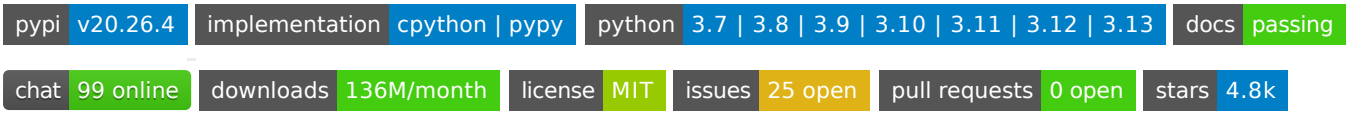
> **Note**
>
> You are not reading the most recent version of this documentation. 20.26.4 is the latest version available.

virtualenv



`virtualenv` is a tool to create isolated Python environments. Since Python `3.3`, a subset of it has been integrated into the standard library under the venv module. The `venv` module does not offer all features of this library, to name just a few more prominent:

- is slower (by not having the `app-data` seed method),
- is not as extendable,
- cannot create virtual environments for arbitrarily installed python versions (and automatically discover these),
- is not upgrade-able via pip,
- does not have as rich programmatic API (describe virtual environments without creating them).

The basic problem being addressed is one of dependencies and versions, and indirectly permissions. Imagine you have an application that needs version `1` of `LibFoo`, but another application requires version `2`. How can you use both these libraries? If you install everything into your host python (e.g. `python3.8`) it's easy to end up in a situation where two packages have conflicting requirements.

Or more generally, what if you want to install an application *and leave it be*? If an application works, any change in its libraries or the versions of those libraries can break the application. Also, what if you can't install packages into the global `site-packages` directory, due to not having permissions to change the host python environment?

In all these cases, `virtualenv` can help you. It creates an environment that has its own installation directories, that doesn't share libraries with other virtualenv environments (and optionally doesn't access the globally installed libraries either).

Useful links

Related projects, that build abstractions on top of virtualenv

- virtualenvwrapper - a useful set of scripts for creating and deleting virtual environments
- pew - provides a set of commands to manage multiple virtual environments
- tox - a generic virtualenv management and test automation command line tool, driven by a configuration file

Skip to content

- l that automates testing in multiple Python environments, similar to tox, driven by a `noxfile.py` configuration file

Tutorials

- Corey Schafer tutorial on how to use it
- Using virtualenv with mod_wsgi

Presenting how the package works from within

- Bernat Gabor: status quo of virtual environments
- Carl Meyer: Reverse-engineering Ian Bicking's brain: inside pip and virtualenv

---

Made with Sphinx and @pradyunsg's Furo
Last updated on 2024-06-22T01:57:47.408024+00:00

Installation

via pipx

virtualenv is a CLI tool that needs a Python interpreter to run. If you already have a `Python 3.7+` interpreter the best is to use pipx to install virtualenv into an isolated environment. This has the added benefit that later you'll be able to upgrade virtualenv without affecting other parts of the system.

```
pipx install virtualenv
virtualenv --help
```

via pip

Alternatively you can install it within the global Python interpreter itself (perhaps as a user package via the `--user` flag). Be cautious if you are using a python install that is managed by your operating system or another package manager. `pip` might not coordinate with those tools, and may leave your system in an inconsistent state. Note, if you go down this path you need to ensure pip is new enough per the subsections below:

```
python -m pip install --user virtualenv
python -m virtualenv --help
```

wheel

Installing virtualenv via a wheel (default with pip) requires an installer that can understand the `python-requires` tag (see PEP-503), with pip this is version `9.0.0` (released 2016 November). Furthermore, in case you're not installing it via the PyPi you need to be using a mirror that correctly forwards the `python-requires` tag (notably the OpenStack mirrors don't do this, or older devpi versions - added with version `4.7.0`).

sdist

When installing via a source distribution you need an installer that handles the PEP-517 specification. In case of `pip` this is version `18.0.0` or later (released on 2018 July). If you cannot upgrade your pip to support this you need to ensure that the build requirements from pyproject.toml are satisfied before triggering the install.

Skip to content

zipapp

You can use virtualenv without installing it too. We publish a Python zipapp, you can just download this from https://bootstrap.pypa.io/virtualenv.pyz and invoke this package with a python interpreter:

```
python virtualenv.pyz --help
```

The root level zipapp is always the current latest release. To get the last supported zipapp against a given python minor release use the link `https://bootstrap.pypa.io/virtualenv/x.y/virtualenv.pyz`, e.g. for the last virtualenv supporting Python 3.11 use https://bootstrap.pypa.io/virtualenv/3.11/virtualenv.pyz.

If you are looking for past version of virtualenv.pyz they are available here:

```
https://github.com/pypa/get-virtualenv/blob/<virtualenv version>/public/<python version>
```

latest unreleased

Installing an unreleased version is discouraged and should be only done for testing purposes. If you do so you'll need a pip version of at least `18.0.0` and use the following command:

```
pip install git+https://github.com/pypa/virtualenv.git@main
```

Python and OS Compatibility

virtualenv works with the following Python interpreter implementations:

- CPython: `3.12 >= python_version >= 3.7`
- PyPy: `3.10 >= python_version >= 3.7`

This means virtualenv works on the latest patch version of each of these minor versions. Previous patch versions are supported on a best effort approach.

CPython is shipped in multiple forms, and each OS repackages it, often applying some customization along the way. Therefore we cannot say universally that we support all platforms, but rather specify some we test against. In case of ones not specified here the support is unknown, though likely will work. If you find some cases please open a feature request on our issue tracker.

Note:

- as of `20.18.0` – `2023-02-06` – we no longer support running under Python `<=3.6`,
- as of `20.22.0` – `2023-04-19` – we no longer support creating environments for Python `<=3.6`.

Linux

- installations from python.org
- Ubuntu 16.04+ (both upstream and deadsnakes builds)
- Fedora

Skip to content

- CentOS
- OpenSuse
- Arch Linux

macOS

In case of macOS we support:

- installations from python.org,
- python versions installed via brew,
- Python 3 part of XCode (Python framework - `/Library/Frameworks/Python3.framework/`).

Windows

- Installations from python.org
- Windows Store Python - note only version 3.7+

User Guide

Quick start

Create the environment (creates a folder in your current directory)

```
virtualenv env_name
```

In Linux or Mac, activate the new python environment

```
source env_name/bin/activate
```

Or in Windows

```
.\env_name\Scripts\activate
```

Confirm that the env is successfully selected

```
which python3
```

Introduction

Virtualenv has one basic command:

```
virtualenv venv
```

This will create a python virtual environment of the same version as virtualenv, installed into the subdirectory `venv`. The command line tool has quite a few of flags that modify the tool's behavior, for a full list make sure to check out CLI flags.

The tool works in two phases:

- **Phase 1** discovers a python interpreter to create a virtual environment from (by default this is the same python as the one `virtualenv` is running from, however we can change this via the `p` option).
- **Phase 2** creates a virtual environment at the specified destination (`dest`), this can be broken down into four further sub-steps:
  - create a python that matches the target python interpreter from phase 1,
  - install (bootstrap) seed packages (one or more of pip, setuptools, wheel) in the created virtual environment,
  - install activation scripts into the binary directory of the virtual environment (these will allow end users to *activate* the virtual environment from various shells).
  - create files that mark the virtual environment as to be ignored by version control systems (currently we support Git only, as Mercurial, Bazaar or SVN do not support ignore files in subdirectories). This step can be skipped with the `no-vcs-ignore` option.

The python in your new virtualenv is effectively isolated from the python that was used to create it.

Python discovery

The first thing we need to be able to create a virtual environment is a python interpreter. This will describe to the tool what type of virtual environment you would like to create, think of it as: version, architecture, implementation.

`virtualenv` being a python application has always at least one such available, the one `virtualenv` itself is using, and as such this is the default discovered element. This means that if you install `virtualenv` under python `3.8`, virtualenv will by default create virtual environments that are also of version `3.8`.

Created python virtual environments are usually not self-contained. A complete python packaging is usually made up of thousands of files, so it's not efficient to install the entire python again into a new folder. Instead virtual environments are mere shells, that contain little within themselves, and borrow most from the system python (this is what you installed, when you installed python itself). This does mean that if you upgrade your system python your virtual environments *might* break, so watch out. The upside of this, referring to the system python, is that creating virtual environments can be fast.

Here we'll describe the built-in mechanism (note this can be extended though by plugins). The CLI flag `p` or `python` allows you to specify a python specifier for what type of virtual environment you would like, the format is either:

- a relative/absolute path to a Python interpreter,
- a specifier identifying the Python implementation, version, architecture in the following format:

```
{python implementation name}{version}{architecture}
```

We have the following restrictions:

- the python implementation is all alphabetic characters (`python` means any implementation, and if is missing it defaults to `python`),
- the version is a dot separated version number,
- the architecture is either `-64` or `-32` (missing means `any`).

For example:

- `python3.8.1` means any python implementation having the version `3.8.1`,
- `3` means any python implementation having the major version `3`,
- `cpython3` means a `CPython` implementation having the version `3`,
- `pypy2` means a python interpreter with the `PyPy` implementation and major version `2`.

Given the specifier `virtualenv` will apply the following strategy to discover/find the system executable:

- If we're on Windows look into the Windows registry, and check if we see any registered Python implementations that match the specification. This is in line with expectation laid out inside PEP-514
- Try to discover a matching python executable within the folders enumerated on the `PATH` environment variable. In this case we'll try to find an executable that has a name roughly similar to the specification (for exact logic, please see the implementation code).

Warning

As detailed above, virtual environments usually just borrow things from the system Python, they don't actually contain all the data from the system Python. The version of the python executable is hardcoded within the python exe itself. Therefore, if you upgrade your system Python, your virtual environment will still report the version before the upgrade, even though now other than the executable all additional content (standard library, binary libs, etc) are of the new version.

Barring any major incompatibilities (rarely the case) the virtual environment will continue working, but other than the
dded within the python executable it will behave like the upgraded version. If such a virtual environment
ified as the target python interpreter, we will create virtual environments that match the new system
on, not the version reported by the virtual environment.

Creators

These are what actually setup the virtual environment, usually as a reference against the system python. virtualenv at the moment has two types of virtual environments:

- `venv` - this delegates the creation process towards the `venv` module, as described in PEP 405. This is only available on Python interpreters having version `3.5` or later, and also has the downside that virtualenv **must** create a process to invoke that module (unless virtualenv is installed in the system python), which can be an expensive operation (especially true on Windows).
- `builtin` - this means `virtualenv` is able to do the creation operation itself (by knowing exactly what files to create and what system files need to be referenced). The creator with name `builtin` is an alias on the first creator that's of this type (we provide creators for various target environments, that all differ in actual create operations, such as CPython 2 on Windows, PyPy2 on Windows, CPython3 on Posix, PyPy3 on Posix, and so on; for a full list see `creator`).

Seeders

These will install for you some seed packages (one or more of: pip, setuptools, wheel) that enables you to install additional python packages into the created virtual environment (by invoking pip). Installing setuptools and wheel is disabled by default on Python 3.12+ environments. There are two main seed mechanisms available:

- `pip` - this method uses the bundled pip with virtualenv to install the seed packages (note, a new child process needs to be created to do this, which can be expensive especially on Windows).
- `app-data` - this method uses the user application data directory to create install images. These images are needed to be created only once, and subsequent virtual environments can just link/copy those images into their pure python library path (the `site-packages` folder). This allows all but the first virtual environment creation to be blazing fast (a `pip` mechanism takes usually 98% of the virtualenv creation time, so by creating this install image that we can just link into the virtual environments install directory we can achieve speedups of shaving the initial 1 minute and 10 seconds down to just 8 seconds in case of a copy, or `0.8` seconds in case symlinks are available - this is on Windows, Linux/macOS with symlinks this can be as low as `100ms` from 3+ seconds). To override the filesystem location of the seed cache, one can use the `VIRTUALENV_OVERRIDE_APP_DATA` environment variable.

Wheels

To install a seed package via either `pip` or `app-data` method virtualenv needs to acquire a wheel of the target package. These wheels may be acquired from multiple locations as follows:

- `virtualenv` ships out of box with a set of embed `wheels` for all three seed packages (pip, setuptools, wheel). These are packaged together with the virtualenv source files, and only change upon upgrading virtualenv. Different Python versions require different versions of these, and because virtualenv supports a wide range of Python versions, the number of embedded wheels out of box is greater than 3. Whenever newer versions of these embedded packages are released upstream `virtualenv` project upgrades them, and does a new release. Therefore, upgrading virtualenv periodically will also upgrade the version of the seed packages.
- However, end users might not be able to upgrade virtualenv at the same speed as we do new releases. Therefore, a user might request to upgrade the list of embedded wheels by invoking virtualenv with the `upgrade-embed-wheels` flag. If the operation is triggered in such a manual way subsequent runs of virtualenv will always use the upgraded embed wheels.

  The operation can trigger automatically too, as a background process upon invocation of virtualenv, if no such upgrade has been performed in the last 14 days. It will only start using automatically upgraded wheel if they have been released for more than 28 days, and the automatic upgrade finished at least an hour ago:

  - the 28 days period should guarantee end users are not pulling in automatically releases that have known bugs within,
  - the one hour period after the automatic upgrade finished is implemented so that continuous integration services do not start using a new embedded versions half way through.

  The automatic behavior might be disabled via the `no-periodic-update` configuration flag/option. To acquire the release date of a package virtualenv will perform the following:

  - lookup `https://pypi.org/pypi/<distribution>/json` (primary truth source),
  - save the date the version was first discovered, and wait until 28 days passed.
- Users can specify a set of local paths containing additional wheels by using the `extra-search-dir` command line argument flag.

When searching for a wheel to use virtualenv performs lookup in the following order:

- embedded wheels,
- upgraded embedded wheels,
- extra search dir.

Bundled wheels are all three above together. If neither of the locations contain the requested wheel version or `download` option is set will use `pip` download to load the latest version available from the index server.

Embed wheels for distributions

Skip to content

butions often want to use their own set of wheel versions to distribute instead of the
.v releases on PyPi. The reason for this is trying to keep the system versions of those packages in sync with what virtualenv uses. In such cases they should patch the module

virtualenv.seed.wheels.embed, making sure to provide the function `get_embed_wheel` (which returns the wheel to use given a distribution/python version). The `BUNDLE_FOLDER`, `BUNDLE_SUPPORT` and `MAX` variables are needed if they want to use virtualenv's test suite to validate.

Furthermore, they might want to disable the periodic update by patching the virtualenv.seed.embed.base_embed.PERIODIC_UPDATE_ON_BY_DEFAULT to `False`, and letting the system update mechanism to handle this. Note in this case the user might still request an upgrade of the embedded wheels by invoking virtualenv via `upgrade-embed-wheels`, but no longer happens automatically, and will not alter the OS provided wheels.

Activators

These are activation scripts that will mangle with your shell's settings to ensure that commands from within the python virtual environment take priority over your system paths. For example, if invoking `pip` from your shell returned the system python's pip before activation, once you do the activation this should refer to the virtual environments `pip`. Note, though that all we do is change priority; so, if your virtual environments `bin`/`Scripts` folder does not contain some executable, this will still resolve to the same executable it would have resolved before the activation.

For a list of shells we provide activators see `activators`. The location of these is right alongside the Python executables: usually `Scripts` folder on Windows, `bin` on POSIX. They are called `activate`, plus an extension that's specific per activator, with no extension for Bash. You can invoke them, usually by source-ing them. The source command might vary by shell - e.g. on Bash it's `source` (or `.`):

```
source venv/bin/activate
```

The activate script prepends the virtual environment's binary folder onto the `PATH` environment variable. It's really just convenience for doing so, since you could do the same yourself.

Note that you don't have to activate a virtual environment to use it. You can instead use the full paths to its executables, rather than relying on your shell to resolve them to your virtual environment.

Activator scripts also modify your shell prompt to indicate which environment is currently active, by prepending the environment name (or the name specified by `--prompt` when initially creating the environment) in brackets, like `(venv)`. You can disable this behavior by setting the environment variable `VIRTUAL_ENV_DISABLE_PROMPT` to any value. You can also get the environment name via the environment variable `VIRTUAL_ENV_PROMPT` if you want to customize your prompt, for example.

The scripts also provision a `deactivate` command that will allow you to undo the operation:

```
deactivate
```

Note

If using Powershell, the `activate` script is subject to the execution policies on the system. By default, Windows 7 and later, the system's execution policy is set to `Restricted`, meaning no scripts like the `activate` script are allowed to be executed.

However, that can't stop us from changing that slightly to allow it to be executed. You may relax the system execution policy to allow running of local scripts without verifying the code signature using the following:

```
Set-ExecutionPolicy RemoteSigned
```

Since the `activate.ps1` script is generated locally for each virtualenv, it is not considered a remote script and can then be executed.

A longer explanation of this can be found within Allison Kaptur's 2013 blog post: There's no magic: virtualenv edition explains how virtualenv uses bash and Python and `PATH` and `PYTHONHOME` to isolate virtual environments' paths.

Programmatic API

At the moment `virtualenv` offers only CLI level interface. If you want to trigger invocation of Python environments from within Python you should be using the `virtualenv.cli_run` method; this takes an `args` argument where you can pass the options the same way you would from the command line. The run will return a session object containing data about the created virtual environment.

```
from virtualenv import cli_run

cli_run(["venv"])
```

`virtualenv.cli_run(args, options=None, setup_logging=True, env=None)`

Create a virtual environment given some command line interface arguments.

PARAMETERS:

- **args** – the command line arguments
- **options** – passing in a `VirtualEnvOptions` object allows return of the parsed options
- **setup_logging** – `True` if setup logging handlers, `False` to use handlers already registered
- **env** – environment variables to use

RETURNS:

the session object of the creation (its structure for now is experimental and might change on short notice)

`virtualenv.session_via_cli(args, options=None, setup_logging=True, env=None)`

Create a virtualenv session (same as cli_run, but this does not perform the creation). Use this if you just want to query what the virtual environment would look like, but not actually create it.

PARAMETERS:

- **args** – the command line arguments
- **options** – passing in a `VirtualEnvOptions` object allows return of the parsed options
- **setup_logging** – `True` if setup logging handlers, `False` to use handlers already registered
- **env** – environment variables to use

RETURNS:

the session object of the creation (its structure for now is experimental and might change on short notice)

`class virtualenv.run.session.Session(verbosity, app_data, interpreter, creator, seeder, activators)`

Skip to content

Represents a virtual environment creation session.

property verbosity

The verbosity of the run.

property interpreter

Create a virtual environment based on this reference interpreter.

property creator

The creator used to build the virtual environment (must be compatible with the interpreter).

property seeder

The mechanism used to provide the seed packages (pip, setuptools, wheel).

property activators

Activators used to generate activations scripts.

CLI interface

CLI flags

`virtualenv` is primarily a command line application.

It modifies the environment variables in a shell to create an isolated Python environment, so you'll need to have a shell to run it. You can type in `virtualenv` (name of the application) followed by flags that control its behavior. All options have sensible defaults, and there's one required argument: the name/path of the virtual environment to create. The default values for the command line options can be overridden via the Configuration file or Environment Variables. Environment variables takes priority over the configuration file values (`--help` will show if a default comes from the environment variable as the help message will end in this case with environment variables or the configuration file).

The options that can be passed to virtualenv, along with their default values and a short description are listed below.

virtualenv [OPTIONS]

Named Arguments		
`--version`		display the version of the virtualenv package and its location, then exit
`--with-traceback`	`False`	on failure also display the stacktrace internals of virtualenv
`--read-only-app-data`	`False`	use app data folder in read-only mode (write operations will fail with error)
`--app-data`	platform specific application data folder	a data folder used as cache by the virtualenv
`--reset-app-data`	`False`	start with empty app data folder
`--upgrade-embed-wheels`	`False`	trigger a manual update of the embedded wheels

verbosity ⇒ verbosity = verbose - quiet, default INFO, mapping => CRITICAL=0, ERROR=1, WARNING=2, INFO=3, DEBUG=4, NOTSET=5		
`-v` `--verbose`	`2`	increase verbosity
`-q` `--quiet`	`0`	decrease verbosity

Skip to content

discovery

core ⇒ options shared across all discovery		
`--discovery`	`builtin`	interpreter discovery method; choice of: `builtin`
`-p`, `--python`	the python executable virtualenv is installed into	interpreter based on what to create environment (path/identifier) - by default use the interpreter where the tool is installed - first found wins
`--try-first-with`	`[]`	try first these interpreters before starting the discovery

creator

core ⇒ options shared across all creator		
`--creator`	`builtin` if exist, else `venv`	create environment via; choice of: `cpython3-mac-brew`, `cpython3-mac-framework`, `cpython3-posix`, `cpython3-win`, `pypy3-posix`, `pypy3-win`, `venv`
`dest`		directory to create virtualenv at
`--clear`	`False`	remove the destination directory if exist before starting (will overwrite files otherwise)
`--no-vcs-ignore`	`False`	don’t create VCS ignore directive in the destination directory
`--system-site-packages`	`False`	give the virtual environment access to the system site-packages dir
`--symlinks`	`True`	try to use symlinks rather than copies, when symlinks are not the default for the platform
`--copies`, `--always-copy`	`False`	try to use copies rather than symlinks, even when symlinks are the default for the platform

seeder

core ⇒ options shared across all seeder		
`--seeder`	`app-data`	seed packages install method; choice of: `app-data`, `pip`
`--no-seed`, `--without-pip`	`False`	do not install seed packages
`--no-download`, `--never-download`	`True`	pass to disable download of the latest pip/setuptools/wheel from PyPI
`--download`	`False`	pass to enable download of the latest pip/setuptools/wheel from PyPI
`--extra-search-dir`	`[]`	a path containing wheels to extend the internal wheel list (can be set 1+ times)
`--pip`	`bundle`	version of pip to install as seed: embed, bundle, none or exact version
`--setuptools`	`none`	version of setuptools to install as seed: embed, bundle, none or exact version
`--wheel`	`none`	version of wheel to install as seed: embed, bundle, none or exact version
`--no-pip`	`False`	do not install pip
`--no-setuptools`	`False`	do not install setuptools
`--no-wheel`	`False`	do not install wheel
`--no-periodic-update`	`False`	disable the periodic (once every 14 days) update of the embedded wheels

app-data ⇒ options specific to seeder app-data		
`--symlink-app-data`	`False`	symlink the python packages from the app-data folder (requires seed pip>=19.3)

activators

core ⇒ options shared across all activators		
`--activators`	comma separated list of activators supported	activators to generate - default is all supported; choice of: `bash`, `batch`, `cshell`, `fish`, `nushell`, `powershell`, `python`
		provides an alternative prompt prefix for this environment (value of . means name of the current working directory)

Defaults

Configuration file

Unless `VIRTUALENV_CONFIG_FILE` is set, virtualenv looks for a standard `virtualenv.ini` configuration file. The exact location depends on the operating system you're using, as determined by platformdirs application configuration definition. It can be overridden by setting the `VIRTUALENV_CONFIG_FILE` environment variable. The configuration file location is printed as at the end of the output when `--help` is passed.

The keys of the settings are derived from the command line option (left strip the `-` characters, and replace `-` with `_`). Where multiple flags are available first found wins (where order is as it shows up under the `--help`).

For example, `--python` would be specified as:

```
[virtualenv]
python = /opt/python-3.8/bin/python
```

Options that take multiple values, like `extra-search-dir` can be specified as:

```
[virtualenv]
extra_search_dir =
    /path/to/dists
    /path/to/other/dists
```

Environment Variables

Default values may be also specified via environment variables. The keys of the settings are derived from the command line option (left strip the `-` characters, and replace `-` with `_`, finally capitalize the name). Where multiple flags are available first found wins (where order is as it shows up under the `--help`).

For example, to use a custom Python binary, instead of the one virtualenv is run with, you can set the environment variable `VIRTUALENV_PYTHON` like:

```
env VIRTUALENV_PYTHON=/opt/python-3.8/bin/python virtualenv
```

Where the option accepts multiple values, for example for `python` or `extra-search-dir`, the values can be separated either by literal newlines or commas. Newlines and commas can not be mixed and if both are present only the newline is used for separating values. Examples for multiple values:

```
env VIRTUALENV_PYTHON=/opt/python-3.8/bin/python,python3.8 virtualenv
env VIRTUALENV_EXTRA_SEARCH_DIR=/path/to/dists\n/path/to/other/dists virtualenv
```

Skip to content

nt CLI-flags based invocation for the above examples would be:

```
virtualenv --python=/opt/python-3.8/bin/python --python=python3.8
virtualenv --extra-search-dir=/path/to/dists --extra-search-dir=/path/to/other/dists
```

Extend functionality

`virtualenv` allows one to extend the builtin functionality via a plugin system. To add a plugin you need to:

- write a python file containing the plugin code which follows our expected interface,
- package it as a python library,
- install it alongside the virtual environment.

Python discovery

The python discovery mechanism is a component that needs to answer the following answer: based on some type of user input give me a Python interpreter on the machine that matches that. The builtin interpreter tries to discover an installed Python interpreter (based on PEP-515 and `PATH` discovery) on the users machine where the user input is a python specification. An alternative such discovery mechanism for example would be to use the popular pyenv project to discover, and if not present install the requested Python interpreter. Python discovery mechanisms must be registered under key `virtualenv.discovery`, and the plugin must implement `virtualenv.discovery.discover.Discover`:

```
virtualenv.discovery =
     pyenv = virtualenv_pyenv.discovery:PyEnvDiscovery
```

class virtualenv.discovery.discover.**Discover**(options)

Discover and provide the requested Python interpreter.

Create a new discovery mechanism.

PARAMETERS:

options – the parsed options as defined within `add_parser_arguments()`

classmethod **add_parser_arguments**(parser)

Add CLI arguments for this discovery mechanisms.

PARAMETERS:

parser – the CLI parser

abstract **run**()

Discovers an interpreter.

RETURNS:

the interpreter ready to use for virtual environment creation

property **interpreter**

RETURNS:

the interpreter as returned by `run()`, cached

Creators

Creators are what actually perform the creation of a virtual environment. The builtin virtual environment creators all achieve this by referencing a global install; but would be just as valid for a creator to install a brand new entire python under the target path; or one could add additional creators that can create virtual environments for other python implementations, such as IronPython. They must be registered under and entry point with key `virtualenv.create` , and the class must implement `virtualenv.create.creator.Creator`:

```
virtualenv.create =
    cpython3-posix = virtualenv.create.via_global_ref.builtin.cpython.cpython3:CPython3
```

class virtualenv.create.creator.**Creator**(*options*, *interpreter*)

A class that given a python Interpreter creates a virtual environment.

Construct a new virtual environment creator.

PARAMETERS:

- **options** – the CLI option as parsed from `add_parser_arguments()`
- **interpreter** – the interpreter to create virtual environment from

classmethod **can_create**(*interpreter*)

Determine if we can create a virtual environment.

PARAMETERS:

interpreter – the interpreter in question

RETURNS:

`None` if we can't create, any other object otherwise that will be forwarded to `add_parser_arguments()`

classmethod **add_parser_arguments**(*parser*, *interpreter*, *meta*, *app_data*)

Add CLI arguments for the creator.

PARAMETERS:

- **parser** – the CLI parser
- **app_data** – the application data folder
- **interpreter** – the interpreter we're asked to create virtual environment for
- **meta** – value as returned by `can_create()`

abstract **create**()

Perform the virtual environment creation.

setup_ignore_vcs()

Generate ignore instructions for version control systems.

Seed mechanism

Seeders are what given a virtual environment will install somehow some seed packages into it. They must be registered under and entry point with key `virtualenv.seed` , and the class must implement `virtualenv.seed.seeder.Seeder`:

```
virtualenv.seed =
     db = virtualenv.seed.fromDb:InstallFromDb
```

class virtualenv.seed.seeder.**Seeder**(*options*, *enabled*)

A seeder will install some seed packages into a virtual environment.

Create.

PARAMETERS:

- **options** – the parsed options as defined within `add_parser_arguments()`
- **enabled** – a flag weather the seeder is enabled or not

classmethod **add_parser_arguments**(*parser*, *interpreter*, *app_data*)

Add CLI arguments for this seed mechanisms.

PARAMETERS:

- **parser** – the CLI parser
- **app_data** – the CLI parser
- **interpreter** – the interpreter this virtual environment is based of

abstract **run**(*creator*)

Perform the seed operation.

PARAMETERS:

creator – the creator (based of `virtualenv.create.creator.Creator`) we used to create this virtual environment

Activation scripts

If you want add an activator for a new shell you can do this by implementing a new activator. They must be registered under an entry point with key `virtualenv.activate` , and the class must implement `virtualenv.activation.activator.Activator`:

```
virtualenv.activate =
     bash = virtualenv.activation.bash:BashActivator
```

class virtualenv.activation.activator.**Activator**(options)

Generates activate script for the virtual environment.

Create a new activator generator.

PARAMETERS:

options – the parsed options as defined within `add_parser_arguments()`

classmethod **supports**(interpreter)

Check if the activation script is supported in the given interpreter.

PARAMETERS:

interpreter – the interpreter we need to support

RETURNS:

`True` if supported, `False` otherwise

classmethod **add_parser_arguments**(parser, interpreter)

Add CLI arguments for this activation script.

PARAMETERS:

- **parser** – the CLI parser
- **interpreter** – the interpreter this virtual environment is based of

abstract **generate**(creator)

Generate activate script for the given creator.

PARAMETERS:

creator – the creator (based of `virtualenv.create.creator.Creator`) we used to create this virtual environment

Development

Getting started

`virtualenv` is a volunteer maintained open source project and we welcome contributions of all forms. The sections below will help you get started with development, testing, and documentation. We're pleased that you are interested in working on virtualenv. This document is meant to get you setup to work on virtualenv and to act as a guide and reference to the development setup. If you face any issues during this process, please open an issue about it on the issue tracker.

Setup

virtualenv is a command line application written in Python. To work on it, you'll need:

- **Source code**: available on GitHub. You can use `git` to clone the repository:

```
git clone https://github.com/pypa/virtualenv
cd virtualenv
```

- **Python interpreter**: We recommend using `CPython`. You can use this guide to set it up.
- tox: to automatically get the projects development dependencies and run the test suite. We recommend installing it using pipx.

Running from source tree

The easiest way to do this is to generate the development tox environment, and then invoke virtualenv from under the `.tox/dev` folder

```
tox -e dev
.tox/dev/bin/virtualenv  # on Linux
.tox/dev/Scripts/virtualenv  # on Windows
```

Running tests

virtualenv's tests are written using the pytest test framework. tox is used to automate the setup and execution of virtualenv's tests.

To run tests locally execute:

```
tox -e py
```


… the test suite for the same Python version as under which `tox` is installed. Alternatively you can specify a specific version of python by using the `pyNN` format, such as:

`py38`, `pypy3`, etc.

`tox` has been configured to forward any additional arguments it is given to `pytest`. This enables the use of pytest's rich CLI. As an example, you can select tests using the various ways that pytest provides:

```
# Using markers
tox -e py -- -m "not slow"
# Using keywords
tox -e py -- -k "test_extra"
```

Some tests require additional dependencies to be run, such is the various shell activators (`bash`, `fish`, `powershell`, etc). These tests will automatically be skipped if these are not present, note however that in CI all tests are run; so even if all tests succeed locally for you, they may still fail in the CI.

Running linters

virtualenv uses pre-commit for managing linting of the codebase. `pre-commit` performs various checks on all files in virtualenv and uses tools that help follow a consistent code style within the codebase. To use linters locally, run:

```
tox -e fix
```

> **Note**
>
> Avoid using `# noqa` comments to suppress linter warnings - wherever possible, warnings should be fixed instead. `# noqa` comments are reserved for rare cases where the recommended style causes severe readability problems.

Building documentation

virtualenv's documentation is built using Sphinx. The documentation is written in reStructuredText. To build it locally, run:

```
tox -e docs
```

The built documentation can be found in the `.tox/docs_out` folder and may be viewed by opening `index.html` within that folder.

Release

virtualenv's release schedule is tied to `pip`, `setuptools` and `wheel`. We bundle the latest version of these libraries so each time there's a new version of any of these, there will be a new virtualenv release shortly afterwards (we usually wait just a few days to avoid pulling in any broken releases).

Contributing

Skip to content

ing pull requests

Submit pull requests against the `main` branch, providing a good description of what you're doing and why. You must have legal permission to distribute any code you contribute to virtualenv and it

must be available under the MIT License. Provide tests that cover your changes and run the tests locally first. virtualenv supports multiple Python versions and operating systems. Any pull request must consider and work on all these platforms.

Pull Requests should be small to facilitate review. Keep them self-contained, and limited in scope. Studies have shown that review quality falls off as patch size grows. Sometimes this will result in many small PRs to land a single large feature. In particular, pull requests must not be treated as "feature branches", with ongoing development work happening within the PR. Instead, the feature should be broken up into smaller, independent parts which can be reviewed and merged individually.

Additionally, avoid including "cosmetic" changes to code that is unrelated to your change, as these make reviewing the PR more difficult. Examples include re-flowing text in comments or documentation, or addition or removal of blank lines or whitespace within lines. Such changes can be made separately, as a "formatting cleanup" PR, if needed.

Automated testing

All pull requests and merges to 'main' branch are tested using Azure Pipelines (configured by `azure-pipelines.yml` file at the root of the repository). You can find the status and results to the CI runs for your PR on GitHub's Web UI for the pull request. You can also find links to the CI services' pages for the specific builds in the form of "Details" links, in case the CI run fails and you wish to view the output.

To trigger CI to run again for a pull request, you can close and open the pull request or submit another change to the pull request. If needed, project maintainers can manually trigger a restart of a job/build.

NEWS entries

The `changelog.rst` file is managed using towncrier and all non trivial changes must be accompanied by a news entry. To add an entry to the news file, first you need to have created an issue describing the change you want to make. A Pull Request itself *may* function as such, but it is preferred to have a dedicated issue (for example, in case the PR ends up rejected due to code quality reasons).

Once you have an issue or pull request, you take the number and you create a file inside of the `docs/changelog` directory named after that issue number with an extension of:

- `feature.rst`,
- `bugfix.rst`,
- `doc.rst`,
- `removal.rst`,
- `misc.rst`.

Thus if your issue or PR number is `1234` and this change is fixing a bug, then you would create a file `docs/changelog/1234.bugfix.rst`. PRs can span multiple categories by creating multiple files (for example, if you added a feature and deprecated/removed the old feature at the same time, you would create `docs/changelog/1234.bugfix.rst` and `docs/changelog/1234.remove.rst`).

Likewise if a PR touches multiple issues/PRs you may create a file for each of them with the same contents and towncrier will deduplicate them.

Contents of a NEWS entry

The contents of this file are reStructuredText formatted text that will be used as the content of the news file entry. You do not need to reference the issue or PR numbers here as towncrier will automatically add a reference to all of the affected issues when rendering the news file.

In order to maintain a consistent style in the `changelog.rst` file, it is preferred to keep the news entry to the point, in sentence case, shorter than 120 characters and in an imperative tone – an entry should complete the sentence `This change will …`. In rare cases, where one line is not enough, use a summary line in an imperative tone followed by a blank line separating it from a description of the feature/change in one or more paragraphs, each wrapped at 120 characters. Remember that a news entry is meant for end users and should only contain details relevant to an end user.

Choosing the type of NEWS entry

A trivial change is anything that does not warrant an entry in the news file. Some examples are: code refactors that don't change anything as far as the public is concerned, typo fixes, white space modification, etc. To mark a PR as trivial a contributor simply needs to add a randomly named, empty file to the `news/` directory with the extension of `.trivial`.

Becoming a maintainer

If you want to become an official maintainer, start by helping out. As a first step, we welcome you to triage issues on virtualenv's issue tracker. virtualenv maintainers provide triage abilities to contributors once they have been around for some time and contributed positively to the project. This is optional and highly recommended for becoming a virtualenv maintainer. Later, when you think you're ready, get in touch with one of the maintainers and they will initiate a vote among the existing maintainers.

> **Note**
>
> Upon becoming a maintainer, a person should be given access to various virtualenv-related tooling across multiple platforms. These are noted here for future reference by the maintainers:
>
> - GitHub Push Access
> - PyPI Publishing Access
> - CI Administration capabilities
> - ReadTheDocs Administration capabilities

Release History

v20.26.3 (2024-06-21)

Bugfixes - 20.26.3

- Upgrade embedded wheels:
  - setuptools to `70.1.0` from `69.5.1`
  - pip to `24.1``from ``24.0` (#2741)

v20.26.2 (2024-05-13)

Bugfixes - 20.26.2

- `virtualenv.pyz` no longer fails when zipapp path contains a symlink - by @HandSonic and @petamas. (#1949)
- Fix bad return code from activate.sh if hashing is disabled - by :user:'fenkes-ibm'. (#2717)

v20.26.1 (2024-04-29)

Bugfixes - 20.26.1

- fix PATH-based Python discovery on Windows - by @ofek. (#2712)

v20.26.0 (2024-04-23)

Bugfixes - 20.26.0

- allow builtin discovery to discover specific interpreters (e.g. `python3.12`) given an unspecific spec (e.g. `python3`) - by @flying-sheep. (#2709)

v20.25.3 (2024-04-17)

Bugfixes - 20.25.3

- Python 3.13.0a6 renamed pathmod to parser. (#2702)

Skip to content

v20.25.2 (2024-04-16)

Bugfixes - 20.25.2

- Upgrade embedded wheels:
  - setuptools of `69.1.0` to `69.5.1`
  - wheel of `0.42.0` to `0.43.0` (#2699)

v20.25.1 (2024-02-21)

Bugfixes - 20.25.1

- Upgrade embedded wheels:
  - setuptools to `69.0.3` from `69.0.2`
  - pip to `23.3.2` from `23.3.1` (#2681)
- Upgrade embedded wheels:
  - pip `23.3.2` to `24.0`,
  - setuptools `69.0.3` to `69.1.0`. (#2691)

Misc - 20.25.1

- #2688

v20.25.0 (2023-12-01)

Features - 20.25.0

- The tests now pass on the CI with Python 3.13.0a2 - by @hroncok. (#2673)

Bugfixes - 20.25.0

- Upgrade embedded wheels:
  - wheel to `0.41.3` from `0.41.2` (#2665)
- Upgrade embedded wheels:
  - wheel to `0.42.0` from `0.41.3`
  - setuptools to `69.0.2` from `68.2.2` (#2669)

v20.24.6 (2023-10-23)

Bugfixes - 20.24.6

- Use get_hookimpls method instead of the private attribute in tests. (#2649)
- Upgrade embedded wheels:
  - setuptools to `68.2.2` from `68.2.0`
  - pip to `23.3.1` from `23.2.1` (#2656)

v20.24.5 (2023-09-08)

Bugfixes - 20.24.5

- Declare PyPy 3.10 support - by @cclauss. (#2638)
- Brew on macOS no longer allows copy builds - disallow choosing this by @gaborbernat. (#2640)
- Upgrade embedded wheels:
  - setuptools to `68.2.0` from `68.1.2` (#2642)

v20.24.4 (2023-08-30)

Bugfixes - 20.24.4

- Upgrade embedded wheels:
  - setuptools to `68.1.2` from `68.1.0` on `3.8+`
  - wheel to `0.41.2` from `0.41.1` on `3.7+` (#2628)

v20.24.3 (2023-08-11)

Bugfixes - 20.24.3

- Fixed ResourceWarning on exit caused by periodic update subprocess (#2472)
- Upgrade embedded wheels:
  - wheel to `0.41.1` from `0.41.0` (#2622)

Misc - 20.24.3

- #2610

v20.24.2 (2023-07-24)

Bugfixes - 20.24.2

- Upgrade embedded wheels:
  - pip to `23.2.1` from `23.2`
  - wheel to `0.41.0` from `0.40.0` (#2614)

v20.24.1 (2023-07-19)

Bugfixes - 20.24.1

Skip to content

mbedded wheels:

- pip to `23.2` from `23.1.2` - by @arielkirkwood (#2611)

v20.24.0 (2023-07-14)

Features - 20.24.0

- Export the prompt prefix as `VIRTUAL_ENV_PROMPT` when activating a virtual environment - by @jimporter. (#2194)

Bugfixes - 20.24.0

- Fix test suite - by @gaborbernat. (#2592)
- Upgrade embedded wheels:
  - setuptools to `68.0.0` from `67.8.0` (#2607)

v20.23.1 (2023-06-16)

Bugfixes - 20.23.1

- update and simplify nushell activation script, fixes an issue on Windows resulting in consecutive command not found - by @melMass. (#2572)
- Upgrade embedded wheels:
  - setuptools to `67.8.0` from `67.7.2` (#2588)

v20.23.0 (2023-04-27)

Features - 20.23.0

- Do not install `wheel` and `setuptools` seed packages for Python 3.12+. To restore the old behavior use:
  - for `wheel` use `VIRTUALENV_WHEEL=bundle` environment variable or `--wheel=bundle` CLI flag,
  - for `setuptools` use `VIRTUALENV_SETUPTOOLS=bundle` environment variable or `--setuptools=bundle` CLI flag.

  By @chrysle. (#2487)
- 3.12 support - by @gaborbernat. (#2558)

Bugfixes - 20.23.0

- Prevent `PermissionError` when using venv creator on systems that deliver files without user write permission - by @kulikjak. (#2543)
- Upgrade setuptools to `67.7.2` from `67.6.1` and pip to `23.1.2` from `23.1` - by @szleb. (#2560)

v20.22.0 (2023-04-19)

Skip to content

s - 20.22.0

- Drop support for creating Python <=3.6 (including 2) interpreters. Removed pip of `20.3.4`, `21.3.1`; wheel of `0.37.1`; setuptools of `59.6.0`, `44.1.1`, `50.3.2` - by @gaborbernat. (#2548)

v20.21.1 (2023-04-19)

Bugfixes - 20.21.1

- Add `tox.ini` to sdist - by @mtelka. (#2511)
- Move the use of ‘let’ in nushell to ensure compatibility with future releases of nushell, where ‘let’ no longer assumes that its initializer is a full expressions. (#2527)
- The nushell command ‘str collect’ has been superseded by the ‘str join’ command. The activate.nu script has been updated to reflect this change. (#2532)
- Upgrade embedded wheels:
  - wheel to `0.40.0` from `0.38.4`
  - setuptools to `67.6.1` from `67.4.0`
  - pip to `23.1` from `23.0.1` (#2546)

v20.21.0 (2023-03-12)

Features - 20.21.0

- Make closure syntax explicitly starts with {||. (#2512)

Bugfixes - 20.21.0

- Add `print` command to nushell print_prompt to ensure compatibility with future release of nushell, where intermediate commands no longer print their result to stdout. (#2514)
- Do not assume the default encoding. (#2515)
- Make `ReentrantFileLock` thread-safe and, thereby, fix race condition in `virtualenv.cli_run` - by @radoering. (#2516)

v20.20.0 (2023-02-28)

Features - 20.20.0

- Change environment variable existence check in Nushell activation script to not use deprecated command. (#2506)

Bugfixes - 20.20.0

- Discover CPython implementations distributed on Windows by any organization - by @faph. (#2504)
- Upgrade embedded setuptools to `67.4.0` from `67.1.0` and pip to `23.0.1` from `23.0` - by @gaborbernat. (#2510)

v20.19.0 (2023-02-07)

Features - 20.19.0

- Allow platformdirs version `3` - by @cdce8p. (#2499)

v20.18.0 (2023-02-06)

Features - 20.18.0

- Drop `3.6` runtime support (can still create `2.7+`) - by @gaborbernat. (#2489)

Bugfixes - 20.18.0

- Fix broken prompt in Nushell when activating virtual environment - by @kubouc. (#2481)
- Bump embedded pip to `23.0` and setuptools to `67.1` - by @gaborbernat. (#2489)

v20.17.1 (2022-12-05)

Bugfixes - 20.17.1

- A `py` or `python` spec means any Python rather than `CPython` - by @gaborbernat. (#2460)
- Make `activate.nu` respect `VIRTUAL_ENV_DISABLE_PROMPT` and not set the prompt if requested - by @m-lima. (#2461)

v20.17.0 (2022-11-27)

Features - 20.17.0

- Change Nushell activation script to be a module meant to be activated as an overlay. (#2422)
- Update operator used in Nushell activation script to be compatible with future versions. (#2450)

Bugfixes - 20.17.0

- Do not use deprecated API from `importlib.resources` on Python 3.10 or later - by @gaborbernat. (#2448)
- Upgrade embedded setuptools to `65.6.3` from `65.5.1` - by @gaborbernat. (#2451)

v20.16.7 (2022-11-12)

Bugfixes - 20.16.7

- Use parent directory of python executable for pyvenv.cfg “home” value per PEP 405 - by @vfazio. (#2440)
- In POSIX virtual environments, try alternate binary names if `sys._base_executable` does not exist - by @vfazio. (#2442)
- Upgrade embedded wheel to `0.38.4` and pip to `22.3.1` from `22.3` and setuptools to `65.5.1` from `65.5.0` - by @gaborbernat. (#2443)

v20.16.6 (2022-10-25)

Features - 20.16.6

- Drop unneeded shims for PyPy3 directory structure (#2426)

Bugfixes - 20.16.6

- Fix selected scheme on debian derivatives for python 3.10 when `python3-distutils` is not installed or the `venv` scheme is not available - by @asottile. (#2350)
- Allow the test suite to pass even with the original C shell (rather than `tcsh`) - by @kulikjak. (#2418)
- Fix fallback handling of downloading wheels for bundled packages - by @schaap. (#2429)
- Upgrade embedded setuptools to `65.5.0` from `65.3.0` and pip to `22.3` from `22.2.2` - by @gaborbernat. (#2434)

v20.16.5 (2022-09-07)

Bugfixes - 20.16.5

- Do not turn echo off for subsequent commands in batch activators (`activate.bat` and `deactivate.bat`) - by @pawelszramowski. (#2411)

v20.16.4 (2022-08-29)

Bugfixes - 20.16.4

- Bump embed setuptools to `65.3` - by @gaborbernat. (#2405)

v20.16.3 (2022-08-04)

Bugfixes - 20.16.3

- Upgrade embedded pip to `22.2.2` from `22.2.1` and setuptools to `63.4.1` from `63.2.0` - by @gaborbernat. (#2395)

v20.16.2 (2022-07-27)

Bugfixes - 20.16.2

- Bump embedded pip from `22.2` to `22.2.1` - by @gaborbernat. (#2391)

v20.16.1 (2022-07-26)

Features - 20.16.1

- Update Nushell activation scripts to version 0.67 - by @kubouch. (#2386)

v20.16.0 (2022-07-25)

Features - 20.16.0

- Drop support for running under Python 2 (still can generate Python 2 environments) - by @gaborbernat. (#2382)
- Upgrade embedded pip to `22.2` from `22.1.2` and setuptools to `63.2.0` from `62.6.0` - by @gaborbernat. (#2383)

v20.15.1 (2022-06-28)

Bugfixes - 20.15.1

- Fix the incorrect operation when `setuptools` plugins output something into `stdout`. (#2335)
- CPython3Windows creator ignores missing `DLLs` dir. (#2368)

v20.15.0 (2022-06-25)

Features - 20.15.0

- Support for Windows embeddable Python package: includes `python<VERSION>.zip` in the
  urces - by @reksarka. (#1774)

Bugfixes - 20.15.0

- Upgrade embedded setuptools to `62.3.3` from `62.6.0` and pip to `22.1.2` from `22.0.4` - by @gaborbernat. (#2348)
- Use `shlex.quote` instead of deprecated `pipes.quote` in Python 3 - by @frenzymadness. (#2351)
- Fix Windows PyPy 3.6 - by @reksarka. (#2363)

v20.14.1 (2022-04-11)

Features - 20.14.1

- Support for creating a virtual environment from a Python 2.7 framework on macOS 12 - by @nickhutchinson. (#2284)

Bugfixes - 20.14.1

- Upgrade embedded setuptools to `62.1.0` from `61.0.0` - by @gaborbernat. (#2327)

v20.14.0 (2022-03-25)

Features - 20.14.0

- Support Nushell activation scripts with nu version `0.60` - by @kubouch. (#2321)

Bugfixes - 20.14.0

- Upgrade embedded setuptools to `61.0.0` from `60.10.0` - by @gaborbernat. (#2322)

v20.13.4 (2022-03-18)

Bugfixes - 20.13.4

- Improve performance of python startup inside created virtualenvs - by @asottile. (#2317)
- Upgrade embedded setuptools to `60.10.0` from `60.9.3` - by @gaborbernat. (#2320)

v20.13.3 (2022-03-07)

Bugfixes - 20.13.3

- Avoid symlinking the contents of `/usr` into PyPy3.8+ virtualenvs - by @stefanor. (#2310)
- Bump embed pip from `22.0.3` to `22.0.4` - by @gaborbernat. (#2311)

v20.13.2 (2022-02-24)

Bugfixes - 20.13.2

- Upgrade embedded setuptools to `60.9.3` from `60.6.0` - by @gaborbernat. (#2306)

v20.13.1 (2022-02-05)

Bugfixes - 20.13.1

- fix “execv() arg 2 must contain only strings” error on M1 MacOS (#2282)
- Upgrade embedded setuptools to `60.5.0` from `60.2.0` - by @asottile. (#2289)
- Upgrade embedded pip to `22.0.3` and setuptools to `60.6.0` - by @gaborbernat and @asottile. (#2294)

v20.13.0 (2022-01-02)

Features - 20.13.0

- Add downloaded wheel information in the relevant JSON embed file to prevent additional downloads of the same wheel. - by @mayeut. (#2268)

Bugfixes - 20.13.0

- Fix `AttributeError: 'bool' object has no attribute 'error'` when creating a Python 2.x virtualenv on macOS - by `moreati`. (#2269)
- Fix `PermissionError: [Errno 1] Operation not permitted` when creating a Python 2.x virtualenv on macOS/arm64 - by `moreati`. (#2271)

v20.12.1 (2022-01-01)

Bugfixes - 20.12.1

- Try using previous updates of `pip`, `setuptools` & `wheel` when inside an update grace period rather than always falling back to embedded wheels - by @mayeut. (#2265)
- New patch versions of `pip`, `setuptools` & `wheel` are now returned in the expected timeframe. - by @mayeut. (#2266)
- Manual upgrades of `pip`, `setuptools` & `wheel` are not discarded by a periodic update - by @mayeut. (#2267)

v20.12.0 (2021-12-31)

ذ - 20.12.0

- Sign the python2 exe on Darwin arm64 - by @tmspicer. (#2233)

Bugfixes - 20.12.0

- Fix `--download` option - by @mayeut. (#2120)
- Upgrade embedded setuptools to `60.2.0` from `60.1.1` - by @gaborbernat. (#2263)

v20.11.2 (2021-12-29)

Bugfixes - 20.11.2

- Fix installation of pinned versions of `pip`, `setuptools` & `wheel` - by @mayeut. (#2203)

v20.11.1 (2021-12-29)

Bugfixes - 20.11.1

- Bump embed setuptools to `60.1.1` from `60.1.0` - by @gaborbernat. (#2258)

v20.11.0 (2021-12-28)

Features - 20.11.0

- Avoid deprecation warning from py-filelock argument - by @ofek. (#2237)
- Upgrade embedded setuptools to `61.1.0` from `58.3.0` - by @gaborbernat. (#2240)
- Drop the runtime dependency of `backports.entry-points-selectable` - by @hroncok. (#2246)
- Fish: PATH variables should not be quoted when being set - by @d3dave. (#2248)

v20.10.0 (2021-11-01)

Features - 20.10.0

- If a `"venv"` install scheme exists in `sysconfig`, virtualenv now uses it to create new virtual environments. This allows Python distributors, such as Fedora, to patch/replace the default install scheme without affecting the paths in new virtual environments. A similar technique was proposed to Python, for the venv module - by `hroncok` (#2208)
- The activated virtualenv prompt is now always wrapped in parentheses. This affects venvs created with the `--prompt` attribute, and matches virtualenv's behavior on par with venv. (#2224)

Bugfixes - 20.10.0

- Fix broken prompt set up by activate.bat - by @SiggyBar. (#2225)

v20.9.0 (2021-10-23)

Features - 20.9.0

- Special-case `--prompt .` to the name of the current directory - by @rkm. (#2220)
- Add libffi-8.dll to pypy windows #2218 - by @mattip

Bugfixes - 20.9.0

- Fixed path collision that could lead to a PermissionError or writing to system directories when using PyPy3.8 - by @mgorny. (#2182)
- Upgrade embedded setuptools to `58.3.0` from `58.1.0` and pip to `21.3.1` from `21.2.4` - by @gaborbernat. (#2205)
- Remove stray closing parenthesis in activate.bat - by @SiggyBar. (#2221)

v20.8.1 (2021-09-24)

Bugfixes - 20.8.1

- Fixed a bug where while creating a venv on top of an existing one, without cleaning, when seeded wheel version mismatch occurred, multiple `.dist-info` directories may be present, confounding entrypoint discovery - by @arcivanov (#2185)
- Bump embed setuptools from `58.0.4` to `58.1.0` - by @gaborbernat. (#2195)

Misc - 20.8.1

- #2189

v20.8.0 (2021-09-16)

- upgrade embedded setuptools to `58.0.4` from `57.4.0` and pip to `21.2.4` from `21.2.3`
- Add nushell activation script

v20.7.2 (2021-08-10)

Bugfixes - 20.7.2

- Upgrade embedded pip to `21.2.3` from `21.2.2` and wheel to `0.37.0` from `0.36.2` - by @gaborbernat. (#2168)

v20.7.1 (2021-08-09)

Skip to content

s - 20.7.1

- Fix unpacking dictionary items in PythonInfo.install_path (#2165)

v20.7.0 (2021-07-31)

Bugfixes - 20.7.0

- upgrade embedded pip to `21.2.2` from `21.1.3` and setuptools to `57.4.0` from `57.1.0` - by @gaborbernat (#2159)

Deprecations and Removals - 20.7.0

- Removed `xonsh` activator due to this breaking fairly often the CI and lack of support from those packages maintainers, upstream is encouraged to continue supporting the project as a plugin - by @gaborbernat. (#2160)

v20.6.0 (2021-07-14)

Features - 20.6.0

- Support Python interpreters without `distutils` (fallback to `syconfig` in these cases) - by @gaborbernat. (#1910)

v20.5.0 (2021-07-13)

Features - 20.5.0

- Plugins now use ‘selectable’ entry points - by @jaraco. (#2093)
- add libffi-7.dll to the hard-coded list of dlls for PyPy (#2141)
- Use the better maintained `platformdirs` instead of `appdirs` - by @gaborbernat. (#2142)

Bugfixes - 20.5.0

- Bump pip the embedded pip `21.1.3` and setuptools to `57.1.0` - by @gaborbernat. (#2135)

Deprecations and Removals - 20.5.0

- Drop python `3.4` support as it has been over 2 years since EOL - by @gaborbernat. (#2141)

v20.4.7 (2021-05-24)

Bugfixes - 20.4.7

- Upgrade embedded pip to `21.1.2` and setuptools to `57.0.0` - by @gaborbernat. (#2123)

v20.4.6 (2021-05-05)

Bugfixes - 20.4.6

- Fix `site.getsitepackages()` broken on python2 on debian - by @freundTech. (#2105)

v20.4.5 (2021-05-05)

Bugfixes - 20.4.5

- Bump pip to `21.1.1` from `21.0.1` - by @gaborbernat. (#2104)
- Fix `site.getsitepackages()` ignoring `--system-site-packages` on python2 - by @freundTech. (#2106)

v20.4.4 (2021-04-20)

Bugfixes - 20.4.4

- Built in discovery class is always preferred over plugin supplied classes. (#2087)
- Upgrade embedded setuptools to `56.0.0` by @gaborbernat. (#2094)

v20.4.3 (2021-03-16)

Bugfixes - 20.4.3

- Bump embedded setuptools from `52.0.0` to `54.1.2` - by @gaborbernat (#2069)
- Fix PyPy3 stdlib on Windows is incorrect - by @gaborbernat. (#2071)

v20.4.2 (2021-02-01)

Bugfixes - 20.4.2

- Running virtualenv `--upgrade-embed-wheels` crashes - by @gaborbernat. (#2058)

v20.4.1 (2021-01-31)

Bugfixes - 20.4.1

- Bump embedded pip and setuptools packages to latest upstream supported (`21.0.1` and `52.0.0`) - by @gaborbernat. (#2060)

v20.4.0 (2021-01-19)

Features - 20.4.0

- On the programmatic API allow passing in the environment variable dictionary to use, defaults to `os.environ` if not specified - by @gaborbernat. (#2054)

Bugfixes - 20.4.0

- Upgrade embedded setuptools to `51.3.3` from `51.1.2` - by @gaborbernat. (#2055)

v20.3.1 (2021-01-13)

Bugfixes - 20.3.1

- Bump embed pip to `20.3.3`, setuptools to `51.1.1` and wheel to `0.36.2` - by @gaborbernat. (#2036)
- Allow unfunctioning of pydoc to fail freely so that virtualenvs can be activated under Zsh with set -e (since otherwise `unset -f` and `unfunction` exit with 1 if the function does not exist in Zsh) - by @d125q. (#2049)
- Drop cached python information if the system executable is no longer present (for example when the executable is a shim and the mapped executable is replaced - such is the case with pyenv) - by @gaborbernat. (#2050)

v20.3.0 (2021-01-10)

Features - 20.3.0

- The builtin discovery takes now a `--try-first-with` argument and is first attempted as valid interpreters. One can use this to force discovery of a given python executable when the discovery order/mechanism raises errors - by @gaborbernat. (#2046)

Bugfixes - 20.3.0

- On Windows python `3.7+` distributions where the exe shim is missing fallback to the old ways - by @gaborbernat. (#1986)
- When discovering interpreters on Windows, via the PEP-514, prefer `PythonCore` releases over other ones. virtualenv is used via pip mostly by this distribution, so prefer it over other such as conda - by @gaborbernat. (#2046)

v20.2.2 (2020-12-07)

Skip to content

s - 20.2.2

- Bump pip to `20.3.1`, setuptools to `51.0.0` and wheel to `0.36.1` - by @gaborbernat. (#2029)

v20.2.1 (2020-11-23)

No significant changes.

v20.2.0 (2020-11-21)

Features - 20.2.0

- Optionally skip VCS ignore directive for entire virtualenv directory, using option `no-vcs-ignore`, by default `False`. (#2003)
- Add `--read-only-app-data` option to allow for creation based on an existing app data cache which is non-writable. This may be useful (for example) to produce a docker image where the app-data is pre-populated.

  ```
  ENV \
      VIRTUALENV_OVERRIDE_APP_DATA=/opt/virtualenv/cache \
      VIRTUALENV_SYMLINK_APP_DATA=1
  RUN virtualenv venv && rm -rf venv
  ENV VIRTUALENV_READ_ONLY_APP_DATA=1
  USER nobody
  # this virtualenv has symlinks into the read-only app-data cache
  RUN virtualenv /tmp/venv
  ```

  Patch by @asottile. (#2009)

Bugfixes - 20.2.0

- Fix processing of the `VIRTUALENV_PYTHON` environment variable and make it multi-value as well (separated by comma) - by @pneff. (#1998)

v20.1.0 (2020-10-25)

Features - 20.1.0

- The python specification can now take one or more values, first found is used to create the virtual environment - by @gaborbernat. (#1995)

v20.0.35 (2020-10-15)

Bugfixes - 20.0.35

- Bump embedded setuptools from `50.3.0` to `50.3.1` - by @gaborbernat. (#1982)
- After importing virtualenv passing cwd to a subprocess calls breaks with `invalid directory` - by @gaborbernat. (#1983)

v20.0.34 (2020-10-12)

Bugfixes - 20.0.34

- Align with venv module when creating virtual environments with builtin creator on Windows 3.7 and later - by @gaborbernat. (#1782)
- Handle Cygwin path conversion in the activation script - by @davidcoghlan. (#1969)

v20.0.33 (2020-10-04)

Bugfixes - 20.0.33

- Fix `None` type error in cygwin if POSIX path in dest - by @danyeaw. (#1962)
- Fix Python 3.4 incompatibilities (added back to the CI) - by @gaborbernat. (#1963)

v20.0.32 (2020-10-01)

Bugfixes - 20.0.32

- For activation scripts always use UNIX line endings (unless it's BATCH shell related) - by @saytosid. (#1818)
- Upgrade embedded pip to `20.2.1` and setuptools to `49.4.0` - by @gaborbernat. (#1918)
- Avoid spawning new windows when doing seed package upgrades in the background on Windows - by @gaborbernat. (#1928)
- Fix a bug that reading and writing on the same file may cause race on multiple processes. (#1938)
- Upgrade embedded setuptools to `50.2.0` and pip to `20.2.3` - by @gaborbernat. (#1939)
- Provide correct path for bash activator in cygwin or msys2 - by @danyeaw. (#1940)
- Relax importlib requirement to allow version<3 - by @usamasadiq (#1953)
- pth files were not processed on CPython2 if $PYTHONPATH was pointing to site-packages/ - by @navytux. (#1959) (#1960)

v20.0.31 (2020-08-17)

Bugfixes - 20.0.31

- Upgrade embedded pip to `20.2.1`, setuptools to `49.6.0` and wheel to `0.35.1` - by @gaborbernat. (#1918)

v20.0.30 (2020-08-04)

s - 20.0.30

- Upgrade pip to `20.2.1` and setuptools to `49.2.1` - by @gaborbernat. (#1915)

v20.0.29 (2020-07-31)

Bugfixes - 20.0.29

- Upgrade embedded pip from version `20.1.2` to `20.2` - by @gaborbernat. (#1909)

v20.0.28 (2020-07-24)

Bugfixes - 20.0.28

- Fix test suite failing if run from system Python - by @gaborbernat. (#1882)
- Provide `setup_logging` flag to python API so that users can bypass logging handling if their application already performs this - by @gaborbernat. (#1896)
- Use `\n` instead if `\r\n` as line separator for report (because Python already performs this transformation automatically upon write to the logging pipe) - by @gaborbernat. (#1905)

v20.0.27 (2020-07-15)

Bugfixes - 20.0.27

- No longer preimport threading to fix support for gpython and gevent - by @navytux. (#1897)
- Upgrade setuptools from `49.2.0` on `Python 3.5+` - by @gaborbernat. (#1898)

v20.0.26 (2020-07-07)

Bugfixes - 20.0.26

- Bump dependency `distutils >= 0.3.1` - by @gaborbernat. (#1880)
- Improve periodic update handling:
  - better logging output while running and enable logging on background process call (`_VIRTUALENV_PERIODIC_UPDATE_INLINE` may be used to debug behavior inline)
  - fallback to unverified context when querying the PyPi for release date,
  - stop downloading wheels once we reach the embedded version,

  by @gaborbernat. (#1883)
- Do not print error message if the application exists with `SystemExit(0)` - by @gaborbernat. (#1885)
- Upgrade embedded setuptools from `47.3.1` to `49.1.0` for Python `3.5+` - by @gaborbernat. (#1887)

v20.0.25 (2020-06-23)

Bugfixes - 20.0.25

- Fix that when the `app-data` seeders image creation fails the exception is silently ignored. Avoid two virtual environment creations to step on each others toes by using a lock while creating the base images. By @gaborbernat. (#1869)

v20.0.24 (2020-06-22)

Features - 20.0.24

- Ensure that the seeded packages do not get too much out of date:
  - add a CLI flag that triggers upgrade of embedded wheels under `upgrade-embed-wheels`
  - periodically (once every 14 days) upgrade the embedded wheels in a background process, and use them if they have been released for more than 28 days (can be disabled via `no-periodic-update`)

  More details under Wheels - by @gaborbernat. (#1821)
- Upgrade embed wheel content:
  - ship wheels for Python `3.9` and `3.10`
  - upgrade setuptools for Python `3.5+` from `47.1.1` to `47.3.1`

  by @gaborbernat. (#1841)
- Display the installed seed package versions in the final summary output, for example:

  ```
  created virtual environment CPython3.8.3.final.0-64 in 350ms
    creator CPython3Posix(dest=/x, clear=True, global=False)
    seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via
      added seed packages: pip==20.1.1, setuptools==47.3.1, wheel==0.34.2
  ```

  by @gaborbernat. (#1864)

Bugfixes - 20.0.24

- Do not generate/overwrite `.gitignore` if it already exists at destination path - by @gaborbernat. (#1862)
- Improve error message for no `.dist-info` inside the `app-data` copy seeder - by @gaborbernat. (#1867)

Improved Documentation - 20.0.24

- How seeding mechanisms discover (and automatically keep it up to date) wheels at Wheels - by @gaborbernat. (#1821)
- How distributions should handle shipping their own embedded wheels at Embed wheels for ıs - by @gaborbernat. (#1840)

v20.0.23 (2020-06-12)

Bugfixes - 20.0.23

- Fix typo in `setup.cfg` - by @RowdyHowell. (#1857)

v20.0.22 (2020-06-12)

Bugfixes - 20.0.22

- Relax `importlib.resources` requirement to also allow version 2 - by @asottile. (#1846)
- Upgrade embedded setuptools to `44.1.1` for python 2 and `47.1.1` for python3.5+ - by @gaborbernat. (#1855)

v20.0.21 (2020-05-20)

Features - 20.0.21

- Generate ignore file for version control systems to avoid tracking virtual environments by default. Users should remove these files if still want to track. For now we support only **git** by @gaborbernat. (#1806)

Bugfixes - 20.0.21

- Fix virtualenv fails sometimes when run concurrently, `--clear-app-data` conflicts with `clear` flag when abbreviation is turned on. To bypass this while allowing abbreviated flags on the command line we had to move it to `reset-app-data` - by @gaborbernat. (#1824)
- Upgrade embedded `setuptools` to `46.4.0` from `46.1.3` on Python `3.5+`, and `pip` from `20.1` to `20.1.1` - by @gaborbernat. (#1827)
- Seeder pip now correctly handles `--extra-search-dir` - by @frenzymadness. (#1834)

v20.0.20 (2020-05-04)

Bugfixes - 20.0.20

- Fix download fails with python 3.4 - by @gaborbernat. (#1809)
- Fixes older CPython2 versions use `_get_makefile_filename` instead of `get_makefile_filename` on `sysconfig` - by @ianw. (#1810)
- Fix download is `True` by default - by @gaborbernat. (#1813)
- Fail `app-data` seed operation when wheel download fails and better error message - by @gaborbernat. (#1814)

v20.0.19 (2020-05-03)

Bugfixes - 20.0.19

- Fix generating a Python 2 environment from Python 3 creates invalid python activator - by @gaborbernat. (#1776)
- Fix pinning seed packages via `app-data` seeder raised `Invalid Requirement` - by @gaborbernat. (#1779)
- Do not stop interpreter discovery if we fail to find the system interpreter for a executable during discovery - by @gaborbernat. (#1781)
- On CPython2 POSIX platforms ensure `syconfig.get_makefile_filename` exists within the virtual environment (this is used by some c-extension based libraries - e.g. numpy - for building) - by @gaborbernat. (#1783)
- Better handling of options `copies` and `symlinks`. Introduce priority of where the option is set to follow the order: CLI, env var, file, hardcoded. If both set at same level prefers copy over symlink. - by @gaborbernat. (#1784)
- Upgrade pip for Python `2.7` and `3.5+` from `20.0.2` to `20.1` - by @gaborbernat. (#1793)
- Fix CPython is not discovered from Windows registry, and discover pythons from Windows registry in decreasing order by version - by @gaborbernat. (#1796)
- Fix symlink detection for creators - by @asottile (#1803)

v20.0.18 (2020-04-16)

Bugfixes - 20.0.18

- Importing setuptools before cli_run could cause our python information query to fail due to setuptools patching `distutils.dist.Distribution` - by @gaborbernat. (#1771)

v20.0.17 (2020-04-09)

Features - 20.0.17

- Extend environment variables checked for configuration to also check aliases (e.g. setting either `VIRTUALENV_COPIES` or `VIRTUALENV_ALWAYS_COPY` will work) - by @gaborbernat. (#1763)

v20.0.16 (2020-04-04)

Bugfixes - 20.0.16

- Allow seed wheel files inside the `extra-search-dir` folders that do not have `Requires-Python` metadata specified, these are considered compatible with all python versions - by rnat. (#1757)

v20.0.15 (2020-03-27)

Features - 20.0.15

- Upgrade embedded setuptools to `46.1.3` from `46.1.1` - by @gaborbernat. (#1752)

v20.0.14 (2020-03-25)

Features - 20.0.14

- Remove `__PYVENV_LAUNCHER__` on macOs for Python `3.7.(<8)` and `3.8.(<3)` on interpreter startup via `pth` file, this pulls in the upstream patch - by @gaborbernat. (#1704)
- Upgrade embedded setuptools for Python `3.5+` to `46.1.1`, for Python `2.7` to `44.1.0` - by @gaborbernat. (#1745)

Bugfixes - 20.0.14

- Fix discovery of interpreter by name from `PATH` that does not match a spec format - by @gaborbernat. (#1746)

v20.0.13 (2020-03-19)

Bugfixes - 20.0.13

- Do not fail when the pyc files is missing for the host Python 2 - by @gaborbernat. (#1738)
- Support broken Packaging pythons that put the include headers under distutils pattern rather than sysconfig one - by @gaborbernat. (#1739)

v20.0.12 (2020-03-19)

Bugfixes - 20.0.12

- Fix relative path discovery of interpreters - by @gaborbernat. (#1734)

v20.0.11 (2020-03-18)

Features - 20.0.11

- Improve error message when the host python does not satisfy invariants needed to create virtual environments (now we print which host files are incompatible/missing and for which creators when no supported creator can be matched, however we found creators that can he given Python interpreter - will still print no supported creator for Jython, however ly what host files do not allow creation of virtual environments in case of CPython/PyPy) - by @gaborbernat. (#1716)

Bugfixes - 20.0.11

- Support Python 3 Framework distributed via XCode in macOs Catalina and before - by @gaborbernat. (#1663)
- Fix Windows Store Python support, do not allow creation via symlink as that's not going to work by design - by @gaborbernat. (#1709)
- Fix `activate_this.py` throws `AttributeError` on Windows when virtual environment was created via cross python mechanism - by @gaborbernat. (#1710)
- Fix `--no-pip`, `--no-setuptools`, `--no-wheel` not being respected - by @gaborbernat. (#1712)
- Allow missing `.py` files if a compiled `.pyc` version is available - by @tucked. (#1714)
- Do not fail if the distutils/setuptools patch happens on a C-extension loader (such as `zipimporter` on Python 3.7 or earlier) - by @gaborbernat. (#1715)
- Support Python 2 implementations that require the landmark files and `site.py` to be in platform standard library instead of the standard library path of the virtual environment (notably some RHEL ones, such as the Docker image `amazonlinux:1`) - by @gaborbernat. (#1719)
- Allow the test suite to pass even when called with the system Python - to help repackaging of the tool for Linux distributions - by @gaborbernat. (#1721)
- Also generate `pipx.y` console script beside `pip-x.y` to be compatible with how pip installs itself - by @gaborbernat. (#1723)
- Automatically create the application data folder if it does not exists - by @gaborbernat. (#1728)

Improved Documentation - 20.0.11

- supports details now explicitly what Python installations we support - by @gaborbernat. (#1714)

v20.0.10 (2020-03-10)

Bugfixes - 20.0.10

- Fix acquiring python information might be altered by distutils configuration files generating incorrect layout virtual environments - by @gaborbernat. (#1663)
- Upgrade embedded setuptools to `46.0.0` from `45.3.0` on Python `3.5+` - by @gaborbernat. (#1702)

Improved Documentation - 20.0.10

- Document requirements (pip + index server) when installing via pip under the installation section - by @gaborbernat. (#1618)
- Document installing from non PEP-518 systems - @gaborbernat. (#1619)
- Document installing latest unreleased version from Github - @gaborbernat. (#1620)

v20.0.9 (2020-03-08)

Bugfixes - 20.0.9

- `pythonw.exe` works as `python.exe` on Windows - by @gaborbernat. (#1686)
- Handle legacy loaders for virtualenv import hooks used to patch distutils configuration load - by @gaborbernat. (#1690)
- Support for python 2 platforms that store landmark files in `platstdlib` over `stdlib` (e.g. RHEL) - by @gaborbernat. (#1694)
- Upgrade embedded setuptools to `45.3.0` from `45.2.0` for Python `3.5+` - by @gaborbernat. (#1699)

v20.0.8 (2020-03-04)

Bugfixes - 20.0.8

- Having distutils configuration files that set `prefix` and `install_scripts` cause installation of packages in the wrong location - by @gaborbernat. (#1663)
- Fix `PYTHONPATH` being overridden on Python 2 — by @jd. (#1673)
- Fix list configuration value parsing from config file or environment variable - by @gaborbernat. (#1674)
- Fix Batch activation script shell prompt to display environment name by default - by @spetafree. (#1679)
- Fix startup on Python 2 is slower for virtualenv - this was due to setuptools calculating it's working set distribution - by @gaborbernat. (#1682)
- Fix entry points are not populated for editable installs on Python 2 due to setuptools working set being calculated before `easy_install.pth` runs - by @gaborbernat. (#1684)
- Fix `attr:` import fails for setuptools - by @gaborbernat. (#1685)

v20.0.7 (2020-02-26)

Bugfixes - 20.0.7

- Disable distutils fixup for python 3 until pypa/pip #7778 is fixed and released - by @gaborbernat. (#1669)

v20.0.6 (2020-02-26)

Bugfixes - 20.0.6

- Fix global site package always being added with bundled macOs python framework builds - by @gaborbernat. (#1561)
- Fix generated scripts use host version info rather than target - by @gaborbernat. (#1600)
- Fix circular prefix reference with single elements (accept these as if they were system executables, print a info about them referencing themselves) - by @gaborbernat. (#1632)
- Handle the case when the application data folder is read-only:
  - the application data folder is now controllable via `app-data`,
  - `clear-app-data` now cleans the entire application data folder, not just the `app-data` seeder path,
  - check if the application data path passed in does not exist or is read-only, and fallback to a temporary directory,
  - temporary directory application data is automatically cleaned up at the end of execution,
  - `symlink-app-data` is always `False` when the application data is temporary

  by @gaborbernat. (#1640)
- Fix PyPy 2 builtin modules are imported from standard library, rather than from builtin - by @gaborbernat. (#1652)
- Fix creation of entry points when path contains spaces - by @nsoranzo. (#1660)
- Fix relative paths for the zipapp (for python `3.7+`) - by @gaborbernat. (#1666)

v20.0.5 (2020-02-21)

Features - 20.0.5

- Also create `pythonX.X` executables when creating pypy virtualenvs - by @asottile (#1612)
- Fail with better error message if trying to install source with unsupported `setuptools`, allow `setuptools-scm >= 2` and move to legacy `setuptools-scm` format to support better older platforms (`CentOS 7` and such) - by @gaborbernat. (#1621)
- Report of the created virtual environment is now split across four short lines rather than one long - by @gaborbernat (#1641)

Bugfixes - 20.0.5

- Add macOs Python 2 Framework support (now we test it with the CI via brew) - by @gaborbernat (#1561)
- Fix losing of libpypy-c.so when the pypy executable is a symlink - by @asottile (#1614)
- Discover python interpreter in a case insensitive manner - by @PrajwalM2212 (#1624)
- Fix cross interpreter support when the host python sets `sys.base_executable` based on `__PYVENV_LAUNCHER__` - by @cjolowicz (#1643)

v20.0.4 (2020-02-14)

Features - 20.0.4

- When aliasing interpreters, use relative symlinks - by @asottile. (#1596)

Bugfixes - 20.0.4

- Allow the use of `/` as pathname component separator on Windows - by `vphilippon` (#1582)
- Lower minimal version of six required to 1.9 - by `ssbarnea` (#1606)

v20.0.3 (2020-02-12)

Bugfixes - 20.0.3

- On Python 2 with Apple Framework builds the global site package is no longer added when the `system-site-packages` is not specified - by @gaborbernat. (#1561)
- Fix system python discovery mechanism when prefixes contain relative parts (e.g. `..`) by resolving paths within the python information query - by @gaborbernat. (#1583)
- Expose a programmatic API as `from virtualenv import cli_run` - by @gaborbernat. (#1585)
- Fix `app-data` `seeder` injects a extra `.dist-info.virtualenv` path that breaks `importlib.metadata`, now we inject an extra `.virtualenv` - by @gaborbernat. (#1589)

Improved Documentation - 20.0.3

- Document a programmatic API as `from virtualenv import cli_run` under Programmatic API - by @gaborbernat. (#1585)

v20.0.2 (2020-02-11)

Features - 20.0.2

- Print out a one line message about the created virtual environment when no `verbose` is set, this can now be silenced to get back the original behavior via the `quiet` flag - by @pradyunsg. (#1557)
- Allow virtualenv's app data cache to be overridden by `VIRTUALENV_OVERRIDE_APP_DATA` - by @asottile. (#1559)
- Passing in the virtual environment name/path is now required (no longer defaults to `venv`) - by @gaborbernat. (#1568)
- Add a CLI flag `with-traceback` that allows displaying the stacktrace of the virtualenv when a failure occurs - by @gaborbernat. (#1572)

Bugfixes - 20.0.2

- Support long path names for generated virtual environment console entry points (such as `pip`) when using the `app-data` `seeder` - by @gaborbernat. (#997)
- Improve python discovery mechanism:
  - do not fail if there are executables that fail to query (e.g. for not having execute access to it) on the `PATH`,
  - beside the prefix folder also try with the platform dependent binary folder within that,

  by @gaborbernat. (#1545)
- When copying (either files or trees) do not copy the permission bits, last access time, last modification time, and flags as access to these might be forbidden (for example in case of the macOs Framework Python) and these are not needed for the user to use the virtual environment - by @gaborbernat. (#1561)
- While discovering a python executables interpreters that cannot be queried are now displayed with info level rather than warning, so now they're no longer shown by default (these can be just executables to which we don't have access or that are broken, don't warn if it's not the target Python we want) - by @gaborbernat. (#1574)
- The `app-data` `seeder` no longer symlinks the packages on UNIX and copies on Windows. Instead by default always copies, however now has the `symlink-app-data` flag allowing users to request this less robust but faster method - by @gaborbernat. (#1575)

Improved Documentation - 20.0.2

- Add link to the legacy documentation for the changelog by @jezdez. (#1547)
- Fine tune the documentation layout: default width of theme, allow tables to wrap around, soft corners for code snippets - by @pradyunsg. (#1548)

v20.0.1 (2020-02-10)

Features - 20.0.1

- upgrade embedded setuptools to `45.2.0` from `45.1.0` for Python `3.4+` - by @gaborbernat. (#1554)

Bugfixes - 20.0.1

- Virtual environments created via relative path on Windows creates bad console executables - by @gaborbernat. (#1552)
- Seems sometimes venvs created set their base executable to themselves; we accept these without question, so we handle virtual environments as system pythons causing issues - by @gaborbernat. (#1553)

v20.0.0. (2020-02-10)

Improved Documentation - 20.0.0.

- Fixes typos, repeated words and inconsistent heading spacing. Rephrase parts of the development documentation and CLI documentation. Expands shorthands like `env var` and `config` to their full forms. Uses descriptions from respective documentation, for projects listed in `related links` - by @pradyunsg. (#1540)

v20.0.0b2 (2020-02-04)

Features - 20.0.0b2

- Improve base executable discovery mechanism:
  - print at debug level why we refuse some candidates,
  - when no candidates match exactly, instead of hard failing fallback to the closest match where the priority of matching attributes is: python implementation, major version, minor version, architecture, patch version, release level and serial (this is to facilitate things to still work when the OS upgrade replace/upgrades the system python with a never version, than what the virtualenv host python was created with),
  - always resolve system_executable information during the interpreter discovery, and the discovered environment is the system interpreter instead of the venv/virtualenv (this happened before lazily the first time we accessed, and caused reporting that the created virtual environment is of type of the virtualenv host python version, instead of the system pythons version - these two can differ if the OS upgraded the system python underneath and the virtualenv host was created via copy),

  by @gaborbernat. (#1515)
- Generate `bash` and `fish` activators on Windows too (as these can be available with git bash, cygwin or mysys2) - by @gaborbernat. (#1527)
- Upgrade the bundled `wheel` package from `0.34.0` to `0.34.2` - by @gaborbernat. (#1531)

Bugfixes - 20.0.0b2

- Bash activation script should have no extensions instead of `.sh` (this fixes the virtualenvwrapper integration) - by @gaborbernat. (#1508)
- Show less information when we run with a single verbosity (`-v`):
  - no longer shows accepted interpreters information (as the last proposed one is always the accepted one),
  - do not display the `str_spec` attribute for `PythonSpec` as these can be deduced from the other attributes,
  - for the `app-data` seeder do not show the type of lock, only the path to the app data directory,

  By @gaborbernat. (#1510)
- Fixed cannot discover a python interpreter that has already been discovered under a different path (such is the case when we have multiple symlinks to the same interpreter) - by @gaborbernat. (#1512)
- Support relative paths for `-p` - by @gaborbernat. (#1514)
- Creating virtual environments in parallel fail with cannot acquire lock within app data - by @gaborbernat. (#1516)
- pth files were not processed under Debian CPython2 interpreters - by @gaborbernat. (#1517)
- Fix prompt not displayed correctly with upcoming fish 3.10 due to us not preserving `$pipestatus` - by @krobelus. (#1530)
- Stable order within `pyenv.cfg` and add `include-system-site-packages` only for creators that reference a global Python - by user:*gaborbernat*. (#1535)

Improved Documentation - 20.0.0b2

- Create the first iteration of the new documentation - by @gaborbernat. (#1465)
- Project readme is now of type MarkDown instead of reStructuredText - by @gaborbernat. (#1531)

v20.0.0b1 (2020-01-28)

- First public release of the rewrite. Everything is brand new and just added.
- `--download` defaults to `False`
- No longer replaces builtin `site` module with custom version baked within virtualenv code itself. A simple shim module is used to fix up things on Python 2 only.

> **Warning**
>
> The current virtualenv is the second iteration of implementation. From version `0.8` all the way to `16.7.9` we numbered the first iteration. Version `20.0.0b1` is a complete rewrite of the package, and as such this release history starts from there. The old changelog is still available in the legacy branch documentation.

virtualenv
==========

.. image:: https://img.shields.io/pypi/v/virtualenv?style=flat-square
  :target: https://pypi.org/project/virtualenv/#history
  :alt: Latest version on PyPI
.. image:: https://img.shields.io/pypi/implementation/virtualenv?style=flat-square
  :alt: PyPI - Implementation
.. image:: https://img.shields.io/pypi/pyversions/virtualenv?style=flat-square
  :alt: PyPI - Python Version
.. image:: https://readthedocs.org/projects/virtualenv/badge/?version=latest&style=flat-square
  :target: https://virtualenv.pypa.io
  :alt: Documentation status
.. image:: https://img.shields.io/discord/803025117553754132
  :target: https://discord.gg/pypa
  :alt: Discord
.. image:: https://img.shields.io/pypi/dm/virtualenv?style=flat-square
  :target: https://pypistats.org/packages/virtualenv
  :alt: PyPI - Downloads
.. image:: https://img.shields.io/pypi/l/virtualenv?style=flat-square
  :target: https://opensource.org/licenses/MIT
  :alt: PyPI - License
.. image:: https://img.shields.io/github/issues/pypa/virtualenv?style=flat-square
  :target: https://github.com/pypa/virtualenv/issues
  :alt: Open issues
.. image:: https://img.shields.io/github/issues-pr/pypa/virtualenv?style=flat-square
  :target: https://github.com/pypa/virtualenv/pulls
  :alt: Open pull requests
.. image:: https://img.shields.io/github/stars/pypa/virtualenv?style=flat-square
  :target: https://pypistats.org/packages/virtualenv
  :alt: Package popularity

``virtualenv`` is a tool to create isolated Python environments. Since Python ``3.3``, a subset of it has been
integrated into the standard library under the `venv module <https://docs.python.org/3/library/venv.html>`_. The
``venv`` module does not offer all features of this library, to name just a few more prominent:

- is slower (by not having the ``app-data`` seed method),
- is not as extendable,
- cannot create virtual environments for arbitrarily installed python versions (and automatically discover these),
- is not upgrade-able via `pip <https://pip.pypa.io/en/stable/installing/>`_,
- does not have as rich programmatic API (describe virtual environments without creating them).

The basic problem being addressed is one of dependencies and versions, and indirectly permissions.
Imagine you have an application that needs version ``1`` of ``LibFoo``, but another application requires version
``2``. How can you use both these libraries? If you install everything into your host python (e.g. ``python3.8``)
it's easy to end up in a situation where two packages have conflicting requirements.

Or more generally, what if you want to install an application *and leave it be*? If an application works, any change
in its libraries or the versions of those libraries can break the application. Also, what if you can't install packages
into the global ``site-packages`` directory, due to not having permissions to change the host python environment?

In all these cases, ``virtualenv`` can help you. It creates an environment that has its own installation directories,
that doesn't share libraries with other virtualenv environments (and optionally doesn't access the globally installed
libraries either).

Useful links

Related projects, that build abstractions on top of virtualenv

* :pypi:`virtualenvwrapper` - a useful set of scripts for creating and deleting virtual environments
* :pypi:`pew` - provides a set of commands to manage multiple virtual environments
* :pypi:`tox` - a generic virtualenv management and test automation command line tool, driven by a

``tox.ini``
  configuration file
* :pypi:`nox` - a tool that automates testing in multiple Python environments, similar to tox,
  driven by a ``noxfile.py`` configuration file

Tutorials

* `Corey Schafer tutorial <https://www.youtube.com/watch?v=N5vscPTWKOk>`_ on how to use it
* `Using virtualenv with mod_wsgi <http://code.google.com/p/modwsgi/wiki/VirtualEnvironments>`_

Presenting how the package works from within

* `Bernat Gabor: status quo of virtual environments <https://www.youtube.com/watch?v=o1Vue9CWRxU>`_
* `Carl Meyer: Reverse-engineering Ian Bicking's brain: inside pip and virtualenv
  <http://pyvideo.org/video/568/reverse-engineering-ian-bicking--39-s-brain--insi>`_

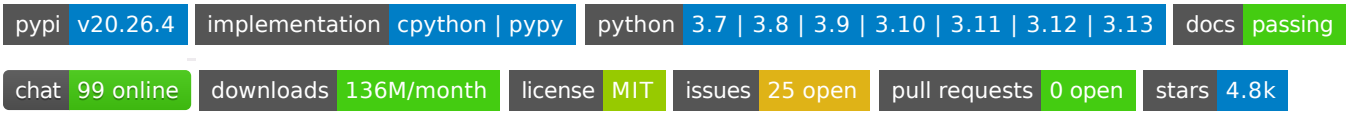
.. comment: split here

.. toctree::
   :hidden:

   installation
   user_guide
   cli_interface
   extend
   development
   changelog

> **Note**
>
> You are not reading the most recent version of this documentation. 20.26.4 is the latest version available.

virtualenv



`virtualenv` is a tool to create isolated Python environments. Since Python `3.3`, a subset of it has been integrated into the standard library under the venv module. The `venv` module does not offer all features of this library, to name just a few more prominent:

- is slower (by not having the `app-data` seed method),
- is not as extendable,
- cannot create virtual environments for arbitrarily installed python versions (and automatically discover these),
- is not upgrade-able via pip,
- does not have as rich programmatic API (describe virtual environments without creating them).

The basic problem being addressed is one of dependencies and versions, and indirectly permissions. Imagine you have an application that needs version `1` of `LibFoo`, but another application requires version `2`. How can you use both these libraries? If you install everything into your host python (e.g. `python3.8`) it's easy to end up in a situation where two packages have conflicting requirements.

Or more generally, what if you want to install an application *and leave it be*? If an application works, any change in its libraries or the versions of those libraries can break the application. Also, what if you can't install packages into the global `site-packages` directory, due to not having permissions to change the host python environment?

In all these cases, `virtualenv` can help you. It creates an environment that has its own installation directories, that doesn't share libraries with other virtualenv environments (and optionally doesn't access the globally installed libraries either).

Useful links

Related projects, that build abstractions on top of virtualenv

- virtualenvwrapper - a useful set of scripts for creating and deleting virtual environments
- pew - provides a set of commands to manage multiple virtual environments
- tox - a generic virtualenv management and test automation command line tool, driven by a configuration file

Skip to content

- l that automates testing in multiple Python environments, similar to tox, driven by a `noxfile.py` configuration file

Tutorials

- Corey Schafer tutorial on how to use it
- Using virtualenv with mod_wsgi

Presenting how the package works from within

- Bernat Gabor: status quo of virtual environments
- Carl Meyer: Reverse-engineering Ian Bicking's brain: inside pip and virtualenv

---

Made with Sphinx and @pradyunsg's Furo
Last updated on 2024-06-22T01:57:47.408024+00:00

Installation
============

via pipx

:pypi:`virtualenv` is a CLI tool that needs a Python interpreter to run. If you already have a ``Python 3.7+``
interpreter the best is to use :pypi:`pipx` to install virtualenv into an isolated environment. This has the added
benefit that later you'll be able to upgrade virtualenv without affecting other parts of the system.

.. code-block:: console

    pipx install virtualenv
    virtualenv --help

via pip

Alternatively you can install it within the global Python interpreter itself (perhaps as a user package via the
``--user`` flag). Be cautious if you are using a python install that is managed by your operating system or
another package manager. ``pip`` might not coordinate with those tools, and may leave your system in an
inconsistent state. Note, if you go down this path you need to ensure pip is new enough per the subsections below:

.. code-block:: console

    python -m pip install --user virtualenv
    python -m virtualenv --help

wheel
~~~~~
Installing virtualenv via a wheel (default with pip) requires an installer that can understand the ``python-requires``
tag (see `PEP-503 <https://www.python.org/dev/peps/pep-0503/>`_), with pip this is version ``9.0.0`` (released 2016
November). Furthermore, in case you're not installing it via the PyPi you need to be using a mirror that correctly
forwards the ``python-requires`` tag (notably the OpenStack mirrors don't do this, or older `devpi <https://github.com/devpi/devpi>`_ versions - added with version ``4.7.0``).

.. _sdist:

sdist
~~~~~
When installing via a source distribution you need an installer that handles the `PEP-517 <https://www.python.org/dev/peps/pep-0517/>`_ specification. In case of ``pip`` this is version ``18.0.0`` or
later (released on 2018 July). If you cannot upgrade your pip to support this you need to ensure that the build
requirements from `pyproject.toml <https://github.com/pypa/virtualenv/blob/main/pyproject.toml#L2>`_ are satisfied
before triggering the install.

via zipapp

You can use virtualenv without installing it too. We publish a Python `zipapp <https://docs.python.org/3/library/zipapp.html>`_, you can just download this from `https://bootstrap.pypa.io/virtualenv.pyz <https://bootstrap.pypa.io/virtualenv.pyz>`_ and invoke this package
with a python interpreter:

.. code-block:: console

    python virtualenv.pyz --help

The root level zipapp is always the current latest release. To get the last supported zipapp against a given python

minor release use the link ``https://bootstrap.pypa.io/virtualenv/x.y/virtualenv.pyz``, e.g. for the last virtualenv
supporting Python 3.11 use
`https://bootstrap.pypa.io/virtualenv/3.11/virtualenv.pyz
<https://bootstrap.pypa.io/virtualenv/3.11/virtualenv.pyz>`_.

If you are looking for past version of virtualenv.pyz they are available here:

.. code-block:: console

    https://github.com/pypa/get-virtualenv/blob/<virtualenv version>/public/<python version>/virtualenv.pyz?raw=true

latest unreleased

Installing an unreleased version is discouraged and should be only done for testing purposes. If you do so you'll need
a pip version of at least ``18.0.0`` and use the following command:

.. code-block:: console

    pip install git+https://github.com/pypa/virtualenv.git@main

.. _compatibility-requirements:

Python and OS Compatibility
----

virtualenv works with the following Python interpreter implementations:

- `CPython <https://www.python.org/>`_: ``3.12 >= python_version >= 3.7``
- `PyPy <https://pypy.org/>`_: ``3.10 >= python_version >= 3.7``

This means virtualenv works on the latest patch version of each of these minor versions. Previous patch versions are
supported on a best effort approach.

CPython is shipped in multiple forms, and each OS repackages it, often applying some customization along the way.
Therefore we cannot say universally that we support all platforms, but rather specify some we test against. In case
of ones not specified here the support is unknown, though likely will work. If you find some cases please open a feature
request on our issue tracker.

Note:

- as of ``20.18.0`` -- ``2023-02-06`` -- we no longer support running under Python ``<=3.6``,
- as of ``20.22.0`` -- ``2023-04-19`` -- we no longer support creating environments for Python ``<=3.6``.

Linux
~~~~~
- installations from `python.org <https://www.python.org/downloads/>`_
- Ubuntu 16.04+ (both upstream and `deadsnakes
<https://launchpad.net/~deadsnakes/+archive/ubuntu/ppa>`_ builds)
- Fedora
- RHEL and CentOS
- OpenSuse
- Arch Linux

macOS
~~~~~
In case of macOS we support:

- installations from `python.org <https://www.python.org/downloads/>`_,
- python versions installed via `brew <https://docs.brew.sh/Homebrew-and-Python>`_,
- Python 3 part of XCode (Python framework - ``/Library/Frameworks/Python3.framework/``).

Windows
~~~~~~~
- Installations from `python.org <https://www.python.org/downloads/>`_
~~~~~~~

- Windows Store Python - note only `version 3.7+ <https://www.microsoft.com/en-us/p/python-38/9mssztt1n39l>`_

User Guide
==========

Quick start

Create the environment (creates a folder in your current directory)
    .. code-block:: console

        virtualenv env_name
In Linux or Mac, activate the new python environment
    .. code-block:: console

        source env_name/bin/activate
Or in Windows
    .. code-block:: console

        .\env_name\Scripts\activate
Confirm that the env is successfully selected
    .. code-block:: console

        which python3

Introduction

Virtualenv has one basic command:

.. code-block:: console

    virtualenv venv

This will create a python virtual environment of the same version as virtualenv, installed into the subdirectory
``venv``. The command line tool has quite a few of flags that modify the tool's behavior, for a full list make sure to check out :ref:`cli_flags`.

The tool works in two phases:

- **Phase 1** discovers a python interpreter to create a virtual environment from (by default this is the same python
  as the one ``virtualenv`` is running from, however we can change this via the :option:`p` option).
- **Phase 2** creates a virtual environment at the specified destination (:option:`dest`), this can be broken down into
  four further sub-steps:

  - create a python that matches the target python interpreter from phase 1,
  - install (bootstrap) seed packages (one or more of :pypi:`pip`, :pypi:`setuptools`, :pypi:`wheel`) in the created
    virtual environment,
  - install activation scripts into the binary directory of the virtual environment (these will allow end users to
    activate the virtual environment from various shells).
  - create files that mark the virtual environment as to be ignored by version control systems (currently we support
    Git only, as Mercurial, Bazaar or SVN do not support ignore files in subdirectories). This step can be skipped
    with the :option:`no-vcs-ignore` option.

The python in your new virtualenv is effectively isolated from the python that was used to create it.

Python discovery

The first thing we need to be able to create a virtual environment is a python interpreter. This will describe to the
tool what type of virtual environment you would like to create, think of it as: version, architecture, implementation.

``virtualenv`` being a python application has always at least one such available, the one ``virtualenv`` itself is

using, and as such this is the default discovered element. This means that if you install ``virtualenv`` under
python ``3.8``, virtualenv will by default create virtual environments that are also of version ``3.8``.

Created python virtual environments are usually not self-contained. A complete python packaging is usually made up of
thousands of files, so it's not efficient to install the entire python again into a new folder. Instead virtual
environments are mere shells, that contain little within themselves, and borrow most from the system python (this is what
you installed, when you installed python itself). This does mean that if you upgrade your system python your virtual
environments *might* break, so watch out. The upside of this, referring to the system python, is that creating virtual
environments can be fast.

Here we'll describe the built-in mechanism (note this can be extended though by plugins). The CLI flag :option:`p` or
:option:`python` allows you to specify a python specifier for what type of virtual environment you would like, the
format is either:

- a relative/absolute path to a Python interpreter,

- a specifier identifying the Python implementation, version, architecture in the following format:

    .. code-block::

       {python implementation name}{version}{architecture}

    We have the following restrictions:

    - the python implementation is all alphabetic characters (``python`` means any implementation, and if is missing it
      defaults to ``python``),
    - the version is a dot separated version number,
    - the architecture is either ``-64`` or ``-32`` (missing means ``any``).

    For example:

    - ``python3.8.1`` means any python implementation having the version ``3.8.1``,
    - ``3`` means any python implementation having the major version ``3``,
    - ``cpython3`` means a ``CPython`` implementation having the version ``3``,
    - ``pypy2`` means a python interpreter with the ``PyPy`` implementation and major version ``2``.

  Given the specifier ``virtualenv`` will apply the following strategy to discover/find the system executable:

   - If we're on Windows look into the Windows registry, and check if we see any registered Python implementations that
     match the specification. This is in line with expectation laid out inside
     `PEP-514 <https://www.python.org/dev/peps/pep-0514/>`_
   - Try to discover a matching python executable within the folders enumerated on the ``PATH`` environment variable.
     In this case we'll try to find an executable that has a name roughly similar to the specification (for exact logic,
     please see the implementation code).

.. warning::

   As detailed above, virtual environments usually just borrow things from the system Python, they don't actually contain
   all the data from the system Python. The version of the python executable is hardcoded within the python exe itself.
   Therefore, if you upgrade your system Python, your virtual environment will still report the version before the
   upgrade, even though now other than the executable all additional content (standard library, binary libs, etc) are
   of the new version.

   Barring any major incompatibilities (rarely the case) the virtual environment will continue working, but other than

the content embedded within the python executable it will behave like the upgraded version. If such a virtual
environment python is specified as the target python interpreter, we will create virtual environments that match the
new system Python version, not the version reported by the virtual environment.

Creators

These are what actually setup the virtual environment, usually as a reference against the system python. virtualenv
at the moment has two types of virtual environments:

- ``venv`` - this delegates the creation process towards the ``venv`` module, as described in
  `PEP 405 <https://www.python.org/dev/peps/pep-0405>`_. This is only available on Python interpreters having version
  ``3.5`` or later, and also has the downside that virtualenv **must** create a process to invoke that module (unless
  virtualenv is installed in the system python), which can be an expensive operation (especially true on Windows).

- ``builtin`` - this means ``virtualenv`` is able to do the creation operation itself (by knowing exactly what files to
  create and what system files need to be referenced). The creator with name ``builtin`` is an alias on the first
  creator that's of this type (we provide creators for various target environments, that all differ in actual create
  operations, such as CPython 2 on Windows, PyPy2 on Windows, CPython3 on Posix, PyPy3 on Posix, and so on; for a full
  list see :option:`creator`).

Seeders

These will install for you some seed packages (one or more of: :pypi:`pip`, :pypi:`setuptools`, :pypi:`wheel`) that
enables you to install additional python packages into the created virtual environment (by invoking pip). Installing
:pypi:`setuptools` and :pypi:`wheel` is disabled by default on Python 3.12+ environments. There are two
main seed mechanisms available:

- ``pip`` - this method uses the bundled pip with virtualenv to install the seed packages (note, a new child process
  needs to be created to do this, which can be expensive especially on Windows).
- ``app-data`` - this method uses the user application data directory to create install images. These images are needed
  to be created only once, and subsequent virtual environments can just link/copy those images into their pure python
  library path (the ``site-packages`` folder). This allows all but the first virtual environment creation to be blazing
  fast (a ``pip`` mechanism takes usually 98% of the virtualenv creation time, so by creating this install image that
  we can just link into the virtual environments install directory we can achieve speedups of shaving the initial
  1 minute and 10 seconds down to just 8 seconds in case of a copy, or ``0.8`` seconds in case symlinks are available -
  this is on Windows, Linux/macOS with symlinks this can be as low as ``100ms`` from 3+ seconds).
  To override the filesystem location of the seed cache, one can use the
  ``VIRTUALENV_OVERRIDE_APP_DATA`` environment variable.

.. _wheels:

Wheels
~~~~~~

To install a seed package via either ``pip`` or ``app-data`` method virtualenv needs to acquire a wheel of the target
package. These wheels may be acquired from multiple locations as follows:

- ``virtualenv`` ships out of box with a set of embed ``wheels`` for all three seed packages (:pypi:`pip`,
  :pypi:`setuptools`, :pypi:`wheel`). These are packaged together with the virtualenv source files, and only change upon
  upgrading virtualenv. Different Python versions require different versions of these, and because
~~~~~~

virtualenv supports a
  wide range of Python versions, the number of embedded wheels out of box is greater than 3. Whenever newer versions of
  these embedded packages are released upstream ``virtualenv`` project upgrades them, and does a new release. Therefore,
  upgrading virtualenv periodically will also upgrade the version of the seed packages.
- However, end users might not be able to upgrade virtualenv at the same speed as we do new releases. Therefore, a user
  might request to upgrade the list of embedded wheels by invoking virtualenv with the :option:`upgrade-embed-wheels`
  flag. If the operation is triggered in such a manual way subsequent runs of virtualenv will always use the upgraded
  embed wheels.

  The operation can trigger automatically too, as a background process upon invocation of virtualenv, if no such upgrade
  has been performed in the last 14 days. It will only start using automatically upgraded wheel if they have been
  released for more than 28 days, and the automatic upgrade finished at least an hour ago:

  - the 28 days period should guarantee end users are not pulling in automatically releases that have known bugs within,
  - the one hour period after the automatic upgrade finished is implemented so that continuous integration services do
    not start using a new embedded versions half way through.

  The automatic behavior might be disabled via the :option:`no-periodic-update` configuration flag/option. To acquire
  the release date of a package virtualenv will perform the following:

  - lookup ``https://pypi.org/pypi/<distribution>/json`` (primary truth source),
  - save the date the version was first discovered, and wait until 28 days passed.
- Users can specify a set of local paths containing additional wheels by using the :option:`extra-search-dir` command
  line argument flag.

When searching for a wheel to use virtualenv performs lookup in the following order:

- embedded wheels,
- upgraded embedded wheels,
- extra search dir.

Bundled wheels are all three above together. If neither of the locations contain the requested wheel version or
:option:`download` option is set will use ``pip`` download to load the latest version available from the index server.

.. _distribution_wheels:

Embed wheels for distributions
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Custom distributions often want to use their own set of wheel versions to distribute instead of the one virtualenv
releases on PyPi. The reason for this is trying to keep the system versions of those packages in sync with what
virtualenv uses. In such cases they should patch the module `virtualenv.seed.wheels.embed
<https://github.com/pypa/virtualenv/tree/main/src/virtualenv/seed/wheels/embed>`_, making sure to provide the function
``get_embed_wheel`` (which returns the wheel to use given a distribution/python version). The ``BUNDLE_FOLDER``,
``BUNDLE_SUPPORT`` and ``MAX`` variables are needed if they want to use virtualenv's test suite to validate.

Furthermore, they might want to disable the periodic update by patching the
`virtualenv.seed.embed.base_embed.PERIODIC_UPDATE_ON_BY_DEFAULT
<https://github.com/pypa/virtualenv/tree/main/src/virtualenv/seed/embed/base_embed.py>`_
to ``False``, and letting the system update mechanism to handle this. Note in this case the user might still request an
upgrade of the embedded wheels by invoking virtualenv via :option:`upgrade-embed-wheels`, but no longer happens
automatically, and will not alter the OS provided wheels.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Activators

These are activation scripts that will mangle with your shell's settings to ensure that commands from within the python
virtual environment take priority over your system paths. For example, if invoking ``pip`` from your shell returned the
system python's pip before activation, once you do the activation this should refer to the virtual environments ``pip``.
Note, though that all we do is change priority; so, if your virtual environments ``bin``/``Scripts`` folder does not
contain some executable, this will still resolve to the same executable it would have resolved before the activation.

For a list of shells we provide activators see :option:`activators`. The location of these is right alongside the Python
executables: usually ``Scripts`` folder on Windows, ``bin`` on POSIX. They are called ``activate``, plus an
extension that's specific per activator, with no extension for Bash. You can invoke them, usually by source-ing them.
The source command might vary by shell - e.g. on Bash it’s ``source`` (or ``.``):

.. code-block:: console

   source venv/bin/activate

The activate script prepends the virtual environment’s binary folder onto the ``PATH`` environment variable. It’s
really just convenience for doing so, since you could do the same yourself.

Note that you don't have to activate a virtual environment to use it. You can instead use the full paths to its
executables, rather than relying on your shell to resolve them to your virtual environment.

Activator scripts also modify your shell prompt to indicate which environment is currently active, by prepending the
environment name (or the name specified by ``--prompt`` when initially creating the environment) in brackets, like
``(venv)``. You can disable this behavior by setting the environment variable ``VIRTUAL_ENV_DISABLE_PROMPT`` to any
value. You can also get the environment name via the environment variable ``VIRTUAL_ENV_PROMPT`` if you want to
customize your prompt, for example.

The scripts also provision a ``deactivate`` command that will allow you to undo the operation:

.. code-block:: console

   deactivate

.. note::

    If using Powershell, the ``activate`` script is subject to the
    `execution policies <http://technet.microsoft.com/en-us/library/dd347641.aspx>`_ on the system. By default, Windows
    7 and later, the system's execution policy is set to ``Restricted``, meaning no scripts like the ``activate`` script
    are allowed to be executed.

    However, that can't stop us from changing that slightly to allow it to be executed. You may relax the system
    execution policy to allow running of local scripts without verifying the code signature using the following:

    .. code-block:: powershell

       Set-ExecutionPolicy RemoteSigned

    Since the ``activate.ps1`` script is generated locally for each virtualenv, it is not considered a remote script and
    can then be executed.

A longer explanation of this can be found within Allison Kaptur's 2013 blog post: `There's no magic: virtualenv
edition <https://www.recurse.com/blog/14-there-is-no-magic-virtualenv-edition>`_ explains how

virtualenv uses bash and
Python and ``PATH`` and ``PYTHONHOME`` to isolate virtual environments' paths.

.. _programmatic_api:

Programmatic API

At the moment ``virtualenv`` offers only CLI level interface. If you want to trigger invocation of Python environments
from within Python you should be using the ``virtualenv.cli_run`` method; this takes an ``args`` argument where you can
pass the options the same way you would from the command line. The run will return a session object containing data
about the created virtual environment.

.. code-block:: python

   from virtualenv import cli_run

   cli_run(["venv"])

.. automodule:: virtualenv
   :members:

.. currentmodule:: virtualenv.run.session

.. autoclass:: Session
    :members:

CLI interface
==========

.. _cli_flags:

CLI flags
~~~~~~~~~

``virtualenv`` is primarily a command line application.

It modifies the environment variables in a shell to create an isolated Python environment, so you'll need to have a
shell to run it. You can type in ``virtualenv`` (name of the application) followed by flags that control its
behavior. All options have sensible defaults, and there's one required argument: the name/path of the virtual
environment to create. The default values for the command line options can be overridden via the :ref:`conf_file` or :ref:`env_vars`. Environment variables takes priority over the configuration file values
(``--help`` will show if a default comes from the environment variable as the help message will end in this case
with environment variables or the configuration file).

The options that can be passed to virtualenv, along with their default values and a short description are listed below.

:command:`virtualenv [OPTIONS]`

.. table_cli::
   :module: virtualenv.run
   :func: build_parser_only

Defaults
~~~~~~~~

.. _conf_file:

Configuration file
^^^^^^^^^^^^^^^^^^

Unless ``VIRTUALENV_CONFIG_FILE`` is set, virtualenv looks for a standard ``virtualenv.ini`` configuration file.
The exact location depends on the operating system you're using, as determined by :pypi:`platformdirs` application
configuration definition. It can be overridden by setting the ``VIRTUALENV_CONFIG_FILE`` environment variable.
The configuration file location is printed as at the end of the output when ``--help`` is passed.

The keys of the settings are derived from the command line option (left strip the ``-`` characters, and replace ``-``
with ``_``). Where multiple flags are available first found wins (where order is as it shows up under the ``--help``).

For example, :option:`--python <python>` would be specified as:

.. code-block:: ini

  [virtualenv]
  python = /opt/python-3.8/bin/python

Options that take multiple values, like :option:`extra-search-dir` can be specified as:

.. code-block:: ini

  [virtualenv]
  extra_search_dir =
      /path/to/dists
      /path/to/other/dists

.. _env_vars:

Environment Variables
^^^^^^^^^^^^^^^^^^^^^

Default values may be also specified via environment variables. The keys of the settings are derived from the
command line option (left strip the ``-`` characters, and replace ``-`` with ``_``, finally capitalize the name). Where
multiple flags are available first found wins (where order is as it shows up under the ``--help``).

For example, to use a custom Python binary, instead of the one virtualenv is run with, you can set the environment
variable ``VIRTUALENV_PYTHON`` like:

.. code-block:: console

   env VIRTUALENV_PYTHON=/opt/python-3.8/bin/python virtualenv

Where the option accepts multiple values, for example for :option:`python` or
:option:`extra-search-dir`, the values can be separated either by literal
newlines or commas. Newlines and commas can not be mixed and if both are
present only the newline is used for separating values. Examples for multiple
values:

.. code-block:: console

   env VIRTUALENV_PYTHON=/opt/python-3.8/bin/python,python3.8 virtualenv
   env VIRTUALENV_EXTRA_SEARCH_DIR=/path/to/dists\n/path/to/other/dists virtualenv

The equivalent CLI-flags based invocation for the above examples would be:

.. code-block:: console

   virtualenv --python=/opt/python-3.8/bin/python --python=python3.8
   virtualenv --extra-search-dir=/path/to/dists --extra-search-dir=/path/to/other/dists

Extend functionality
====================

``virtualenv`` allows one to extend the builtin functionality via a plugin system. To add a plugin you need to:

- write a python file containing the plugin code which follows our expected interface,
- package it as a python library,
- install it alongside the virtual environment.

Python discovery

The python discovery mechanism is a component that needs to answer the following answer: based on some type of user
input give me a Python interpreter on the machine that matches that. The builtin interpreter tries to discover
an installed Python interpreter (based on PEP-515 and ``PATH`` discovery) on the users machine where the user input is a
python specification. An alternative such discovery mechanism for example would be to use the popular
`pyenv <https://github.com/pyenv/pyenv>`_ project to discover, and if not present install the requested Python
interpreter. Python discovery mechanisms must be registered under key ``virtualenv.discovery``, and the plugin must
implement :class:`virtualenv.discovery.discover.Discover`:

.. code-block:: ini

   virtualenv.discovery =
        pyenv = virtualenv_pyenv.discovery:PyEnvDiscovery

.. currentmodule:: virtualenv.discovery.discover

.. autoclass:: Discover
    :undoc-members:
    :members:

Creators

Creators are what actually perform the creation of a virtual environment. The builtin virtual environment creators
all achieve this by referencing a global install; but would be just as valid for a creator to install a brand new
entire python under the target path; or one could add additional creators that can create virtual environments for other
python implementations, such as IronPython. They must be registered under and entry point with key
``virtualenv.create`` , and the class must implement :class:`virtualenv.create.creator.Creator`:

.. code-block:: ini

   virtualenv.create =
        cpython3-posix = virtualenv.create.via_global_ref.builtin.cpython.cpython3:CPython3Posix

.. currentmodule:: virtualenv.create.creator

.. autoclass:: Creator
    :undoc-members:
    :members:
    :exclude-members: run, set_pyenv_cfg, debug_script, debug_script, validate_dest, debug

Seed mechanism

Seeders are what given a virtual environment will install somehow some seed packages into it. They must be registered
under and entry point with key ``virtualenv.seed`` , and the class must implement
:class:`virtualenv.seed.seeder.Seeder`:

.. code-block:: ini

```
   virtualenv.seed =
        db = virtualenv.seed.fromDb:InstallFromDb

.. currentmodule:: virtualenv.seed.seeder

.. autoclass:: Seeder
    :undoc-members:
    :members:

Activation scripts
------------------
If you want add an activator for a new shell you can do this by implementing a new activator. They
must be registered
under an entry point with key ``virtualenv.activate`` , and the class must implement
:class:`virtualenv.activation.activator.Activator`:

.. code-block:: ini

   virtualenv.activate =
        bash = virtualenv.activation.bash:BashActivator

.. currentmodule:: virtualenv.activation.activator

.. autoclass:: Activator
    :undoc-members:
    :members:
```

Development
===========

Getting started

``virtualenv`` is a volunteer maintained open source project and we welcome contributions of all forms. The sections
below will help you get started with development, testing, and documentation. We’re pleased that you are interested in
working on virtualenv. This document is meant to get you setup to work on virtualenv and to act as a guide and reference
to the development setup. If you face any issues during this process, please
`open an issue <https://github.com/pypa/virtualenv/issues/new?
title=Trouble+with+development+environment>`_ about it on
the issue tracker.

Setup
~~~~~

virtualenv is a command line application written in Python. To work on it, you'll need:

- **Source code**: available on `GitHub <https://github.com/pypa/virtualenv>`_. You can use ``git`` to clone the
    repository:

  .. code-block:: console

      git clone https://github.com/pypa/virtualenv
      cd virtualenv

- **Python interpreter**: We recommend using ``CPython``. You can use
  `this guide <https://realpython.com/installing-python/>`_ to set it up.

- :pypi:`tox`: to automatically get the projects development dependencies and run the test suite. We recommend
  installing it using `pipx <https://pipxproject.github.io/pipx/>`_.

Running from source tree
~~~~~~~~~~~~~~~~~~~~~~~~

The easiest way to do this is to generate the development tox environment, and then invoke virtualenv from under the
``.tox/dev`` folder

.. code-block:: console

    tox -e dev
    .tox/dev/bin/virtualenv # on Linux
    .tox/dev/Scripts/virtualenv # on Windows

Running tests
~~~~~~~~~~~~~

virtualenv's tests are written using the :pypi:`pytest` test framework. :pypi:`tox` is used to automate the setup
and execution of virtualenv's tests.

To run tests locally execute:

.. code-block:: console

    tox -e py

This will run the test suite for the same Python version as under which ``tox`` is installed. Alternatively you can
specify a specific version of python by using the ``pyNN`` format, such as: ``py38``, ``pypy3``, etc.

``tox`` has been configured to forward any additional arguments it is given to ``pytest``. This enables the use of pytest's
`rich CLI <https://docs.pytest.org/en/latest/usage.html#specifying-tests-selecting-tests>`_. As an example, you can
~~~~~~~~~~~~~

select tests using the various ways that pytest provides:

.. code-block:: console

    # Using markers
    tox -e py -- -m "not slow"
    # Using keywords
    tox -e py -- -k "test_extra"

Some tests require additional dependencies to be run, such is the various shell activators (``bash``, ``fish``,
``powershell``, etc). These tests will automatically be skipped if these are not present, note however that in CI
all tests are run; so even if all tests succeed locally for you, they may still fail in the CI.

Running linters
~~~~~~~~~~~~~~~

virtualenv uses :pypi:`pre-commit` for managing linting of the codebase. ``pre-commit`` performs various checks on all
files in virtualenv and uses tools that help follow a consistent code style within the codebase. To use linters locally,
run:

.. code-block:: console

    tox -e fix

.. note::

    Avoid using ``# noqa`` comments to suppress linter warnings - wherever possible, warnings should be fixed instead.
    ``# noqa`` comments are reserved for rare cases where the recommended style causes severe readability problems.

Building documentation
~~~~~~~~~~~~~~~~~~~~~~

virtualenv's documentation is built using :pypi:`Sphinx`. The documentation is written in reStructuredText. To build it
locally, run:

.. code-block:: console

    tox -e docs

The built documentation can be found in the ``.tox/docs_out`` folder and may be viewed by opening ``index.html`` within
that folder.

Release
~~~~~~~

virtualenv's release schedule is tied to ``pip``, ``setuptools`` and ``wheel``. We bundle the latest version of these
libraries so each time there's a new version of any of these, there will be a new virtualenv release shortly afterwards
(we usually wait just a few days to avoid pulling in any broken releases).

Contributing

Submitting pull requests
~~~~~~~~~~~~~~~~~~~~~~~~

Submit pull requests against the ``main`` branch, providing a good description of what you're doing and why. You must
have legal permission to distribute any code you contribute to virtualenv and it must be available under the MIT
License. Provide tests that cover your changes and run the tests locally first. virtualenv
:ref:`supports <compatibility-requirements>` multiple Python versions and operating systems. Any pull request must
consider and work on all these platforms.

Pull Requests should be small to facilitate review. Keep them self-contained, and limited in scope.
`Studies have shown
<https://www.kessler.de/prd/smartbear/BestPracticesForPeerCodeReview.pdf>`_ that review quality
falls off as patch size
grows. Sometimes this will result in many small PRs to land a single large feature. In particular,
pull requests must
not be treated as "feature branches", with ongoing development work happening within the PR.
Instead, the feature should
be broken up into smaller, independent parts which can be reviewed and merged individually.

Additionally, avoid including "cosmetic" changes to code that is unrelated to your change, as these
make reviewing the
PR more difficult. Examples include re-flowing text in comments or documentation, or addition or
removal of blank lines
or whitespace within lines. Such changes can be made separately, as a "formatting cleanup" PR, if
needed.

Automated testing
~~~~~~~~~~~~~~~~~

All pull requests and merges to 'main' branch are tested using
`Azure Pipelines <https://azure.microsoft.com/en-gb/services/devops/pipelines/>`_ (configured by
``azure-pipelines.yml`` file at the root of the repository). You can find the status and results to
the CI runs for your
PR on GitHub's Web UI for the pull request. You can also find links to the CI services' pages for
the specific builds in
the form of "Details" links, in case the CI run fails and you wish to view the output.

To trigger CI to run again for a pull request, you can close and open the pull request or submit
another change to the
pull request. If needed, project maintainers can manually trigger a restart of a job/build.

NEWS entries
~~~~~~~~~~~~

The ``changelog.rst`` file is managed using :pypi:`towncrier` and all non trivial changes must be
accompanied by a news
entry. To add an entry to the news file, first you need to have created an issue describing the
change you want to
make. A Pull Request itself *may* function as such, but it is preferred to have a dedicated issue
(for example, in case
the PR ends up rejected due to code quality reasons).

Once you have an issue or pull request, you take the number and you create a file inside of the
``docs/changelog``
directory named after that issue number with an extension of:

- ``feature.rst``,
- ``bugfix.rst``,
- ``doc.rst``,
- ``removal.rst``,
- ``misc.rst``.

Thus if your issue or PR number is ``1234`` and this change is fixing a bug, then you would create
a file
``docs/changelog/1234.bugfix.rst``. PRs can span multiple categories by creating multiple files
(for instance, if you
added a feature and deprecated/removed the old feature at the same time, you would create
``docs/changelog/1234.bugfix.rst`` and ``docs/changelog/1234.remove.rst``). Likewise if a PR
touches multiple issues/PRs
you may create a file for each of them with the same contents and :pypi:`towncrier` will
deduplicate them.

Contents of a NEWS entry
^^^^^^^^^^^^^^^^^^^^^^^^

The contents of this file are reStructuredText formatted text that will be used as the content of
the news file entry.
You do not need to reference the issue or PR numbers here as towncrier will automatically add a
reference to all of
the affected issues when rendering the news file.

In order to maintain a consistent style in the ``changelog.rst`` file, it is preferred to keep the
news entry to the

point, in sentence case, shorter than 120 characters and in an imperative tone -- an entry should complete the sentence
``This change will …``. In rare cases, where one line is not enough, use a summary line in an imperative tone followed
by a blank line separating it from a description of the feature/change in one or more paragraphs, each wrapped
at 120 characters. Remember that a news entry is meant for end users and should only contain details relevant to an end
user.

Choosing the type of NEWS entry
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

A trivial change is anything that does not warrant an entry in the news file. Some examples are: code refactors that
don't change anything as far as the public is concerned, typo fixes, white space modification, etc. To mark a PR
as trivial a contributor simply needs to add a randomly named, empty file to the ``news/`` directory with the extension
of ``.trivial``.

Becoming a maintainer
~~~~~~~~~~~~~~~~~~~~~

If you want to become an official maintainer, start by helping out. As a first step, we welcome you to triage issues on
virtualenv's issue tracker. virtualenv maintainers provide triage abilities to contributors once they have been around
for some time and contributed positively to the project. This is optional and highly recommended for becoming a
virtualenv maintainer. Later, when you think you're ready, get in touch with one of the maintainers and they will
initiate a vote among the existing maintainers.

.. note::

    Upon becoming a maintainer, a person should be given access to various virtualenv-related tooling across
    multiple platforms. These are noted here for future reference by the maintainers:

    - GitHub Push Access
    - PyPI Publishing Access
    - CI Administration capabilities
    - ReadTheDocs Administration capabilities
~~~~~~~~~~~~~~~~~~~~~

Release History
===============

.. include:: _draft.rst

.. towncrier release notes start

v20.26.3 (2024-06-21)

Bugfixes - 20.26.3
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * setuptools to ``70.1.0`` from ``69.5.1``
  * pip to ``24.1``from ``24.0`` (:issue:`2741`)

v20.26.2 (2024-05-13)

Bugfixes - 20.26.2
~~~~~~~~~~~~~~~~~~
- ``virtualenv.pyz`` no longer fails when zipapp path contains a symlink - by :user:`HandSonic` and
:user:`petamas`. (:issue:`1949`)
- Fix bad return code from activate.sh if hashing is disabled - by :user:'fenkes-ibm'.
(:issue:`2717`)

v20.26.1 (2024-04-29)

Bugfixes - 20.26.1
~~~~~~~~~~~~~~~~~~
- fix PATH-based Python discovery on Windows - by :user:`ofek`. (:issue:`2712`)

v20.26.0 (2024-04-23)

Bugfixes - 20.26.0
~~~~~~~~~~~~~~~~~~
- allow builtin discovery to discover specific interpreters (e.g. ``python3.12``) given an
unspecific spec (e.g. ``python3``) - by :user:`flying-sheep`. (:issue:`2709`)

v20.25.3 (2024-04-17)

Bugfixes - 20.25.3
~~~~~~~~~~~~~~~~~~
- Python 3.13.0a6 renamed pathmod to parser. (:issue:`2702`)

v20.25.2 (2024-04-16)

Bugfixes - 20.25.2
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  - setuptools of ``69.1.0`` to ``69.5.1``
  - wheel of ``0.42.0`` to ``0.43.0`` (:issue:`2699`)

v20.25.1 (2024-02-21)

Bugfixes - 20.25.1
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * setuptools to ``69.0.3`` from ``69.0.2``
  * pip to ``23.3.2`` from ``23.3.1`` (:issue:`2681`)
- Upgrade embedded wheels:

  - pip ``23.3.2`` to ``24.0``,
  - setuptools ``69.0.3`` to ``69.1.0``. (:issue:`2691`)

Misc - 20.25.1
~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~
- :issue:`2688`

v20.25.0 (2023-12-01)

Features - 20.25.0
~~~~~~~~~~~~~~~~~~
- The tests now pass on the CI with Python 3.13.0a2 - by :user:`hroncok`. (:issue:`2673`)

Bugfixes - 20.25.0
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * wheel to ``0.41.3`` from ``0.41.2`` (:issue:`2665`)
- Upgrade embedded wheels:

  * wheel to ``0.42.0`` from ``0.41.3``
  * setuptools to ``69.0.2`` from ``68.2.2`` (:issue:`2669`)

v20.24.6 (2023-10-23)

Bugfixes - 20.24.6
~~~~~~~~~~~~~~~~~~
- Use get_hookimpls method instead of the private attribute in tests. (:issue:`2649`)
- Upgrade embedded wheels:

  * setuptools to ``68.2.2`` from ``68.2.0``
  * pip to ``23.3.1`` from ``23.2.1`` (:issue:`2656`)

v20.24.5 (2023-09-08)

Bugfixes - 20.24.5
~~~~~~~~~~~~~~~~~~
- Declare PyPy 3.10 support - by :user:`cclauss`. (:issue:`2638`)
- Brew on macOS no longer allows copy builds - disallow choosing this by :user:`gaborbernat`.
(:issue:`2640`)
- Upgrade embedded wheels:

  * setuptools to ``68.2.0`` from ``68.1.2`` (:issue:`2642`)

v20.24.4 (2023-08-30)

Bugfixes - 20.24.4
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * setuptools to ``68.1.2`` from ``68.1.0`` on ``3.8+``
  * wheel to ``0.41.2`` from ``0.41.1`` on ``3.7+`` (:issue:`2628`)

v20.24.3 (2023-08-11)

Bugfixes - 20.24.3
~~~~~~~~~~~~~~~~~~
- Fixed ResourceWarning on exit caused by periodic update subprocess (:issue:`2472`)
- Upgrade embedded wheels:

  * wheel to ``0.41.1`` from ``0.41.0`` (:issue:`2622`)

Misc - 20.24.3
~~~~~~~~~~~~~~
- :issue:`2610`

v20.24.2 (2023-07-24)

Bugfixes - 20.24.2
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * pip to ``23.2.1`` from ``23.2``
  * wheel to ``0.41.0`` from ``0.40.0`` (:issue:`2614`)

v20.24.1 (2023-07-19)

Bugfixes - 20.24.1
~~~~~~~~~~~~~~~~~~
- Upgrade embedded wheels:

  * pip to ``23.2`` from ``23.1.2`` - by :user:`arielkirkwood` (:issue:`2611`)

v20.24.0 (2023-07-14)

Features - 20.24.0
~~~~~~~~~~~~~~~~~~
- Export the prompt prefix as ``VIRTUAL_ENV_PROMPT`` when activating a virtual
  environment - by :user:`jimporter`. (:issue:`2194`)

Bugfixes - 20.24.0
~~~~~~~~~~~~~~~~~~
- Fix test suite - by :user:`gaborbernat`. (:issue:`2592`)
- Upgrade embedded wheels:

  * setuptools to ``68.0.0`` from ``67.8.0`` (:issue:`2607`)

v20.23.1 (2023-06-16)

Bugfixes - 20.23.1
~~~~~~~~~~~~~~~~~~
- update and simplify nushell activation script, fixes an issue on Windows resulting in consecutive
command not found - by :user:`melMass`. (:issue:`2572`)
- Upgrade embedded wheels:

  * setuptools to ``67.8.0`` from ``67.7.2`` (:issue:`2588`)

v20.23.0 (2023-04-27)

Features - 20.23.0
~~~~~~~~~~~~~~~~~~
- Do not install ``wheel`` and ``setuptools`` seed packages for Python 3.12+. To restore the old
behavior use:

  - for ``wheel`` use ``VIRTUALENV_WHEEL=bundle`` environment variable or ``--wheel=bundle`` CLI
flag,
  - for ``setuptools`` use ``VIRTUALENV_SETUPTOOLS=bundle`` environment variable or ``--
setuptools=bundle`` CLI flag.

  By :user:`chrysle`. (:issue:`2487`)
- 3.12 support - by :user:`gaborbernat`. (:issue:`2558`)

Bugfixes - 20.23.0
~~~~~~~~~~~~~~~~~~
- Prevent ``PermissionError`` when using venv creator on systems that deliver files without user
write
  permission - by :user:`kulikjak`. (:issue:`2543`)
- Upgrade setuptools to ``67.7.2`` from ``67.6.1`` and pip to ``23.1.2`` from ``23.1`` - by
:user:`szleb`. (:issue:`2560`)

v20.22.0 (2023-04-19)

~~~~~~~~~~~~~~~~~~

Features - 20.22.0
~~~~~~~~~~~~~~~~~~
- Drop support for creating Python <=3.6 (including 2) interpreters. Removed pip of ``20.3.4``, ``21.3.1``; wheel of
  ``0.37.1``; setuptools of ``59.6.0``, ``44.1.1``, ``50.3.2``- by :user:`gaborbernat`. (:issue:`2548`)

v20.21.1 (2023-04-19)

Bugfixes - 20.21.1
~~~~~~~~~~~~~~~~~~
- Add ``tox.ini`` to sdist - by :user:`mtelka`. (:issue:`2511`)
- Move the use of 'let' in nushell to ensure compatibility with future releases of nushell, where 'let' no longer
  assumes that its initializer is a full expressions. (:issue:`2527`)
- The nushell command 'str collect' has been superseded by the 'str join' command. The activate.nu script has
  been updated to reflect this change. (:issue:`2532`)
- Upgrade embedded wheels:

  * wheel to ``0.40.0`` from ``0.38.4``
  * setuptools to ``67.6.1`` from ``67.4.0``
  * pip to ``23.1`` from ``23.0.1`` (:issue:`2546`)

v20.21.0 (2023-03-12)

Features - 20.21.0
~~~~~~~~~~~~~~~~~~
- Make closure syntax explicitly starts with {||. (:issue:`2512`)

Bugfixes - 20.21.0
~~~~~~~~~~~~~~~~~~
- Add ``print`` command to nushell print_prompt to ensure compatibility with future release of nushell,
  where intermediate commands no longer print their result to stdout. (:issue:`2514`)
- Do not assume the default encoding. (:issue:`2515`)
- Make ``ReentrantFileLock`` thread-safe and,
  thereby, fix race condition in ``virtualenv.cli_run`` - by :user:`radoering`. (:issue:`2516`)

v20.20.0 (2023-02-28)

Features - 20.20.0
~~~~~~~~~~~~~~~~~~
- Change environment variable existence check in Nushell activation script to not use deprecated command. (:issue:`2506`)

Bugfixes - 20.20.0
~~~~~~~~~~~~~~~~~~
- Discover CPython implementations distributed on Windows by any organization - by :user:`faph`. (:issue:`2504`)
- Upgrade embedded setuptools to ``67.4.0`` from ``67.1.0`` and pip to ``23.0.1`` from ``23.0`` - by :user:`gaborbernat`. (:issue:`2510`)

v20.19.0 (2023-02-07)

Features - 20.19.0
~~~~~~~~~~~~~~~~~~
- Allow platformdirs version ``3`` - by :user:`cdce8p`. (:issue:`2499`)

v20.18.0 (2023-02-06)

Features - 20.18.0
~~~~~~~~~~~~~~~~~~
- Drop ``3.6`` runtime support (can still create ``2.7+``) - by :user:`gaborbernat`.

(:issue:`2489`)

Bugfixes - 20.18.0
~~~~~~~~~~~~~~~~~~
- Fix broken prompt in Nushell when activating virtual environment - by :user:`kubouc`.
(:issue:`2481`)
- Bump embedded pip to ``23.0`` and setuptools to ``67.1`` - by :user:`gaborbernat`.
(:issue:`2489`)

v20.17.1 (2022-12-05)

Bugfixes - 20.17.1
~~~~~~~~~~~~~~~~~~
- A ``py`` or ``python`` spec means any Python rather than ``CPython`` - by :user:`gaborbernat`.
(`#2460 <https://github.com/pypa/virtualenv/issues/2460>`_)
- Make ``activate.nu`` respect ``VIRTUAL_ENV_DISABLE_PROMPT`` and not set the prompt if requested -
by :user:`m-lima`. (`#2461 <https://github.com/pypa/virtualenv/issues/2461>`_)

v20.17.0 (2022-11-27)

Features - 20.17.0
~~~~~~~~~~~~~~~~~~
- Change Nushell activation script to be a module meant to be activated as an overlay. (`#2422
<https://github.com/pypa/virtualenv/issues/2422>`_)
- Update operator used in Nushell activation script to be compatible with future versions. (`#2450
<https://github.com/pypa/virtualenv/issues/2450>`_)

Bugfixes - 20.17.0
~~~~~~~~~~~~~~~~~~
- Do not use deprecated API from ``importlib.resources`` on Python 3.10 or later - by
:user:`gaborbernat`. (`#2448 <https://github.com/pypa/virtualenv/issues/2448>`_)
- Upgrade embedded setuptools to ``65.6.3`` from ``65.5.1`` - by :user:`gaborbernat`. (`#2451
<https://github.com/pypa/virtualenv/issues/2451>`_)

v20.16.7 (2022-11-12)

Bugfixes - 20.16.7
~~~~~~~~~~~~~~~~~~
- Use parent directory of python executable for pyvenv.cfg "home" value per PEP 405 - by
:user:`vfazio`. (`#2440 <https://github.com/pypa/virtualenv/issues/2440>`_)
- In POSIX virtual environments, try alternate binary names if ``sys._base_executable`` does not
exist - by :user:`vfazio`. (`#2442 <https://github.com/pypa/virtualenv/issues/2442>`_)
- Upgrade embedded wheel to ``0.38.4`` and pip to ``22.3.1`` from ``22.3`` and setuptools to
``65.5.1`` from
  ``65.5.0`` - by :user:`gaborbernat`. (`#2443 <https://github.com/pypa/virtualenv/issues/2443>`_)

v20.16.6 (2022-10-25)

Features - 20.16.6
~~~~~~~~~~~~~~~~~~
- Drop unneeded shims for PyPy3 directory structure (`#2426
<https://github.com/pypa/virtualenv/issues/2426>`_)

Bugfixes - 20.16.6
~~~~~~~~~~~~~~~~~~
- Fix selected scheme on debian derivatives for python 3.10 when ``python3-distutils`` is not
installed or the ``venv`` scheme is not available - by :user:`asottile`. (`#2350
<https://github.com/pypa/virtualenv/issues/2350>`_)
- Allow the test suite to pass even with the original C shell (rather than ``tcsh``) - by
:user:`kulikjak`. (`#2418 <https://github.com/pypa/virtualenv/issues/2418>`_)
- Fix fallback handling of downloading wheels for bundled packages - by :user:`schaap`. (`#2429
<https://github.com/pypa/virtualenv/issues/2429>`_)
- Upgrade embedded setuptools to ``65.5.0`` from ``65.3.0`` and pip to ``22.3`` from ``22.2.2`` -
by :user:`gaborbernat`. (`#2434 <https://github.com/pypa/virtualenv/issues/2434>`_)
~~~~~~~~~~~~~~~~~~

v20.16.5 (2022-09-07)

Bugfixes - 20.16.5
~~~~~~~~~~~~~~~~~~
- Do not turn echo off for subsequent commands in batch activators
  (``activate.bat`` and ``deactivate.bat``) - by :user:`pawelszramowski`. (`#2411
<https://github.com/pypa/virtualenv/issues/2411>`_)

v20.16.4 (2022-08-29)

Bugfixes - 20.16.4
~~~~~~~~~~~~~~~~~~
- Bump embed setuptools to ``65.3`` - by :user:`gaborbernat`. (`#2405
<https://github.com/pypa/virtualenv/issues/2405>`_)

v20.16.3 (2022-08-04)

Bugfixes - 20.16.3
~~~~~~~~~~~~~~~~~~
- Upgrade embedded pip to ``22.2.2`` from ``22.2.1`` and setuptools to ``63.4.1`` from ``63.2.0`` -
by :user:`gaborbernat`. (`#2395 <https://github.com/pypa/virtualenv/issues/2395>`_)

v20.16.2 (2022-07-27)

Bugfixes - 20.16.2
~~~~~~~~~~~~~~~~~~
- Bump embedded pip from ``22.2`` to ``22.2.1`` - by :user:`gaborbernat`. (`#2391
<https://github.com/pypa/virtualenv/issues/2391>`_)

v20.16.1 (2022-07-26)

Features - 20.16.1
~~~~~~~~~~~~~~~~~~
- Update Nushell activation scripts to version 0.67 - by :user:`kubouch`. (`#2386
<https://github.com/pypa/virtualenv/issues/2386>`_)

v20.16.0 (2022-07-25)

Features - 20.16.0
~~~~~~~~~~~~~~~~~~
- Drop support for running under Python 2 (still can generate Python 2 environments) - by
:user:`gaborbernat`. (`#2382 <https://github.com/pypa/virtualenv/issues/2382>`_)
- Upgrade embedded pip to ``22.2`` from ``22.1.2`` and setuptools to ``63.2.0`` from ``62.6.0`` -
  by :user:`gaborbernat`. (`#2383 <https://github.com/pypa/virtualenv/issues/2383>`_)

v20.15.1 (2022-06-28)

Bugfixes - 20.15.1
~~~~~~~~~~~~~~~~~~
- Fix the incorrect operation when ``setuptools`` plugins output something into ``stdout``. (`#2335
<https://github.com/pypa/virtualenv/issues/2335>`_)
- CPython3Windows creator ignores missing ``DLLs`` dir. (`#2368
<https://github.com/pypa/virtualenv/issues/2368>`_)

v20.15.0 (2022-06-25)

Features - 20.15.0
~~~~~~~~~~~~~~~~~~
- Support for Windows embeddable Python package: includes ``python<VERSION>.zip`` in the creator

sources
  - by :user:`reksarka`. (`#1774 <https://github.com/pypa/virtualenv/issues/1774>`_)

Bugfixes - 20.15.0
~~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``62.3.3`` from ``62.6.0`` and pip to ``22.1.2`` from ``22.0.4``
  - by :user:`gaborbernat`. (`#2348 <https://github.com/pypa/virtualenv/issues/2348>`_)
- Use ``shlex.quote`` instead of deprecated ``pipes.quote`` in Python 3 - by :user:`frenzymadness`.
(`#2351 <https://github.com/pypa/virtualenv/issues/2351>`_)
- Fix Windows PyPy 3.6 - by :user:`reksarka`. (`#2363
<https://github.com/pypa/virtualenv/issues/2363>`_)

v20.14.1 (2022-04-11)

Features - 20.14.1
~~~~~~~~~~~~~~~~~~
- Support for creating a virtual environment from a Python 2.7 framework on macOS 12 - by
:user:`nickhutchinson`. (`#2284 <https://github.com/pypa/virtualenv/issues/2284>`_)

Bugfixes - 20.14.1
~~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``62.1.0`` from ``61.0.0`` - by :user:`gaborbernat`. (`#2327
<https://github.com/pypa/virtualenv/issues/2327>`_)

v20.14.0 (2022-03-25)

Features - 20.14.0
~~~~~~~~~~~~~~~~~~
- Support Nushell activation scripts with nu version ``0.60`` - by :user:`kubouch`. (`#2321
<https://github.com/pypa/virtualenv/issues/2321>`_)

Bugfixes - 20.14.0
~~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``61.0.0`` from ``60.10.0`` - by :user:`gaborbernat`. (`#2322
<https://github.com/pypa/virtualenv/issues/2322>`_)

v20.13.4 (2022-03-18)

Bugfixes - 20.13.4
~~~~~~~~~~~~~~~~~~
- Improve performance of python startup inside created virtualenvs - by :user:`asottile`. (`#2317
<https://github.com/pypa/virtualenv/issues/2317>`_)
- Upgrade embedded setuptools to ``60.10.0`` from ``60.9.3`` - by :user:`gaborbernat`. (`#2320
<https://github.com/pypa/virtualenv/issues/2320>`_)

v20.13.3 (2022-03-07)

Bugfixes - 20.13.3
~~~~~~~~~~~~~~~~~~
- Avoid symlinking the contents of ``/usr`` into PyPy3.8+ virtualenvs - by :user:`stefanor`.
(`#2310 <https://github.com/pypa/virtualenv/issues/2310>`_)
- Bump embed pip from ``22.0.3`` to ``22.0.4`` - by :user:`gaborbernat`. (`#2311
<https://github.com/pypa/virtualenv/issues/2311>`_)

v20.13.2 (2022-02-24)

Bugfixes - 20.13.2
~~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``60.9.3`` from ``60.6.0`` - by :user:`gaborbernat`. (`#2306
<https://github.com/pypa/virtualenv/issues/2306>`_)

v20.13.1 (2022-02-05)

Bugfixes - 20.13.1

~~~~~~~~~~~~~~~~~~
- fix "execv() arg 2 must contain only strings" error on M1 MacOS (`#2282
<https://github.com/pypa/virtualenv/issues/2282>`_)
- Upgrade embedded setuptools to ``60.5.0`` from ``60.2.0`` - by :user:`asottile`. (`#2289
<https://github.com/pypa/virtualenv/issues/2289>`_)
- Upgrade embedded pip to ``22.0.3`` and setuptools to ``60.6.0`` - by :user:`gaborbernat` and
:user:`asottile`. (`#2294 <https://github.com/pypa/virtualenv/issues/2294>`_)

v20.13.0 (2022-01-02)

Features - 20.13.0
~~~~~~~~~~~~~~~~~~
- Add downloaded wheel information in the relevant JSON embed file to
  prevent additional downloads of the same wheel. - by :user:`mayeut`. (`#2268
<https://github.com/pypa/virtualenv/issues/2268>`_)

Bugfixes - 20.13.0
~~~~~~~~~~~~~~~~~~
- Fix ``AttributeError: 'bool' object has no attribute 'error'`` when creating a
  Python 2.x virtualenv on macOS - by ``moreati``. (`#2269
<https://github.com/pypa/virtualenv/issues/2269>`_)
- Fix ``PermissionError: [Errno 1] Operation not permitted`` when creating a
  Python 2.x virtualenv on macOS/arm64 - by ``moreati``. (`#2271
<https://github.com/pypa/virtualenv/issues/2271>`_)

v20.12.1 (2022-01-01)

Bugfixes - 20.12.1
~~~~~~~~~~~~~~~~~~
- Try using previous updates of ``pip``, ``setuptools`` & ``wheel``
  when inside an update grace period rather than always falling back
  to embedded wheels - by :user:`mayeut`. (`#2265
<https://github.com/pypa/virtualenv/issues/2265>`_)
- New patch versions of ``pip``, ``setuptools`` & ``wheel`` are now
  returned in the expected timeframe. - by :user:`mayeut`. (`#2266
<https://github.com/pypa/virtualenv/issues/2266>`_)
- Manual upgrades of ``pip``, ``setuptools`` & ``wheel`` are
  not discarded by a periodic update - by :user:`mayeut`. (`#2267
<https://github.com/pypa/virtualenv/issues/2267>`_)

v20.12.0 (2021-12-31)

Features - 20.12.0
~~~~~~~~~~~~~~~~~~
- Sign the python2 exe on Darwin arm64 - by :user:`tmspicer`. (`#2233
<https://github.com/pypa/virtualenv/issues/2233>`_)

Bugfixes - 20.12.0
~~~~~~~~~~~~~~~~~~
- Fix ``--download`` option - by :user:`mayeut`. (`#2120
<https://github.com/pypa/virtualenv/issues/2120>`_)
- Upgrade embedded setuptools to ``60.2.0`` from ``60.1.1`` - by :user:`gaborbernat`. (`#2263
<https://github.com/pypa/virtualenv/issues/2263>`_)

v20.11.2 (2021-12-29)

Bugfixes - 20.11.2
~~~~~~~~~~~~~~~~~~
- Fix installation of pinned versions of ``pip``, ``setuptools`` & ``wheel`` - by :user:`mayeut`.
(`#2203 <https://github.com/pypa/virtualenv/issues/2203>`_)

v20.11.1 (2021-12-29)

Bugfixes - 20.11.1
~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~
- Bump embed setuptools to ``60.1.1`` from ``60.1.0`` - by :user:`gaborbernat`. (`#2258 <https://github.com/pypa/virtualenv/issues/2258>`_)

v20.11.0 (2021-12-28)

Features - 20.11.0
~~~~~~~~~~~~~~~~~~
- Avoid deprecation warning from py-filelock argument - by :user:`ofek`. (`#2237 <https://github.com/pypa/virtualenv/issues/2237>`_)
- Upgrade embedded setuptools to ``61.1.0`` from ``58.3.0`` - by :user:`gaborbernat`. (`#2240 <https://github.com/pypa/virtualenv/issues/2240>`_)
- Drop the runtime dependency of ``backports.entry-points-selectable`` - by :user:`hroncok`. (`#2246 <https://github.com/pypa/virtualenv/issues/2246>`_)
- Fish: PATH variables should not be quoted when being set - by :user:`d3dave`. (`#2248 <https://github.com/pypa/virtualenv/issues/2248>`_)

v20.10.0 (2021-11-01)

Features - 20.10.0
~~~~~~~~~~~~~~~~~~
- If a ``"venv"`` install scheme exists in ``sysconfig``, virtualenv now uses it to create new virtual environments.
  This allows Python distributors, such as Fedora, to patch/replace the default install scheme without affecting
  the paths in new virtual environments.
  A similar technique `was proposed to Python, for the venv module <https://bugs.python.org/issue45413>`_ - by ``hroncok`` (`#2208 <https://github.com/pypa/virtualenv/issues/2208>`_)
- The activated virtualenv prompt is now always wrapped in parentheses. This
  affects venvs created with the ``--prompt`` attribute, and matches virtualenv's
  behavior on par with venv. (`#2224 <https://github.com/pypa/virtualenv/issues/2224>`_)

Bugfixes - 20.10.0
~~~~~~~~~~~~~~~~~~
- Fix broken prompt set up by activate.bat - by :user:`SiggyBar`. (`#2225 <https://github.com/pypa/virtualenv/issues/2225>`_)

v20.9.0 (2021-10-23)

Features - 20.9.0
~~~~~~~~~~~~~~~~~
- Special-case ``--prompt .`` to the name of the current directory - by :user:`rkm`. (`#2220 <https://github.com/pypa/virtualenv/issues/2220>`_)
- Add libffi-8.dll to pypy windows `#2218 <https://github.com/pypa/virtualenv/issues/2218>`_ - by :user:`mattip`

Bugfixes - 20.9.0
~~~~~~~~~~~~~~~~~
- Fixed path collision that could lead to a PermissionError or writing to system
  directories when using PyPy3.8 - by :user:`mgorny`. (`#2182 <https://github.com/pypa/virtualenv/issues/2182>`_)
- Upgrade embedded setuptools to ``58.3.0`` from ``58.1.0`` and pip to ``21.3.1`` from ``21.2.4`` - by
  :user:`gaborbernat`. (`#2205 <https://github.com/pypa/virtualenv/issues/2205>`_)
- Remove stray closing parenthesis in activate.bat - by :user:`SiggyBar`. (`#2221 <https://github.com/pypa/virtualenv/issues/2221>`_)

v20.8.1 (2021-09-24)

Bugfixes - 20.8.1
~~~~~~~~~~~~~~~~~
- Fixed a bug where while creating a venv on top of an existing one, without cleaning, when seeded
  wheel version mismatch occurred, multiple ``.dist-info`` directories may be present, confounding entrypoint
  discovery - by :user:`arcivanov` (`#2185 <https://github.com/pypa/virtualenv/issues/2185>`_)
~~~~~~~~~~~~~~~~~

- Bump embed setuptools from ``58.0.4`` to ``58.1.0`` - by :user:`gaborbernat`. (`#2195 <https://github.com/pypa/virtualenv/issues/2195>`_)

Misc - 20.8.1
~~~~~~~~~~~~~
- `#2189 <https://github.com/pypa/virtualenv/issues/2189>`_

v20.8.0 (2021-09-16)
----

* upgrade embedded setuptools to ``58.0.4`` from ``57.4.0`` and pip to ``21.2.4`` from ``21.2.3``
* Add nushell activation script

v20.7.2 (2021-08-10)

Bugfixes - 20.7.2
~~~~~~~~~~~~~~~~~
- Upgrade embedded pip to ``21.2.3`` from ``21.2.2`` and wheel to ``0.37.0`` from ``0.36.2`` - by :user:`gaborbernat`. (`#2168 <https://github.com/pypa/virtualenv/issues/2168>`_)

v20.7.1 (2021-08-09)

Bugfixes - 20.7.1
~~~~~~~~~~~~~~~~~
- Fix unpacking dictionary items in PythonInfo.install_path (`#2165 <https://github.com/pypa/virtualenv/issues/2165>`_)

v20.7.0 (2021-07-31)

Bugfixes - 20.7.0
~~~~~~~~~~~~~~~~~
- upgrade embedded pip to ``21.2.2`` from ``21.1.3`` and setuptools to ``57.4.0`` from ``57.1.0`` - by :user:`gaborbernat` (`#2159 <https://github.com/pypa/virtualenv/issues/2159>`_)

Deprecations and Removals - 20.7.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Removed ``xonsh`` activator due to this breaking fairly often the CI and lack of support from those packages
  maintainers, upstream is encouraged to continue supporting the project as a
  `plugin <https://github.com/xonsh/xonsh/issues/3689>`_ - by :user:`gaborbernat`. (`#2160 <https://github.com/pypa/virtualenv/issues/2160>`_)

v20.6.0 (2021-07-14)

Features - 20.6.0
~~~~~~~~~~~~~~~~~
- Support Python interpreters without ``distutils`` (fallback to ``syconfig`` in these cases) - by :user:`gaborbernat`. (`#1910 <https://github.com/pypa/virtualenv/issues/1910>`_)

v20.5.0 (2021-07-13)

Features - 20.5.0
~~~~~~~~~~~~~~~~~
- Plugins now use 'selectable' entry points - by :user:`jaraco`. (`#2093 <https://github.com/pypa/virtualenv/issues/2093>`_)
- add libffi-7.dll to the hard-coded list of dlls for PyPy (`#2141 <https://github.com/pypa/virtualenv/issues/2141>`_)
- Use the better maintained ``platformdirs`` instead of ``appdirs`` - by :user:`gaborbernat`. (`#2142 <https://github.com/pypa/virtualenv/issues/2142>`_)

Bugfixes - 20.5.0
~~~~~~~~~~~~~~~~~
- Bump pip the embedded pip ``21.1.3`` and setuptools to ``57.1.0`` - by :user:`gaborbernat`. (`#2135 <https://github.com/pypa/virtualenv/issues/2135>`_)

Deprecations and Removals - 20.5.0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Drop python ``3.4`` support as it has been over 2 years since EOL - by :user:`gaborbernat`.
(`#2141 <https://github.com/pypa/virtualenv/issues/2141>`_)

v20.4.7 (2021-05-24)

Bugfixes - 20.4.7
~~~~~~~~~~~~~~~~~
- Upgrade embedded pip to ``21.1.2`` and setuptools to ``57.0.0`` - by :user:`gaborbernat`. (`#2123
<https://github.com/pypa/virtualenv/issues/2123>`_)

v20.4.6 (2021-05-05)

Bugfixes - 20.4.6
~~~~~~~~~~~~~~~~~
- Fix ``site.getsitepackages()`` broken on python2 on debian - by :user:`freundTech`. (`#2105
<https://github.com/pypa/virtualenv/issues/2105>`_)

v20.4.5 (2021-05-05)

Bugfixes - 20.4.5
~~~~~~~~~~~~~~~~~
- Bump pip to ``21.1.1`` from ``21.0.1`` - by :user:`gaborbernat`. (`#2104
<https://github.com/pypa/virtualenv/issues/2104>`_)
- Fix ``site.getsitepackages()`` ignoring ``--system-site-packages`` on python2 - by
:user:`freundTech`. (`#2106 <https://github.com/pypa/virtualenv/issues/2106>`_)

v20.4.4 (2021-04-20)

Bugfixes - 20.4.4
~~~~~~~~~~~~~~~~~
- Built in discovery class is always preferred over plugin supplied classes. (`#2087
<https://github.com/pypa/virtualenv/issues/2087>`_)
- Upgrade embedded setuptools to ``56.0.0`` by :user:`gaborbernat`. (`#2094
<https://github.com/pypa/virtualenv/issues/2094>`_)

v20.4.3 (2021-03-16)

Bugfixes - 20.4.3
~~~~~~~~~~~~~~~~~
- Bump embedded setuptools from ``52.0.0`` to ``54.1.2`` - by :user:`gaborbernat` (`#2069
<https://github.com/pypa/virtualenv/issues/2069>`_)
- Fix PyPy3 stdlib on Windows is incorrect - by :user:`gaborbernat`. (`#2071
<https://github.com/pypa/virtualenv/issues/2071>`_)

v20.4.2 (2021-02-01)

Bugfixes - 20.4.2
~~~~~~~~~~~~~~~~~
- Running virtualenv ``--upgrade-embed-wheels`` crashes - by :user:`gaborbernat`. (`#2058
<https://github.com/pypa/virtualenv/issues/2058>`_)

v20.4.1 (2021-01-31)

Bugfixes - 20.4.1
~~~~~~~~~~~~~~~~~
- Bump embedded pip and setuptools packages to latest upstream supported (``21.0.1`` and
``52.0.0``) - by :user:`gaborbernat`. (`#2060 <https://github.com/pypa/virtualenv/issues/2060>`_)

v20.4.0 (2021-01-19)

Features - 20.4.0
~~~~~~~~~~~~~~~~~
- On the programmatic API allow passing in the environment variable dictionary to use, defaults to ``os.environ`` if not
  specified - by :user:`gaborbernat`. (`#2054 <https://github.com/pypa/virtualenv/issues/2054>`_)

Bugfixes - 20.4.0
~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``51.3.3`` from ``51.1.2`` - by :user:`gaborbernat`. (`#2055 <https://github.com/pypa/virtualenv/issues/2055>`_)

v20.3.1 (2021-01-13)

Bugfixes - 20.3.1
~~~~~~~~~~~~~~~~~
- Bump embed pip to ``20.3.3``, setuptools to ``51.1.1`` and wheel to ``0.36.2`` - by :user:`gaborbernat`. (`#2036 <https://github.com/pypa/virtualenv/issues/2036>`_)
- Allow unfunctioning of pydoc to fail freely so that virtualenvs can be
  activated under Zsh with set -e (since otherwise ``unset -f`` and
  ``unfunction`` exit with 1 if the function does not exist in Zsh) - by
  :user:`d125q`. (`#2049 <https://github.com/pypa/virtualenv/issues/2049>`_)
- Drop cached python information if the system executable is no longer present (for example when the executable is a
  shim and the mapped executable is replaced - such is the case with pyenv) - by :user:`gaborbernat`. (`#2050 <https://github.com/pypa/virtualenv/issues/2050>`_)

v20.3.0 (2021-01-10)

Features - 20.3.0
~~~~~~~~~~~~~~~~~
- The builtin discovery takes now a ``--try-first-with`` argument and is first attempted as valid interpreters. One can
  use this to force discovery of a given python executable when the discovery order/mechanism raises errors -
  by :user:`gaborbernat`. (`#2046 <https://github.com/pypa/virtualenv/issues/2046>`_)

Bugfixes - 20.3.0
~~~~~~~~~~~~~~~~~
- On Windows python ``3.7+`` distributions where the exe shim is missing fallback to the old ways - by :user:`gaborbernat`. (`#1986 <https://github.com/pypa/virtualenv/issues/1986>`_)
- When discovering interpreters on Windows, via the PEP-514, prefer ``PythonCore`` releases over other ones. virtualenv
  is used via pip mostly by this distribution, so prefer it over other such as conda - by :user:`gaborbernat`. (`#2046 <https://github.com/pypa/virtualenv/issues/2046>`_)

v20.2.2 (2020-12-07)

Bugfixes - 20.2.2
~~~~~~~~~~~~~~~~~
- Bump pip to ``20.3.1``, setuptools to ``51.0.0`` and wheel to ``0.36.1`` - by :user:`gaborbernat`. (`#2029 <https://github.com/pypa/virtualenv/issues/2029>`_)

v20.2.1 (2020-11-23)

No significant changes.

v20.2.0 (2020-11-21)

Features - 20.2.0

~~~~~~~~~~~~~~~~~
- Optionally skip VCS ignore directive for entire virtualenv directory, using option :option:`no-vcs-ignore`, by default ``False``. (`#2003 <https://github.com/pypa/virtualenv/issues/2003>`_)
- Add ``--read-only-app-data`` option to allow for creation based on an existing
  app data cache which is non-writable. This may be useful (for example) to
  produce a docker image where the app-data is pre-populated.

  .. code-block:: dockerfile

      ENV \
          VIRTUALENV_OVERRIDE_APP_DATA=/opt/virtualenv/cache \
          VIRTUALENV_SYMLINK_APP_DATA=1
      RUN virtualenv venv && rm -rf venv
      ENV VIRTUALENV_READ_ONLY_APP_DATA=1
      USER nobody
      # this virtualenv has symlinks into the read-only app-data cache
      RUN virtualenv /tmp/venv

  Patch by :user:`asottile`. (`#2009 <https://github.com/pypa/virtualenv/issues/2009>`_)

Bugfixes - 20.2.0
~~~~~~~~~~~~~~~~~
- Fix processing of the ``VIRTUALENV_PYTHON`` environment variable and make it
  multi-value as well (separated by comma) - by :user:`pneff`. (`#1998 <https://github.com/pypa/virtualenv/issues/1998>`_)

v20.1.0 (2020-10-25)

Features - 20.1.0
~~~~~~~~~~~~~~~~~
- The python specification can now take one or more values, first found is used to create the virtual environment - by
  :user:`gaborbernat`. (`#1995 <https://github.com/pypa/virtualenv/issues/1995>`_)

v20.0.35 (2020-10-15)

Bugfixes - 20.0.35
~~~~~~~~~~~~~~~~~~
- Bump embedded setuptools from ``50.3.0`` to ``50.3.1`` - by :user:`gaborbernat`. (`#1982 <https://github.com/pypa/virtualenv/issues/1982>`_)
- After importing virtualenv passing cwd to a subprocess calls breaks with ``invalid directory`` - by :user:`gaborbernat`. (`#1983 <https://github.com/pypa/virtualenv/issues/1983>`_)

v20.0.34 (2020-10-12)

Bugfixes - 20.0.34
~~~~~~~~~~~~~~~~~~
- Align with venv module when creating virtual environments with builtin creator on Windows 3.7 and later
  - by :user:`gaborbernat`. (`#1782 <https://github.com/pypa/virtualenv/issues/1782>`_)
- Handle Cygwin path conversion in the activation script - by :user:`davidcoghlan`. (`#1969 <https://github.com/pypa/virtualenv/issues/1969>`_)

v20.0.33 (2020-10-04)

Bugfixes - 20.0.33
~~~~~~~~~~~~~~~~~~
- Fix ``None`` type error in cygwin if POSIX path in dest - by :user:`danyeaw`. (`#1962 <https://github.com/pypa/virtualenv/issues/1962>`_)
- Fix Python 3.4 incompatibilities (added back to the CI) - by :user:`gaborbernat`. (`#1963 <https://github.com/pypa/virtualenv/issues/1963>`_)

v20.0.32 (2020-10-01)

Bugfixes - 20.0.32
~~~~~~~~~~~~~~~~~~
- For activation scripts always use UNIX line endings (unless it's BATCH shell related) - by :user:`saytosid`. (`#1818 <https://github.com/pypa/virtualenv/issues/1818>`_)
- Upgrade embedded pip to ``20.2.1`` and setuptools to ``49.4.0`` - by :user:`gaborbernat`. (`#1918 <https://github.com/pypa/virtualenv/issues/1918>`_)
- Avoid spawning new windows when doing seed package upgrades in the background on Windows - by :user:`gaborbernat`. (`#1928 <https://github.com/pypa/virtualenv/issues/1928>`_)
- Fix a bug that reading and writing on the same file may cause race on multiple processes. (`#1938 <https://github.com/pypa/virtualenv/issues/1938>`_)
- Upgrade embedded setuptools to ``50.2.0`` and pip to ``20.2.3`` - by :user:`gaborbernat`. (`#1939 <https://github.com/pypa/virtualenv/issues/1939>`_)
- Provide correct path for bash activator in cygwin or msys2 - by :user:`danyeaw`. (`#1940 <https://github.com/pypa/virtualenv/issues/1940>`_)
- Relax importlib requirement to allow version<3 - by :user:`usamasadiq` (`#1953 <https://github.com/pypa/virtualenv/issues/1953>`_)
- pth files were not processed on CPython2 if $PYTHONPATH was pointing to site-packages/ - by :user:`navytux`. (`#1959 <https://github.com/pypa/virtualenv/issues/1959>`_) (`#1960 <https://github.com/pypa/virtualenv/issues/1960>`_)

v20.0.31 (2020-08-17)

Bugfixes - 20.0.31
~~~~~~~~~~~~~~~~~~
- Upgrade embedded pip to ``20.2.1``, setuptools to ``49.6.0`` and wheel to ``0.35.1`` - by :user:`gaborbernat`. (`#1918 <https://github.com/pypa/virtualenv/issues/1918>`_)

v20.0.30 (2020-08-04)

Bugfixes - 20.0.30
~~~~~~~~~~~~~~~~~~
- Upgrade pip to ``20.2.1`` and setuptools to ``49.2.1`` - by :user:`gaborbernat`. (`#1915 <https://github.com/pypa/virtualenv/issues/1915>`_)

v20.0.29 (2020-07-31)

Bugfixes - 20.0.29
~~~~~~~~~~~~~~~~~~
- Upgrade embedded pip from version ``20.1.2`` to ``20.2`` - by :user:`gaborbernat`. (`#1909 <https://github.com/pypa/virtualenv/issues/1909>`_)

v20.0.28 (2020-07-24)

Bugfixes - 20.0.28
~~~~~~~~~~~~~~~~~~
- Fix test suite failing if run from system Python - by :user:`gaborbernat`. (`#1882 <https://github.com/pypa/virtualenv/issues/1882>`_)
- Provide ``setup_logging`` flag to python API so that users can bypass logging handling if their application already
  performs this - by :user:`gaborbernat`. (`#1896 <https://github.com/pypa/virtualenv/issues/1896>`_)
- Use ``\n`` instead if ``\r\n`` as line separator for report (because Python already performs this transformation
  automatically upon write to the logging pipe) - by :user:`gaborbernat`. (`#1905 <https://github.com/pypa/virtualenv/issues/1905>`_)

v20.0.27 (2020-07-15)

Bugfixes - 20.0.27
~~~~~~~~~~~~~~~~~~
- No longer preimport threading to fix support for `gpython <https://pypi.org/project/pygolang/#gpython>`_ and `gevent <https://www.gevent.org/>`_ - by :user:`navytux`. (`#1897 <https://github.com/pypa/virtualenv/issues/1897>`_)
- Upgrade setuptools from ``49.2.0`` on ``Python 3.5+`` - by :user:`gaborbernat`. (`#1898

<https://github.com/pypa/virtualenv/issues/1898>`_)

v20.0.26 (2020-07-07)

Bugfixes - 20.0.26
~~~~~~~~~~~~~~~~~~
- Bump dependency ``distutils >= 0.3.1`` - by :user:`gaborbernat`. (`#1880
<https://github.com/pypa/virtualenv/issues/1880>`_)
- Improve periodic update handling:

  - better logging output while running and enable logging on background process call (
    ``_VIRTUALENV_PERIODIC_UPDATE_INLINE`` may be used to debug behavior inline)
  - fallback to unverified context when querying the PyPi for release date,
  - stop downloading wheels once we reach the embedded version,

  by :user:`gaborbernat`. (`#1883 <https://github.com/pypa/virtualenv/issues/1883>`_)
- Do not print error message if the application exists with ``SystemExit(0)`` - by
:user:`gaborbernat`. (`#1885 <https://github.com/pypa/virtualenv/issues/1885>`_)
- Upgrade embedded setuptools from ``47.3.1`` to ``49.1.0`` for Python ``3.5+`` - by
:user:`gaborbernat`. (`#1887 <https://github.com/pypa/virtualenv/issues/1887>`_)

v20.0.25 (2020-06-23)

Bugfixes - 20.0.25
~~~~~~~~~~~~~~~~~~
- Fix that when the ``app-data`` seeders image creation fails the exception is silently ignored.
Avoid two virtual environment creations to step on each others toes by using a lock while creating
the base images. By :user:`gaborbernat`. (`#1869
<https://github.com/pypa/virtualenv/issues/1869>`_)

v20.0.24 (2020-06-22)

Features - 20.0.24
~~~~~~~~~~~~~~~~~~
- Ensure that the seeded packages do not get too much out of date:

  - add a CLI flag that triggers upgrade of embedded wheels under :option:`upgrade-embed-wheels`
  - periodically (once every 14 days) upgrade the embedded wheels in a background process, and use
them if they have been
    released for more than 28 days (can be disabled via :option:`no-periodic-update`)

  More details under :ref:`wheels` - by :user:`gaborbernat`. (`#1821
<https://github.com/pypa/virtualenv/issues/1821>`_)
- Upgrade embed wheel content:

  - ship wheels for Python ``3.9`` and ``3.10``
  - upgrade setuptools for Python ``3.5+`` from ``47.1.1`` to ``47.3.1``

  by :user:`gaborbernat`. (`#1841 <https://github.com/pypa/virtualenv/issues/1841>`_)
- Display the installed seed package versions in the final summary output, for example:

  .. code-block:: console

      created virtual environment CPython3.8.3.final.0-64 in 350ms
        creator CPython3Posix(dest=/x, clear=True, global=False)
        seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy,
app_data_dir=/y/virtualenv)
          added seed packages: pip==20.1.1, setuptools==47.3.1, wheel==0.34.2

  by :user:`gaborbernat`. (`#1864 <https://github.com/pypa/virtualenv/issues/1864>`_)

Bugfixes - 20.0.24
~~~~~~~~~~~~~~~~~~
- Do not generate/overwrite ``.gitignore`` if it already exists at destination path - by
:user:`gaborbernat`. (`#1862 <https://github.com/pypa/virtualenv/issues/1862>`_)
- Improve error message for no ``.dist-info`` inside the ``app-data`` copy seeder - by
:user:`gaborbernat`. (`#1867 <https://github.com/pypa/virtualenv/issues/1867>`_)

Improved Documentation - 20.0.24
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- How seeding mechanisms discover (and automatically keep it up to date) wheels at :ref:`wheels` - by :user:`gaborbernat`. (`#1821 <https://github.com/pypa/virtualenv/issues/1821>`_)
- How distributions should handle shipping their own embedded wheels at :ref:`distribution_wheels` - by :user:`gaborbernat`. (`#1840 <https://github.com/pypa/virtualenv/issues/1840>`_)

v20.0.23 (2020-06-12)

Bugfixes - 20.0.23
~~~~~~~~~~~~~~~~~~
- Fix typo in ``setup.cfg`` - by :user:`RowdyHowell`. (`#1857 <https://github.com/pypa/virtualenv/issues/1857>`_)

v20.0.22 (2020-06-12)

Bugfixes - 20.0.22
~~~~~~~~~~~~~~~~~~
- Relax ``importlib.resources`` requirement to also allow version 2 - by :user:`asottile`. (`#1846 <https://github.com/pypa/virtualenv/issues/1846>`_)
- Upgrade embedded setuptools to ``44.1.1`` for python 2 and ``47.1.1`` for python3.5+ - by :user:`gaborbernat`. (`#1855 <https://github.com/pypa/virtualenv/issues/1855>`_)

v20.0.21 (2020-05-20)

Features - 20.0.21
~~~~~~~~~~~~~~~~~~
- Generate ignore file for version control systems to avoid tracking virtual environments by default. Users should
  remove these files if still want to track. For now we support only **git** by :user:`gaborbernat`. (`#1806 <https://github.com/pypa/virtualenv/issues/1806>`_)

Bugfixes - 20.0.21
~~~~~~~~~~~~~~~~~~
- Fix virtualenv fails sometimes when run concurrently, ``--clear-app-data`` conflicts with :option:`clear` flag when
  abbreviation is turned on. To bypass this while allowing abbreviated flags on the command line we had to move it to
  :option:`reset-app-data` - by :user:`gaborbernat`. (`#1824 <https://github.com/pypa/virtualenv/issues/1824>`_)
- Upgrade embedded ``setuptools`` to ``46.4.0`` from ``46.1.3`` on Python ``3.5+``, and ``pip`` from ``20.1`` to ``20.1.1`` - by :user:`gaborbernat`. (`#1827 <https://github.com/pypa/virtualenv/issues/1827>`_)
- Seeder pip now correctly handles ``--extra-search-dir`` - by :user:`frenzymadness`. (`#1834 <https://github.com/pypa/virtualenv/issues/1834>`_)

v20.0.20 (2020-05-04)

Bugfixes - 20.0.20
~~~~~~~~~~~~~~~~~~
- Fix download fails with python 3.4 - by :user:`gaborbernat`. (`#1809 <https://github.com/pypa/virtualenv/issues/1809>`_)
- Fixes older CPython2 versions use ``_get_makefile_filename`` instead of ``get_makefile_filename`` on ``sysconfig`` - by :user:`ianw`. (`#1810 <https://github.com/pypa/virtualenv/issues/1810>`_)
- Fix download is ``True`` by default - by :user:`gaborbernat`. (`#1813 <https://github.com/pypa/virtualenv/issues/1813>`_)
- Fail ``app-data`` seed operation when wheel download fails and better error message - by :user:`gaborbernat`. (`#1814 <https://github.com/pypa/virtualenv/issues/1814>`_)

v20.0.19 (2020-05-03)

Bugfixes - 20.0.19
~~~~~~~~~~~~~~~~~~
- Fix generating a Python 2 environment from Python 3 creates invalid python activator - by
~~~~~~~~~~~~~~~~~~

:user:`gaborbernat`. (`#1776 <https://github.com/pypa/virtualenv/issues/1776>`_)
- Fix pinning seed packages via ``app-data`` seeder raised ``Invalid Requirement`` - by
:user:`gaborbernat`. (`#1779 <https://github.com/pypa/virtualenv/issues/1779>`_)
- Do not stop interpreter discovery if we fail to find the system interpreter for a executable
during discovery
  - by :user:`gaborbernat`. (`#1781 <https://github.com/pypa/virtualenv/issues/1781>`_)
- On CPython2 POSIX platforms ensure ``syconfig.get_makefile_filename`` exists within the virtual
environment (this is used by some c-extension based libraries - e.g. numpy - for building) - by
:user:`gaborbernat`. (`#1783 <https://github.com/pypa/virtualenv/issues/1783>`_)
- Better handling of options :option:`copies` and :option:`symlinks`. Introduce priority of where
the option is set
  to follow the order: CLI, env var, file, hardcoded. If both set at same level prefers copy over
symlink. - by
  :user:`gaborbernat`. (`#1784 <https://github.com/pypa/virtualenv/issues/1784>`_)
- Upgrade pip for Python ``2.7`` and ``3.5+`` from ``20.0.2`` to ``20.1`` - by :user:`gaborbernat`.
(`#1793 <https://github.com/pypa/virtualenv/issues/1793>`_)
- Fix CPython is not discovered from Windows registry, and discover pythons from Windows registry
in decreasing order
  by version - by :user:`gaborbernat`. (`#1796 <https://github.com/pypa/virtualenv/issues/1796>`_)
- Fix symlink detection for creators - by :user:`asottile` (`#1803
<https://github.com/pypa/virtualenv/issues/1803>`_)

v20.0.18 (2020-04-16)

Bugfixes - 20.0.18
~~~~~~~~~~~~~~~~~~
- Importing setuptools before cli_run could cause our python information query to fail due to
setuptools patching
  ``distutils.dist.Distribution`` - by :user:`gaborbernat`. (`#1771
<https://github.com/pypa/virtualenv/issues/1771>`_)

v20.0.17 (2020-04-09)

Features - 20.0.17
~~~~~~~~~~~~~~~~~~
- Extend environment variables checked for configuration to also check aliases (e.g. setting either
  ``VIRTUALENV_COPIES`` or ``VIRTUALENV_ALWAYS_COPY`` will work) - by :user:`gaborbernat`. (`#1763
<https://github.com/pypa/virtualenv/issues/1763>`_)

v20.0.16 (2020-04-04)

Bugfixes - 20.0.16
~~~~~~~~~~~~~~~~~~
- Allow seed wheel files inside the :option:`extra-search-dir` folders that do not have ``Requires-
Python``
  metadata specified, these are considered compatible with all python versions - by
:user:`gaborbernat`. (`#1757 <https://github.com/pypa/virtualenv/issues/1757>`_)

v20.0.15 (2020-03-27)

Features - 20.0.15
~~~~~~~~~~~~~~~~~~
- Upgrade embedded setuptools to ``46.1.3`` from ``46.1.1`` - by :user:`gaborbernat`. (`#1752
<https://github.com/pypa/virtualenv/issues/1752>`_)

v20.0.14 (2020-03-25)

Features - 20.0.14
~~~~~~~~~~~~~~~~~~
- Remove ``__PYVENV_LAUNCHER__`` on macOs for Python ``3.7.(<8)`` and ``3.8.(<3)`` on interpreter
startup via ``pth``
  file, this pulls in the `upstream patch <https://github.com/python/cpython/pull/9516>`_ - by
:user:`gaborbernat`. (`#1704 <https://github.com/pypa/virtualenv/issues/1704>`_)
- Upgrade embedded setuptools for Python ``3.5+`` to ``46.1.1``, for Python ``2.7`` to ``44.1.0`` -
~~~~~~~~~~~~~~~~~~

by :user:`gaborbernat`. (`#1745 <https://github.com/pypa/virtualenv/issues/1745>`_)

Bugfixes - 20.0.14
~~~~~~~~~~~~~~~~~~
- Fix discovery of interpreter by name from ``PATH`` that does not match a spec format - by
:user:`gaborbernat`. (`#1746 <https://github.com/pypa/virtualenv/issues/1746>`_)

v20.0.13 (2020-03-19)

Bugfixes - 20.0.13
~~~~~~~~~~~~~~~~~~
- Do not fail when the pyc files is missing for the host Python 2 - by :user:`gaborbernat`. (`#1738
<https://github.com/pypa/virtualenv/issues/1738>`_)
- Support broken Packaging pythons that put the include headers under distutils pattern rather than
sysconfig one
  - by :user:`gaborbernat`. (`#1739 <https://github.com/pypa/virtualenv/issues/1739>`_)

v20.0.12 (2020-03-19)

Bugfixes - 20.0.12
~~~~~~~~~~~~~~~~~~
- Fix relative path discovery of interpreters - by :user:`gaborbernat`. (`#1734
<https://github.com/pypa/virtualenv/issues/1734>`_)

v20.0.11 (2020-03-18)

Features - 20.0.11
~~~~~~~~~~~~~~~~~~
- Improve error message when the host python does not satisfy invariants needed to create virtual
environments (now we
  print which host files are incompatible/missing and for which creators when no supported creator
can be matched, however
  we found creators that can describe the given Python interpreter - will still print no supported
creator for Jython,
  however print exactly what host files do not allow creation of virtual environments in case of
CPython/PyPy)
  - by :user:`gaborbernat`. (`#1716 <https://github.com/pypa/virtualenv/issues/1716>`_)

Bugfixes - 20.0.11
~~~~~~~~~~~~~~~~~~
- Support Python 3 Framework distributed via XCode in macOs Catalina and before - by
:user:`gaborbernat`. (`#1663 <https://github.com/pypa/virtualenv/issues/1663>`_)
- Fix Windows Store Python support, do not allow creation via symlink as that's not going to work
by design
  - by :user:`gaborbernat`. (`#1709 <https://github.com/pypa/virtualenv/issues/1709>`_)
- Fix ``activate_this.py`` throws ``AttributeError`` on Windows when virtual environment was
created via cross python
  mechanism - by :user:`gaborbernat`. (`#1710 <https://github.com/pypa/virtualenv/issues/1710>`_)
- Fix ``--no-pip``, ``--no-setuptools``, ``--no-wheel`` not being respected - by
:user:`gaborbernat`. (`#1712 <https://github.com/pypa/virtualenv/issues/1712>`_)
- Allow missing ``.py`` files if a compiled ``.pyc`` version is available - by :user:`tucked`.
(`#1714 <https://github.com/pypa/virtualenv/issues/1714>`_)
- Do not fail if the distutils/setuptools patch happens on a C-extension loader (such as
``zipimporter`` on Python 3.7 or
  earlier) - by :user:`gaborbernat`. (`#1715 <https://github.com/pypa/virtualenv/issues/1715>`_)
- Support Python 2 implementations that require the landmark files and ``site.py`` to be in
platform standard library
  instead of the standard library path of the virtual environment (notably some RHEL ones, such as
the Docker
  image ``amazonlinux:1``) - by :user:`gaborbernat`. (`#1719
<https://github.com/pypa/virtualenv/issues/1719>`_)
- Allow the test suite to pass even when called with the system Python - to help repackaging of the
tool for Linux
  distributions - by :user:`gaborbernat`. (`#1721
<https://github.com/pypa/virtualenv/issues/1721>`_)
- Also generate ``pipx.y`` console script beside ``pip-x.y`` to be compatible with how pip installs
itself -
  by :user:`gaborbernat`. (`#1723 <https://github.com/pypa/virtualenv/issues/1723>`_)
~~~~~~~~~~~~~~~~~~

- Automatically create the application data folder if it does not exists - by :user:`gaborbernat`.
(`#1728 <https://github.com/pypa/virtualenv/issues/1728>`_)

Improved Documentation - 20.0.11
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- :ref:`supports <compatibility-requirements>` details now explicitly what Python installations we
support
  - by :user:`gaborbernat`. (`#1714 <https://github.com/pypa/virtualenv/issues/1714>`_)

v20.0.10 (2020-03-10)

Bugfixes - 20.0.10
~~~~~~~~~~~~~~~~~~
- Fix acquiring python information might be altered by distutils configuration files generating
incorrect layout virtual
  environments - by :user:`gaborbernat`. (`#1663
<https://github.com/pypa/virtualenv/issues/1663>`_)
- Upgrade embedded setuptools to ``46.0.0`` from ``45.3.0`` on Python ``3.5+`` - by
:user:`gaborbernat`. (`#1702 <https://github.com/pypa/virtualenv/issues/1702>`_)

Improved Documentation - 20.0.10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Document requirements (pip + index server) when installing via pip under the installation section
- by
  :user:`gaborbernat`. (`#1618 <https://github.com/pypa/virtualenv/issues/1618>`_)
- Document installing from non PEP-518 systems - :user:`gaborbernat`. (`#1619
<https://github.com/pypa/virtualenv/issues/1619>`_)
- Document installing latest unreleased version from Github - :user:`gaborbernat`. (`#1620
<https://github.com/pypa/virtualenv/issues/1620>`_)

v20.0.9 (2020-03-08)

Bugfixes - 20.0.9
~~~~~~~~~~~~~~~~~
- ``pythonw.exe`` works as ``python.exe`` on Windows - by :user:`gaborbernat`. (`#1686
<https://github.com/pypa/virtualenv/issues/1686>`_)
- Handle legacy loaders for virtualenv import hooks used to patch distutils configuration load - by
:user:`gaborbernat`. (`#1690 <https://github.com/pypa/virtualenv/issues/1690>`_)
- Support for python 2 platforms that store landmark files in ``platstdlib`` over ``stdlib`` (e.g.
RHEL) - by
  :user:`gaborbernat`. (`#1694 <https://github.com/pypa/virtualenv/issues/1694>`_)
- Upgrade embedded setuptools to ``45.3.0`` from ``45.2.0`` for Python ``3.5+`` - by
:user:`gaborbernat`. (`#1699 <https://github.com/pypa/virtualenv/issues/1699>`_)

v20.0.8 (2020-03-04)

Bugfixes - 20.0.8
~~~~~~~~~~~~~~~~~
- Having `distutils configuration <https://docs.python.org/3/install/index.html#distutils-
configuration-files>`_
  files that set ``prefix`` and ``install_scripts`` cause installation of packages in the wrong
location -
  by :user:`gaborbernat`. (`#1663 <https://github.com/pypa/virtualenv/issues/1663>`_)
- Fix ``PYTHONPATH`` being overridden on Python 2 — by :user:`jd`. (`#1673
<https://github.com/pypa/virtualenv/issues/1673>`_)
- Fix list configuration value parsing from config file or environment variable - by
:user:`gaborbernat`. (`#1674 <https://github.com/pypa/virtualenv/issues/1674>`_)
- Fix Batch activation script shell prompt to display environment name by default - by
:user:`spetafree`. (`#1679 <https://github.com/pypa/virtualenv/issues/1679>`_)
- Fix startup on Python 2 is slower for virtualenv - this was due to setuptools calculating it's
working set distribution
  - by :user:`gaborbernat`. (`#1682 <https://github.com/pypa/virtualenv/issues/1682>`_)
- Fix entry points are not populated for editable installs on Python 2 due to setuptools working
set being calculated
  before ``easy_install.pth`` runs - by :user:`gaborbernat`. (`#1684
<https://github.com/pypa/virtualenv/issues/1684>`_)
- Fix ``attr:`` import fails for setuptools - by :user:`gaborbernat`. (`#1685
<https://github.com/pypa/virtualenv/issues/1685>`_)
~~~~~~~~~~~~~~~~~

v20.0.7 (2020-02-26)

Bugfixes - 20.0.7
~~~~~~~~~~~~~~~~~
- Disable distutils fixup for python 3 until `pypa/pip #7778
<https://github.com/pypa/pip/issues/7778>`_ is fixed and
  released - by :user:`gaborbernat`. (`#1669 <https://github.com/pypa/virtualenv/issues/1669>`_)

v20.0.6 (2020-02-26)

Bugfixes - 20.0.6
~~~~~~~~~~~~~~~~~
- Fix global site package always being added with bundled macOs python framework builds - by
:user:`gaborbernat`. (`#1561 <https://github.com/pypa/virtualenv/issues/1561>`_)
- Fix generated scripts use host version info rather than target - by :user:`gaborbernat`. (`#1600
<https://github.com/pypa/virtualenv/issues/1600>`_)
- Fix circular prefix reference with single elements (accept these as if they were system
executables, print a info about
  them referencing themselves) - by :user:`gaborbernat`. (`#1632
<https://github.com/pypa/virtualenv/issues/1632>`_)
- Handle the case when the application data folder is read-only:

  - the application data folder is now controllable via :option:`app-data`,
  - ``clear-app-data`` now cleans the entire application data folder, not just the ``app-data``
seeder path,
  - check if the application data path passed in does not exist or is read-only, and fallback to a
temporary directory,
  - temporary directory application data is automatically cleaned up at the end of execution,
  - :option:`symlink-app-data` is always ``False`` when the application data is temporary

  by :user:`gaborbernat`. (`#1640 <https://github.com/pypa/virtualenv/issues/1640>`_)
- Fix PyPy 2 builtin modules are imported from standard library, rather than from builtin - by
:user:`gaborbernat`. (`#1652 <https://github.com/pypa/virtualenv/issues/1652>`_)
- Fix creation of entry points when path contains spaces - by :user:`nsoranzo`. (`#1660
<https://github.com/pypa/virtualenv/issues/1660>`_)
- Fix relative paths for the zipapp (for python ``3.7+``) - by :user:`gaborbernat`. (`#1666
<https://github.com/pypa/virtualenv/issues/1666>`_)

v20.0.5 (2020-02-21)

Features - 20.0.5
~~~~~~~~~~~~~~~~~
- Also create ``pythonX.X`` executables when creating pypy virtualenvs - by :user:`asottile`
(`#1612 <https://github.com/pypa/virtualenv/issues/1612>`_)
- Fail with better error message if trying to install source with unsupported ``setuptools``, allow
``setuptools-scm >= 2``
  and move to legacy ``setuptools-scm`` format to support better older platforms (``CentOS 7`` and
such) - by :user:`gaborbernat`. (`#1621 <https://github.com/pypa/virtualenv/issues/1621>`_)
- Report of the created virtual environment is now split across four short lines rather than one
long - by :user:`gaborbernat` (`#1641 <https://github.com/pypa/virtualenv/issues/1641>`_)

Bugfixes - 20.0.5
~~~~~~~~~~~~~~~~~
- Add macOs Python 2 Framework support (now we test it with the CI via brew) - by
:user:`gaborbernat` (`#1561 <https://github.com/pypa/virtualenv/issues/1561>`_)
- Fix losing of libpypy-c.so when the pypy executable is a symlink - by :user:`asottile` (`#1614
<https://github.com/pypa/virtualenv/issues/1614>`_)
- Discover python interpreter in a case insensitive manner - by :user:`PrajwalM2212` (`#1624
<https://github.com/pypa/virtualenv/issues/1624>`_)
- Fix cross interpreter support when the host python sets ``sys.base_executable`` based on
``__PYVENV_LAUNCHER__`` -
  by :user:`cjolowicz` (`#1643 <https://github.com/pypa/virtualenv/issues/1643>`_)

v20.0.4 (2020-02-14)

Features - 20.0.4

~~~~~~~~~~~~~~~~~
- When aliasing interpreters, use relative symlinks - by :user:`asottile`. (`#1596 <https://github.com/pypa/virtualenv/issues/1596>`_)

Bugfixes - 20.0.4
~~~~~~~~~~~~~~~~~
- Allow the use of ``/`` as pathname component separator on Windows - by ``vphilippon`` (`#1582 <https://github.com/pypa/virtualenv/issues/1582>`_)
- Lower minimal version of six required to 1.9 - by ``ssbarnea`` (`#1606 <https://github.com/pypa/virtualenv/issues/1606>`_)

v20.0.3 (2020-02-12)

Bugfixes - 20.0.3
~~~~~~~~~~~~~~~~~
- On Python 2 with Apple Framework builds the global site package is no longer added when the
  :option:`system-site-packages` is not specified - by :user:`gaborbernat`. (`#1561 <https://github.com/pypa/virtualenv/issues/1561>`_)
- Fix system python discovery mechanism when prefixes contain relative parts (e.g. ``..``) by resolving paths within the
  python information query - by :user:`gaborbernat`. (`#1583 <https://github.com/pypa/virtualenv/issues/1583>`_)
- Expose a programmatic API as ``from virtualenv import cli_run`` - by :user:`gaborbernat`. (`#1585 <https://github.com/pypa/virtualenv/issues/1585>`_)
- Fix ``app-data`` :option:`seeder` injects a extra ``.dist-info.virtualenv`` path that breaks ``importlib.metadata``,
  now we inject an extra ``.virtualenv`` - by :user:`gaborbernat`. (`#1589 <https://github.com/pypa/virtualenv/issues/1589>`_)

Improved Documentation - 20.0.3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Document a programmatic API as ``from virtualenv import cli_run`` under :ref:`programmatic_api` -
  by :user:`gaborbernat`. (`#1585 <https://github.com/pypa/virtualenv/issues/1585>`_)

v20.0.2 (2020-02-11)

Features - 20.0.2
~~~~~~~~~~~~~~~~~
- Print out a one line message about the created virtual environment when no :option:`verbose` is set, this can now be
  silenced to get back the original behavior via the :option:`quiet` flag - by :user:`pradyunsg`. (`#1557 <https://github.com/pypa/virtualenv/issues/1557>`_)
- Allow virtualenv's app data cache to be overridden by ``VIRTUALENV_OVERRIDE_APP_DATA`` - by :user:`asottile`. (`#1559 <https://github.com/pypa/virtualenv/issues/1559>`_)
- Passing in the virtual environment name/path is now required (no longer defaults to ``venv``) - by :user:`gaborbernat`. (`#1568 <https://github.com/pypa/virtualenv/issues/1568>`_)
- Add a CLI flag :option:`with-traceback` that allows displaying the stacktrace of the virtualenv when a failure occurs
  - by :user:`gaborbernat`. (`#1572 <https://github.com/pypa/virtualenv/issues/1572>`_)

Bugfixes - 20.0.2
~~~~~~~~~~~~~~~~~
- Support long path names for generated virtual environment console entry points (such as ``pip``) when using the
  ``app-data`` :option:`seeder` - by :user:`gaborbernat`. (`#997 <https://github.com/pypa/virtualenv/issues/997>`_)
- Improve python discovery mechanism:

  - do not fail if there are executables that fail to query (e.g. for not having execute access to it) on the ``PATH``,
  - beside the prefix folder also try with the platform dependent binary folder within that,

  by :user:`gaborbernat`. (`#1545 <https://github.com/pypa/virtualenv/issues/1545>`_)
- When copying (either files or trees) do not copy the permission bits, last access time, last modification time, and
  flags as access to these might be forbidden (for example in case of the macOs Framework Python) and these are not needed
  for the user to use the virtual environment - by :user:`gaborbernat`. (`#1561 <https://github.com/pypa/virtualenv/issues/1561>`_)
- While discovering a python executables interpreters that cannot be queried are now displayed with

info level rather
  than warning, so now they're no longer shown by default (these can be just executables to which we don't have access
  or that are broken, don't warn if it's not the target Python we want) - by :user:`gaborbernat`. (`#1574 <https://github.com/pypa/virtualenv/issues/1574>`_)
- The ``app-data`` :option:`seeder` no longer symlinks the packages on UNIX and copies on Windows. Instead by default
  always copies, however now has the :option:`symlink-app-data` flag allowing users to request this less robust but faster
  method - by :user:`gaborbernat`. (`#1575 <https://github.com/pypa/virtualenv/issues/1575>`_)

Improved Documentation - 20.0.2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Add link to the `legacy documentation <https://virtualenv.pypa.io/en/legacy>`_ for the changelog by :user:`jezdez`. (`#1547 <https://github.com/pypa/virtualenv/issues/1547>`_)
- Fine tune the documentation layout: default width of theme, allow tables to wrap around, soft corners for code snippets
  - by :user:`pradyunsg`. (`#1548 <https://github.com/pypa/virtualenv/issues/1548>`_)

v20.0.1 (2020-02-10)

Features - 20.0.1
~~~~~~~~~~~~~~~~~
- upgrade embedded setuptools to ``45.2.0`` from ``45.1.0`` for Python ``3.4+`` - by :user:`gaborbernat`. (`#1554 <https://github.com/pypa/virtualenv/issues/1554>`_)

Bugfixes - 20.0.1
~~~~~~~~~~~~~~~~~
- Virtual environments created via relative path on Windows creates bad console executables - by :user:`gaborbernat`. (`#1552 <https://github.com/pypa/virtualenv/issues/1552>`_)
- Seems sometimes venvs created set their base executable to themselves; we accept these without question, so we handle
  virtual environments as system pythons causing issues - by :user:`gaborbernat`. (`#1553 <https://github.com/pypa/virtualenv/issues/1553>`_)

v20.0.0. (2020-02-10)

Improved Documentation - 20.0.0.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Fixes typos, repeated words and inconsistent heading spacing. Rephrase parts of the development documentation and CLI
  documentation. Expands shorthands like ``env var`` and ``config`` to their full forms. Uses descriptions from respective
  documentation, for projects listed in ``related links`` - by :user:`pradyunsg`. (`#1540 <https://github.com/pypa/virtualenv/issues/1540>`_)

v20.0.0b2 (2020-02-04)

Features - 20.0.0b2
~~~~~~~~~~~~~~~~~~~
- Improve base executable discovery mechanism:

  - print at debug level why we refuse some candidates,
  - when no candidates match exactly, instead of hard failing fallback to the closest match where the priority of
    matching attributes is: python implementation, major version, minor version, architecture, patch version,
    release level and serial (this is to facilitate things to still work when the OS upgrade replace/upgrades the system
    python with a never version, than what the virtualenv host python was created with),
  - always resolve system_executable information during the interpreter discovery, and the discovered environment is the
    system interpreter instead of the venv/virtualenv (this happened before lazily the first time we accessed, and caused
    reporting that the created virtual environment is of type of the virtualenv host python version, instead of the
    system pythons version - these two can differ if the OS upgraded the system python underneath and the virtualenv
    host was created via copy),
~~~~~~~~~~~~~~~~~~~

  by :user:`gaborbernat`. (`#1515 <https://github.com/pypa/virtualenv/issues/1515>`_)
- Generate ``bash`` and ``fish`` activators on Windows too (as these can be available with git bash, cygwin or mysys2)
  - by :user:`gaborbernat`. (`#1527 <https://github.com/pypa/virtualenv/issues/1527>`_)
- Upgrade the bundled ``wheel`` package from ``0.34.0`` to ``0.34.2`` - by :user:`gaborbernat`. (`#1531 <https://github.com/pypa/virtualenv/issues/1531>`_)

Bugfixes - 20.0.0b2
~~~~~~~~~~~~~~~~~~~
- Bash activation script should have no extensions instead of ``.sh`` (this fixes the :pypi:`virtualenvwrapper`
  integration) - by :user:`gaborbernat`. (`#1508 <https://github.com/pypa/virtualenv/issues/1508>`_)
- Show less information when we run with a single verbosity (``-v``):

  - no longer shows accepted interpreters information (as the last proposed one is always the accepted one),
  - do not display the ``str_spec`` attribute for ``PythonSpec`` as these can be deduced from the other attributes,
  - for the ``app-data`` seeder do not show the type of lock, only the path to the app data directory,

  By :user:`gaborbernat`. (`#1510 <https://github.com/pypa/virtualenv/issues/1510>`_)
- Fixed cannot discover a python interpreter that has already been discovered under a different path (such is the case
  when we have multiple symlinks to the same interpreter) - by :user:`gaborbernat`. (`#1512 <https://github.com/pypa/virtualenv/issues/1512>`_)
- Support relative paths for ``-p`` - by :user:`gaborbernat`. (`#1514 <https://github.com/pypa/virtualenv/issues/1514>`_)
- Creating virtual environments in parallel fail with cannot acquire lock within app data - by :user:`gaborbernat`. (`#1516 <https://github.com/pypa/virtualenv/issues/1516>`_)
- pth files were not processed under Debian CPython2 interpreters - by :user:`gaborbernat`. (`#1517 <https://github.com/pypa/virtualenv/issues/1517>`_)
- Fix prompt not displayed correctly with upcoming fish 3.10 due to us not preserving ``$pipestatus`` - by
  :user:`krobelus`. (`#1530 <https://github.com/pypa/virtualenv/issues/1530>`_)
- Stable order within ``pyenv.cfg`` and add ``include-system-site-packages`` only for creators that reference a global
  Python - by user:`gaborbernat`. (`#1535 <https://github.com/pypa/virtualenv/issues/1535>`_)

Improved Documentation - 20.0.0b2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
- Create the first iteration of the new documentation - by :user:`gaborbernat`. (`#1465 <https://github.com/pypa/virtualenv/issues/1465>`_)
- Project readme is now of type MarkDown instead of reStructuredText - by :user:`gaborbernat`. (`#1531 <https://github.com/pypa/virtualenv/issues/1531>`_)

v20.0.0b1 (2020-01-28)
----

* First public release of the rewrite. Everything is brand new and just added.
* ``--download`` defaults to ``False``
* No longer replaces builtin ``site`` module with `custom version baked within virtualenv code itself <https://github.com/pypa/virtualenv/blob/legacy/virtualenv_embedded/site.py>`_. A simple shim module is used to fix up things on Python 2 only.

.. warning::

   The current virtualenv is the second iteration of implementation. From version ``0.8`` all the way to ``16.7.9``
   we numbered the first iteration. Version ``20.0.0b1`` is a complete rewrite of the package, and as such this release
   history starts from there. The old changelog is still available in the
   `legacy branch documentation <https://virtualenv.pypa.io/en/legacy/changes.html>`_.