Poetry Cheatsheet

Basic Concepts

Poetry is a tool for dependency management and packaging in Python. It handles virtual environments, dependency resolution, and publishing in a single tool.

- Configuration: `pyproject.toml` (Standard-compliant file for project metadata).
- Lock File: `poetry.lock` (Ensures reproducible environments).

Project Management

- `poetry init`: Interactively create a `pyproject.toml` in an existing project.
- `poetry new [project_name]`: Create a new project structure.
- `poetry install`: Install dependencies from `pyproject.toml` and `poetry.lock`.
- `poetry update`: Update dependencies to latest compatible versions and refresh lock file.

Dependency Management

- `poetry add [package]`: Install and add a package to requirements.
  - `--group dev`: Add to development group (useful for testing tools).
- `poetry remove [package]`: Uninstall and remove a package.
- `poetry show`: List all installed packages.
  - `--tree`: Display dependency tree.

Environment and Execution

- `poetry run [command]`: Run a command within the context of the virtual environment (e.g., `poetry run python main.py`).
- `poetry shell`: Activate the project's virtual environment.
- `poetry env info`: Show information about the current virtual environment.
- `poetry env list`: List all virtual environments associated with the project.
- `poetry env use python3.10`: Switch to a specific Python version.

Building and Publishing

- `poetry build`: Build source and wheel distributions.
- `poetry publish`: Upload the built package to a remote repository (default: PyPI).
- `poetry config`: Manage poetry configuration settings (e.g., API tokens).

Configuration (pyproject.toml)

```
[tool.poetry]
name = "my_project"
version = "0.1.0"
description = ""
authors = ["Your Name
<email@example.com>"]

[tool.poetry.dependencies]
python = "^3.10"
requests = "^2.28"

[tool.poetry.group.dev.dependencies]
pytest = "^7.0"
```

Version Constraints

- `^2.1.0`: Compatible with `2.1.0` (up to `<3.0.0`).
- `~1.2`: Compatible with `1.2.x` (up to `<1.3.0`).
- `⩾1.5`: Any version greater than or equal to `1.5`.
- `*`: Any version.

Pro Tips

Virtual Environment Location

By default, Poetry creates virtual environments in a centralized folder. Change this to the project root for IDE compatibility: `poetry config virtualenvs.in-project true`

Exporting Requirements

Convert your dependencies to standard `requirements.txt` format: `poetry export -f requirements.txt --output requirements.txt`

Reproducibility

Always commit your `poetry.lock` file to version control. This ensures every developer stays on the exact same package versions.