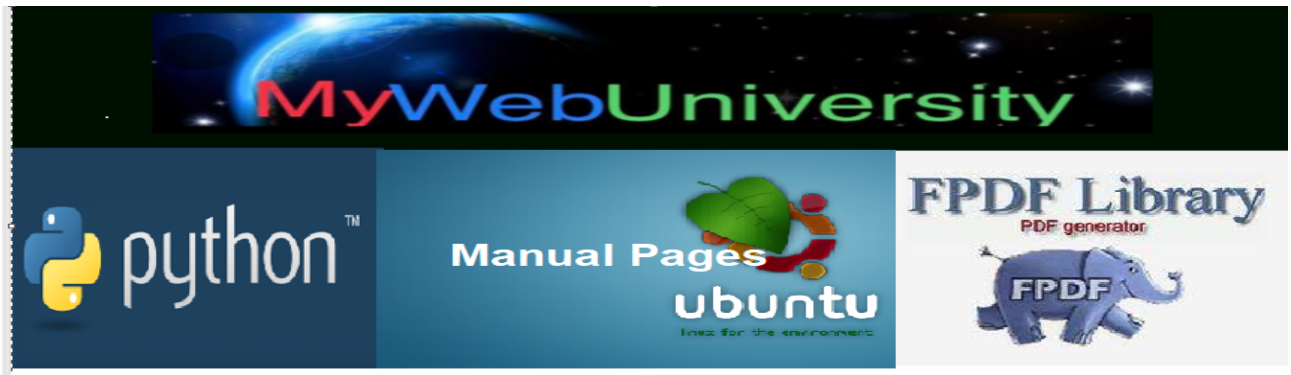

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'pip3-download.1'

$ man pip3-download.1

PIP3-DOWNLOAD(1) pip PIP3-DOWNLOAD(1)

NAME

pip3-download - description of pip3 download command

DESCRIPTION

Download packages from:

? PyPI (and other indexes) using requirement specifiers.

? VCS project urls.

? Local project directories.

? Local or remote source archives.

pip also supports downloading from "requirements files", which provide an easy way to spec?

ify a whole environment to be downloaded.

USAGE

python -m pip download [options] <requirement specifier> [package-index-options] ...

python -m pip download [options] -r <requirements file> [package-index-options] ...

python -m pip download [options] <vcs project url> ...

python -m pip download [options] <local project path> ...

python -m pip download [options] <archive url/path> ...

OPTIONS

-c, --constraint <file>

Constrain versions using the given constraints file. This option can be used multiple

times.

(environment variable: PIP_CONSTRAINT)

-r, --requirement <file>

Install from the given requirements file. This option can be used multiple times.

(environment variable: PIP_REQUIREMENT)

--no-deps

Don't install package dependencies.

(environment variable: PIP_NO_DEPS, PIP_NO_DEPENDENCIES)

--global-option <options>

Extra global options to be supplied to the setup.py call before the install or bdist_wheel command.

(environment variable: PIP_GLOBAL_OPTION)

--no-binary <format_control>

Do not use binary packages. Can be supplied multiple times, and each time adds to the existing value. Accepts either ":all:" to disable all binary packages, ":none:" to empty the set (notice the colons), or one or more package names with commas between them (no colons). Note that some packages are tricky to compile and may fail to in? stall when this option is used on them.

(environment variable: PIP_NO_BINARY)

--only-binary <format_control>

Do not use source packages. Can be supplied multiple times, and each time adds to the existing value. Accepts either ":all:" to disable all source packages, ":none:" to empty the set, or one or more package names with commas between them. Packages with? out binary distributions will fail to install when this option is used on them.

(environment variable: PIP_ONLY_BINARY)

--prefer-binary

Prefer binary packages over source packages, even if the source packages are newer.

(environment variable: PIP_PREFER_BINARY)

--src <dir>

Directory to check out editable projects into. The default in a virtualenv is "<venv path>/src". The default for global installs is "<current dir>/src".

(environment variable: PIP_SRC, PIP_SOURCE, PIP_SOURCE_DIR, PIP_SOURCE_DIRECTORY)

--pre Include pre-release and development versions. By default, pip only finds stable ver? sions.

(environment variable: PIP_PRE)

--require-hashes

Require a hash to check each requirement against, for repeatable installs. This op?
tion is implied when any package in a requirements file has a --hash option.
(environment variable: PIP_REQUIRE_HASHES)

--progress-bar <progress_bar>
Specify whether the progress bar should be used [on, off, raw] (default: on)
(environment variable: PIP_PROGRESS_BAR)

--no-build-isolation
Disable isolation when building a modern source distribution. Build dependencies
specified by PEP 518 must be already installed if this option is used.
(environment variable: PIP_NO_BUILD_ISOLATION)

--use-pep517
Use PEP 517 for building source distributions (use --no-use-pep517 to force legacy
behaviour).
(environment variable: PIP_USE_PEP517)

--check-build-dependencies
Check the build dependencies when PEP517 is used.
(environment variable: PIP_CHECK_BUILD_DEPENDENCIES)

--ignore-requires-python
Ignore the Requires-Python information.
(environment variable: PIP_IGNORE_REQUIRES_PYTHON)

-d, --dest <dir>
Download packages into <dir>.
(environment variable: PIP_DEST, PIP_DESTINATION_DIR, PIP_DESTINATION_DIRECTORY)

--platform <platform>
Only use wheels compatible with <platform>. Defaults to the platform of the running
system. Use this option multiple times to specify multiple platforms supported by the
target interpreter.
(environment variable: PIP_PLATFORM)

--python-version <python_version>
The Python interpreter version to use for wheel and "Requires-Python" compatibility
checks. Defaults to a version derived from the running interpreter. The version can
be specified using up to three dot-separated integers (e.g. "3" for 3.0.0, "3.7" for
3.7.0, or "3.7.3"). A major-minor version can also be given as a string without dots

(e.g. "37" for 3.7.0).

(environment variable: PIP_PYTHON_VERSION)

--implementation <implementation>

Only use wheels compatible with Python implementation <implementation>, e.g. 'pp', 'jy', 'cp', or 'ip'. If not specified, then the current interpreter implementation is used. Use 'py' to force implementation-agnostic wheels.

(environment variable: PIP_IMPLEMENTATION)

--abi <abi>

Only use wheels compatible with Python abi <abi>, e.g. 'pypy_41'. If not specified, then the current interpreter abi tag is used. Use this option multiple times to specify multiple abis supported by the target interpreter. Generally you will need to specify --implementation, --platform, and --python-version when using this option.

(environment variable: PIP_ABI)

--group <[path:]group>

Install a named dependency-group from a "pyproject.toml" file. If a path is given, the name of the file must be "pyproject.toml". Defaults to using "pyproject.toml" in the current directory.

(environment variable: PIP_GROUP)

--no-clean

Don't clean up build directories.

(environment variable: PIP_NO_CLEAN)

AUTHOR

pip developers

COPYRIGHT