Roadmap de Preparación para Entrevistas Python

2 horas diarias - 12 semanas (84 días)

SEMANA 1: Fundamentos Python + Algoritmos Básicos

Objetivo: Dominar las bases sólidas del lenguaje

Día 1-2: Fundamentos Python Core

- **1h Teoría:** Estructuras de datos nativas, complejidades temporales
- **1h Práctica:** Ejercicios con listas, diccionarios, sets, tuplas
- **Recursos:** Python docs, implementar operaciones básicas

Día 3-4: Python Avanzado

- **1h Teoría:** Comprensiones, generadores, decoradores
- **1h Práctica:** Crear decoradores personalizados, generators
- **Ejercicios:** LeetCode Easy (arrays, strings)

Día 5-6: Manejo de Excepciones y Context Managers

- **1h Teoría:** Try/except, context managers, *args/**kwargs
- **1h Práctica:** Implementar context managers personalizados
- **Ejercicios:** Manejo de archivos, custom exceptions

Día 7: Repaso y Evaluación

- **2h Práctica:** Resolver 8-10 ejercicios combinando todos los conceptos
- **Mini-evaluación:** Simular entrevista básica

SEMANA 2: Algoritmos y Complejidad

Objetivo: Dominar análisis de algoritmos y estructuras básicas

Día 8-9: Big O y Análisis de Complejidad

- **1h Teoría:** Notación Big O, análisis temporal y espacial
- **1h Práctica:** Analizar complejidad de algoritmos existentes
- **Ejercicios:** Calcular Big O de diferentes soluciones

Día 10-11: Algoritmos de Búsqueda y Ordenamiento

- **1h Teoría:** Binary search, bubble sort, merge sort, quick sort
- **1h Práctica:** Implementar desde cero cada algoritmo
- **Ejercicios:** LeetCode Medium (sorting, searching)

Día 12-13: Estructuras de Datos Básicas

- **1h Teoría:** Arrays, linked lists, stacks, queues
- **1h Práctica:** Implementar desde cero cada estructura
- **Ejercicios:** Problemas específicos de cada estructura

Día 14: Consolidación

- **2h Práctica:** Resolver problemas que combinen múltiples conceptos
- **Simulacro:** 45min de coding interview

SEMANA 3: Estructuras de Datos Avanzadas

Objetivo: Manejar estructuras complejas y sus aplicaciones

Día 15-16: Árboles Binarios

- **1h Teoría:** BST, recorridos (inorder, preorder, postorder)
- **1h Práctica:** Implementar BST con todas las operaciones
- **Ejercicios:** Tree traversal, BST validation

Día 17-18: Heaps y Priority Queues

- **1h Teoría:** Min-heap, max-heap, heapify operations
- **1h Práctica:** Implementar heap desde cero
- **Ejercicios:** Top K problems, merge K sorted lists

Día 19-20: Hash Tables y Grafos Básicos

- **1h Teoría:** Hash functions, collision handling, graph representation
- **1h Práctica:** Implementar hash table, adjacency list/matrix
- **Ejercicios:** Graph problems básicos

Día 21: Práctica Intensiva

- **2h:** Resolver 6-8 problemas Medium de LeetCode
- **Focus:** Combinar todas las estructuras aprendidas

SEMANA 4: Programación Orientada a Objetos + Testing

Objetivo: Dominar OOP y testing strategies

Día 22-23: OOP Fundamentals

- **1h Teoría:** Clases, herencia, polimorfismo, encapsulación
- **1h Práctica:** Diseñar sistema de clases complejo
- **Ejercicios:** Design patterns básicos (Singleton, Factory)

Día 24-25: Python OOP Avanzado

- **1h Teoría:** Métodos especiales, properties, descriptors
- **1h Práctica:** Implementar clase con todos los magic methods
- **Ejercicios:** Custom containers, operator overloading

Día 26-27: Testing y TDD

- **1h Teoría:** unittest, pytest, mocking, TDD principles
- **1h Práctica:** Escribir tests comprehensivos para código existente
- **Ejercicios:** Test-driven development para mini-proyecto

Día 28: Project Day

- **2h:** Crear mini-aplicación con OOP + tests completos
- **Entregable:** Sistema de biblioteca con tests al 100%

SEMANA 5: Concurrencia y Base de Datos

Objetivo: Manejar concurrencia y bases de datos

Día 29-30: Threading y Multiprocessing

- **1h Teoría:** GIL, threading vs multiprocessing, locks
- **1h Práctica:** Implementar ejemplos con ambos enfoques
- **Ejercicios:** Producer-consumer, thread synchronization

Día 31-32: Async/Await y AsyncIO

- **1h Teoría:** Event loop, coroutines, async libraries

- **1h Práctica:** Convertir código sync a async
- **Ejercicios:** Async web scraping, concurrent API calls

Día 33-34: SQL y Bases de Datos

- **1h Teoría:** JOINs, subqueries, índices, normalización
- **1h Práctica:** Resolver problemas complejos de SQL
- **Ejercicios:** HackerRank SQL, diseño de esquemas

Día 35: Integration Day

- **2h:** Proyecto que combine concurrencia + database
- **Challenge:** API que procese requests concurrentemente

SEMANA 6: Django Fundamentals

Objetivo: Dominar Django core concepts

Día 36-37: Django Architecture

- **1h Teoría:** MVT, settings, URL routing, apps
- **1h Práctica:** Setup proyecto Django desde cero
- **Ejercicios:** Crear app básica con múltiples views

Día 38-39: Models y Django ORM

- **1h Teoría:** Model fields, relationships, Meta options
- **1h Práctica:** Diseñar modelos complejos con relationships
- **Ejercicios:** Queries complejas, select_related/prefetch_related

Día 40-41: Views y Templates

- **1h Teoría:** CBV vs FBV, template inheritance, forms
- **1h Práctica:** Implementar CRUD completo
- **Ejercicios:** Custom forms, template filters

Día 42: Django Project

- **2h:** Blog completo con comments, users, admin
- **Features:** Authentication, permissions, search

SEMANA 7: Django REST Framework + Avanzado

Objetivo: API development con DRF

Día 43-44: DRF Basics

- **1h Teoría:** Serializers, ViewSets, APIViews
- **1h Práctica:** Convertir Django app a API REST
- **Ejercicios:** CRUD API con diferentes ViewSets

Día 45-46: DRF Advanced

- **1h Teoría:** Authentication, permissions, throttling
- **1h Práctica:** Implementar JWT auth, custom permissions
- **Ejercicios:** API versioning, filtering, pagination

Día 47-48: Django Performance

- **1h Teoría:** Query optimization, caching, database indexing
- **1h Práctica:** Profile y optimizar queries lentas
- **Ejercicios:** Django debug toolbar, query analysis

Día 49: DRF Project

- **2h:** E-commerce API con auth, products, orders
- **Testing:** Tests comprehensivos para toda la API

SEMANA 8: Flask + FastAPI

Objetivo: Comparar frameworks, dominar alternativas

Día 50-51: Flask Fundamentals

- **1h Teoría:** Application factory, blueprints, context
- **1h Práctica:** API REST básica con Flask
- **Ejercicios:** Flask extensions (SQLAlchemy, JWT)

Día 52-53: Flask Advanced

- **1h Teoría:** Custom middleware, error handling, testing
- **1h Práctica:** Flask app completa con database
- **Ejercicios:** Flask-Login, Flask-WTF

Día 54-55: FastAPI Deep Dive

- **1h Teoría:** Type hints, Pydantic, dependency injection
- **1h Práctica:** Migrar Flask API a FastAPI
- **Ejercicios:** Async endpoints, WebSocket, background tasks

Día 56: Framework Comparison

- **2h:** Implementar misma API en Django/Flask/FastAPI
- **Analysis:** Performance, features, trade-offs

SEMANA 9: Database Deep Dive + Migrations

Objetivo: Dominar ORMs y estrategias de migración

Día 57-58: SQLAlchemy Advanced

- **1h Teoría:** Core vs ORM, relationships, lazy loading
- **1h Práctica:** Complex queries, session management
- **Ejercicios:** Performance optimization, custom types

Día 59-60: Django Migrations

- **1h Teoría:** Migration files, operations, data migrations
- **1h Práctica:** Complex migration scenarios
- **Ejercicios:** Zero-downtime migrations, rollback strategies

Día 61-62: Alembic + Migration Strategies

- **1h Teoría:** Alembic operations, branching, offline migrations
- **1h Práctica:** Flask/FastAPI migration setup
- **Ejercicios:** Production migration strategies

Día 63: Database Project

- **2h:** Multi-tenant application con complex migrations
- **Challenge:** Backward compatibility, data migration

SEMANA 10: System Design + Architecture

Objetivo: Pensar en sistemas escalables

Día 64-65: Design Patterns

- **1h Teoría:** SOLID principles, common patterns
- **1h Práctica:** Refactor código con patterns
- **Ejercicios:** Observer, Strategy, Command patterns

Día 66-67: System Design Basics

- **1h Teoría:** Load balancing, caching, message queues
- **1h Práctica:** Diseñar arquitectura de chat app
- **Ejercicios:** Trade-offs, scaling strategies

Día 68-69: Microservices + APIs

- **1h Teoría:** Service communication, API Gateway
- **1h Práctica:** Splitting monolith to microservices
- **Ejercicios:** Service discovery, fault tolerance

Día 70: Architecture Project

- **2h:** Design Instagram-like system
- **Deliverable:** Complete architecture diagram + justifications

---

SEMANA 11: DevOps + Performance

Objetivo: Production-ready applications

Día 71-72: Containerization

- **1h Teoría:** Docker, multi-stage builds, orchestration
- **1h Práctica:** Dockerizar aplicaciones Django/FastAPI
- **Ejercicios:** Docker Compose, health checks

Día 73-74: Performance Optimization

- **1h Teoría:** Profiling, memory management, caching
- **1h Práctica:** Profile y optimizar aplicación real
- **Ejercicios:** Redis caching, query optimization

Día 75-76: Monitoring + Security

- **1h Teoría:** Logging, metrics, security best practices
- **1h Práctica:** Setup monitoring stack
- **Ejercicios:** Sentry integration, security headers

Día 77: Production Deployment

- **2h:** Deploy complete application to cloud
- **Infrastructure:** CI/CD pipeline, monitoring, backups

SEMANA 12: Interview Preparation + Advanced Topics

Objetivo: Preparación final y temas plus

Día 78-79: Advanced Algorithms

- **1h Teoría:** Dynamic programming, graph algorithms
- **1h Práctica:** Hard LeetCode problems
- **Ejercicios:** Dijkstra, A*, complex DP

Día 80-81: Data Engineering Basics

- **1h Teoría:** ETL, data pipelines, pandas optimization

- **1h Práctica:** Build data processing pipeline
- **Ejercicios:** Apache Airflow, data validation

Día 82-83: Mock Interviews

- **2h/día:** Simulacros completos de entrevista
- **Scenarios:** Technical + system design + behavioral
- **Focus:** Communication, problem-solving approach

Día 84: Final Review

- **1h:** Repaso de conceptos débiles identificados
- **1h:** Preparación de historias STAR para behavioral
- **Checklist:** Portfolio de proyectos completados

Estructura Diaria Recomendada:

Primera Hora (Teoría/Conceptos):

- 15min: Repaso día anterior
- 35min: Nuevo contenido teórico
- 10min: Notas y síntesis

Segunda Hora (Práctica):

- 45min: Coding/implementación
- 10min: Testing de lo implementado
- 5min: Documentación/reflexión

Recursos Recomendados:

Plataformas de Practice:

- LeetCode (3 problems/semana mínimo)
- HackerRank (SQL y algorithms)
- Codewars (Python-specific)
- InterviewBit (system design)

Documentación Oficial:

- Python Documentation
- Django Documentation
- FastAPI Documentation
- Flask Documentation

Libros de Apoyo:

- "Effective Python" - Brett Slatkin
- "Architecture Patterns with Python" - Harry Percival
- "Two Scoops of Django" - Daniel Roy Greenfeld

Proyectos Sugeridos:

1. **Semana 4:** Sistema de biblioteca (OOP + Testing)
2. **Semana 6:** Blog con Django (CRUD completo)
3. **Semana 7:** E-commerce API con DRF
4. **Semana 8:** Comparativa de frameworks (misma API)
5. **Semana 9:** Multi-tenant app con migrations
6. **Semana 11:** Aplicación containerizada en cloud

Métricas de Progress:

Checkpoints Semanales:

- Tests teóricos (80%+ accuracy)
- Coding challenges completados
- Proyectos funcionales entregados
- Mock interview performance

Indicadores de Éxito:

- Resolver Medium LeetCode en <30min
- Explicar trade-offs técnicos claramente
- Implementar API REST completa en <2h
- Diseñar system architecture coherente

Tips para Maximizar 2h/día:

1. **Preparación Previa:** Ten environment setup y recursos listos
2. **Pomodoro Technique:** 25min focus + 5min break
3. **Active Learning:** Explica conceptos en voz alta
4. **Code Review:** Revisa tu código del día anterior
5. **Portfolio Building:** Documenta todos los proyectos
6. **Network Effect:** Comparte progress en LinkedIn/GitHub

¡Éxito en tu preparación! 🚀