# User's Guide » Installation on Linux

## Using software packages

For installing distribution relative packages .deb/.rpm/…

### Ubuntu / Kubuntu / Xubuntu / Lubuntu (Saucy and above)

1. Add one of the PPAs as you prefer

| | |
|---|---|
| **stable builds:** | $ sudo add-apt-repository ppa:kivy-team/kivy |
| **nightly builds:** | $ sudo add-apt-repository ppa:kivy-team/kivy-daily |

2. Update your package list using your package manager

   $ sudo apt-get update

3. Install Kivy

   **Python2 - python-kivy:**

   $ sudo apt-get install python-kivy

   **Python3 - python3-kivy:**

   $ sudo apt-get install python3-kivy

   **optionally the examples - kivy-examples:**

   $ sudo apt-get install kivy-examples

### Debian (Jessie or newer)

1. Add one of the PPAs to your sources.list in apt manually or via Synaptic

| | |
|---|---|
| **stable builds:** | deb http://ppa.launchpad.net/kivy-team/kivy/ubuntu xenial main |
| **daily builds:** | deb http://ppa.launchpad.net/kivy-team/kivy-daily/ubuntu xenial main |

   **Notice**: Wheezy is not supported - You'll need to upgrade to Jessie at least!

2. Add the GPG key to your apt keyring by executing

   as user:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys A863D2D6
```

   as root:

```
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys A863D2D6
```

3. Refresh your package list and install **python-kivy** and/or **python3-kivy** and optionally the examples found in **kivy-examples**

### Linux Mint

1. Find out on which Ubuntu release your installation is based on, using this overview.
2. Continue as described for Ubuntu above, depending on which version your installation is based on.

### Bodhi Linux

1. Find out which version of the distribution you are running and use the table below to find out on which Ubuntu LTS it is based.

| | |
|---|---|
| **Bodhi 1:** | Ubuntu 10.04 LTS aka Lucid (No packages, just manual install) |
| **Bodhi 2:** | Ubuntu 12.04 LTS aka Precise |
| **Bodhi 3:** | Ubuntu 14.04 LTS aka Trusty |
| **Bodhi 4:** | Ubuntu 16.04 LTS aka Xenial |

2. Continue as described for Ubuntu above, depending on which version your installation is based on.

#### OpenSuSE

1. To install kivy go to http://software.opensuse.org/package/python-Kivy and use the "1 Click Install" for your openSuse version. You might need to make the latest kivy version appear in the list by clicking on "Show unstable packages". We prefer to use packages by " devel:languages:python".
2. If you would like access to the examples, please select **python-Kivy-examples** in the upcoming installation wizard.

#### Gentoo

1. There is a kivy ebuild (kivy stable version)

   emerge Kivy

2. available USE-flags are:

   *cairo: Standard flag, let kivy use cairo graphical libraries. camera: Install libraries needed to support camera. doc: Standard flag, will make you build the documentation locally. examples: Standard flag, will give you kivy examples programs. garden: Install garden tool to manage user maintained widgets. gstreamer: Standard flag, kivy will be able to use audio/video streaming libraries. spell: Standard flag, provide enchant to use spelling in kivy apps.*

#### Other

For other distros, we recommend installing via pip as shown below.

## Installation in a Virtual Environment

### Common dependencies

#### Cython

Different versions of Kivy have only been tested up to a certain Cython version. It may or may not work with a later version.

| Kivy | Cython |
|---|---|
| 1.8 | 0.20.2 |
| 1.9 | 0.21.2 |
| 1.9.1 | 0.23 |
| 1.10.0 | 0.25.2 |

### Dependencies with SDL2

#### Ubuntu example

In the following command use "python" and "python-dev" for Python 2, or "python3" and "python3-dev" for Python 3.

```
# Install necessary system packages
sudo apt-get install -y \
    python-pip \
    build-essential \
    git \
    python \
    python-dev \
    ffmpeg \
    libsdl2-dev \
    libsdl2-image-dev \
    libsdl2-mixer-dev \
    libsdl2-ttf-dev \
    libportmidi-dev \
    libswscale-dev \
    libavformat-dev \
    libavcodec-dev \
    zlib1g-dev

# Install gstreamer for audio, video (optional)
sudo apt-get install -y \
    libgstreamer1.0 \
    gstreamer1.0-plugins-base \
    gstreamer1.0-plugins-good
```

**Note:** Depending on your Linux version, you may receive error messages related to the "ffmpeg" package. In this scenario, use "libav-tools " in place of "ffmpeg " (above), or use a PPA (as shown below):

```
- sudo add-apt-repository ppa:mc3man/trusty-media
- sudo apt-get update
- sudo apt-get install ffmpeg
```

### Installation

```
# Make sure Pip, Virtualenv and Setuptools are updated
sudo pip install --upgrade pip virtualenv setuptools

# Then create a virtualenv named "kivyinstall" by either:

# 1. using the default interpreter
virtualenv --no-site-packages kivyinstall

# or 2. using a specific interpreter
# (this will use the interpreter in /usr/bin/python2.7)
virtualenv --no-site-packages -p /usr/bin/python2.7 kivyinstall

# Enter the virtualenv
. kivyinstall/bin/activate

# Use correct Cython version here
pip install Cython==0.28.2

# Install stable version of Kivy into the virtualenv
pip install kivy
# For the development version of Kivy, use the following command instead
# pip install git+https://github.com/kivy/kivy.git@master
```

## Dependencies with legacy PyGame

### Ubuntu example

```
# Install necessary system packages
sudo apt-get install -y \
    python-pip \
    build-essential \
    mercurial \
    git \
    python \
    python-dev \
    ffmpeg \
    libsdl-image1.2-dev \
    libsdl-mixer1.2-dev \
    libsdl-ttf2.0-dev \
    libsmpeg-dev \
    libsdl1.2-dev \
    libportmidi-dev \
    libswscale-dev \
    libavformat-dev \
    libavcodec-dev \
    zlib1g-dev
```

### Fedora

```
$ sudo yum install \
    make \
    mercurial \
    automake \
    gcc \
    gcc-c++ \
    SDL_ttf-devel \
    SDL_mixer-devel \
    khrplatform-devel \
    mesa-libGLES \
    mesa-libGLES-devel \
    gstreamer-plugins-good \
    gstreamer \
    gstreamer-python \
    mtdev-devel \
    python-devel \
    python-pip
```

### OpenSuse

```
$ sudo zypper install \
    python-distutils-extra \
    python-gstreamer-0_10 \
    python-enchant \
    gstreamer-0_10-plugins-good \
    python-devel \
    Mesa-devel \
    python-pip
$ sudo zypper install -t pattern devel_C_C++
```

### Installation

```
# Make sure Pip, Virtualenv and Setuptools are updated
sudo pip install --upgrade pip virtualenv setuptools

# Then create a virtualenv named "kivyinstall" by either:

# 1. using the default interpreter
virtualenv --no-site-packages kivyinstall

# or 2. using a specific interpreter
# (this will use the interpreter in /usr/bin/python2.7)
virtualenv --no-site-packages -p /usr/bin/python2.7 kivyinstall

# Enter the virtualenv
. kivyinstall/bin/activate

pip install numpy

pip install Cython==0.28.2

# If you want to install pygame backend instead of sdl2
# you can install pygame using command below and enforce using
# export USE_SDL2=0. If kivy's setup can't find sdl2 libs it will
# automatically set this value to 0 then try to build using pygame.
pip install hg+http://bitbucket.org/pygame/pygame



# Install stable version of Kivy into the virtualenv
pip install kivy
# For the development version of Kivy, use the following command instead
pip install git+https://github.com/kivy/kivy.git@master
```

Install additional Virtualenv packages

```
# Install development version of buildozer into the virtualenv
pip install git+https://github.com/kivy/buildozer.git@master

# Install development version of plyer into the virtualenv
pip install git+https://github.com/kivy/plyer.git@master

# Install a couple of dependencies for KivyCatalog
pip install -U pygments docutils
```

## Start from the Command Line

We ship some examples that are ready-to-run. However, these examples are packaged inside the package. This means you must first know where easy_install has installed your current kivy package, and then go to the examples directory:

```
$ python -c "import pkg_resources; print(pkg_resources.resource_filename('kivy', '../share/kiv
```

And you should have a path similar to:

```
/usr/local/lib/python2.6/dist-packages/Kivy-1.0.4_beta-py2.6-linux-x86_64.egg/share/kivy-examp
```

Then you can go to the example directory, and run it:

```
# launch touchtracer
$ cd <path to kivy-examples>
$ cd demo/touchtracer
$ python main.py

# launch pictures
$ cd <path to kivy-examples>
$ cd demo/pictures
$ python main.py
```

If you are familiar with Unix and symbolic links, you can create a link directly in your home directory for easier access. For example:

1. Get the example path from the command line above
2. Paste into your console:

   ```
   $ ln -s <path to kivy-examples> ~/
   ```

3. Then, you can access to kivy-examples directly in your home directory:

   ```
   $ cd ~/kivy-examples
   ```

If you wish to start your Kivy programs as scripts (by typing *./main.py*) or by double-clicking them, you will want to define the correct version of Python by linking to it. Something like:

```
$ sudo ln -s /usr/bin/python2.7 /usr/bin/kivy
```

Or, if you are running Kivy inside a virtualenv, link to the Python interpreter for it, like:

```
$ sudo ln -s /home/your_username/Envs/kivy/bin/python2.7 /usr/bin/kivy
```

Then, inside each main.py, add a new first line:

```
#!/usr/bin/kivy
```

NOTE: Beware of Python files stored with Windows-style line endings (CR-LF). Linux will not ignore the <CR> and will try to use it as part of the file name. This makes confusing error messages. Convert to Unix line endings.

## Device permissions

When you app starts, Kivy uses Mtdev to scan for available multi-touch devices to use for input. Access to these devices is typically restricted to users or group with the appropriate permissions.

If you do not have access to these devices, Kivy will log an error or warning specifying these devices, normally something like:

```
Permission denied:'/dev/input/eventX'
```

In order to use these devices, you need to grant the user or group permission. This can be done via:

```
$ sudo chmod u+r /dev/input/eventX
```

for the user or:

```
$ sudo chmod g+r /dev/input/eventX
```

for the group. These permissions will only be effective for the duration of your current session. A more permanent solution is to add the user to a group that has these permissions. For example, in Ubuntu, you can add the user to the ‘input’ group:

```
$ sudo adduser $USER input
```

Note that you need to log out then back in again for these permissions to be applied.

« Installation on OS X Installation on Android »