vms Software

Python for VSI OpenVMS I64

February 2021

1.   Introduction

Thank you for your interest in this port of Python for OpenVMS (I64). This release of Python for OpenVMS is based on the Python 3.8.2 Open Source distribution.

Python (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 of 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 such as VSI DATATRIEVE and the Oracle RDB database, 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 (https://pypi.org/) or elsewhere using the Python `PIP` package installer.

2.   What's new in this release

This release of Python 3.8.2 for VSI OpenVMS I64 provides various bug fixes and provides minor new functionality such as support for the RTL `LIB$SET_SYMBOL`, `LIB$GET_SYMBOL`, and `LIB$DELETE_SYMBOL` functions. The packages kit includes a number of new packages, including S3cmd, which can be used to interact with AWS S3-based cloud storage, and the WSIT package, which to implement Python-based interfaces to legacy OpenVMS-based application code. The ReportLab package (see https://www.reportlab.com/opensource/) for generating PDFs and graphics is also included in this release of the Python packages kit.

3.   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.

- VSI OpenVMS Version 8.4-1H1 I64 or higher

- 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)

- It is recommended that Python source files are of a record type of STREAM_LF

  ```
  $ SET FILE/ATTR=RFM=STMLF <_file_>
  ```

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.

4.   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.python3^.8.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.

5.   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:

- `VSI-I64VMS-PYTHON-B0308-2-1.PCSI$COMPRESSED`

- `VSI-I64VMS-PYTHWHLS-A0000-5-1.PCSI$COMPRESSED`

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.

5.1.   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 I64VMS PYTHON B3.8-2              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 I64VMS PYTHON B3.8-2: Python for OpenVMS is based on
Python Version 3.8.2

    (C) Copyright 2020 VMS Software Inc.

    VSI Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS PYTHON B3.8-2              DISK$I64SYS:[VMS$COMMON.]

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

The following product has been installed:
    VSI I64VMS PYTHON B3.8-2              Layered Product

VSI I64VMS PYTHON B3.8-2: Python for OpenVMS is based on Python
Version 3.8.2 ()

    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'

    To shut down the Python runtime at system shutdown time, add
    the following lines to SYS$MANAGER:SYSHUTDWN.COM:

        $ file := SYS$STARTUP:PYTHON$SHUTDOWN.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]python3.exe
$ set default python$root:[lib.python3^.8]
$ 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]PYTHON3.EXE
```

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).

5.2.   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 I64VMS PYTHWHLS A0.0-5             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 I64VMS PYTHWHLS A0.0-5: Python wheels collection for
OpenVMS

    (C) Copyright 2020 VMS Software Inc.

    VSI Software Inc.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS PYTHWHLS A0.0-5             DISK$I64SYS:[VMS$COMMON.]

Portion done: 0%...10%...60%...70%...90%...100%

The following product has been installed:
    VSI I64VMS PYTHWHLS A0.0-5             Layered Product
```

errors may be safely ignored and will be eliminated in figure releases.

```
VSI I64VMS PYTHWHLS A0.0-5: Python wheels collection for OpenVMS ()

   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$startup:wheels$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, 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.

6. 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 such as VSI DATATRIEVE and the

Oracle RDB database, 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	rdb
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
dtr (DATATRIEVE)	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$root:[lib.python3^.8.test]`, and several examples for the OpenVMS RMS extension can also be found in `python$root:[lib.python3^.8.vms]`.

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/, the Python Package Index repository.

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.

Automat-20.2.0	idna-2.10	pytest_httpbin-1.0.0
Brotli-1.0.9	incremental-17.5.0	pytest_httpserver-0.3.6
CherryPy-18.6.0	iniconfig-1.1.1	pytest_localserver-0.5.0
Cython-3.0	iso8601-0.1.13	python_dateutil-2.8.1
Flask-1.1.2	itsdangerous-1.1.0	python_interface-1.6.0
Jinja2-2.11.2	jaraco.classes-3.1.0	python_magic-0.4.18
MarkupSafe-1.1.1	jaraco.collections-3.0.0	pytz-2020.4
Pillow-8.2.0	jaraco.functools-3.0.1	raven-6.10.0
PyYAML-5.3.1	jaraco.text-3.2.0	reportlab-3.5.59
Pygments-2.7.2	jmespath-0.10.0	requests-2.24.0
Werkzeug-1.0.1	jsonpointer-2.0	responses-0.12.0
apispec-4.0.0	jsonschema-2.5.1	rfc3987-1.3.8
apispec_webframeworks-0.5.2	jsonschema-3.2.0	rsa-4.5
appdirs-1.4.4	marshmallow-3.9.1	s3cmd-2.1.0
async_generator-1.10	mistune-0.8.4	s3transfer-0.3.4
attrs-20.3.0	mock-1.3.0	semver-2.13.0
awscli-1.18.192	mock-4.0.2	service_identity-18.1.0
behave-1.2.5	more_itertools-8.6.0	setuptools-50.3.2
betamax-0.8.1	nose-1.3.7	setuptools_scm-4.1.2
blinker-1.4	numpy-1.21.0	simplejson-3.17.2
botocore-1.19.32	outcome-1.0.1	six-1.15.0
bottle-0.12.18	packaging-20.4	sniffio-1.2.0
certifi-2020.11.8	packaging-20.8	sortedcontainers-2.3.0
cffi-1.14.3	parse-1.19.0	strict_rfc3339-0.7
chardet-3.0.4	parse_type-0.5.2	suds_py3-1.4.1.0
cheroot-8.4.5	path-15.0.0	sure-1.4.11
click-7.1.2	pbr-5.5.1	tempora-4.0.1
colorama-0.4.3	pika-1.1.0	teradata-15.10.0.21
constantly-15.1.0	pluggy-0.13.1	toml-0.10.2
cryptography-3.2	portend-2.6	tox-2.9.1
cryptography_vectors-3.2.1	prance-0.19.0	trustme-0.6.0
decorator-4.4.2	pretend-1.0.9	urllib3-1.25.11
distlib-0.3.1	py-1.10.0	validate_email-1.3
docutils-0.15.2	py-1.9.0	virtualenv-20.4.0
elementpath-2.0.4	pyOpenSSL-20.0.1	wcwidth-0.2.5
filelock-3.0.12	pyasn1-0.4.8	wheel-0.24.0
flasgger-0.9.5	pyasn1_modules-0.2.8	wheel-0.35.1
flex-6.14.1	pycparser-2.20	wsit-1.0.17
freezegun-1.0.0	pyparsing-2.4.7	xmlschema-1.2.5
ftputil-4.0.0	pyrsistent-0.17.3	zc.lockfile-2.0
httpbin-0.9.2	pytest-6.0.1	
hypothesis-5.35.3	pytest_flask-1.1.0	

It is hoped that future releases of Python for OpenVMS will 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.

6.1.   *A note on using S3cmd*

As noted previously, this release of the Python packages kit includes the S3cmd package. 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.