Oracle Linux 8

Installing and Managing Python

F36403-12
January 2026

Oracle Linux 8 Installing and Managing Python,

F36403-12

Copyright © 2020, 2026, Oracle and/or its affiliates.

Documentation License
The content in this document is licensed under the Creative Commons Attribution–Share Alike 4.0 (CC-BY-SA) license. In accordance with CC-BY-SA, if you distribute this content or an adaptation of it, you must provide attribution to Oracle and retain the original copyright notices.

Contents

Copyright © 2020, 2026, Oracle and/or its affiliates.

Preface

Oracle Linux 8: Installing and Managing Python describes how to install and configure a Python runtime environment so that you can run applications and scripting tools that require a Python interpreter to function.

Copyright © 2020, 2026, Oracle and/or its affiliates.

1

About Python

Python is a high-level general purpose programming language that relies on an interpreter to fulfill scripted functions. On Oracle Linux 8, many system utilities, tools for data analysis and web applications rely on the presence of a Python runtime environment to function.

Python 2 is no longer maintained by the Python community; Python 2 is also no longer supported on Oracle Linux 8. Any existing Python 2 scripts must be migrated to Python 3 or run inside a container. For more information about running scripts inside containers, see Oracle Linux: Podman User's Guide.

To learn about the `2to3` automated migration tool visit https://docs.python.org/3/library/2to3.html. See also the porting guide at https://portingguide.readthedocs.io/en/latest/ for more in-depth information.

You can read more information about creating Python scripts at https://www.python.org/doc/.

> **Note**
>
> The `python` command isn't aliased by default in Oracle Linux 8. Specify the Python interpreter version with the `python3` command when you run Python scripts.
>
> For more information, see Installing Python.

Copyright © 2020, 2026, Oracle and/or its affiliates.

2 Installing Python

To install Python 3.6 on an Oracle Linux 8 system:

```
sudo dnf install python3
```

The `python3` command points to Python 3.6 if it's installed on the system. To verify that behavior, run the following command:

```
python3 --version
```

The `python` command isn't aliased by default in Oracle Linux 8. Specify the Python interpreter version with the `python3` command when you run Python scripts.

If you need to alias the `python` command to fix compatibility problems with existing scripts and applications, you can set it manually. For example, to set Python 3 as the default interpreter version, run the following command:

```
sudo alternatives --set python /usr/bin/python3
```

For more information about using the `python` command, see the `python(1)` manual page.

> **Note**
>
> Python 3.6 is maintained for the full lifespan of Oracle Linux 8.
>
> Application Stream packages, such as Python 2.7 and more recent versions of Python 3, have their own major version releases and have shorter maintenance lifespans. For more information, see Oracle Linux: Product Life Cycle Information.

If the system is running Oracle Linux 8.8 or later, you can optionally also install Python 3.11:

```
sudo dnf install python3.11
```

If the system is running Oracle Linux 8.10 or later, you can optionally also install Python 3.12:

```
sudo dnf install python3.12
```

The `python3` command is aliased to Python 3.6 by default, so for newer versions of Python 3 you must explicitly reference the correct binary. For example, the following command verifies that Python 3.12 is installed on the system:

```
python3.12 --version
```


Copyright © 2020, 2026, Oracle and/or its affiliates.

Or you can edit the default Python 3 version by using the `alternatives` command. For example, to remap the `python3` command to use Python 3.12, run the following command:

```
sudo alternatives --set python3 /usr/bin/python3.12
```

Don't uninstall Python 3.6 after making that change, as the command must be reverted when the newer version of Python 3 reaches the end of its maintenance lifespan. Keeping Python 3.6 in place also ensures that the command can be reverted to troubleshoot or resolve problems with systems and components in Oracle Linux 8 that were tested against Python 3.6.

> **Note**
>
> Newer versions of Python might be available in the `ol8_developer_EPEL` yum repository. Those packages are intended for development purposes only, so we recommend that you only install them in development environments. In that scenario you could optionally use `pip` to install extra Python libraries that haven't been packaged yet. For more information, see Installing Third-Party Packages.

Installing Extra Python Libraries

You can also install extra dependencies from the Oracle Linux yum server. For example, to install the `requests` library for the default runtime version of Python 3, you would install the `python3-requests` package:

```
sudo dnf install python3-requests
```

To install dependency packages for specific Python versions, add the runtime version to the package name. For example, to install the `requests` library for Python 3.12, run the following command:

```
sudo dnf install python3.12-requests
```

Dependencies that are installed in this way are available for any compatible Python installations on the same system. In addition, any matching packages can also be removed without also removing existing Python installations.

Copyright © 2020, 2026, Oracle and/or its affiliates.

3

Installing Third-Party Packages

Before installing a third-party package, verify if you can install the Python library you need from the Oracle Linux yum server. For example, to check if the `requests` library has been provided for Python 3.12, run the following command:

```
sudo dnf search python3.12-requests
```

For more information about installing extra Python libraries from the Oracle Linux yum server, read Installing Extra Python Libraries.

If you can't find a particular dependency on the Oracle Linux yum server, or if the script that you need to run requires a newer version of the dependency than the installed package already provides, you can optionally use the `pip` package manager to install it from a third-party source.

To ensure that the system remains supported, for each project you can install and run third-party packages in an isolated virtual environment created with the `venv` Python module.

To learn more about installing third-party packages inside Python virtual environments, visit https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/.

Installing Pip Libraries With Python 3

1. Install base packages for the `pip3` command:

   ```
   sudo dnf install python3-pip python3-setuptools python3-wheel
   ```

2. Create a Python virtual environment. For example, the following command creates a Python 3 virtual environment named `example3`:

   ```
   python3 -m venv --system-site-packages example3
   ```

3. You can now activate the `example3` environment and begin installing third-party dependencies. For example, to install a newer version of the `requests` library for Python 3:

   ```
   source example3/bin/activate
   ```

   ```
   python3 -m pip install --user requests
   ```


Copyright © 2020, 2026, Oracle and/or its affiliates.

NOT_SUPPORTED

Using the `pip3` command outside of a Python virtual environment applies changes system-wide, and that can impact compatibility with some installed packages in an Oracle Linux 8 installation.

Add the `--user` flag to any `pip3 install` commands to ensure that dependency packages are only available to the current user.

4. To run compatible scripts with the third-party packages that have been installed, run them from within the same Python virtual environment.

Installing Pip Libraries With Versioned Python 3

You can also install third-party dependencies for newer releases of Python 3 by using versioned `pip` commands. For example, to install Pip libraries for Python 3.12, follow these instructions:

1. Install base packages for the `pip3.12` command:

   ```
   sudo dnf install python3.12-pip python3.12-setuptools python3.12-wheel
   ```

2. Create a Python virtual environment. For example, the following command creates a Python 3.12 virtual environment named `example4`:

   ```
   python3.12 -m venv --system-site-packages example4
   ```

3. You can now activate the `example4` environment and begin installing third-party dependencies. For example, to install a newer version of the `requests` library for Python 3.12:

   ```
   source example4/bin/activate
   ```

   ```
   python3.12 -m pip install --user requests
   ```

   NOT_SUPPORTED

   Using the `pip3.12` command outside of a Python virtual environment applies changes system-wide.

   Add the `--user` flag to any `pip3.12 install` commands to ensure that dependency packages are only available to the current user.

4. To run compatible scripts with the third-party packages that have been installed, run them from within the same Python virtual environment.

Copyright © 2020, 2026, Oracle and/or its affiliates.