Migration Guide: setup.py to pyproject.toml

Benefits of the Modern Approach

1. **Single Configuration File**: All project metadata, dependencies, and tool configurations in one place
2. **Standardized Format**: Following PEP 621 standard for Python packaging
3. **Better Tool Integration**: Modern tools like black, ruff, pytest all support pyproject.toml
4. **Cleaner Structure**: No need for setup.cfg, MANIFEST.in in most cases
5. **Better Dependency Resolution**: More reliable builds and installations

Migration Steps

1. Create pyproject.toml

Copy the pyproject.toml file from the artifact above to your project root.

2. Update version.py

Update your `schwab_extra/version.py` to use the simpler format shown above.

3. Keep Minimal setup.py (Optional)

Keep the minimal setup.py for backward compatibility with older tools.

4. Remove setup.cfg

Most of the configuration from setup.cfg is now in pyproject.toml. You can remove it after migration.

5. Test the Build

bash

```bash
# Install build tool
pip install build

# Build the package
python -m build

# This creates dist/ folder with wheel and source distribution
```

6. Install in Development Mode

```bash
# From your project directory
pip install -e .

# Or with dev dependencies
pip install -e ".[dev]"
```

Key Differences

Old Way (setup.py):

- Configuration spread across setup.py, setup.cfg, MANIFEST.in
- Version read from file with custom code
- Tool configs in separate files

New Way (pyproject.toml):

- All configuration in one file
- Version handled by setuptools dynamically
- Tool configurations included (black, ruff, pytest, coverage)

Modern Development Tools

The pyproject.toml includes configurations for:

- **black**: Code formatting (replaces brunette)
- **ruff**: Fast linting (replaces flake8)
- **pytest**: Testing with coverage
- **mypy**: Type checking

Building and Publishing

```bash
# Build the package
python -m build

# Upload to PyPI (when ready)
python -m twine upload dist/*
```

Windows Specific Notes

Since you're on Windows, make sure to:

1. Use `python -m` prefix for commands
2. Scripts will be installed to `Scripts/` folder in your Python environment
3. All your console scripts will work the same way

Rich Console Integration

Your use of the `rich` package for console interaction works perfectly with this setup. No changes needed to your actual code!