Python and Python Wheels for OpenVMS

Release Notes

Publication Date: January 2026

Operating Systems: VSI OpenVMS IA-64 Version 8.4-2L1 or higher
VSI OpenVMS x86-64 Version 9.2-3

Software Version: Python V3.10-10 and Python Wheels V1.1-7

Kit Names: VSI-I64VMS-PYTHON-V0310-10-1.PCSI$COMPRESSED
VSI-X86VMS-PYTHON-V0310-10-1.PCSI$COMPRESSED
VSI-I64VMS-PYTHWHLS-V0101-7-1.PCSI$COMPRESSED
VSI-X86VMS-PYTHWHLS-V0101-7-1.PCSI$COMPRESSED

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

Table of Contents

1. Introduction

Thank you for your interest in this port of Python for OpenVMS x86-64. This release of Python for OpenVMS is based on the Python 3.10.0 Open Source distribution and is the first full release of Python for OpenVMS x86-64. Note that other than Python proper, this release also includes a dedicated Python packages kit (for details, see *Section 4, "Installing the Kits"*).

Python (https://www.python.org/ [https://www.python.org/]) is an interpreted high-level programming language for general-purpose programming that emphasizes code readability. It provides constructs that enable clear programming of both small- and large-scale software applications. Python features a dynamic type system and automatic memory management. The Python language supports multiple programming paradigms, including object-oriented, functional, and procedural, and has a large and comprehensive runtime library.

This OpenVMS port of Python comprises two kits, namely a core Python kit and a Python packages kit. The core Python kit includes the core binary Python distribution and various OpenVMS-specific extensions, including interfaces for OpenVMS products, along with interfaces for many OpenVMS system services and library calls. The packages kit includes a substantial collection of commonly used Python packages, including modules for web application development, integration, and testing. It is anticipated that additional packages will be included in future releases. It is also possible for users to readily download and install many other Python packages directly from the Python Package Index repository at https://pypi.org/ or elsewhere using the Python PIP package installer.

2. Requirements

The kit you are receiving has been compiled and built using the operating system and compiler versions listed below. While it is highly likely that you will have no problems installing and using the kit on systems running higher versions of the products listed, we cannot say for sure that you will be so lucky if your system is running older versions.

- One of the following versions of OpenVMS:
  - VSI OpenVMS IA-64 Version 8.4-2L1 or higher
  - VSI OpenVMS x86-64 Version 9.2-3
- The software must be installed on an ODS-5 enabled disk. (The installation will fail if this requirement is not met.)
- VSI TCP/IP or HPE TCP/IP Services for OpenVMS

  It has not been verified whether the kit works with the MultiNet TCP/IP stack, but there is a good chance that it will.
- C compiler (optional; required only if you intend to develop your own extensions)

In addition to the above requirements, it is assumed that the reader has a good knowledge of OpenVMS and of software development in the OpenVMS environment.

3. Recommended Reading

It is recommended that software developers read some of the excellent tutorials and other documentation available via the Python web site (https://www.python.org/). The web site provides numerous links

to books, technical guides, tutorials, and many other documents that provide useful information on developing applications using Python.

In addition, the VSI wiki provides the following pages, which provide details regarding many of the OpenVMS-specific modules, and details of compatibility issues that may be relevant to developing and using Python applications on VSI OpenVMS.

- https://wiki.vmssoftware.com/VMS_specific_Python_modules
- https://wiki.vmssoftware.com/VMS_Python_compatibility_issues

It should be noted that the documentation for OpenVMS-specific modules is at this time incomplete. It is anticipated that this documentation will be updated over the coming months to include details for all such modules.

The core Python kit also includes a number of simple example and test programs in the directory PYTHON$ROOT:[LIB.PYTHON^.10.TEST] that serve to illustrate the usage of some of some of the OpenVMS specific extensions. It is envisaged that additional examples will be added over time.

4. Installing the Kits

As noted previously, this OpenVMS port of Python comprises two compressed PCSI kits, namely a core Python kit and a packages kit (also referred to as the “wheels” kit).

Note that the packages kit is based on Python *wheels* (see https://pythonwheels.com/), which provides a flexible and efficient means of installing optional Python packages. While it would be possible to combine the core and packages kits into a single installable kit, it is unlikely that all packages in the packages kit will be of interest to all users. It is also envisaged that the list of packages and package versions will be updated more regularly than the core distribution.

Note

The Pandas module is *not* included in this release.

4.1. Installing the Core Kit

The core Python for VSI OpenVMS kit can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL PYTHON
```

The installation will then proceed as follows (output may differ slightly from that shown):

```
The following product has been selected:
    VSI X86VMS PYTHON V3.10-10             Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product
and for any products that may be installed to satisfy software dependency
 requirements.
```

```
Configuring VSI X86VMS PYTHON V3.10-10: Python for OpenVMS is based on
 Python Version 3.10

    (C) Copyright 2025 VMS Software Inc.

    VMS Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI X86VMS PYTHON V3.10-10             DISK$X86SYS:[VMS$COMMON.]

Portion done:
 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

The following product has been installed:
    VSI X86VMS PYTHON V3.10-10             Layered Product

VSI X86VMS PYTHON V3.10-10: Python for OpenVMS is based on Python Version
 3.10

    Post-installation tasks are required.

    To define the Python runtime at system boot time, add the
    following lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

        $ file := SYS$STARTUP:PYTHON$STARTUP.COM
        $ if f$search("''file'") .nes. "" then @'file'
```

After the installation has successfully completed, include the commands displayed at the end of the installation procedure into SYSTARTUP_VMS.COM to ensure that the logical names required in order for users to use the software are defined system-wide at start-up.

To enhance performance, Python includes a facility to compile Python modules to byte codes when they are first used. The compiled files are created in the Python installation area. To ensure that users who do not have permission to write to the installation area do not get errors when using Python, it is recommended that all modules are compiled following installation. This can be done by executing the following commands [1]. It is assumed that the runtime environment has been previously defined as described above by running PYTHON$STARTUP.COM. Compilation should take less than two minutes on most systems.

```
$ set process/parse_style=extended
$ python :== $python$root:[bin]python.exe
$ set default python$root:[lib.python3^.10]
$ python compileall.py ./
```

If new libraries are added into to the installation area the above commands can be re-run in order to compile any new or modified modules.

Users will then be able to use the Python interpreter by defining a foreign command as follows:

```
$ PYTHON :== $PYTHON$ROOT:[BIN]PYTHON.EXE
```

[1]Note that some test scripts relating to UTF-8 encoding will currently fail to compile correctly; these errors may be safely ignored and will be eliminated in figure releases.

Generally speaking there are no special quota or privilege requirements required for users wishing to develop applications using Python, although it should be noted that some extensions may have special requirements (for example, networking extensions may require an elevated BYTLM quota).

4.2. Installing the Packages Kit

The Python packages kit can be installed by a suitably privileged user using the following command:

```
$ PRODUCT INSTALL PYTHWHLS
```

The installation will then proceed as follows (output may differ slightly from that shown):

```
The following product has been selected:
    VSI X86VMS PYTHWHLS V1.1-7             Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product
and for any products that may be installed to satisfy software dependency
 requirements.

Configuring VSI X86VMS PYTHWHLS V1.1-7: Python wheels collection for
 OpenVMS Python 3.10

    (C) Copyright 2026 VMS Software Inc.

    VMS Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI X86VMS PYTHWHLS V1.1-7             DISK$X86SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...50%...60%...80%...90%...100%

The following product has been installed:
    VSI X86VMS PYTHWHLS V1.1-7             Layered Product

VSI X86VMS PYTHWHLS V1.1-7: Python wheels collection for OpenVMS Python
 3.10

    Post-installation tasks are required.

    To define the Wheels for Python runtime at system boot time, add
    The following lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

        $ file := sys$startup:python_wheels$startup.com
        $ if f$search("''file'") .nes. "" then @'file'
```

After the installation has successfully completed, include the command displayed at the end of the installation procedure into SYSTARTUP_VMS.COM to ensure that the logical names PYTHON_WHEELS$ROOT and PIP_FIND_LINKS are defined system-wide at system start-up. The logical name PYTHON_WHEELS$ROOT specifies the location of the installable wheels packages and the logical name PIP_FIND_LINKS is used by the PIP Python package installer to find these packages.

After installing the packages kit and ensuring the logical names PYTHON_WHEELS$ROOT and PIP_FIND_LINKS are correctly defined, individual packages can be installed as follows by a suitably privileged user, where *module-name* is the name of the package that you wish to install.

```
$ python -m pip install --no-index module-name
```

A complete list of the packages included in the PYTHWHLS kit can be found below in *Section 5, "Included Modules"* and details about these various packages can be found at https://pypi.org/.

Note that the **`--no-index`** option needs to be specified when installing packages from the PYTHWHLS kit in order to prevent the PIP Python package installer from instead trying to download packages from the internet. Alternatively to specifying this option you can define the logical name PIP_NO_INDEX as shown below:

```
$ define PIP_NO_INDEX 1
```

When installing packages you should also ensure that the logical name PYTHONCASEOK is not defined.

It should be noted that the installation of some packages can consume considerable temporary disk space. By default, the logical name SYS$SCRATCH will be used to determine the location of this temporary storage; however, an alternative location may be specified by defining the logical name TMPDIR to point to the desired location. For similar reasons it may also be desirable to define the logical name PIP_CACHE_DIR, which determines where the PIP Python package installer caches temporary dependency data when installing packages. Values (directory specifications) for these logical names should be specified using UNIX syntax.

5. Included Modules

As noted previously, this OpenVMS port of Python comprises a core Python kit and a Python packages kit.

The core Python kit includes the core binary distribution and various OpenVMS-specific extensions, including interfaces for OpenVMS products, along with various facility definitions (item codes and error codes) and interfaces for many OpenVMS system services and library calls. The following table summarizes the OpenVMS-specific packages:

accdef	ile3	prvdef
acldef	iledef	prxdef
acrdef	impdef	pscandef
armdef	indexedfile	psldef
brkdef	initdef	pxbdef
capdef	iodef	quidef
chpdef	issdef	rabdef
ciadef	jbcmsgdef	clidef
jpidef	rec	cmbdef
kgbdef	regdef	cvtfnmdef
lckdef	rmidef	dcdef
lib, libdef	rms	decc

libclidef	rmsdef	dmtdef
libdtdef	rsdmdef	dpsdef
libfisdef	sdvdef	dscdef
lkidef	sjcdef	lnmdef
ssdef	dvidef	maildef
statedef	dvsdef	mntdef
stenvdef	efndef	nsadef
stsdef	eradef	ossdef
syidef	fabdef	pcbdef
sys	fdldef	ppropdef
uafdef	fpdef	pqldef
uaidef	fscndef	prcdef
prvdef	iccdef	prdef
prxdef		

Several simple example and test programs for some of these extensions can be found in PYTHON$ROOT:[LIB.PYTHON^.10.TEST], and several examples for the OpenVMS RMS extension can also be found in PYTHON$ROOT:[LIB.PYTHON^.10.VMS].

Note that the core distribution also includes the following two default packages:

- pip
- setuptools

The PYTHWHLS packages kit includes a collection of commonly used Python packages, including packages for web application development, integration, and testing. The following table lists the various packages that are included in this release of the PYTHWHLS kit. It is envisaged that this list will be periodically updated to include additional packages as well as new versions of existing packages. Details about each of these packages may be found at https://pypi.org/.

Note that if a particular package is not currently included in the PYTHWHLS kit and your OpenVMS system is able to access the internet, in most cases it will be possible to download and install the desired package directly using the **`python -m pip install <module-name>`** command.

anyio-3.6.2	apipkg-3.0.1	apispec-6.0.2
apispec-webframeworks-0.5.2	appdirs-1.4.4	async-generator-1.10
attrs-22.2.0	autocommand-2.2.2	Automat-22.10.0
awscli-1.19.50	backports.entry-points-selectable-1.2.0	behave-1.2.6
betamax-0.8.1	blinker-1.5	botocore-1.20.50
bottle-0.12.23	Brotli-1.0.9	certifi-2024.2.2
cffi-1.14.5	chardet-3.0.4	charset-normalizer-3.0.1
cheroot-9.0.0	CherryPy-18.8.0	click-8.1.3

colorama-0.4.3	constantly-15.1.0	cryptography-3.4.7
cryptography-vectors-39.0.0	Cython-0.29.22	decorator-5.1.1
distlib-0.3.6	docutils-0.15.2	elementpath-3.0.2
exceptiongroup-1.1.0	execnet-1.9.0	filelock-3.9.0
flaky-3.7.0	flasgger-0.9.5	Flask-2.0.0
flex-6.14.1	freezegun-1.2.2	ftputil-5.0.4
h11-0.14.0	httpbin-0.7.0	httpcore-0.16.3
httpx-0.23.3	hypothesis-6.65.1	idna-2.8
incremental-22.10.0	inflect-6.0.2	iniconfig-2.0.0
iso8601-1.1.0	itsdangerous-2.1.2	jaraco.classes-3.2.3
jaraco.collections-3.8.0	jaraco.context-4.3.0	jaraco.functools-3.5.2
jaraco.text-3.11.1	Jinja2-3.1.2	jmespath-0.10.0
jsonpointer-2.3	jsonschema-4.17.3	MarkupSafe-2.1.2
marshmallow-3.19.0	mistune-2.0.4	mock-5.0.1
more-itertools-9.0.0	nose-1.3.7	numpy-1.20.3
outcome-1.2.0	packaging-21.3	parse-1.19.0
parse-type-0.6.0	path-16.6.0	pbr-5.11.1
perfdat-4.3.1	pika-1.3.1	Pillow-8.2.0
pip-21.2.3	platformdirs-2.6.2	pluggy-0.13.1
portend-3.1.0	prance-0.20.0	pretend-1.0.9
py-1.11.0	pyasn1-0.4.8	pyasn1-modules-0.2.8
pycparser-2.21	pydantic-1.10.4	Pygments-2.14.0
pyOpenSSL-21.0.0	pyparsing-3.0.9	pypdf-3.3.0
PyPDF2-3.0.1	pyrsistent-0.18.0	pytest-6.2.4
pytest-flask-1.2.0	pytest-forked-1.4.0	pytest-httpbin-1.0.2
pytest-httpserver-1.0.6	pytest-localserver-0.7.0	pytest-subtests-0.5.0
pytest-xdist-3.1.0	python-dateutil-2.8.2	python-interface-1.6.1
python-magic-0.4.27	pytz-2022.7.1	pywebui-0.1
PyYAML-5.4.1	raven-6.10.0	reportlab-3.5.66
requests-2.22.0	responses-0.22.0	rfc3986-1.5.0
rfc3987-1.3.8	rsa-4.7.2	ruamel.yaml-0.17.21
ruamel.yaml.clib-0.2.8	s3cmd-2.3.0	s3transfer-0.3.7
semantic-version-2.10.0	semver-2.13.0	service-identity-21.1.0
setuptools-57.4.0	setuptools-rust-0.12.1	setuptools-scm-7.1.0
simplejson-3.17.5	six-1.16.0	sniffio-1.3.0

sortedcontainers-2.4.0	starlette-0.23.1	strict-rfc3339-0.7
suds-py3-1.4.5.0	sure-2.0.0	tempora-5.2.1
toml-0.10.2	tomli-2.0.1	trustme-0.9.0
types-toml-0.10.8.1	typing_extensions-4.4.0	urllib3-1.25.11
uvicorn-0.20.0	validate-email-1.3	virtualenv-20.17.1
wcwidth-0.2.6	Werkzeug-2.0.3	wheel-0.42.0
wsit-1.0.19	xmlschema-2.1.1	zc.lockfile-2.0

Future releases of Python for OpenVMS may include a more comprehensive suite of such modules along with accompanying documentation and example programs.

Developers can implement their own dynamic Python modules (shareable images) by including in their projects the Python header files found in PYTHON$ROOT:[INCLUDE] and by linking with the shareable image PYTHON$ROOT:[LIB]PYTHON$SHR.EXE. Note that the logical name PYTHON$SHR is defined by the Python start-up procedure and may be used to link with the shareable image. Symbols in the shareable image are mixed-case, and developers should use the C compiler option **`/NAMES=(AS_IS,SHORTENED)`** or include in their code appropriate `#pragma` directives to ensure that symbols are correctly resolved when linking.

5.1. A Note on Using S3cmd

S3cmd (see https://s3tools.org/s3cmd) is a Python-based command line utility for uploading, downloading, and managing files in AWS S3 storage and equivalent services provided by other cloud storage service providers that support the S3 protocol.

In order to use S3cmd, in addition to having access to S3-based storage, some basic level of configuration is required. The following notes describe the basic steps that must be followed in order to configure and use S3cmd to interact with your S3 storage service (for a more detailed description of configuring and using S3cmd refer to the documentation provided at https://s3tools.org/s3cmd).

1. Locate certificate file:

   ```
   $ python -c "import certifi; print(certifi.where())"
   ```

2. Configure s3cmd:

   ```
   $ python /sys$login/local/bin/s3cmd --configure - --ca-
   certs=<output_from_the_previous_command>
   ```

You can then run S3cmd as follows:

```
$ python /sys$login/local/bin/s3cmd <command>
```

It should be noted that S3cmd on VSI OpenVMS can currently only be used with files are in STREAM-LF format. This limitation may be removed in the future.