# Secure Password Generator (CLI)

## A Cryptographically-Secure Python Command-Line Tool

## Project Overview

### Python-Based CLI Tool

Leverages Python for cross-platform compatibility and ease of development.

### Cryptographic Security

Utilizes Python's `secrets` module for generating cryptographically strong random numbers.

### Cross-Platform Compatibility

Functions across Linux, macOS, and Windows terminals, providing broad accessibility.

### Zero External Dependencies

Operates independently, requiring no additional libraries for installation and execution.

## Why It's Needed

**Weak Passwords Lead to Security Risks**

Weak passwords cause compromised accounts and data breaches. Robust password generation is essential.

**Users Often Reuse or Manually Create Predictable Passwords**

Users often reuse or create predictable passwords for convenience. This dramatically increases vulnerability.

## Key Features: Comprehensive Customization

**Fully Customizable Password Generation**

Tailor password complexity and character sets precisely, including length and character inclusion.

**Supports Lowercase, Uppercase, Digits, Symbols**

Choose from lowercase, uppercase, digits, and symbols to increase password strength.

**Custom Symbol Sets**

Define custom special characters for greater control over password composition.

**Exclude Characters & Remove Ambiguous Characters**

Exclude specific characters and remove ambiguous ones (e.g., O/0, I/l/1) for better readability.

**Enforce At Least One of Each Chosen Character Class**

Guarantee passwords contain a mix of selected character types, meeting complex requirements.

**No-Repeat Mode & URL-Safe Mode**

Generate passwords with no duplicate characters (no-repeat mode) or URL-friendly characters (URL-safe mode).

**Prefix and Suffix Support**

Add strings at the beginning or end of passwords for branding or identification.

**Generate One or Many Passwords**

Produce single passwords or batches of credentials with one command, ideal for bulk account creation.

## Security Features

### Uses `secrets.choice()` and `secrets.randbelow`

Employs Python's `secrets` module for cryptographically secure random number generation.

### Avoids Python's Insecure `random` Module

Avoids Python's `random` module, which is not suitable for cryptography.

### Supports Entropy Calculation

Measures password randomness and unpredictability. Allows users to assess password strength.

### Clipboard Integration for Secure Transfer

Copies passwords to the system clipboard using native tools (`wl-copy`, `xclip`, `pbcopy`). Minimizes exposure and manual errors.

## Output Formats: Adaptable and Convenient

**Plain Text**

Simple, unformatted output for quick viewing.

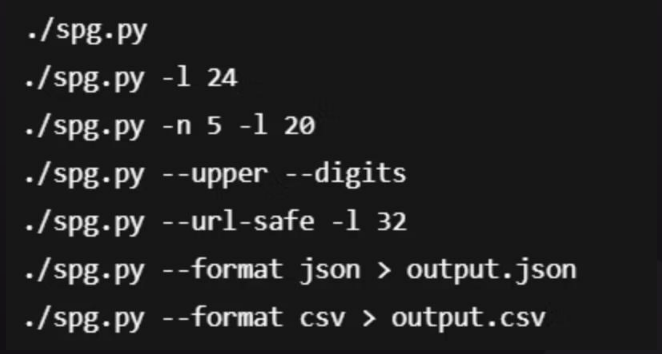
**JSON (JavaScript Object Notation)**

Structured data output, ideal for programmatic parsing and integration.

**CSV (Comma-Separated Values)**

Tabular data output, perfect for spreadsheets and databases.

## Example Commands



```
./spg.py
./spg.py -l 24
./spg.py -n 5 -l 20
./spg.py --upper --digits
./spg.py --url-safe -l 32
./spg.py --format json > output.json
./spg.py --format csv > output.csv
```

**Generate a basic 12-character password:**

```
passwordgen -l 12
```

**Generate a 16-character password with symbols and no ambiguous characters:**

```
passwordgen -l 16 -s --no-ambiguous
```

**Generate 5 passwords, 20 characters long, with a custom symbol set:**

```
passwordgen -n 5 -l 20 -c "!@#$%^&*"
```

## How It Works Internally

### Builds Character Pool

A pool of eligible characters is created based on user options.

### Generates Requested Number of Passwords

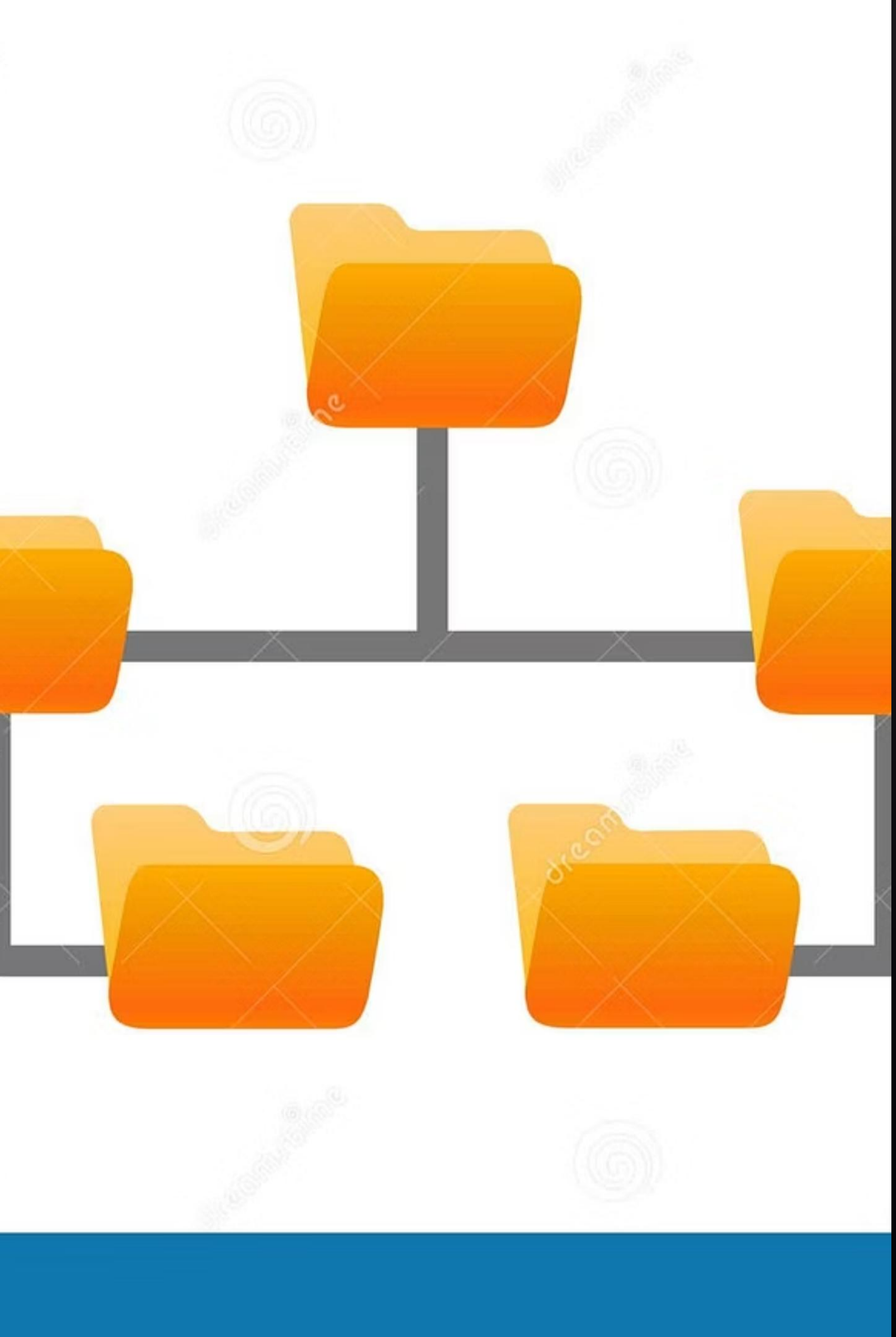
Passwords are constructed from the shuffled pool. The process repeats if multiple passwords are requested.

### Ensures Required Classes and Constraints

The tool verifies all specified requirements are met, such as including diverse character types.

### Uses Secure Shuffle Algorithm

A cryptographically secure shuffling algorithm rearranges the character pool for randomness and to prevent pattern detection.



## Project Structure

```
spg.py        # main CLI script
README.md     # documentation
```

1 **Root Directory**

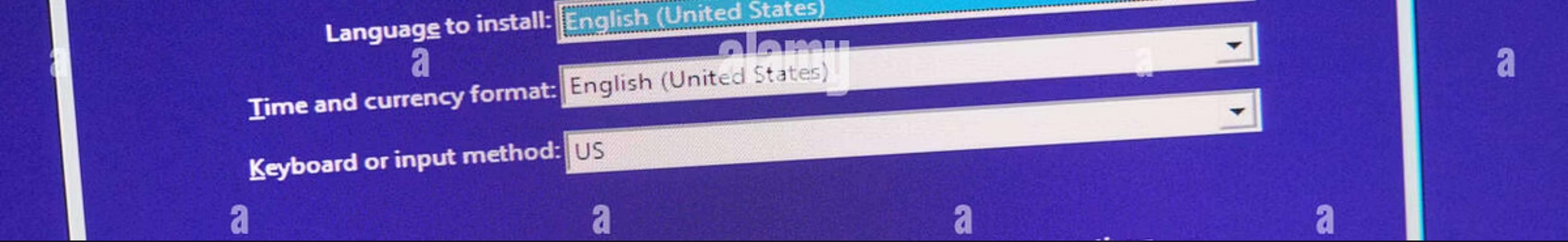
Main application files, configuration, and documentation.

2 **Docs/**

Usage instructions, examples, and contributing guidelines.

3 **Tests/**

Unit and integration tests for code quality and functionality.


Language to install: English (United States)
Time and currency format: English (United States)
Keyboard or input method: US

## Installation

Getting started with the Secure Password Generator is quick and straightforward, allowing you to begin generating strong passwords in minutes.

```
chmod +x spg.py
sudo mv spg.py /usr/local/bin/spg
```

01

### Clone the Repository

Clone the project repository from GitHub to your local machine.

`git clone [repository-url]`

02

### Navigate to Project Directory

Change your current directory to the newly cloned project folder.

`cd secure-password-generator`

03

### Run the Tool

Execute the Python script directly from your terminal.

`python passwordgen.py --help`