

# Python

Pieter P

This page explains how to build Python 3 from source on Ubuntu.

## Install dependencies and tools

---

First, install GCC, GNU Make and GNU Wget if you haven't already.

```
$ sudo apt update
$ sudo apt install gcc g++ make wget
```

Also install the dependencies to build Python and its modules.

```
$ sudo apt install zlib1g-dev libbz2-dev libssl-dev uuid-dev libffi-dev libreadline-dev libsqlite3-dev tk-dev libbz2-dev
libncurses5-dev libreadline6-dev libgdbm-dev liblzma-dev
```

On Ubuntu 18.04 and later, you'll need compatibility development files for GNU dbm.

```
$ sudo apt install libgdbm-compat-dev
```

You can try to build Python without these dependencies, but then some of the optional modules will not be built.

## Download and extract the source code

---

Next, download and extract the Python source code.

```
$ version="3.8.2"
$ python="Python-$version"
$ cd /tmp
$ wget "https://www.python.org/ftp/python/$version/$python.tgz" # Download
$ tar xf $python.tgz # Extract
```

## Configure the build settings

---

You can now tune the settings for your build now. I'll use the standard version with optimizations, link-time optimizations, and IPv6 enabled. **--enable-shared** builds the shared libraries for Python. This allows other programs to use and embed Python.

The installation location is **~/local**. This is a user-level installation, it's just for the current user, doesn't require sudo, and won't overwrite the Python version that comes with your Linux distribution.

On most distributions, **~/local/lib** is not in the runtime linker's search path. Therefore, we need to specify the **rpath** during the linking stage.

```
$ cd "$python"
$ ./configure --prefix="$HOME/local" \
  --enable-ipv6 \
  --enable-shared \
  --with-lto --enable-optimizations \
  'LDFLAGS=-Wl,-rpath,\${$ORIGIN}/../lib'
```

To see all options, run the following command.

```
$ ./configure --help
```

## Build Python

---

Actually build Python. This can easily take up to an hour, especially if you have optimizations enabled, because then it will run all tests.

The **-j** option tells make to compile multiple files in parallel, **nproc** gives the number of CPU cores of the system.

```
$ make -j$(nproc)
```

## Install Python

---

Finally, install Python to the location specified as **prefix** in the configure step.

There are two possible install options: Either you install Python as the main/default version: this means that it will be installed as **python3**, and it will replace the previous default Python 3 version at the install location. The version you're installing will become the new default.

The second option is to install Python as an "alternative" version. The default Python 3 version will be preserved, and the new version will be installed as **python3.8**.

```
$ make install # Replace default version
```

```
$ make altinstall # Install alongside existing version, preserve default
```

## Adding Python to the PATH

If the installation location `~/.local/bin` is not in your **PATH**, you'll have to add it yourself.

```
$ export PATH="$HOME/.local/bin:$PATH"
```

To make it permanent, add it to your `~/.profile` file, so it is added to your **PATH** every time you log in.

```
$ echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.profile
```

## Finding the shared libraries

Python itself will find its shared libraries without problems, because of the **rpath** linker option we added previously. However, if you are using other programs that require these libraries, you'll have to add `~/.local/lib` to your **LD\_LIBRARY\_PATH** environment variable.

```
$ export LD_LIBRARY_PATH="$HOME/.local/lib"
```

Setting **LD\_LIBRARY\_PATH** is not the most elegant solution, so if you have root privileges, you can add the `~/.local/lib` folder to the **ld** configuration folder:

```
$ echo "$HOME/.local/lib" | sudo tee -a /etc/ld.so.conf.d/home.local.conf  
$ sudo ldconfig
```

## Shell Script

---

Here's a shell script that executes the previous steps for you.

## python.sh

```
7 version="3.8.2"
8 builddir="/tmp"
9 python="Python-$version"
10 prefix="$HOME/.local"
11
12 # Install dependencies and build tools
13 sudo apt-get update
14 sudo apt-get install -y \
15     zlib1g-dev libbz2-dev libssl-dev uuid-dev libffi-dev libreadline-dev \
16     libsqlite3-dev tk-dev libbz2-dev libncurses5-dev libreadline6-dev \
17     libgdbm-dev liblzma-dev \
18     gcc g++ make wget
19
20 # For Ubuntu 18.04 and later, another dependency is required for GNU dbm
21 source /etc/os-release
22 if (( $(echo "$VERSION_ID >= 18.04" | bc -l) ));
23 then
24     sudo apt-get install libgdbm-compat-dev
25 fi
26
27 # Download and extract the Python source code
28 mkdir -p "$builddir"
29 cd $builddir
30 if [ ! -d "$python" ]
31 then
32     wget "https://www.python.org/ftp/python/$version/$python.tgz"
33     tar xf $python.tgz
34 fi
35
36 cd "$python"
37 ./configure --prefix="$prefix" \
38     --enable-ipv6 \
39     --enable-shared \
40     --with-lto --enable-optimizations \
41     'LDFLAGS=-Wl,-rpath,${$ORIGIN}/../lib'
42
43 make -j$(($(nproc) * 2))
44 make altinstall
```

You can download it [here](#). Then allow execution and run it:

```
$ chmod +x python.sh
$ ./python.sh
```

## Tested on

---

- Ubuntu 16.04 - Python 3.7.3
- Ubuntu 16.04 - Python 3.8.0
- Ubuntu 18.04 - Python 3.7.4
- Ubuntu 19.10 - Python 3.8.1
- Ubuntu 19.10 - Python 3.8.2