

VITUX

Linux Compendium

[Home](#) [Linux](#) [CentOS](#) [Debian](#) [Ubuntu](#) [Shell](#)

Install Python3 on Ubuntu 18.04 and Set Up a Virtual Programming Environment



Adobe Creative Cloud for

Teams starting at \$29.99 per month. ads via Carbon

Search



Python is an object-oriented, interpreted, high-level programming language created by Guido van Rossum and was first released in 1991. It reduces the cost of program maintenance with its easy to learn syntax and high user readability. It encourages program modularity and thus code reuse by supporting modules and packages based programming concept. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Programmers often prefer Python over other languages as in Python there is no separate compilation step. This increases the productivity for programmers as the edit-test-debug cycle becomes pretty fast. Python just seems to be getting more and more popular with Linux developers and is arguably the best general-purpose language currently available. So as Linux users, you need to get a hold of how to install it and start writing your Python applications.

In this article, we will install the latest version of Python3 on our Ubuntu system and then set up a virtual programming environment where you can write and execute your Python application programs. The article will also help you in writing and running your first Python program, that will get you started with developing your own complex Python applications.

We have run the commands and procedures mentioned in this article on an Ubuntu 18.04 LTS system.

We are using the Ubuntu command line, the Terminal, for installation and set up a virtual programming environment. You can open the Terminal either through the system Dash or the Ctrl+Alt+T shortcut.

About This Site

Vitux.com aims to become a Linux compendium with lots of unique and up to date tutorials.

Most Popular

[How to Uninstall Programs from your Ubuntu System](#)

posted on November 6, 2018 | under [Linux](#), [Ubuntu](#)

[Get Linux System and Hardware Details on the Command Line](#)

posted on August 7, 2018 | under [CentOS](#), [Debian](#), [Linux](#), [Shell](#), [Ubuntu](#)

[How to Change sudo Password in Ubuntu](#)

posted on November 29, 2018 | under [Linux](#), [Ubuntu](#)

[How to Install Ubuntu 18.04 along with Windows 10](#)

posted on June 29, 2018 | under [Ubuntu](#)

[How to Install XAMPP on your Ubuntu 18.04 LTS System](#)

posted on October 2, 2018 | under [Linux](#), [Ubuntu](#)

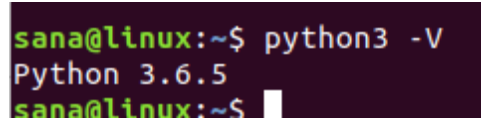
Check Current Python Version

Checking the current version of a software not only helps you get the version number of that software installed on your system but also verifies if the software is indeed installed on your system. We will do the same for Python by running the following command in our Terminal:

```
$ python3 -V
```

or

```
$ python3 --version
```



```
sana@linux:~$ python3 -V
Python 3.6.5
sana@linux:~$
```

The version number will appear as shown in the above output, depending on when you updated your system.

You might also have several versions of Python installed on your system. The following command will help you get a list of all Python versions you have on your system:

[How to Install Wine on Ubuntu 18.04 LTS](#)

posted on September 20, 2018 | under [Linux](#), [Ubuntu](#)

[Install NFS Server and Client on Ubuntu 18.04 LTS](#)

posted on November 13, 2018 | under [Linux](#), [Ubuntu](#)

[6 Ways to Open Folders in Ubuntu 18.04 LTS](#)

posted on February 14, 2019 | under [Linux](#), [Ubuntu](#)

[5 Ways to Check Available Memory in Ubuntu](#)

posted on December 6, 2018 | under [Linux](#), [Ubuntu](#)

[How to Install VirtualBox on Ubuntu 18.04 LTS](#)

posted on November 5, 2018 | under [Desktop](#), [Linux](#), [Shell](#), [Ubuntu](#)

```
$ apt list --installed | grep python
```

```
sana@linux:~$ apt list --installed | grep python
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

libpython2.7/bionic,now 2.7.15~rc1-1 amd64 [installed,upgradable to: 2.7.15~rc1-1ubuntu0.1]
libpython2.7-minimal/bionic,now 2.7.15~rc1-1 amd64 [installed,upgradable to: 2.7.15~rc1-1ubuntu0.1]
libpython2.7-stdlib/bionic,now 2.7.15~rc1-1 amd64 [installed,upgradable to: 2.7.15~rc1-1ubuntu0.1]
libpython3-stdlib/bionic,now 3.6.5-3 amd64 [installed,upgradable to: 3.6.7-1~18.04]
libpython3.6/bionic,now 3.6.5-3 amd64 [installed,upgradable to: 3.6.7-1~18.04]
libpython3.6-minimal/bionic,now 3.6.5-3 amd64 [installed,upgradable to: 3.6.7-1~18.04]
libpython3.6-stdlib/bionic,now 3.6.5-3 amd64 [installed,upgradable to: 3.6.7-1~18.04]
python-apt-common/bionic,bionic,now 1.6.0 all [installed,upgradable to: 1.6.3]
python-talloc/bionic,now 2.1.10-2ubuntu1 amd64 [installed]
python3/bionic,now 3.6.5-3 amd64 [installed,upgradable to: 3.6.7-1~18.04]
python3-apport/bionic,bionic,now 2.20.9-0ubuntu7 all [installed,upgradable to: 2.20.9-0ubuntu7.5]
python3-apt/bionic,now 1.6.0 amd64 [installed,upgradable to: 1.6.3]
```

Install Python through apt-get

Installing Python through the apt-get command is pretty simple. First, you need to update your system repository index with that of the Internet so that the latest available version can be installed. Run the following command as sudo in order to do so:

```
$ sudo apt-get update
```

```
File Edit View Search Terminal Help
sana@linux:~$ sudo apt-get update
[sudo] password for sana:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
```

```
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates I
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports
Fetched 247 kB in 3s (86.9 kB/s)
Reading package lists... Done
```

Since we already have Python installed on our system, as verified in the previous section, we only need to upgrade it to the latest version as follows:

```
$ sudo apt-get upgrade python3
```

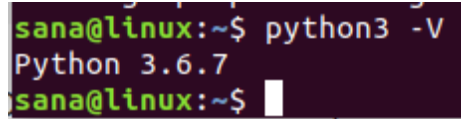
```
sana@linux:~$ sudo apt-get upgrade python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.6.7-1~18.04).
Calculating upgrade... Done
The following packages will be upgraded:
  desktop-file-utils python3-distupgrade python3-update-manager
  ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk unattended-
  update-manager update-manager-core
8 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 772 kB/823 kB of archives.
After this operation, 21.5 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

The system might ask you the password for sudo as only an authorized user can add/remove and upgrade software on Ubuntu.

The system will also prompt you with a y/n option in order to confirm the upgrade; please enter Y and then hit Enter to continue.

The latest available version of Python will now be installed on your system.

Now when you check the version number of Python, you will see an updated installation:

A terminal window with a dark background. The prompt is 'sana@linux:~\$'. The user enters 'python3 -V'. The output is 'Python 3.6.7'. The prompt returns to 'sana@linux:~\$' with a cursor.

```
sana@linux:~$ python3 -V
Python 3.6.7
sana@linux:~$
```

In case you did not have Python installed in the first place, you can install it as sudo through the following command after running apt-get update:

```
$ sudo apt-get install python3
```

Manually Install Python from Source

Python's website maintains a list of all Python releases on this link:

<https://www.python.org/downloads/source/>

So if you choose to install Python manually through the source, you have the freedom to install whichever build you want to choose. The website also contains the latest versions that you can not even get through the apt-get command.

We visited the website to see that Python-3.7.1 was the latest available version, so we will download its .tgz file through the following command:

```
$ wget https://www.python.org/ftp/python/3.7.1/Python-3.7.1.tgz
```

```
sana@linux:~$ wget https://www.python.org/ftp/python/3.7.1/Python-3.7.1.tgz
--2018-11-27 06:01:10-- https://www.python.org/ftp/python/3.7.1/Python-3.7.1.tgz
Resolving www.python.org (www.python.org)... 151.101.140.223, 2a04:4e42:21::223
Connecting to www.python.org (www.python.org)|151.101.140.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22802018 (22M) [application/octet-stream]
Saving to: 'Python-3.7.1.tgz'

Python-3.7.1.tgz  100%[=====] 21.75M  446KB/s  in 28s

2018-11-27 06:01:38 (800 KB/s) - 'Python-3.7.1.tgz' saved [22802018/22802018]
```

When the file download is complete, please run the following command in order to extract the resources:

```
$ tar -xvf Python-3.7.1.tgz
```

```
sana@linux:~$ tar -xvf Python-3.7.1.tgz
Python-3.7.1/
Python-3.7.1/Doc/
Python-3.7.1/Doc/c-api/
Python-3.7.1/Doc/c-api/sys.rst
Python-3.7.1/Doc/c-api/conversion.rst
Python-3.7.1/Doc/c-api/marshal.rst
Python-3.7.1/Doc/c-api/coro.rst
Python-3.7.1/Doc/c-api/method.rst
Python-3.7.1/Doc/c-api/index.rst
Python-3.7.1/Doc/c-api/bytearray.rst
```

```
Python-3.7.1/Doc/c-api/bytes.rst  
Python-3.7.1/Doc/c-api/none.rst  
Python-3.7.1/Doc/c-api/long.rst  
Python-3.7.1/Doc/c-api/number.rst  
Python-3.7.1/Doc/c-api/code.rst  
Python-3.7.1/Doc/c-api/allocation.rst  
Python-3.7.1/Doc/c-api/list.rst  
Python-3.7.1/Doc/c-api/datetime.rst  
Python-3.7.1/Doc/c-api/set.rst
```

Once the resources are extracted, you need to run the c program “configure” to check the built. For this, you need to have the C compiler gcc installed on your system. If you do not have it available, please install it through the following command:

```
$ sudo apt-get install gcc
```

Change the directory to Python-3.7.1, or to whatever download version you have extracted:

```
$ cd Python-3.7.1
```

Now run the following command to run the configuration script:

```
$ ./configure
```

```
sana@linux:~/Python-3.7.1$ ./configure  
checking build system type... x86_64-pc-linux-gnu  
checking host system type... x86_64-pc-linux-gnu
```



```
checking host system type... x86_64-pc-linux-gnu
checking for python3.7... no
checking for python3... python3
checking for --enable-universalsdk... no
checking for --with-universal-archs... no
checking MACHDEP... checking for --without-gcc... no
checking for --with-icc... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.o
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
```

Now is the time to install Python.

```
$ make
```

If you can not run the make command, you might need to install make through the following command:

```
$ sudo apt-get make
```

```
sana@linux:~/Python-3.7.1$ make
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -W
    -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-fi
```

```
initializers -Werror=implicit-function-declaration -I. -I./Include -DPy
LD_CORE -o Programs/python.o ./Programs/python.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -W
    -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-fi
initializers -Werror=implicit-function-declaration -I. -I./Include -DPy
LD_CORE -o Parser/acceler.o Parser/acceler.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -W
    -std=c99 -Wextra -Wno-unused-result -Wno-unused-parameter -Wno-missing-fi
initializers -Werror=implicit-function-declaration -I. -I./Include -DPy
LD_CORE -o Parser/grammar1.o Parser/grammar1.c
gcc -pthread -c -Wno-unused-result -Wsign-compare -DNDEBUG -g -fwrapv -O3 -W
```

Also, run the following command for Python installation:

```
$ sudo make install
```

```
Looking in links: /tmp/tmpuhd_neod
Collecting setuptools
Collecting pip
Installing collected packages: setuptools, pip
Successfully installed pip-10.0.1 setuptools-39.0.1
sana@linux:~/Python-3.7.1$
```

The downloaded version of Python from the website will be installed on your system.

Errors that might be encountered during installation

Error 1

When you run the “*sudo make install*” command, you might encounter the following error:

```
return _run_pip(args + [p[0] for p in _PROJECTS], additional_paths)
File "/home/sana/Python-3.7.1/Lib/ensurepip/__init__.py", line 27, in _run_pip
import pip. internal
zipimport.ZipImportError: can't decompress data; zlib not available
Makefile:1122: recipe for target 'install' failed
make: *** [install] Error 1
sana@linux:~/Python-3.7.1$
```

This would mean that a package named `zlib1g-dev` is missing from your system as you might have never needed it before.

Solution:

Run the following command as `sudo` in order to install the missing `zlib1g-dev` package:

```
$ sudo apt install zlib1g-dev
```

Then run the following command in order to complete the Python installation:

```
$ sudo make install
```

Error 2

When might also get the following error when you run the “`sudo make install`” command:

```
from ctypes import Union, Structure, Array
ModuleNotFoundError: No module named '_ctypes'
Makefile:1122: recipe for target 'install' failed
make: *** [install] Error 1
```

This would mean that a package named `libffi-dev` is missing from your system as you might have never needed it before.

Solution:

Run the following command as `sudo` in order to install the missing `libffi-dev` package:

```
$ sudo apt-get install libffi-dev
```

Then run the following command in order to complete the Python installation:

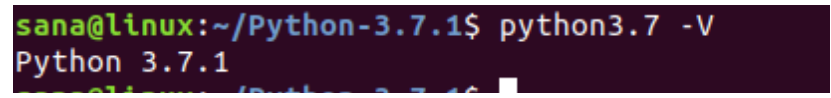
```
$ sudo make install
```

Upgrade Python to The Latest Version

Before manually installing Python from the source, the version number of our Python installation was 3.6.7

When I checked the version number of Python3.7, it gives the following output:

```
$ python3.7 -V
```



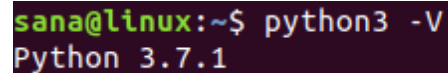
```
sana@linux:~/Python-3.7.1$ python3.7 -V  
Python 3.7.1
```



Since I want to upgrade the version of Python3 to this installed version, I will run the following command:

```
$ sudo apt-get upgrade python3
```

Now you can see that the updated Python version on my system is 3.7.1; the one that I installed manually from the source.



Setup Virtual Programming Environment for Python3

First, let us get familiar with what is a Virtual Programming Environment for Python projects. You can assume it as an isolated space on your system where you can create Python projects having their own set of dependencies that do not affect anything outside of the project. When you are inside this environment, you can make use of Python and pip commands directly instead of using pip3 and Python3 commands. However, outside of this environment, you will have to use the pip3 and Python3 commands to develop and run your applications.

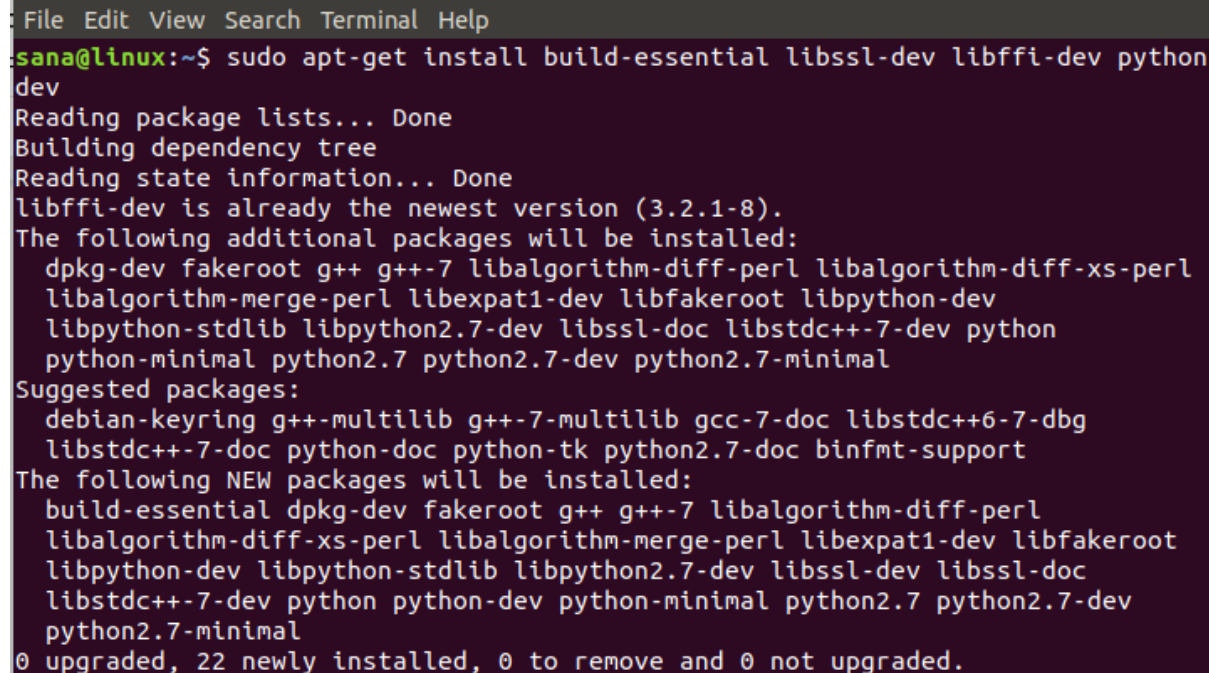
here the step by step procedure for you to create and activate a new virtual

programming environment for Python:

Step 1: Install the Prerequisites

Before installing pip, you will need to add a few prerequisites that will help you in setting up your virtual space. Run the following command as sudo in order to install the build-essential, libssl-dev, libffi-dev and python-dev packages to your system:

```
$ sudo apt-get install build-essential libssl-dev libffi-dev  
python-dev
```



```
File Edit View Search Terminal Help  
sana@linux:~$ sudo apt-get install build-essential libssl-dev libffi-dev python-  
dev  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
libffi-dev is already the newest version (3.2.1-8).  
The following additional packages will be installed:  
  dpkg-dev fakeroot g++ g++-7 libalgorithm-diff-perl libalgorithm-diff-xs-perl  
  libalgorithm-merge-perl libexpat1-dev libfakeroot libpython-dev  
  libpython-stdlib libpython2.7-dev libssl-doc libstdc++-7-dev python  
  python-minimal python2.7 python2.7-dev python2.7-minimal  
Suggested packages:  
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg  
  libstdc++-7-doc python-doc python-tk python2.7-doc binfmt-support  
The following NEW packages will be installed:  
  build-essential dpkg-dev fakeroot g++ g++-7 libalgorithm-diff-perl  
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libexpat1-dev libfakeroot  
  libpython-dev libpython-stdlib libpython2.7-dev libssl-dev libssl-doc  
  libstdc++-7-dev python python-dev python-minimal python2.7 python2.7-dev  
  python2.7-minimal  
0 upgraded, 22 newly installed, 0 to remove and 0 not upgraded.
```

```
Need to get 41.3 MB/42.8 MB of archives.  
After this operation, 103 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

Please click Y and then hit Enter when the system prompts you with a y/n option to continue installation.

All these packages will then be installed to your system.

Step 2: Install pip3 if it is already not installed on your system

You can verify if pip3 is installed on your system or not simply by checking its version number. Please run the following command to check the version:

```
$ pip3 -V
```

```
sana@linux:~$ pip3 -V  
pip 10.0.1 from /usr/local/lib/python3.7/site-packages/pip (python 3.7)  
sana@linux:~$
```

The above output shows that pip 10.0.1 is already installed on my system.

If your output suggests that pip is not installed on your system, please run the following commands as sudo to install the latest pip3 package:

```
$ sudo apt-get update
```

And then,

```
$ sudo apt install python3-pip
```

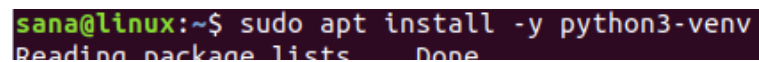
Now that pip3 is installed on your system, you can install any pip package by using the following command syntax:

```
$ pip3 install [package-name]
```

Step 3: Create a virtual environment through Python3-venv

In order to create the virtual environment, you need the Python3-venv package installed on your system. Please run the following command as sudo in order to install it:

```
$ sudo apt install -y python3-venv
```

A terminal window with a dark background. The prompt is 'sana@linux:~\$'. The command entered is 'sudo apt install -y python3-venv'. The output shows 'Reading package lists...' followed by 'Done' on the next line.

```
sana@linux:~$ sudo apt install -y python3-venv
Reading package lists... Done
```



```
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  python-pip-whl python3-distutils python3-lib2to3 python3.6-venv  
The following NEW packages will be installed:  
  python-pip-whl python3-distutils python3-lib2to3 python3-venv  
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.  
Need to get 1,877 kB of archives.  
After this operation, 4,014 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python-pip-whl all 9.0.1-2.3~ubuntu1 [1,652 kB]
```

Now we will create a folder for your Python virtual environments where you can create your stand-alone virtual environments. You can use the following syntax to create your own working directory:

```
$ mkdir [environment_dir_name]
```

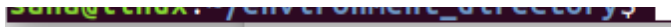
Example:

```
$ mkdir environment_directory
```

Now change the working directory to the environments directory that you just created:

```
$ cd environment_directory
```

```
sana@linux:~$ mkdir environment_directory  
sana@linux:~$ cd environment_directory  
sana@linux:~/environment_directory$
```



In the environments directory, we will be creating a new virtual environment where you can write your Python programs and create projects.

Syntax:

\$ python3 -m venv environment_name

Example:

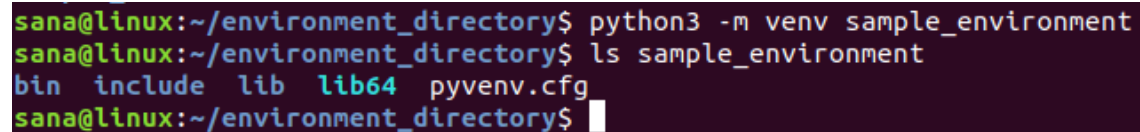
```
$ python3 -m venv sample_environment
```

When you list the contents of your Python environment through the ls command, you will be able to see the following basic contents:

bin include lib lib64 pyvenv.cfg

Example:

```
$ ls sample_environment
```



```
sana@linux:~/environment_directory$ python3 -m venv sample_environment
sana@linux:~/environment_directory$ ls sample_environment
bin  include  lib  lib64  pyvenv.cfg
sana@linux:~/environment_directory$
```

This means that your environment is successfully set up.

Step 4: Activate the Python Virtual Environment

When you want to use the newly created virtual environment, you first need to activate it. Use the following command to syntax to do so:

Syntax:

\$ source environment_name/bin/activate

Example:

```
$ source sample_environment/bin/activate
```

```
sana@linux:~/environment_directory$ source sample_environment/bin/activate  
(sample_environment) sana@linux:~/environment_directory$
```

When you activate the environment, you will see how your environment name appears inside brackets, suggesting that you are now inside the environment.

Whenever you want to deactivate the environment, you can use the following command:

```
$ deactivate
```

```
(sample_environment) sana@linux:~/environment_directory$ deactivate  
sana@linux:~/environment_directory$
```

This will deactivate the virtual environment and you can work outside of it.

Your First Python Program

You can create and run your first Python program both inside and outside of the Virtual Working environment. In this example, we will tell you how to write a sample Python program inside the virtual environment you just created.

In order to get inside the environment, first change the directory to your environments folder and then activate whichever virtual environment you want to activate.

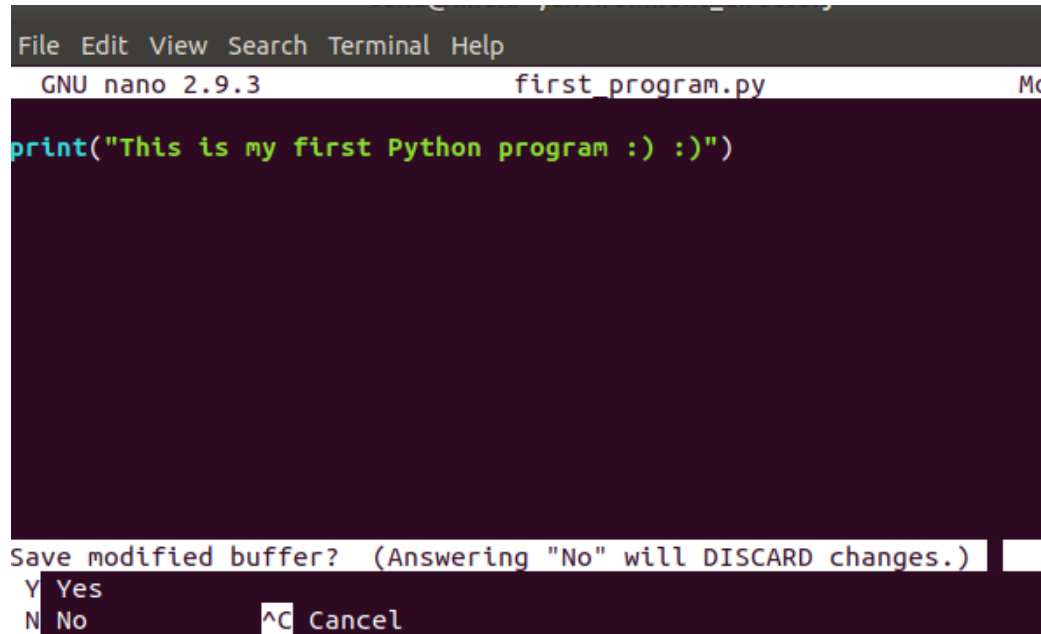
Once you are inside the virtual environment, you can use your favorite text editor to create your first Python program. In this example, we are using the Nano editor to create a program.

```
$ nano first_program.py
```

This command will open a blank text file by the name of first_program.py

Write or paste the following line in your first Python program:

```
print("This is my first Python program :) :)")
```

A screenshot of the nano text editor interface. The top menu bar shows 'File Edit View Search Terminal Help'. Below it, the status bar indicates 'GNU nano 2.9.3' and the filename 'first_program.py'. The main editing area has a dark background with the code 'print("This is my first Python program :) :)")' written in a light green font. At the bottom, a prompt 'Save modified buffer? (Answering "No" will DISCARD changes.)' is displayed, with 'Y Yes' and 'N No' as options, and '^C Cancel' as a keyboard shortcut.

Save the file by hitting Ctrl+X, then entering Y and hitting Enter. Your program is now saved in your virtual environment.

Run the following command in order to execute the Python program:

```
$ python [program_name.py]
```

Example:

```
$ python [first_program.py]
```

```
(sample_environment) sana@linux:~/environment_directory$ python first_prog  
ram.py  
This is my first Python program :) :)
```

You can then deactivate the environment. Please remember that when you want to execute this program outside the virtual environment, you might have to use the Python3 commands instead of Python commands.

Conclusion

Most versions of Ubuntu already have Python and Pip3 installed in them but after reading this article you will know how to download and upgrade to the latest versions of each. You have also learned how to create your own Python virtual environment where you can write your independent Python programs and projects. Hope your first program will serve as a basis for you to move to more useful and complex Python applications. Happy programming!

📅 November 28, 2018 📁 Linux, Shell, Ubuntu

← [How to Install Joomla! CMS on Debian 9](#)

[How to Change sudo Password in Ubuntu](#) →

