Complete Python API Learning Curriculum

From Beginner to Advanced - Project-Based Learning Path

Phase 1: Foundations (Week 1-2)

Project 1: Hello World API

Goal: Understand basic API concepts and Flask setup

What You'll Learn:

- What APIs are and how they work
- HTTP methods (GET, POST, PUT, DELETE)
- Status codes
- JSON responses
- Flask basics

Project: Build a simple greeting API

```python
# Features to implement:
- GET /hello - returns "Hello World"
- GET /hello/{name} - returns personalized greeting
- POST /hello - accepts name in JSON, returns greeting
```

Skills Gained:

- Setting up Flask
- Creating endpoints
- Handling different HTTP methods
- JSON serialization/deserialization
- Testing APIs with Postman/curl

Project 2: Personal Task Manager API

Goal: Learn CRUD operations and data persistence

What You'll Learn:

- CRUD operations
- Request validation

- Error handling
- File-based storage
- API design principles

Project: Task management system

```python
# Features to implement:
- GET /tasks - list all tasks
- GET /tasks/{id} - get specific task
- POST /tasks - create new task
- PUT /tasks/{id} - update task
- DELETE /tasks/{id} - delete task
```

Skills Gained:

- Data validation with marshmallow
- Error handling and custom exceptions
- File I/O for data persistence
- RESTful API design
- Request/response schemas

Phase 2: Database Integration (Week 3-4)

Project 3: Library Management System

Goal: Learn database integration and relationships

What You'll Learn:

- SQLAlchemy ORM
- Database models and relationships
- Migrations
- Query optimization
- Environment configuration

Project: Library book management

```python
# Features to implement:
- Books CRUD (title, author, ISBN, availability)
- Authors management
- Book borrowing system
- Search and filtering
- Database relationships (one-to-many, many-to-many)
```

Skills Gained:

- Database design and modeling
- SQLAlchemy relationships
- Database migrations with Alembic
- Environment variables and configuration
- Advanced querying

Project 4: Blog API with User System

Goal: Learn authentication and authorization

What You'll Learn:

- User authentication
- JWT tokens
- Password hashing
- Role-based access control
- Session management

Project: Multi-user blog platform

```python
# Features to implement:
- User registration/login
- JWT authentication
- Blog post CRUD (authenticated users only)
- Comments system
- User roles (admin, author, reader)
- Password reset functionality
```

Skills Gained:

- JWT implementation

- bcrypt for password hashing
- Decorators for authentication
- Role-based permissions
- Secure API design

Phase 3: Advanced Features (Week 5-6)

Project 5: E-commerce API

Goal: Learn complex business logic and transactions

What You'll Learn:

- Database transactions
- Complex relationships
- Business logic implementation
- Data validation
- API versioning

Project: Online store backend

```python
# Features to implement:
- Product catalog with categories
- Shopping cart management
- Order processing
- Inventory tracking
- Payment integration (mock)
- Admin dashboard endpoints
```

Skills Gained:

- Database transactions
- Complex data relationships
- Business logic separation
- API versioning strategies
- Advanced error handling

Project 6: Social Media API

Goal: Learn real-time features and performance optimization

What You'll Learn:

- File uploads
- Image processing
- Caching
- Background tasks
- API rate limiting

Project: Social networking platform

```python
# Features to implement:
- User profiles with image uploads
- Post creation with media
- Follow/unfollow system
- News feed generation
- Like/comment system
- Search functionality
```

Skills Gained:

- File upload handling
- Image processing with Pillow
- Redis for caching
- Celery for background tasks
- Rate limiting implementation

Phase 4: Production Ready (Week 7-8)

Project 7: Real-time Chat API

Goal: Learn WebSockets and real-time communication

What You'll Learn:

- WebSocket implementation
- Real-time messaging
- Socket.IO integration
- Connection management
- Message queuing

Project: Chat application backend

```python
# Features to implement:
- Real-time messaging
- Chat rooms
- Private messaging
- Online status tracking
- Message history
- File sharing in chats
```

Skills Gained:

- WebSocket with Flask-SocketIO
- Real-time event handling
- Connection state management
- Message broadcasting
- Real-time data synchronization

Project 8: Microservices Architecture

Goal: Learn microservices and API integration

What You'll Learn:

- Microservices design
- Service communication
- API gateway
- Load balancing
- Service discovery

Project: Distributed e-learning platform

```python
# Services to build:
- User Service (authentication)
- Course Service (course management)
- Progress Service (learning progress)
- Notification Service (alerts)
- API Gateway (request routing)
```

Skills Gained:

- Microservices architecture
- Inter-service communication
- API gateway implementation
- Docker containerization
- Service orchestration

Phase 5: Advanced Topics & Deployment (Week 9-10)

Project 9: Analytics Dashboard API

Goal: Learn data processing and visualization APIs

What You'll Learn:

- Data aggregation
- Analytics endpoints
- Chart data preparation
- Performance optimization
- Caching strategies

Project: Business analytics platform

```python
# Features to implement:
- Data ingestion endpoints
- Real-time analytics
- Custom dashboard creation
- Report generation
- Data export (CSV, PDF)
- Scheduled reports
```

Skills Gained:

- Data aggregation techniques
- Performance optimization
- Report generation
- Scheduled tasks
- Data visualization preparation

Project 10: API Marketplace

Goal: Learn API management and monitoring

What You'll Learn:

- API documentation
- Rate limiting
- API keys management
- Usage analytics
- Monitoring and logging

Project: API management platform

```python
# Features to implement:
- API key generation and management
- Usage tracking and analytics
- Rate limiting per API key
- Automatic API documentation
- Health monitoring
- Billing and usage reports
```

Skills Gained:

- API key management
- Usage tracking
- Automatic documentation with Swagger
- Monitoring and alerting
- API monetization concepts

Learning Resources & Tools

Essential Libraries

- **Flask/FastAPI**: Web framework
- **SQLAlchemy**: ORM
- **Marshmallow**: Serialization/validation
- **Flask-JWT-Extended**: JWT authentication
- **Celery**: Background tasks
- **Redis**: Caching and message broker
- **pytest**: Testing

- **Alembic**: Database migrations

Development Tools

- **Postman**: API testing
- **Docker**: Containerization
- **GitHub Actions**: CI/CD
- **Swagger/OpenAPI**: Documentation
- **Prometheus**: Monitoring
- **Nginx**: Reverse proxy

Testing Strategy

Each project should include:

- Unit tests for business logic
- Integration tests for endpoints
- Load testing for performance
- Security testing for vulnerabilities

Deployment Progression

1. **Local development** (Projects 1-3)
2. **Heroku deployment** (Projects 4-6)
3. **Docker containers** (Projects 7-8)
4. **Cloud deployment** (AWS/GCP) (Projects 9-10)

Assessment Milestones

Beginner Level (After Project 2)

- Can create basic CRUD APIs
- Understands HTTP methods and status codes
- Can handle JSON requests/responses

Intermediate Level (After Project 6)

- Implements authentication and authorization
- Works with databases and relationships
- Handles file uploads and processing

Advanced Level (After Project 10)

- Designs microservices architecture
- Implements real-time features
- Understands production deployment

Bonus Projects (Optional Advanced Practice)

AI/ML API Integration

- Integrate OpenAI API
- Image recognition service
- Natural language processing

IoT Data Collection API

- Sensor data ingestion
- Real-time monitoring
- Device management

Blockchain Integration

- Cryptocurrency price tracking
- Smart contract interaction
- NFT marketplace backend

Weekly Time Commitment

- **Study Time**: 10-15 hours per week
- **Project Development**: 8-12 hours per week
- **Testing & Documentation**: 3-5 hours per week

Success Metrics

By completion, you should be able to:

- Design and implement RESTful APIs
- Handle authentication and authorization
- Work with databases and complex relationships
- Implement real-time features
- Deploy production-ready applications
- Debug and optimize API performance
- Write comprehensive tests

- Create proper API documentation

Next Steps After Completion

- Contribute to open-source API projects
- Build APIs for real client projects
- Explore GraphQL as alternative to REST
- Learn cloud-native development (Kubernetes, serverless)
- Specialize in specific domains (fintech, healthcare, etc.)