WhatsApp Automation with Python Using Neonize

Build powerful WhatsApp bots and automation scripts using Neonize's Python-native API backed by Go performance. For Python developers and automation engineers.

Python Instructor: Ethan Carter • 5 sections • 53 lessons • 15.75h total

View this course online Browse all courses

About this course

WhatsApp automation has become a core tool for businesses, support teams, and developers who need programmatic access to messaging. But building reliable bots from scratch? That's where most projects stall out. Neonize solves this by wrapping the well-tested Whatsmeow Go library in a clean Python API, giving you real-time message handling, media support, and group management without wrestling with low-level protocol details.

This course walks you through Neonize from first installation to production-ready bots. You'll set up your environment, connect to WhatsApp, handle messages and media events, manage groups, work with async patterns, and deploy real projects like notification systems and customer support flows. Every lesson includes working code, practical examples, and patterns you can adapt to your own use cases.

Course curriculum

Introduction to Neonize and WhatsApp Automation

- Why WhatsApp Automation Matters in 2026 (12m)
  Explore the current state of WhatsApp automation, common business use cases, and why programmatic messaging is now a standard tool for support, notifications, and workflows.
- Neonize Architecture: Python Meets Go Performance (18m)
  Understand how Neonize wraps the Whatsmeow Go library to deliver speed and reliability through a Python-native interface. Covers the client-event model and data flow.
- Comparing Neonize to Other WhatsApp Libraries (15m)
  See how Neonize stacks up against alternatives like pywhatkit, whatsapp-web.js wrappers, and raw API approaches. Learn when Neonize is the right choice and when it isn't.
- Core Concepts: Events, Messages, and Connections (20m)
  Get familiar with the three pillars of Neonize: event-driven architecture, message objects, and persistent connections. This foundation will carry you through every later section.
- Understanding End-to-End Encryption in Neonize (14m)
  Learn how WhatsApp's encryption works and how Neonize handles session keys, device registration, and encrypted message delivery behind the scenes.
- What You'll Build: Course Project Overview (10m)
  Preview the hands-on projects across this course: a notification bot, a customer support agent, a group management tool, and a media processing pipeline.
- Setting Expectations: Limitations and Responsible Use (11m)
  Honest discussion about rate limits, WhatsApp's terms of service, and how to build automation that's useful without crossing into spam or abuse territory.

Installation and Environment Setup

- Installing Python 3.8+ and Preparing Your System (10m)
  Step-by-step setup for your Python environment. Covers version checks, virtual environments, and OS-specific considerations for Windows, macOS, and Linux.
- Installing Neonize via pip (8m)
  Run the pip install command, verify the package installed correctly, and understand what dependencies Neonize pulls in automatically.
- Building Neonize from Source with Go (Optional) (22m)
  For developers who need the latest features or custom builds. Install Go, clone the repository, and compile the library from scratch.
- Project Structure for WhatsApp Automation Scripts (15m)
  Set up a clean project directory with config files, credential storage, logging output, and a main entry point. Includes a ready-to-use template.
- Your First Neonize Client: Connection Walkthrough (18m)
  Initialize NewClient, scan the QR code for authentication, and confirm your bot connects to WhatsApp. The essential first successful run.

- Handling the QR Code Authentication Flow (14m)
  Understand how device linking works, manage session persistence so you don't re-scan every restart, and troubleshoot common authentication failures.
- Configuration Files and Credential Management (16m)
  Store bot names, database paths, and API tokens securely. Learn patterns for environment variables, .env files, and config modules.
- Verifying Your Setup with a Simple Echo Bot (12m)
  Build a minimal bot that echoes back any message it receives. Use this to confirm your installation, connection, and message handling all work end to end.
- Common Installation Errors and How to Fix Them (15m)
  Troubleshoot dependency conflicts, platform-specific build errors, connection timeouts, and the most frequent issues new users hit.
- Setting Up Logging and Debug Output (13m)
  Configure Neonize's built-in logging to track connection status, message flow, and errors. Covers log levels, output formats, and file-based logging.

Basic Functionality and Core Operations

- The Event System: Listening for WhatsApp Events (20m)
  Deep dive into Neonize's event-driven architecture. Register handlers for MessageEv, ConnectedEv, and other core events. Understand event data structures.
- Sending Text Messages Programmatically (14m)
  Use client methods to send messages to individual contacts. Covers phone number formatting, message objects, and basic delivery confirmation.
- Receiving and Parsing Incoming Messages (18m)
  Extract sender info, message content, and timestamps from incoming MessageEv events. Handle both conversation messages and extended text messages.
- Replying to Messages and Threaded Conversations (15m)
  Use reply_message to respond in context. Build simple conversational flows where your bot answers questions based on what users send.
- Message Receipts and Read Status Tracking (16m)
  Track when messages are delivered and read. Understand receipt types and how to use them for confirmation workflows.
- Handling Presence and Typing Indicators (12m)
  Show typing indicators before your bot responds. Send presence updates so contacts know your bot is active or available.
- Error Handling Patterns for Message Operations (18m)
  Build try-except blocks around send operations, handle network disconnections gracefully, and implement retry logic for failed message deliveries.
- Building a Simple Command-Based Bot (22m)
  Combine everything so far into a bot that responds to /help, /status, and /echo commands. Includes command parsing, argument handling, and formatted responses.

Advanced Features and Customization

- Sending Images and Photos (18m)
  Load images from local files and URLs, attach them to messages, and add captions. Handles format conversion and file size considerations.
- Sending Videos and Audio Files (16m)
  Extend media handling to video and audio. Covers supported formats, duration metadata, and streaming large files without blocking your bot.
- Sending Documents and File Attachments (14m)
  Attach PDFs, spreadsheets, and other document types to messages. Set filenames and MIME types so recipients see proper file previews.
- Receiving and Saving Incoming Media (20m)
  Download images, videos, and documents from incoming messages. Save them to disk with proper file extensions and organize by contact or date.
- Group Management: Creating and Configuring Groups (22m)
  Create new WhatsApp groups programmatically, set group names and descriptions, and configure group settings through the API.
- Adding and Removing Group Participants (18m)
  Manage group membership by adding contacts, removing participants, and handling join requests. Includes permission checks and error cases.
- Group Messaging and Announcement Patterns (16m)
  Send messages to groups, handle group-specific events like participant joins, and build announcement broadcast systems.
- Managing Admin Permissions in Groups (14m)
  Promote and demote group admins programmatically. Build moderation bots that enforce rules based on admin status.
- Building Async Bots with NewAClient (25m)
  Switch from synchronous to async patterns using NewAClient. Handle multiple conversations concurrently without blocking, which is essential for high-traffic bots.
- Event-Driven Patterns for Complex Workflows (22m)
  Chain multiple events together to build workflows. Example: receive a message, process it, query a database, and send a formatted response.
- Database Integration with SQLite for Session Storage (28m)
  Store conversation state, user preferences, and message history in SQLite. Covers schema design, connection pooling, and query patterns for bot data.
- Using PostgreSQL for Production Database Storage (24m)
  Scale up from SQLite to PostgreSQL for production deployments. Set up connections, handle migrations, and optimize queries for message-heavy workloads.
- Contact and User Information Retrieval (15m)
  Query contact details, check if numbers are registered on WhatsApp, and fetch profile information to personalize bot interactions.

- Blocklist Management and Spam Prevention (12m)
  Implement block and unblock operations. Build rate limiting and spam detection to protect your bot from abuse and keep your account in good standing.

Real-World Applications and Best Practices

- Project: Automated Notification Bot (32m)
  Build a complete notification bot that sends alerts based on external triggers. Integrates with a webhook receiver and handles message queuing.
- Project: Customer Support Bot with Menu Flow (28m)
  Create a support bot with menu-driven navigation, FAQ responses, and escalation to human agents when the bot can't answer a question.
- Project: Group Management and Moderation Tool (30m)
  Build a group admin bot that welcomes new members, enforces rules, removes spam, and provides group statistics on request.
- Project: Media Processing and Auto-Response Bot (25m)
  Process incoming images and documents automatically. Extract text from images, convert file formats, and respond with processed results.
- Project: Scheduling and Reminder System (22m)
  Build a bot that accepts natural language time inputs and sends scheduled reminders. Covers cron-style scheduling and timezone handling.
- Deploying Neonize Bots on a VPS (20m)
  Move your bot from local development to a cloud server. Covers Linux setup, systemd service configuration, and automatic restart on failure.
- Containerizing Your Bot with Docker (24m)
  Write a Dockerfile for your Neonize bot, manage persistent session data with volumes, and use docker-compose for multi-service deployments.
- Monitoring Bot Health and Uptime (18m)
  Set up health checks, uptime monitoring, and alerting. Know when your bot disconnects or stops responding before your users do.
- Logging Best Practices for Production Bots (16m)
  Structure logs for debugging and analysis. Covers log rotation, structured JSON logging, and integrating with log aggregation services.
- Performance Optimization for High-Traffic Bots (22m)
  Handle thousands of messages per hour. Covers connection pooling, message batching, memory management, and profiling your bot's bottlenecks.
- Security Hardening for WhatsApp Automation (18m)
  Protect session credentials, validate incoming message content, prevent injection attacks, and follow security best practices for internet-facing bots.
- Testing Strategies for Neonize Bots (20m)
  Write unit tests for message handlers, mock WhatsApp events for integration testing, and set up CI pipelines that validate your bot code automatically.

- Contributing to Neonize and Using Community Resources (14m)
  Navigate the Neonize GitHub repository, report issues effectively, submit pull requests, and tap into community examples and support channels.
- Where to Go Next: Scaling and Expanding Your Automation (12m)
  Review what you've built and explore next steps: multi-account management, integration with business tools, and patterns for growing your automation projects.

Discover our blog

Learn more with free articles written by our professors and guest experts.

Latest article: **Flight Attendant Preparation: What to Study Before You Apply**

Visit the blog All courses This course