Flask: SQLite and SQLAlchemy
- Setting up a database with Flask-SQLAlchemy
- Creating models and performing CRUD operations
- Introduction to Flask extensions (e.g., Flask-Login, Flask-Migrate)

Hands-on:

Set up a SQLite database for a Flask app
Create a model for a "Task" (e.g., id, title, description)
Perform CRUD operations (Create, Read, Update, Delete) on tasks

Exercise:

Build a simple task manager app:

- Users can add, view, edit, and delete tasks
- Tasks are stored in a SQLite database
- Use Flask-SQLAlchemy for database operations

Final Projects (Around 3 Days)

1. Weather Dashboard

Create a dashboard that fetches weather data from a public API (e.g., OpenWeatherMap) and displays it in a user-friendly format.

Core Features:

- Fetch weather data for a specific city using an API.
- Display current weather conditions (temperature, humidity, wind speed).
- Use Flask to create a web interface for the dashboard.

Advanced Features (Optional):

- Allow users to search for the weather by city.
- Display a 5-day weather forecast.

OPTIMUM PARTNERS

Final Projects (Around 3 Days)

2. Blogging Platform

Build a simple blogging platform where users can create, read, update, and delete blog posts.

Core Features:

- Create, read, update, and delete blog posts.
- Store blog posts in a SQLite database using Flask-SQLAlchemy.
- Use Flask to create a web interface with templates (Jinja2).

Advanced Features (Optional):

- Add user authentication (Flask-Login) to allow only registered users to post.
- Implement comments and likes functionality.
- Allow users to upload images for blog posts.

Final Projects (Around 3 Days)

3. Task Manager with Reminders

Build a task manager application where users can add tasks, set deadlines, and receive reminders.

Core Features:

- Add tasks with details (title, description, deadline).
- View a list of tasks sorted by deadline.
- Mark tasks as completed.
- Store task data in an SQLite database using Flask-SQLAlchemy.

Advanced Features (Optional):

- Add a Flask-based web interface for better user interaction.
- Send email reminders for upcoming deadlines using smtplib.
- Allow users to categorize tasks (e.g., work, personal).

OPTIMUM PARTNERS

Final Projects (Around 3 Days)

4. Online Quiz Application

Build an online quiz platform where users can take quizzes and view their scores.

Core Features:

- Create a set of questions with multiple-choice answers.
- Allow users to take quizzes and submit answers.
- Calculate and display the user's score.
- Store quiz data in an SQLite database using Flask-SQLAlchemy.

Advanced Features (Optional):

- Add a timer for quizzes.
- Implement user authentication (Flask-Login) to track individual scores.
- Generate a leaderboard for top scorers.
- Allow admins to add, edit, or delete questions.

Final Projects (Around 3 Days)

5. Recipe Finder

Create an application that allows users to search for recipes based on ingredients they have.

Core Features:

- Allow users to input a list of ingredients.
- Search for recipes that include the specified ingredients.
- Display recipe details (name, ingredients, instructions).
- Store recipe data in an SQLite database using Flask-SQLAlchemy.

Advanced Features (Optional):

- Allow users to save favorite recipes.
- Generate a shopping list for missing ingredients.
- Add a Flask-based web interface for better user interaction.

Python FastAPI

OPTIMUM
PARTNERS

Day 25: Introduction to FastAPI

Topics:

- What is FastAPI? Why use FastAPI?
- FastAPI vs Flask vs Django
- Installation and setup
- Creating your first FastAPI app
- Understanding the uvicorn server

Hands-on:

Create a simple FastAPI app with a "Hello, World!" endpoint
Run the app using uvicorn

Exercise:

Add a new endpoint /greet/{name} that returns a personalized greeting

Day 26: Path Parameters, Query Parameters, and Request Body

Topics:

- Path parameters (e.g., /items/{item_id})
- Query parameters (e.g., /items?skip=0&limit=10)
- Request body (using Pydantic models)
- Combining path, query, and body parameters

Hands-on:

Create an endpoint to fetch items by ID and filter them using query parameters
Add an endpoint to create a new item using a request body

Exercise:

Build an endpoint /users/{user_id}/posts that returns posts by a user, filtered by a query parameter status (e.g., published or draft)

OPTIMUM PARTNERS

Day 27: Pydantic Models and Data Validation

Topics:

- Introduction to Pydantic
- Creating Pydantic models for request and response validation
- Nested models and complex data structures
- Custom validators in Pydantic

Hands-on:

Create a Pydantic model for a Product with fields like name, price, and category

Validate incoming requests using the Product model

Exercise:

Add a custom validator to ensure the price field is greater than 0

OPTIMUM
PARTNERS

Day 28: Error Handling and Custom Responses

Topics:

- Built-in HTTP exceptions (e.g., HTTPException)
- Custom error handling
- Returning custom responses (e.g., JSON, HTML, files)

Hands-on:

Create an endpoint that raises a 404 Not Found error if an item doesn’t exist
Return a custom JSON response with a success message

Exercise:

Add error handling for invalid input data (e.g., negative prices)

Day 29: Dependency Injection

Topics:

- What is dependency injection?
- Using dependencies for reusable logic (e.g., authentication, database sessions)
- Dependency injection with Depends

Hands-on:

Create a dependency to verify an API key in the request headers
Use the dependency in multiple endpoints

Exercise:

Add a dependency to validate user roles (e.g., admin, user)

OPTIMUM PARTNERS

Day 30: Dockerizing FastAPI

Topics:

- Introduction to Docker
- Creating a Dockerfile for FastAPI
- Using Docker Compose for multi-container setups (FastAPI + PostgreSQL + Redis)

Hands-on:

Dockerize your FastAPI app

Set up a PostgreSQL database and Redis using Docker Compose

Exercise:

Add a health check endpoint and test the Dockerized app

OPTIMUM PARTNERS

Day 31: Introduction to SQLAlchemy and Database Setup

Topics:

- What is SQLAlchemy?
- Setting up a database (SQLite, PostgreSQL)
- Creating models with SQLAlchemy ORM

Hands-on:

Create a User model with fields like id, username, and email

Set up a SQLite database and perform CRUD operations

Exercise:

Add a Post model with a relationship to the User model

Day 32: Async Database Drivers and SQLAlchemy

Topics:

- Async database drivers (e.g., asyncpg, databases)
- Using SQLAlchemy with async drivers
- Performing async CRUD operations

Hands-on:

Set up an async PostgreSQL database

Perform async CRUD operations on the User model

Exercise:

Add an endpoint to fetch all posts by a user asynchronously

OPTIMUM PARTNERS

Day 33: Database Migrations with Alembic

Topics:

- What is Alembic?
- Setting up Alembic for database migrations
- Creating and applying migrations

Hands-on:

Add a new field created_at to the User model and create a migration

Exercise:

Rollback a migration and reapply it

Day 34: Authentication with OAuth2 and JWT

Topics:

- Introduction to OAuth2 and JWT
- Implementing JWT-based authentication in FastAPI
- Securing endpoints with OAuth2 password flow

Hands-on:

Create a /login endpoint that returns a JWT token

Secure an endpoint to allow only authenticated users

Exercise:

Add role-based access control (e.g., only admin users can delete posts)

Day 35: Advanced Authentication

Topics:

- Refresh tokens
- Social login (e.g., Google, GitHub)
- Two-factor authentication (2FA)

Hands-on:

Implement a refresh token mechanism
Add Google OAuth2 login

Exercise:

Add 2FA using a library like pyotp

OPTIMUM PARTNERS

Day 36: OpenAPI and Automatic Docs

Topics:

- What is OpenAPI?
- Customizing the automatic docs (Swagger UI, ReDoc)
- Adding descriptions and examples to endpoints

Hands-on:

Add descriptions and examples to your existing endpoints

Exercise:

Customize the Swagger UI theme

OPTIMUM PARTNERS

Day 37: Monitoring and Logging

Topics:

- Setting up logging in FastAPI
- Monitoring with Prometheus and Grafana
- Error tracking with Sentry

Hands-on:

Add logging to your FastAPI app
Set up Prometheus to monitor API metrics

Exercise:

Integrate Sentry for error tracking

OPTIMUM PARTNERS

Day 38: Security Best Practices

Topics:

- CORS (Cross-Origin Resource Sharing)
- CSRF protection
- Rate limiting
- Input validation and sanitization

Hands-on:

Enable CORS in your FastAPI app

Add rate limiting to your endpoints

Exercise:

Implement CSRF protection

OPTIMUM PARTNERS

Day 39: Performance Optimization

Topics:

- Caching with Redis
- Using async tasks with Celery
- Optimizing database queries

Hands-on:

Add Redis caching to your FastAPI app
Offload heavy tasks to Celery

Exercise:

Optimize a slow database query using indexing

Day 40: File Uploads and Static Files

Topics:

- Handling file uploads in FastAPI
- Serving static files (e.g., images, CSS)

Hands-on:

Create an endpoint to upload and save files

Serve static files from a directory

Exercise:

Add validation to allow only specific file types (e.g., images)

Day 41: WebSockets

Topics:

- Introduction to WebSockets
- Creating WebSocket endpoints in FastAPI
- Real-time communication use cases

Hands-on:

Create a WebSocket endpoint for a chat application

Exercise:

Add a broadcast feature to send messages to all connected clients

OPTIMUM PARTNERS

Day 42: Background Tasks

Topics:

- What are background tasks?
- Using BackgroundTasks in FastAPI
- Offloading heavy tasks (e.g., sending emails, processing data)

Hands-on:

Add a background task to send an email after user registration

Exercise:

Process and save uploaded files in the background.

OPTIMUM PARTNERS

Day 43: GraphQL Integration

Topics:

- Introduction to GraphQL
- Integrating GraphQL with FastAPI using Strawberry or Graphene

Hands-on:

Create a GraphQL endpoint to fetch and mutate data

Exercise:

Add a GraphQL query to fetch nested data (e.g., user and their posts)

OPTIMUM
PARTNERS

Day 44: Unit Testing in FastAPI

Topics:

- Writing unit tests for FastAPI endpoints
- Using TestClient to simulate requests
- Mocking dependencies (e.g., database, external APIs)

Hands-on:

Write tests for your FastAPI app

Exercise:

Add tests for authentication and authorization logic

Final Projects (Around 4 Days)

1. E-Commerce API

Core Features:

- CRUD operations for products, users, and orders.
- JWT-based authentication for users (login, signup, and role-based access).
- Endpoints for browsing products with filters (e.g., category, price range).
- Order creation with validation (e.g., stock availability).

Advanced Features (Optional):

- Payment gateway integration (e.g., Stripe).
- Redis caching for frequently accessed product data.
- Background tasks for sending order confirmation emails.
- File upload for product images.
- GraphQL endpoint for flexible product queries.

OPTIMUM PARTNERS

Final Projects (Around 4 Days)

2. Online Learning Platform

Core Features:

- CRUD operations for courses, lessons, and enrollments.
- JWT-based authentication for students and instructors.
- Endpoints for fetching courses by category or instructor.
- Enrollment validation (e.g., course capacity).

Advanced Features (Optional):

- File upload for course materials (e.g., PDFs, videos).
- Background tasks for sending enrollment confirmation emails.
- Rate limiting to prevent abuse of enrollment endpoints.
- Integration with a payment gateway for course purchases.
- GraphQL API for fetching nested data (e.g., courses with lessons).

Final Projects (Around 4 Days)

3. Job Board API

Core Features:

- CRUD operations for job postings and applications.
- JWT-based authentication for employers and job seekers.
- Endpoints for filtering jobs by location, category, or salary.
- Application validation (e.g., one application per user per job).

Advanced Features (Optional):

- File upload for resumes and cover letters.
- Background tasks for sending application confirmation emails.
- Rate limiting to prevent spam applications.
- Full-text search for job postings.
- GraphQL API for fetching nested data (e.g., jobs with applications).

OPTIMUM PARTNERS

Final Projects (Around 4 Days)

4. Fitness Tracker API

Core Features:

- CRUD operations for workouts, exercises, and user profiles.
- JWT-based authentication.
- Endpoints for tracking progress (e.g., calories burned, steps taken).
- Validation for workout data (e.g., duration, intensity).

Advanced Features (Optional):

- Integration with wearable device APIs (e.g., Fitbit).
- Background tasks for generating weekly progress reports.
- Redis caching for frequently accessed workout data.
- File upload for workout images or videos.
- GraphQL API for fetching nested data (e.g., workouts with exercises).

OPTIMUM PARTNERS

Final Projects (Around 4 Days)

5. Inventory Management System

Core Features:

- CRUD operations for products, suppliers, and orders.
- JWT-based authentication for admins and staff.
- Endpoints for filtering products by category or supplier.
- Validation for stock levels and reorder points.

Advanced Features (Optional):

- Background tasks for generating low-stock alerts.
- File upload for product manuals or specifications.
- Rate limiting to prevent abuse of order endpoints.
- Integration with a barcode scanning API.
- GraphQL API for fetching nested data (e.g., products with suppliers).

Final Projects (Around 4 Days)

6. Travel Booking API

Core Features:

- CRUD operations for flights, hotels, and bookings.
- JWT-based authentication for users and admins.
- Endpoints for filtering flights/hotels by date, location, or price.
- Booking validation (e.g., seat availability).

Advanced Features (Optional):

- Background tasks for sending booking confirmation emails.
- File upload for travel documents (e.g., passports).
- Rate limiting to prevent abuse of booking endpoints.
- Integration with a payment gateway for bookings.
- GraphQL API fetches nested data (e.g., flights with bookings).