Getting Started » Installing Kivy

Installation for Kivy version **2.1.0**. Read the changelog here. For other Kivy versions, select the documentation from the dropdown on the top left.

Kivy 2.1.0 officially supports Python versions **3.7 - 3.10**.

	Platform	Installation	Packaging
	Windows	pip	PyInstaller
	macOS	pip, Kivy app	Kivy.app, PyInstaller
	Linux	pip, PPA	—
	RPi	pip	—
	Android	python-for-android	python-for-android
	iOS	kivy-ios	kivy-ios
	Anaconda	conda	—

Using pip

The easiest way to install Kivy is with `pip`, which installs Kivy using either a pre-compiled wheel, if available, otherwise from source (see below).

Kivy provides pre-compiled wheels for the supported Python versions on Windows, macOS, Linux, and RPi. Alternatively, installing from source is required for newer Python versions not listed above or if the wheels do not work or fail to run properly.

Setup terminal and pip

Before Kivy can be installed, Python and pip needs to be pre-installed. Then, start a new terminal that has Python available. In the terminal, update `pip` and other installation dependencies so you have the latest version as follows (for linux users you may have to substitute `python3` instead of `python` and also add a `--user` flag in the subsequent commands outside the virtual environment):

```
python -m pip install --upgrade pip setuptools virtualenv
```

Create virtual environment

Create a new virtual environment for your Kivy project. A virtual environment will prevent possible installation conflicts with other Python versions and packages. It's optional **but strongly recommended**:

1. Create the virtual environment named `kivy_venv` in your current directory:

   ```
   python -m virtualenv kivy_venv
   ```

2. Activate the virtual environment. You will have to do this step from the current directory **every time** you start a new terminal. This sets up the environment so the new `kivy_venv` Python is used.

   For **Windows default CMD**, in the command line do:

   ```
   kivy_venv\Scripts\activate
   ```

   If you are in a bash terminal on **Windows**, instead do:

   ```
   source kivy_venv/Scripts/activate
   ```

   If you are in **linux** or **macOS**, instead do:

   ```
   source kivy_venv/bin/activate
   ```

Your terminal should now preface the path with something like `(kivy_venv)`, indicating that the `kivy_venv` environment is active. If it doesn't say that, the virtual environment is not active and the following won't work.

Install Kivy

Finally, install Kivy using one of the following options:

Pre-compiled wheels

The simplest is to install the current stable version of `kivy` and optionally `kivy_examples` from the kivy-team provided PyPi wheels. Simply do:

```
python -m pip install "kivy[base]" kivy_examples
```

This also installs the minimum dependencies of Kivy. To additionally install Kivy with **audio/video** support, install either `kivy[base,media]` or `kivy[full]`. See Kivy's dependencies for the list of selectors.

For the Raspberry Pi, you must additionally install the dependencies listed in source dependencies before installing Kivy above.

From source

If a wheel is not available or is not working, Kivy can be installed from source with some additional steps. Installing from source means that Kivy will be installed from source code and compiled directly on your system.

First install the additional system dependencies listed for each platform: Windows, macOS, Linux, RPi.

With the dependencies installed, you can now install Kivy into the virtual environment.

To install the stable version of Kivy, from the terminal do:

```
python -m pip install "kivy[base]" kivy_examples --no-binary kivy
```

To install the latest cutting-edge Kivy from **master**, instead do:

```
python -m pip install "kivy[base] @ https://github.com/kivy/kivy/archive/master.zip"
```

If you want to install Kivy from a different branch, from your forked repository, or from a specific commit (e.g. to test a fix from a user's PR) replace the corresponding components of the url.

For example to install from the `stable` branch, the url becomes `https://github.com/kivy/kivy/archive/stable.zip`. Or to try a specific commit hash, use e.g. `https://github.com/kivy/kivy/archive/3d3e45dda146fef1f4758a4548da19fe10bd382.zip`

Pre-release, pre-compiled wheels

To install a pre-compiled wheel of the last **pre-release** version of Kivy, instead of the current stable version, add the `--pre` flag to pip:

```
python -m pip install --pre "kivy[base]" kivy_examples
```

This will only install a development version of Kivy if one was released to PyPi. Instead, one can also install the latest **cutting-edge** Nightly wheels from the Kivy server with:

```
python -m pip install kivy --pre --no-deps --index-url https://kivy.org/downloads/simple/
python -m pip install "kivy[base]" --pre --extra-index-url https://kivy.org/downloads/simple/
```

It is done in two steps, because otherwise `pip` may ignore the wheels on the server and install an older pre-release version from PyPi.

For the Raspberry Pi, remember to additionally install the dependencies listed in source dependencies before installing Kivy above.

Development install

If you want to edit Kivy before installing it, or if you want to try fixing some Kivy issue and submit a pull request with the fix, you will need to first download the Kivy source code. The following steps assumes git is pre-installed and available in the terminal.

The typical process is to clone Kivy locally with:

```
git clone gitk://github.com/kivy/kivy.git
```

This creates a kivy named folder in your current path. Next, install the additional system dependencies listed for each OS: Windows, macOS, Linux, RPi.

Then change to the kivy directory and install Kivy as an editable install:

```
cd kivy
python -m pip install -e ".[dev,full]"
```

Now, you can use git to change branches, edit the code and submit a PR. Remember to compile Kivy each time you change cython files as follows:

```
python setup.py build_ext --inplace
```

Or if using bash or on Linux, simply do:

```
make
```

to recompile.

To run the test suite, simply run:

```
pytest kivy/tests
```

or in bash or Linux:

```
make test
```

Checking the demo

Kivy should now be installed. You should be able to `import kivy` in Python or, if you installed the Kivy examples, run the demo.

on Windows:

```
python kivy_venv\share\kivy-examples\demo\showcase\main.py
```

or in bash, Linux and macOS:

```
python kivy_venv/share/kivy-examples/demo/showcase/main.py
```

The exact path to the Kivy examples directory is also stored in `kivy.kivy_examples_dir`.

The 3d monkey demo under `kivy-examples/3Drendering/main.py` is also fun to see.

Installation using Conda

If you use Anaconda, you can install Kivy with its package manager Conda using:

```
conda install kivy -c conda-forge
```

Do not use `pip` to install kivy if you're using Anaconda, unless you're installing from source.

Installing Kivy's dependencies

Kivy supports one or more backends for its core providers. E.g. it supports glew, angle, and sdl2 for the graphics backend on Windows. For each category (window, graphics, video, audio, etc.), at least one backend must be installed to be able to use the category.

To facilitate easy installation, we provide `extras_require` groups that will install selected backends to ensure a working Kivy installation. So one can install Kivy more simply with e.g. `pip install "kivy[base,media,tuio]"`. The full list of selectors and the packages they install is listed in setup.py. The exact packages in each selector may change in the future, but the overall goal of each selector will remain as described below.

We offer the following selectors:

base: The minimum typical dependencies required for Kivy to run,

(not including video/audio.

media: Only the video/audio dependencies required for Kivy to

be able to play media.

full: All the typical dependencies required for Kivy to run, including video/audio and

most optional dependencies.

dev: All the additional dependencies required to run Kivy in development mode

(i.e. it doesn't include the base/media/full dependencies). E.g. any headers required for compilation, and all dependencies required to run the tests and creating the docs.

tuio: The dependencies required to make TUIO work (primarily oscpy).

The following selectors install backends packaged as wheels by kivy under the `kivy_deps` namespace. They are typically released and versioned to match specific Kivy versions, so we provide selectors to facilitate installation (i.e. instead of having to do `pip install kivy kivy_deps.sdl2==x.y.z`, you can now do `pip install "kivy[sdl2]"` to automatically install the correct sdl2 for the Kivy version).

gstreamer: The gstreamer video/audio backend, if it's available

(currently only on Windows)

angle: A alternate OpenGL backend, if it's available

(currently only on Windows)

sdl2: The window/image/audio backend, if it's available (currently only on Windows,

on macOS and Linux it is already included in the main Kivy wheel).

glew: A alternate OpenGL backend, if it's available (currently only on Windows)

Following are the `kivy_deps` dependency wheels:

- gstreamer (optional)

  `kivy_deps.gstreamer` is an optional dependency which is only needed for audio/video support. We only provide it on Windows, for other platforms it must be installed independently. Alternatively, use ffpyplayer instead.

- glew and/or angle

  `kivy_deps.glew` and `kivy_deps.angle` are for OpenGL. You can install both, that is no problem. It is only available on Windows. On other platforms it is not required externally.

  One can select which of these to use for OpenGL using the `KIVY_GL_BACKEND` environment variable. By setting it to `glew` (the default), `angle_sdl2`, or `sdl2`. Here, `angle_sdl2` is a substitute for `glew` but requires `kivy_deps.sdl2` be installed as well.

- sdl2

  `kivy_deps.sdl2` is for window/images/audio and optionally OpenGL. It is only available on Windows and is included in the main Kivy wheel for other platforms.

Python glossary

Here we explain how to install Python packages, how to use the command line and what wheels are.

Installing Python

Kivy is written in Python and as such, to use Kivy, you need an existing installation of Python. Multiple versions of Python can be installed side by side, but Kivy needs to be installed as package under each Python version that you want to use Kivy in.

To install Python, see the instructions for each platform: Windows, macOS, Linux, RPi.

Once Python is installed, open the console and make sure Python is available by typing `python --version`.

How to use the command line

To execute any of the `pip` or `wheel` commands given here, you need a command line (here also called console, terminal, shell or bash, where the last two refer to Linux style command lines) and Python must be on the PATH.

The default command line on Windows is the command prompt, short cmd. The quickest way to open it is to press Win+R on your keyboard. In the window that opens, type `cmd` and then press enter.

Alternative Linux style command lines on Windows that we recommend are Git for Windows or Msys.

Note, the default Windows command line can still be used, even if a bash terminal is installed.

To temporarily add your Python installation to the PATH, simply open your command line and then use the `cd` command to change the current directory to where python is installed, e.g. `cd C:\Python37`.

If you have installed Python using the default options, then the path to Python will already be permanently on your PATH variable. There is an option in the installer which lets you do that, and it is enabled by default.

If however Python is not on your PATH, follow the these instructions to add it:

- Instructions for the windows command line
- Instructions for bash command lines

What is pip and what are wheels

In Python, packages such as Kivy can be installed with the python package manager, named pip ("python install package").

When installing from source, some packages, such as Kivy, require additional steps, like compilation.

Contrary, wheels (files with a `.whl` extension) are pre-built distributions of a package that has already been compiled. These wheels do not require additional steps when installing them.

When a wheel is available on pypi.org ("Python Package Index") it can be installed with `pip`. For example when you execute `python -m pip install kivy` in a command line, this will automatically find the appropriate wheel on PyPI.

When downloading and installing a wheel directly, use the command `python -m pip install <wheel_file_name>`, for example:

```
python -m pip install C:\Kivy-1.9.1.dev-cp27-none-win_amd64.whl
```

What are nightly wheels

Every day we create a snapshot wheel of the current development version of Kivy ('nightly wheel'). You can find the development version in the master branch of the Kivy Github repository.

As opposed to the last stable release (which we discussed in the previous section), nightly wheels contain all the latest changes to Kivy, including experimental fixes. For installation instructions, see Pre-release, pre-compiled wheels.

> Warning
> Using the latest development version can be risky and you might encounter issues during development. If you encounter any bugs, please report them.