[`PIP` Commands] (CheatSheet)

1. Basic Package Installation

- **Install a Package**: pip install package_name
- **Install a Specific Version**: pip install package_name==1.0.4
- **Install Packages from Requirements File**: pip install -r requirements.txt
- **Install to User Space**: pip install --user package_name

2. Package Removal

- **Uninstall a Package**: pip uninstall package_name
- **Uninstall Multiple Packages**: pip uninstall package_one package_two

3. Package Information

- **Show Package Information**: pip show package_name
- **List Installed Packages**: pip list
- **List Outdated Packages**: pip list --outdated
- **Search PyPI for Packages**: pip search search_query

4. Package Versions and History

- **Check a Specific Package's Available Versions**: pip install package_name==
- **View Installation History**: pip freeze
- **Save Installed Packages to Requirements File**: pip freeze > requirements.txt

5. Managing Package Sources

- **Install from a Specific Source**: pip install -i https://pypi.org/simple/package_name
- **Install from a Git Repository**: pip install git+https://github.com/user/repo.git
- **Install from a Local Directory**: pip install /path/to/directory

6. Handling Dependencies

- **Ignore Dependencies**: pip install --no-deps package_name

By: Waleed Mousa

- **Force Reinstall a Package and Dependencies**: pip install --force-reinstall package_name
- **Upgrade a Package and its Dependencies**: pip install --upgrade package_name

7. Environment and Configuration

- **Using Virtual Environments**: (Typically used with virtualenv or venv)
- **Configuration with pip.conf**: (Edit pip.conf for custom configuration)
- **Check pip Configuration**: pip config list

8. Advanced Installation Options

- **Install Pre-release Version**: pip install --pre package_name
- **Choose Python Version**: pip2 or pip3 for specific Python versions
- **Install with Compile Option**: pip install --compile package_name
- **Use Wheel to Install**: pip install --use-wheel package_name

9. Handling Requirements Files

- **Compile a Requirements File**: pip-compile
- **Upgrade All Packages in Requirements File**: pip install -r requirements.txt --upgrade
- **Generate a requirements file from an environment**: pip freeze > requirements.txt

10. Caching

- **Disable Cache**: pip install --no-cache-dir package_name
- **Inspect Cache Info**: pip cache info

11. Downloading Packages

- **Download a Package Without Installing**: pip download package_name
- **Download All Dependencies**: pip download package_name -d /path/to/directory

12. Working with Wheels

- **Building Wheels**: pip wheel --wheel-dir=/path/to/wheelhouse -r requirements.txt

By: Waleed Mousa

- **Installing from Wheels**: `pip install --use-wheel --no-index --find-links=/path/to/wheelhouse/ package_name`
- **Prefer Wheel over Source**: `pip install --prefer-binary package_name`
- **Install from Local Wheel Archive**: `pip install --find-links=/path/to/wheel/directory/ package_name`
- **Forcing Source Distribution Installation**: `pip install --no-binary=:all: package_name`
- **Ignore the Wheel Cache**: `pip install --no-cache-dir package_name`

13. Environment Management

- **Export Current Environment**: `pip freeze > requirements.txt`
- **Install Packages to a Specific Python Version**: `python -m pip install package_name`
- **Using Pip with Conda Environments**: (Conda specific commands)

14. Security

- **Check for Vulnerable Packages**: `pip-audit`
- **Check Installed Package Safety**: `pip install safety; safety check`

15. Miscellaneous

- **Show Pip Version**: `pip --version`
- **List All Available Commands**: `pip --help`
- **Improve Install Speed**: `pip install package_name -vvv`
- **Verbose Mode for More Output**: `pip install --verbose package_name`
- **Use a Mirror**: `pip install -i https://<mirror_url> package_name`

16. Proxy Configuration

- **Install Using a Proxy**: `pip install --proxy http://user:password@proxy.server:port package_name`

17. Troubleshooting and Help

- **Check for pip Problems**: `pip check`
- **Show Help for Specific Command**: `pip install --help`

18. Upgrading Pip

By: Waleed Mousa

- **Upgrade Pip Itself**: `pip install --upgrade pip`

19. Pip and Python Interaction

- **Show Where Python Packages Are Installed**: `pip show -f package_name`
- **Execute a Python Script with Pip's Python**: `python -m pip execute_script.py`

20. Build Isolation

- **Disable Build Isolation**: `pip install --no-build-isolation package_name`

21. Hash Checking

- **Install with Hash Checking**: `pip install --require-hashes -r requirements.txt`

22. Wheel Building

- **Building Wheel for a Package**: `pip wheel package_name`

23. Using Different Python Versions

- **Install Package for Python2 or Python3 Specifically**: `python2 -m pip install package_name`, `python3 -m pip install package_name`

24. Output Control

- **Quiet Mode (Less Output)**: `pip install --quiet package_name`
- **Verbose Mode (More Output)**: `pip install --verbose package_name`

25. Requirements File Options

- **Only Install if Needed**: `pip install -r requirements.txt --ignore-installed`

26. Using Tags and Extras

- **Install Extras of a Package**: `pip install package_name[extra1,extra2]`

27. Environment Variables for Pip

By: Waleed Mousa

- **Setting Custom Cache Directory**: `PIP_CACHE_DIR='/path/to/cache' pip install package_name`
- **Setting Custom Config File Location**: `PIP_CONFIG_FILE='/path/to/pip.conf' pip install package_name`

28. Source Control Systems

- **Install from VCS (Version Control System)**: `pip install git+https://git.repo/some_pkg.git`

29. Platform Specific Options

- **Only Binary or Source**: `pip install --only-binary=:all: package_name`, `pip install --no-binary=:all: package_name`

30. Logging and Reporting

- **Log Output to File**: `pip install package_name --log log.txt`

31. Running Scripts

- **Running a Python Script using Pip's Python**: `python -m pip run script.py`

32. Setting Custom Target Directory

- **Install to a Specific Directory**: `pip install --target=/path/to/directory package_name`

33. Working with Multiple Python Environments

- **Specify Python Version via Pip**: `pip2` or `pip3`

34. Integrity and Hash Checking

- **Verify Package Integrity with Hashes**: `pip install package_name --hash=sha256:hashvalue`

35. Dependency Resolution

- **Control Dependency Resolution Strategy**: `pip install --use-deprecated=legacy-resolver package_name`

By: Waleed Mousa

36. Performance and Optimization

- **Avoiding Repetitive Work**: `pip install --no-deps --force-reinstall package_name`

37. Specifying Package Indexes

- **Install from Specific Index**: `pip install --index-url https://simple.crate.io/ package_name`

38. Managing Scripts and Entry Points

- **Inspecting Entry Points**: `pip show --entry-points package_name`

39. Locking Dependencies

- **Generating a Lock File for Dependencies**: `pip freeze > requirements.lock`

40. Cleaning Up Cache

- **Remove Cache**: `pip cache purge`

41. Dry Runs and Simulations

- **Dry Run Installation**: `pip install --dry-run package_name`

42. Excluding Packages

- **Exclude Certain Packages During Install**: `pip install package_name --exclude package_to_exclude`

43. Specifying a Platform

- **Install for a Specific Platform**: `pip install package_name --platform=manylinux1_x86_64`

44. Custom Install Commands

- **Custom Install Command with Setuptools**: `pip install --install-option='custom_option' package_name`

By: Waleed Mousa

45. Editing Mode

- **Install a Package in Editable Mode**: `pip install -e .`

46. Working with Archived Packages

- **Installing from a .whl File**: `pip install some_package.whl`

47. Managing Dependency Links

- **Install Using Dependency Links**: `pip install --process-dependency-links package_name`

48. Ignoring Installed Packages

- **Ignore Already Installed Packages**: `pip install --ignore-installed package_name`

49. Path and Environment Management

- **Print Python and Pip Executable Path**: `pip --version`

50. Working with Pre-releases

- **Install Pre-releases**: `pip install --pre package_name`

51. Customizing Build Options

- **Customize Build with Global Options**: `pip install --global-option=build_ext --global-option="-I/usr/local/include" package_name`

52. Package Discovery

- **Discovering Package Resources**: `pip show --files package_name`
- **Inspecting Package Dependencies**: `pip show --requires package_name`

53. Requirement Specifiers

- **Specifying Exact, Minimum, Compatible Versions**: `pip install "package_name>=1.0.4,<2.0"`

By: Waleed Mousa

54. Handling SSL Certificate Verification

- **Disable SSL Certificate Verification**: `pip install --trusted-host pypi.org package_name`

55. Path Manipulation for Modules

- **Modifying Python Path**: `PYTHONPATH=/path/to/library pip <command>`

56. Isolating Installation

- **Using `--user` to Isolate Installations**: `pip install --user package_name`

57. Verbose and Quiet Modes

- **Increase Verbosity**: `pip -v install package_name`
- **Decrease Verbosity**: `pip -q install package_name`

58. Working with Local Project

- **Install Package from Local Source in Editable Mode**: `pip install -e path/to/SomeProject`

59. Using Constraints Files

- **Installing with Constraints**: `pip install -c constraints.txt package_name`

60. Managing Local Packages

- **List All Local Packages**: `pip list --local`

61. Using Environment Markers

- **Specifying Environment Markers**: `pip install "package_name; platform_system == 'Windows'"`

62. Using Post-Install Scripts

- **Running Post-Install Scripts**: `pip install --install-option="--post-install-script" package_name`

By: Waleed Mousa

63. Handling Permissions and User Site

- **Prevent Installing to the User Site**: `pip install --no-user package_name`

64. Customizing Python Used by Pip

- **Customize Python Interpreter**: `pip --python-version 3.x <command>`

65. Cleaning Up After Installation

- **Cleaning Unneeded Files Post Installation**: `pip install --clean package_name`

66. Handling Configuration Files

- **Using a Custom Configuration File**: `pip --config-file /path/to/pip.ini <command>`

67. Controlling Installation from Local or Remote

- **Prefer Local/Remote Packages**: `pip install --prefer-local/--prefer-remote package_name`

68. Handling Network Issues and Retries

- **Specify Number of Retries for a Command**: `pip --retries 5 install package_name`

69. Working with Proxy Servers

- **Install Using a Proxy Server**: `pip --proxy username:password@proxy.server:port install package_name`

70. Specifying Package Index and Extra Indexes

- **Using Specific Package Indexes**: `pip install --index-url URL --extra-index-url EXTRA_URL package_name`

71. Working with Legacy Resolvers

By: Waleed Mousa

- **Use Legacy Dependency Resolver**: `pip install --use-deprecated=legacy-resolver package_name`

72. Handling Warnings and Errors

- **Ignoring All Pip Warnings**: `pip install --no-warn-script-location package_name`