vms Software

Python for OpenVMS I64 (BETA RELEASE)

January 2020

Note: This is a Beta release of Python 3.5 for VSI OpenVMS. It is likely that this kit will be superseded by a newer version of Python in the near future. Every attempt will be made to ensure maximum compatibility between these releases, within the constraints of any differences between the Python versions.

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 3.5.0 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 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 includes the core binary distribution and several extensions, including modules for web application development, a simple interface to the Oracle RDB database, and various other useful extensions (listed below). It is anticipated that additional modules will be included in future releases.

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.

- 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.

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 numerous other documents that provide useful information on developing applications using Python.

4. Installing the kit

The kit is provided as an OpenVMS PCSI kit (`VSI-I64VMS-PYTHON-V0305-0A-1.PCSI`) that 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):

```
Performing product kit validation of signed kits ...

The following product has been selected:
    VSI I64VMS PYTHON V3.5-0A              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 V3.5-0A: Python for OpenVMS is based on
Python Version 3.5.0

    © Copyright 2019 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 V3.5-0A
DISK$IA21SYSDISK:[VMS$COMMON.]

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

The following product has been installed:
    VSI I64VMS PYTHON V3.5-0A              Layered Product

VSI I64VMS PYTHON V3.5-0A: Python for OpenVMS is based on Python
Version 3.5.0

    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 shutdown 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'
```

4.1. Post-installation steps

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. It is assumed that the runtime environment has been previously defined as described above by running `PYTHON$STARTUP.COM`. Compilation should take less than a minute on most systems.

```
$ set process/parse_style=extended
$ python :== $python$root:[bin]python.exe
$ set default python$root:[lib.python3^.5]
$ 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
```

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. What is missing

There are a number of components that are either missing in this beta release or that do not function correctly. Most notably, the Python pip package installation facility will not function correctly, particularly for packages that include 3GL code. It is anticipated that this and other related issues will be resolved in future releases.

However, the kit does include the C header files that are required for the implementation of custom dynamic modules and it is therefore possible to manually build and install extensions that include native code.

6. Included modules

As noted previously, this release of Python for OpenVMS includes the core binary distribution and several extensions. Included additional modules are listed below.

- cheroot

  A high-performance, pure-Python HTTP server used by the CherryPy web development framework. See https://github.com/cherrypy/cheroot for additional information.

- cherrypy

  A proven Python-based object-oriented HTTP framework for the development small or large web applications. See https://github.com/cherrypy/cherrypy for more details.

- click

  A Python package for creating command line interfaces with as little code as necessary (not fully supported on OpenVMS). See https://github.com/pallets/click.

- flask

  A micro-web framework written in Python. See http://flask.pocoo.org/ for additional information.

- ftputil

  The ftputil library is a high-level interface to the standard Python ftplib module. See https://ftputil.sschwarzer.net/trac for examples and additional information.

- itsdangerous

  Various helper classes and methods to pass data to untrusted environments and to get it back safely. Data is cryptographically signed to ensure that a token has not been tampered with. See https://github.com/pallets/itsdangerous.

- jinja2

  A templating language for Python developers that can be used to create HTML, XML, or other mark-up formats that are returned via an HTTP request. See http://jinja.pocoo.org/ for details.

- markupsafe

  MarkupSafe implements a text object that escapes characters so it is safe to use in HTML and XML. See https://palletsprojects.com/p/markupsafe/ for details.

- pika

  Pika is a pure-Python implementation RabbitMQ AMQP 0-9-1 client library. For additional information see https://pika.readthedocs.io and https://www.rabbitmq.com/.

- portend

  Portend is a Python package that can be used to monitor TCP ports for bound or unbound states. See https://pypi.org/project/portend/ for additional information.

- pytz

  World time zone definitions for Python. Facilitates accurate and cross platform time zone calculations and solves the issue of ambiguous times at the end of daylight saving time. See http://pytz.sourceforge.net/ for more information.

- requests

  Requests is an HTTP library intended to make HTTP requests simpler and more human-friendly too implement in Python. See http://docs.python-requests.org/en/master/ for more information.

- six

  Provides simple utilities for wrapping over differences between Python 2 and Python 3 and is intended to support codebases that work on both Python 2 and 3 without modification. See https://pythonhosted.org/six/.

- tempora

  Objects and routines pertaining to date and time operations including measuring, profiling, and event scheduling. See https://tempora.readthedocs.io/en/latest/.

- werkzeug

  A powerful and widely used WSGI utility library for Python. See http://werkzeug.pocoo.org/ or additional details.

Additionally there is a simple Python interface for Oracle RDB and interfaces to various OpenVMS System Services, RMS, and RTL routines. Many of these modules are still under development and are subject to change. It is hoped that future releases of Python for OpenVMS will include a more comprehensive suite of such modules and accompanying documentation and example programs.